ANSI COBOL 198X: A SNEAK PREVIEW

Greg Gloss
Hewlett-Packard Information Systems Division

ABSTRACT

The ANSI (American National Standards Institute) X3J4 technical committee is in the final stages of work on the next version of the COBOL standard. This paper will discuss some of the major new features which are expected to be included such as structured programming constructs together with those items which will no longer be allowed in the next standard. The standardization process will also be covered briefly.

BACKGROUND

The current version of ANSI COBOL was adopted in 1974. Since 1977, the ANSI X3J4 committee has been working on the next version of the COBOL standard. Since it is not clear how long this process will take, I will refer to the next standard as COBOL '8X.

OVERVIEW

The next standard will have changes in the following categories:

a. New Features
b. Transitional Features
c. Deleted Features
d. Specification Changes
e. New Reserved Words

A significant effort has been put into incorporating structured programming constructs into COBOL. In addition, other new facilities have been added to make programming in COBOL easier. Some current features have been flagged for deletion either in the new standard or in the subsequent standard. Those features which are in the new standard, but which are not expected to be in the subsequent standard are called transitional. There have also been some changes to the rules and additional reserved words included which may affect existing programs.

STRUCTURED PROGRAMMING

The new structured programming constructs which have been defined for COBOL include Scope Terminators, PERFORM statement enhancements, the EVALUATE statement, and the CONTINUE statement.

Scope Terminators

Under COBOL '74, conditional statements could not be included with the statement group following a conditional phrase such as

AT END or ON SIZE ERROR. New reserved words have been added such that any conditional statement can be turned into an imperative statement and used as part of the conditional statement group. For example,

```
READ FILE-IN AT END
   ADD A TO B ON SIZE ERROR
      PERFORM OVERFLOW-ROUTINE
   END-ADD
   MOVE SPACES TO REC-IN.
```

Under COBOL '74, it is not legal to specify the ON SIZE ERROR phrase in the above example because it turns the ADD statement into a conditional statement and only imperative statements are allowed following the AT END phrase. However, with the scope terminator, END-ADD, the ADD statement with the SIZE ERROR option becomes an imperative statement and is legal in this situation. The MOVE statement is the second imperative statement to be executed if the AT END branch is taken and the period terminates the READ statement. If the READ itself were nested under a conditional such as an IF, it would be terminated by a END-READ instead of the period.

PERFORM Statement Enhancements

The PERFORM statement has been enhanced to allow a list of imperative statements to be embedded within the statement instead of paragraph names and to allow the programmer to specify whether the UNTIL conditions are to be tested before or after the specified set of statements has been executed.

An example of an in-line PERFORM is shown below:

```
PERFORM 10 TIMES
   ADD A TO B
   ADD 1 TO A
END-PERFORM.
```

The two ADD statements will be executed 10 times.

Under COBOL '74, the UNTIL conditions are always tested before executing the specified paragraphs. The new specifications will allow the test to be made afterwards. For example,

```
PERFORM READ-LOOP
   WITH TEST AFTER
   UNTIL EOF-FLAG.
```

Control will always transfer to READ-LOOP at least once. The test option may also be specified with an in-line PERFORM.

EVALUATE Statement

The EVALUATE statement adds a multi-condition CASE construct to COBOL. The EVALUATE statement causes a set of subjects to be evaluated and compared with a set of objects. If the comparisons are all true, a specified group of statements is executed. For example,

```
EVALUATE HOURS-WORKED EXEMPT
   WHEN 0           ANY  PERFORM NO-PAY
   WHEN 1  THRU 40  ANY  PERFORM REG-PAY
   WHEN 41 THRU 80  "N"  PERFORM OVERTIME-PAY
   WHEN 41 THRU 80  "Y"  PERFORM REG-PAY
   WHEN OTHER            PERFORM PAY-ERROR.
```

The above example evaluates two data items, HOURS-WORKED and EXEMPT. If HOURS-WORKED is 0, any value for EXEMPT will be true and NO-PAY will be performed. If HOURS-WORKED is between 1 and 40, REG-PAY will be performed. If HOURS-WORKED is between 41 and 80 and EXEMPT contains "N", OVERTIME-PAY will be performed. If HOURS-WORKED is between 41 and 80 and EXEMPT contains a "Y", REG-PAY is performed. If none of the above conditions are true, PAY-ERROR is executed.

