CONVERTING FROM COBOL/3000 TO COBOL II/3000

Anne Vermilion
Hewlett-Packard General Systems Division

INTRODUCTION

COBOL II/3000 is a new COBOL compiler offered by HP, based on the ANSI 1974 standard. It contains many important new features while at the same time retaining all the functional capability of COBOL/3000. In addition, many compute-bound applications may execute significantly faster and have smaller data areas when compiled under COBOL II. Therefore, users of COBOL/3000 should find it advantageous to convert existing programs to COBOL II. To aid in this conversion, HP has provided a utility, COBCNV, that flags elements in a COBOL/3000 program that require conversion. This utility, use of a new COPY library editor, and some of the new features in COBOL II will be discussed in this paper.

USING THE COBCNV SYNTAX ANALYSIS PROGRAM

The utility program COBCNV checks the syntax of COBOL/3000 programs and flags items requiring conversion for COBOL II. Running the utility is similar to doing a COBOL/3000 compile using the :RUN COBOL option. File equations for text, master, copylib, and list files may be given, as required. These files must be equated to CNVTEXT, CNVMAST, CNVLIB, and CNVLIST, respectively. The PARM option of the :RUN command is used to communicate to COBCNV which file equations have been specified. (Please refer to the COBOL/3000 to COBOL II CONVERSION GUIDE.)

COBCNV will produce a source listing (unless $CONTROL NOLIST is specified) and flags items requiring conversion with a message and a pointer to the previously flagged item for conversion (if any). A summary of the number of items requiring conversion is printed at the end of the listing. The listing, messages, and summary are very similar to what is produced by a COBOL/3000 compile. (See APPENDIX A for sample output from COBCNV).

There are four types of changes that may be required when converting from COBOL/3000 to COBOL II. These are:

(1) Changes in COBOL statements
(2) Changes in reserved words
(3) Changes in commenting conventions
(4) Changes in segmentation and segment names

In addition, non-KSAM copylibs must be converted to KSAM. Items (1) through (3) are checked by COBCNV and will be discussed below, along with changes in segmentation. Conversion of copylibs will be discussed under the topic of the COPY library editor.

CHANGES IN COBOL STATEMENTS

Some generally minor changes may be required in the ENVIRONMENT, DATA, AND PROCEDURE DIVISIONs of COBOL/3000 programs. Most of the changes are for constructs which occur only infrequently, such as specifying "L", "/", or "=" as a substitute currency symbol. Also, most conversions are simple to make and do not require changes in program logic or reprogramming. There are two exceptions: one is a change in the OCCURS integer-1 TO integer-2 TIMES clause and the other involves index data items in record descriptions.

The 1974 standard for COBOL requires that if the OCCURS clause has the variable option (OCCURS integer-1 to integer-2 TIMES) then

(1) there must be a DEPENDING ON clause, and
(2) integer-1 must not be zero.

This may or may not require changes in program logic, depending on how the program data is structured.

Another change that could cause problems has to do with generation of slack bytes for index data items in record descriptions. COBOL II will align index data items on word boundaries for efficiency reasons, making the item look like a SYNCHRONIZED unsigned COMPUTATIONAL integer. In COBOL/3000, the index data item was not SYNCHRONIZED unless the SYNCHRONIZED clause was explicitly given for the item. This change will not generally cause difficulty in converted programs, unless there is an un-SYNCHRONIZED index data item preceded by an odd number of bytes in a file record. In this case, data carried in a file may be misaligned with the record description.

CHANGES IN RESERVED WORDS

Changes in the list of reserved words is the most likely source of incompatibility between COBOL/3000 and COBOL II. There are 107 new reserved words in COBOL II, some of which may appear as names of data items in COBOL/3000 programs. Some new COBOL II reserved words (these are part of the ANSI 1974 standard) are: DATE, TIME, START, MESSAGE, LENGTH, TERMINAL, and TEXT. If you are still coding in COBOL/3000 but anticipate converting to COBOL II in the future, be sure that you don't use these words as data names. A complete list of new COBOL II reserved words is given in APPENDIX B. A simple name change should make the item compatible.

CHANGES IN COMMENT CONVENTIONS

In the ANSI 1974 standard, the REMARKS paragraph and the NOTE paragraph and sentence were eliminated. These were replaced by the convention of simply putting an asterisk in column 7 of any line to be considered as a comment. This commenting facility was available in COBOL/3000, but the REMARKS and NOTEs were still allowed. COBOL II allows REMARKS but does not recognize NOTE paragraphs and sentences. NOTEs should be converted by using the asterisk in column 7 of each comment line.

