Accredited Standards Committee
X3 INFORMATION PROCESSING SYSTEMS

Document No. CIB-25

Chairman, X3J4
CBEMA
311 First Street NW, Suite 500
Washington, D. C. 20001-2178
Tel: (202) 737-8888

COBOL INFORMATION BULLETIN

NUMBER 25

INTERPRETATIONS OF ANSI X3.23-1985

APRIL 1990

TABLE OF CONTENTS

SECTION 1: INTRODUCTION

1.1 COBOL INFORMATION BULLETINS

COBOL Information Bulletins (CIBs) are published by the Computer and Business Equipment Manufacturers Association (CBEMA) on behalf of the X3J4 COBOL Technical Committee to provide a method of interacting with COBOL users who are involved with ANSI COBOL X3.23-1985 and are interested in the work of X3J4. To comment on the work of X3J4, to receive notification of the availability of subsequent COBOL Information Bulletins, or to receive notification of the availability of draft COBOL standards for public review, contact:

Chairman, X3J4
CBEMA
311 First Street NW, Suite 500
Washington, D. C. 20001-2178

This COBOL Information Bulletin, CIB-25, contains the second group of interpretations to ANSI COBOL X3.23-1985 resulting from inquiries received and processed by X3J4 since publication of ANSI COBOL X3.23-1985. The first group of interpretations was published in CIB-24. A comprehensive index of all interpretations of ANSI COBOL X3.23-1985 begins on page 117.

Copies of CIB-24 and CIB-25 can be ordered from Global Engineering Documents, Inc., using telephone number (800) 854-7179 from within the USA, (714) 261-1455 from outside the USA, or fax number (714) 261-7892.

Copies of American National Standard COBOL X3.23-1985 can be obtained by corresponding directly with:

American National Standards Institute, Inc.
1430 Broadway
New York, New York 10018

1.2 X3J4 SCOPE AND PROGRAM OF WORK

An outline of the current scope and program of work of the X3J4 COBOL Technical Committee is being provided in order that the reader of this COBOL Information Bulletin can understand the type of work that X3J4 does and the tasks which it hopes to accomplish.

The scope of X3J4 can be divided into three distinct areas. The first part of the scope is to provide a mechanism for the solicitation and review of all activities regarding the current national COBOL Standard. The second part of the scope is to carry out the procedures necessary to maintain the continued responsiveness of the COBOL Standard to user needs. The third part of the scope calls for the continued support and publication of

the COBOL Information Bulletin to interact with the COBOL community and disseminate relevant information about the status of the COBOL Standard, and any clarifications which may impact implementation of COBOL compilers.

The program of work for X3J4 includes several activities of interest which are outlined below:

1. Support the current COBOL Standard by responding to inquiries about the Standard and requests for clarifications.

2. Support the COBOL language as defined in both the CODASYL COBOL Journal of Development and the COBOL Standard to determine what a subsequent revision to the current COBOL Standard might contain or, more importantly, what it might not contain.

3. Maximize compatibility of any future revision of the COBOL Standard with other American National Standards.

1.3 RELATIONSHIP BETWEEN X3J4 AND PARENT COMMITTEE X3

The X3J4 COBOL Committee is the technical committee for COBOL operating under the Accredited Standards Committee X3 on Information Processing Systems. X3 is a committee that has under it the standardization activities that are concerned with all aspects of information processing systems. The secretariat for X3, i.e., the body that supports the coordination and administration for both X3 and its related technical committees, is the Computer and Business Equipment Manufacturers Association (CBEMA).

1.4 RELATIONSHIP BETWEEN X3J4 AND INTERNATIONAL ORGANIZATIONS

Throughout the COBOL standardization activity, close liaison with various international groups has been maintained. In fact, the impetus for the entire undertaking of standardization in the area of computers and information processing can be traced directly to international sources. The American National Standards Institute represents the United States in the joint international committee of the International Standards Organization (ISO) and the International Electrotechnical Commission (IEC); this joint international committee is called ISO/IEC Joint Technical Committee 1 (JTC1), Information Processing Systems. ISO/IEC JTC1 Subcommittee 22 (SC22) is an ISO/IEC JTC1 subcommittee for application system environments and programming languages. Members of the X3J4 COBOL Technical Committee participate actively in meetings of the COBOL working group WG4 associated with ISO/IEC JTC1 Subcommittee 22. Thus, influence of and requirements for international considerations are present in the COBOL Standard.

1.5 RELATIONSHIP BETWEEN X3J4 AND CODASYL COBOL COMMITTEE

ANSI COBOL X3.23-1985 was derived from the CODASYL COBOL Journal of Development and the previous COBOL Standard, ANSI COBOL X3.23-1974. The CODASYL COBOL Journal of Development is the product of the CODASYL COBOL Committee (CCC), a separate distinct body from the X3J4 COBOL Technical Committee. Because these two entities are frequently confused in the public mind, it is appropriate to briefly describe each one and its function.