CONTINUE Statement

The CONTINUE statement is a no operation statement which indicates that no executable statement is present. It may be used anywhere a conditional statement or an imperative statement may be used. For example,

```
IF A < B THEN
   IF A < C THEN
      CONTINUE
   ELSE
      MOVE ZERO TO A
   END-IF
   ADD B TO C.
SUBTRACT C FROM D.
```

The CONTINUE statement allows control to go to the ADD statement following the IF when A is less than C. If the NEXT SENTENCE option had been used, control would have transferred to the SUBTRACT statement instead.

OTHER NEW FEATURES

There is a long list of other new features which should make the job of the COBOL programmer easier. The more significant ones are listed here.

Reference Modification

Reference modification allows you to reference a portion of a data item by specifying a leftmost character position and a length. For example,

```
MOVE A (3:5) TO B.
```

will move the third through seventh characters of A to B.

INITIALIZE Statement

The INITIALIZE statement provides the ability to set selected types of data fields to predetermined values. Assume RECORD-1 was described as follows:

```
01  RECORD-1.
    05  EMP-NO     PIC 9(6).
    05  EMP-NAME   PIC X(20).
    05  EMP-PAY    PIC 9(5)V99.
    05  JOB-TITLE  PIC X(20).
```

The following INITIALIZE statements in the Procedure Division could be used to put values into the record:

```
INITIALIZE RECORD-1 REPLACING NUMERIC BY ZERO.
INITIALIZE RECORD-1 REPLACING ALPHANUMERIC BY SPACES.
```

The effect would be the same as:

```
MOVE ZERO TO EMP-NO EMP-PAY.
MOVE SPACES TO EMP-NAME JOB-TITLE.
```

De-editing

Under COBOL '74, it is not legal to move from an edited field to a numeric or numeric edited field. The new specifications will allow moving from a numeric edited item to either a numeric or numeric edited item. The edited item which is the sending item will be converted to its numeric value and moved to the receiving field.

REPLACE Statement

The REPLACE statement function is similar to that of a COPY... REPLACING except that the REPLACE statement operates on all source program text, not just text in libraries. Thus, if one of the new reserved words is used heavily in an existing program, you may want to use a REPLACE statement to change it. For example,

```
REPLACE ==TEST== BY ==TESTT==
```

will replace all subsequent occurrences of TEST by TESTT in the source program until another REPLACE statement, a REPLACE OFF statement, or the end of the source program.

Optional FILLER

The word FILLER is now optional for data items which need not be named.

```
01  A.
    05  B  PIC X(5).
    05     PIC X(5) VALUE "NAME:".
```

INITIAL Attribute

The INITIAL clause in the PROGRAM-ID paragraph indicates that every time the program is called, the internal data is initialized. This function is the same as the $CONTROL DYNAMIC option on the HP-3000.

```
PROGRAM-ID.  SUB-PROG  INITIAL.
```

EXTERNAL Attribute

The EXTERNAL clause specifies that a data item or file is available to every program in the run unit which describes the data item or file.

```
FD  FILE-1  IS EXTERNAL.
```

SYMBOLIC CHARACTERS Clause

The SYMBOLIC CHARACTERS clause in the SPECIAL-NAMES paragraph of the Environment Division allows the programmer to equate a name to a specific character. This feature can be useful for non-printable characters. For example,

```
SYMBOLIC CHARACTERS BELL IS 7, CARRIAGE-RETURN IS 13.
```

This clause would allow a MOVE statement such as

```
MOVE BELL TO A.
```

ADD Statement Enhancement

Under COBOL '74, the ADD statement allows either a TO or a GIVING format, but a statement of the form

```
ADD A TO B GIVING C
```

is not allowed. The new specifications will allow the TO before the last operand when the GIVING option is used.