SEGMENTATION

Changes in segmentation conventions may occasionally affect existing stream files that invoke the SEGMENTER following a COBOL compile, or may occasionally produce code segments that are too large to be run. The segmentation changes are:

(1) All sections with the same segment or priority number (in the SECTION header) will be put into the same segment.
(2) The initialization module produced by the compiler will share the same segment as the last (or only) PROCEDURE DIVISION segment.
(3) The name of the initialization module for the main program will not contain a trailing apostrophe (').

If a converted program produces a code segment that is too large, the segmenter may be used to resegment the program, or the segment numbers in the source may be changed and the program recompiled. STREAM files that use the old naming convention for the initialization module should be modified.

CONVERSION SUMMARY

The utility COBCNV.PUB.SYS may be used to check COBOL/3000 programs for conversion to COBOL II. Experience so far indicates that most COBOL/3000 programs require little, if any, conversion. The most common item requiring conversion is use of data names that appear in the new COBOL II reserved word list. A great deal of effort has been expended to make COBOL II highly compatible with COBOL/3000. It is hoped that the conversion task in most installations will cause a minimum of time and effort.

THE COBEDIT COPY LIBRARY EDITOR

The ANSI 1974 COBOL standard allows for the use of a number of COPY libraries with several modules or "text-names" within each library. One format of the new COPY statement is:

```
COPY text-name OF library-name
     REPLACING identifier-1 BY identifier-2.
```

The text-name is one module or set of records within the file library-name. The COPY statement may appear anywhere within a program and does not cause the problems with periods sometimes encountered with the ANSI 1968 COPY statement. (See p. 8-12 of COBOL/3000 reference manual.)

The COBEDIT utility allows users to create, edit, and delete COPY libraries and modules within these libraries. To use COBEDIT to edit a module or library, the library must be a KSAM file with columns 73 to 80 containing the text-name (module name). COPY libraries in regular MPE file format are easily converted to this format using only the BUILD command in COBEDIT. Commands available in COBEDIT are:

BUILD	Builds a new COPYLIB file.
COPY	Copies modules into a library.
EDIT	Edits a module to be added to a library.
EXIT	Leave the COBEDIT program.
HELP	Briefly explains all the COBEDIT commands.
KEEP	Adds a module to a library.
LIBRARY	Activates an already existing COPYLIB file
LIST	Lists contents of a module or lists all module names in a library.
PURGE	Purges a module of a library, or purges a library.
SHOW	Shows the name of a library, its key file and the latest module accessed.

An example of the use of COBEDIT to convert an old COPYLIB file is given in APPENDIX C.

NEW FEATURES IN COBOL II

There are a number of useful features in COBOL II that are not available in COBOL/3000. The 1974 ANSI Standard provides a number of features including string handling, two new I-O modules, alternate collating sequences, and new signed display data types. COBOL II provides some additional features that should prove useful. These include:

(1) An expanded CALL statement that allows MPE intrinsics to be called directly. It also allows passing byte addresses of data items (for example, when a parameter of an intrinsic or SPL procedure must be a byte array), passing null parameters (for SPL option variable procedures), and for passing data items by value.

(2) Condition-codes can also be checked using the new COBOL II reserved word CONDITION-CODE. This is often useful after calling MPE intrinsics.

(3) Octal literals may be specified by preceding an octal number with a per cent sign (%).

(4) ACCEPT FREE can be used to enter low volume free format data from the SYSIN device or the console. Unlike ACCEPT without the FREE option, the number of characters input does not have to exactly equal the length of the data item, and COMP, COMP-3, and NUMERIC edited data types are allowed. Conversion takes place for numeric items as in elementary moves. Numeric data items are aligned on the decimal point with zero fill taking place. No conversion is done for alphanumeric data types (ie. no alphanumeric editing), but right or left justification with blank fill is done.

(5) Printer tape channel controls, denoted by the reserved words C01 through C12, allow control of spacing on line printers as described in the MPE intrinsic FWRITE. C01 is equivalent to tape channel 1 (page-eject), C02 to tape channel 2 (skip to bottom of form), etc. For example, the following could be coded:

```
ENVIRONMENT DIVISION.
      .
      .
      .
SPECIAL-NAMES.
     MID-PAGE-POS IS C06.
      .
      .
      .
PROCEDURE DIVISION.
      .
      .
      .
WRITE TITLE-REC AFTER ADVANCING MID-PAGE-POS.
```

This program segment would print a title half-way down a page after doing a page-eject.

(6) A macro facility has been added that allows a name to be associated with a string of text. For example, if you use a long sequence many times in your code, and want to use a shorthand notation, you may wish to define and then invoke a macro. The advantages of a macro over a COPY statement are that an extra file does not have to set up for something that may be used in only one program, and parts of words or literals may be replaced.

An example of a simple macro definition is:

```
$DEFINE %DIV = DIVISION#
```

This macro could be invoked by the following:

```
DATA %DIV.
```

which would expand to

```
DATA DIVISION.
```

Macros with parameters can also be defined.

(7) Several debugging aids are available. These include a mapping of verbs to their location in

object code ($CONTROL VERBS) and a cross reference facility ($CONTROL CROSSREF).

SUMMARY

In summary, converting an existing program to COBOL II may allow the user to realize runtime performance improvement and makes available improved facilities for program enhancement and maintenance. Programmers writing new applications in COBOL II will find a wide range of new features available to them, and should find COBOL programming more productive and efficient.

APPENDIX A. SAMPLE OUTPUT FROM UTILITY COBCNV

The following very simple program demonstrates the type of output to be expected when running the utility COBCNV as an aid in conversion to COBOL II.

```
:FILE CNVTEXT=DEMO
:RUN COBCNV.PUB.SYS;PARM=1


PAGE 1 HP COBOL CONVERSION 32233A.00.00 (C) HEWLETT-PACKARD


          001100 IDENTIFICATION DIVISION.
          001200 PROGRAM-ID. DEMO.
          001300
          001400 ENVIRONMENT DIVISION.
          001500
          001600 DATA DIVISION.
          001700
          001800 PROCEDURE DIVISION.
          001900 START.
                         ^
====== CONVERSION WARNING #807  A COBOL II RESERVED WORD HAS
                                BEEN USED.
          002000      DISPLAY "THIS IS A DEMO".
          002100 FINISH.
          002200      STOP RUN.

    CPU TIME = 0:00:00.  WALL TIME = 0:00:04.
    END COBOL CONVERSION CHECKER. 001 CONVERSION WARNING.
    NO COBOL'68 ERRORS.  NO COBOL'68 WARNINGS.
    SEE 001900

END OF PROGRAM
```

APPENDIX B. NEW COBOL II RESERVED WORDS

ALSO
BOTTOM
C01
C02
C03
C04
C05
C06
C07
C08
C09
C10
C11
C12
CANCEL
CD
CHARACTER
CODE-SET
COLLATING
COMMUNICATING
CONDITION-CODE
CONDITIONALLY
COUNT
DATE
DAY
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DELETE
DELIMITED
DELIMITER
DESTINATION
DISABLE
DUPLICATES
DYNAMIC
EBCDIC
EGI
EMI
ENABLE
END-OF-PAGE
EOP
ESI
EXCEPTION
EXCLUSIVE
EXDATE
EXTEND
FREE
INITIAL
INSPECT
INTRINSIC
LABELS
LENGTH
LINAGE
LINAGE-COUNTER
MERGE
MESSAGE
NATIVE
NOLIST
ORGANIZATION
OVERFLOW
POINTER
PRINTING
PROCEDURES
QUEUE
RECEIVE
REFERENCES
RELATIVE
REMOVAL
REWRITE
SEGMENT
SEND
SEPARATE
SEQ
SEQUENCE
SORT-MERGE
STANDARD-1
START
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUPPRESS
SW0
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SYMBOLIC
TABLE
TERMINAL
TEXT
TIME
TRAILING
UN-EXCLUSIVE
UNSTRING
VOL
WHEN-COMPILED

APPENDIX C. SAMPLE USE OF COPYLIB EDITOR COBEDIT.

A non-KSAM COPYLIB must be converted to a KSAM file to be compatible with COBOL II. This is easily done using the BUILD command of COBEDIT. The following example demonstrates how to convert an existing COPYLIB file named OLDLIB. All commands and responses issued by the user are displayed in all capital letters.

```
:RUN COBEDIT.PUB.SYS

HP32233A.00.00 COPYLIB EDITOR - COBEDIT JAN 8, 1980, 11:23AM
(C) HEWLETT-PACKARD CO. 1979

Type "HELP" For a list of commands.

>BUILD NEWLIB
Name a key file to be used with Newlib:  NEWLIBK
Library file created.
To copy a file into Newlib now, enter the file name.
File name? OLDLIB
Is the file in copylib format? YES
4 records copied to library file.
Do you wish to copy more files?
Respond Yes or No:  NO
Requested file(s) copied.

>EXIT

End of program
```

If there is more than one module in the old copylib file, it is very important to respond 'YES' when asked if the file is in copylib format; otherwise, you will be prompted for a text-name and columns 73-80 of every record in the new library will contain the text-name you supply to the prompt.

When running COBOL II, the user in the above example could either issue the file equation

```
:FILE COPYLIB=NEWLIB
```

or purge OLDLIB and rename NEWLIB to OLDLIB.