Alphabetic Tests

Two new alphabetic class tests have been defined:

1. ALPHABETIC-UPPER will be true if the data item being tested contains only A-Z and spaces.

2. ALPHABETIC-LOWER will be true if the data item being tested contains only a-z and spaces.

TRANSITIONAL CATEGORY

There are some features of the current standard which are scheduled for a phased deletion. Implementations must still support these features in the new standard, but not in the subsequent standard. Certain clauses have been moved from the file control entry in the Environment Division to the file description entry in the Data Division and vice versa. The old locations are specified as transitional elements so implementations of the new standard must support programs which contain the clauses in either the old or the new locations. The following Environment Division clauses are included in the transitional category:

```
FILE-STATUS
RECORD KEY
ALTERNATE RECORD KEY
ACCESS MODE
```

The following Data Divison clauses are included in the transitional category:

```
BLOCK CONTAINS
CODE-SET
```

The Identification Division paragraphs are included in the transitional category in favor of the more general comment facility (* in column 7).

The INSPECT...TALLYING...REPLACING format of the INSPECT statement is included in the transitional category since the same function can be accomplished with two separate INSPECT statements.

DELETED FEATURES

The following features are not included in the next standard:

1. The ALTER statement.
2. The ENTER statement.
3. The MEMORY SIZE clause.

OTHER CHANGES

New status code values for file errors are being defined. These codes will cover situations which violate the standard but for which no standard status code was defined. For example, trying to open an indexed file in a program which declares it to be a relative file.

The order of the steps in a multi-conditional PERFORM...VARYING statement has been changed. Under COBOL '74, the statement

```
PERFORM PAR-1 VARYING I FROM 1 BY 1 UNTIL I>10
              AFTER   J FROM I BY 1 UNTIL J>10
```

would set I to 1 and vary J from 1 to 10 and then set J to 1, increment I to 2 and vary J until 10. The new specifications will increment I to 2 before setting J to I. Thus, on the second cycle, J will vary from 2 to 10 instead of 1 to 10 as under COBOL '74.

The new reserved word ALPHABET is required in the alphabet clause of the SPECIAL-NAMES paragraph.

```
ALPHABET ASCII IS STANDARD-1.
```

RESERVED WORDS

The following new reserved words have been added:

ALPHABET	END-DELETE	END-UNSTRING
ALPHABETIC-LOWER	END-DIVIDE	END-WRITE
ALPHABETIC-UPPER	END-EVALUATE	EVALUATE
ALPHANUMERIC	END-IF	EXTERNAL
ALPHANUMERIC-EDITED	END-MULTIPLY	FALSE
ANY	END-PERFORM	INITIALIZE
CONTENT	END-READ	NUMERIC-EDITED
CONTINUE	END-RECEIVE	OTHER
CONVERSION	END-RETURN	PADDING
CONVERTING	END-REWRITE	REPLACE
DAY-OF-WEEK	END-SEARCH	STANDARD-2
END-ADD	END-START	TEST
END-CALL	END-STRING	TRUE
END-COMPUTE	END-SUBTRACT	

STANDARDIZATION PROCESS

There are two committees which work on defining COBOL. The CODASYL COBOL Committee has the responsibility of developing the language. The ANSI X3J4 committee has the responsibility of standardizing the language. When working on a new standard, X3J4 can adopt specifications from either the previous standard or from the Journal of Development which reflects the work of the CODASYL COBOL Committee. If there is a problem with the JOD specifications, X3J4 must either subset the specifications from the JOD so that the problem does not appear in the standard or request that CCC resolve the problem. Both committees have representatives from implementors, users, and government. X3J4 currently has 23 members and holds six 4-day meetings each year. The work on the next standard is nearing completion as the committee is currently processing comments on the formal letter ballot of its members. Assuming that a two-thirds vote in favor the proposed draft is achieved, the document is forwaded to X3 who, in turn, votes to send it out for an official public comment period of at least four months. The X3J4 committee will review all comments received during this period. After all negative comments have been processed, the X3 committee votes on sending the draft proposed standard to ANSI to be formally processed as a new standard.

During the standard revision process, X3J4 has published information concerning its work in COBOL Information Bulletins. The latest one was CIB 19 which was published in May, 1980. Comments concerning the draft standard will be officially requested during the public review period; however, comments may be submitted earlier to:

Chairperson, X3J4
CBEMA
1828 L St. N.W.
Washington, D.C. 20036