1.5.1 CODASYL COBOL Committee (CCC)

The CODASYL COBOL Committee is a committee within CODASYL (Conference on Data Systems Languages). It is the development body that both originates and maintains the specifications of the COBOL language. These are reflected in the CODASYL COBOL Journal of Development, which, as its name suggested, is a continually evolving document. The CODASYL COBOL Committee may be contacted by writing to:

Chairman, CODASYL COBOL Committee
50 Presidents Lane
Quincy, Massachusetts 02169

1.5.2 X3J4 COBOL Technical Committee

The X3J4 COBOL Technical Committee is concerned with standardizing COBOL from the output of the CODASYL COBOL Committee. X3J4 takes as its basic input the CODASYL COBOL Journal of Development as of a certain point in time and "freezes" it prior to molding it into a format suitable for a formal standard specification. X3J4 can be contacted by writing to:

Chairman, X3J4
CBEMA
311 First Street NW, Suite 500
Washington, D. C. 20001-2178

1.5.3 Impact of CODASYL COBOL on ANSI COBOL X3.23-1985

When the X3J4 COBOL Technical Committee updates the COBOL Standard, ANSI COBOL X3.23-1985, the current method of operation permits inclusion of additional language specifications only from the CODASYL COBOL Journal of Development. This is unlike the other X3 language standardization committees which perform both the development and the standardization functions.

SECTION 2: RESPONSES TO INTERPRETATION REQUESTS

2.1 INTRODUCTION

This section of the COBOL Information Bulletin contains information related to actions taken by X3J4 in regard to questions that have been asked concerning American National Standard COBOL X3.23-1985 since its publication in 1985. The purpose of providing this information is to keep the public informed as to the current thinking and philosophy of the committee in regard to questions or discussions which involve American National Standard COBOL X3.23-1985.

Each of the responses is presented in a form showing the reference in American National Standard COBOL X3.23-1985, the question, and the response of the X3J4 COBOL Technical Committee. The reference number prefixed by the letter A is the document number used by X3J4 to identify the response to the question.

If the response to a question results in X3J4 formulating a proposed correction to American National Standard COBOL X3.23-1985, then X3J4's interpretation document includes a cross reference to the proposed correction as documented in the X3J4 working document called the Draft Proposed Correction Addendum to ANSI X3.23-1985 (see Section 3 beginning on page 91).

An index of all interpretations made relative to American National Standard COBOL X3.23-1985 is located in Section 4 beginning on page 117. This index is arranged in order by subject matter within American National Standard COBOL X3.23-1985.

2.2 DISCLAIMER

Recognizing the need for a uniform approach to the responsibility for disseminating the interpretations to approved American National Standards, the Accredited Standards Committee on Information Processing Systems, X3, has authorized the publication of COBOL Information Bulletins.

These interpretations are issued in response to questions that have been raised regarding certain specifications contained in American National Standard COBOL, X3.23-1985.

These interpretations were prepared by X3J4, a technical committee of X3, that developed American National Standard COBOL X3.23-1985, and were authorized for release by X3 in order to provide interpretations as quickly as possible in response to questions raised.

These interpretations, while reflecting the technical opinion of X3J4, are intended solely as supplementary information to users of American National Standard COBOL X3.23-1985.

American National Standard COBOL X3.23-1985 was approved through the publication and voting procedures of the American National Standards Institute and is not altered by this bulletin. Any subsequent revision of American National Standard COBOL X3.23-1985 may or may not reflect the content of these interpretations.

X3J4 DOCUMENT A-6

SUBJECT: Assigning File to Internal Storage of Object Computer

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-7, paragraph 2.3.3, syntax rule 3, file control entry
2. Page VII-8, paragraph 2.3.4, general rule 3, file control entry
3. Page VIII-8, paragraph 2.3.3, syntax rule 3, file control entry
4. Page VIII-9, paragraph 2.3.4, general rule 4, file control entry
5. Page IX-8, paragraph 2.3.3, syntax rule 3, file control entry
6. Page IX-9, paragraph 2.3.4, general rule 5, file control entry
7. Page XI-2, paragraph 2.3.4, general rule 1, file control entry

DATE: April 9, 1986

QUESTION:

Is it allowed to assign a file, including a sort or merge file, to the internal storage of the object computer?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the Standard neither prohibits nor requires the capability to assign a file to the internal storage of the object computer (see references 1 and 2 for the Sequential I-O module; references 3 and 4 for the Relative I-O module; references 5 and 6 for the Indexed I-O module; reference 7 for the Sort-Merge module).

X3J4 DOCUMENT A-7

SUBJECT: USE BEFORE REPORTING Ambiguity

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page X-35, paragraph 5.6.4, general rules 1a, 1b, and 1c, USE BEFORE REPORTING statement
2. Page XIII-78, paragraph 4.9.3, syntax rule 2, USE BEFORE REPORTING statement

DATE: August 31, 1986

QUESTION:

Reference 2 states that identifier-1 must not appear in more than one USE BEFORE REPORTING statement. Does this mean one per compilation or one per contained program?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the same identifier-1 can be specified at most once in a containing program, and at most once, in each contained program. When the GLOBAL phrase was added to the USE BEFORE REPORTING statement, general rules 1a, 1b, and 1c were specified to designate the order of precedence for selecting a declarative when programs are contained within other programs. These rules indicate that the intent is that the same identifier-1 can be specified in both the containing and contained programs. Reference 2 limits the specification of identifier-1 to one occurrence for each containing or contained program.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-7. Page 104 of this document contains the proposed correction to page XIII-78 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-9

SUBJECT: Nested IF Statements and Scope Terminators

REFERENCES:

American National Standard COBOL X3.23-1985

1. Pages VI-90 and 91, paragraph 6.16, IF statement
2. Page IV-40, paragraph 6.4.3, scope of statements

DATE: October 2, 1986

QUESTION:

The statement format of the IF statement and the rules for statement termination seem to be ambiguous.

Given the following example:

Example 1a:

```
(1)  IF A = B THEN
(2)      IF C = D THEN
(3)        IF E = F THEN
(4)            MOVE X TO Y
(5)        ELSE
(6)            NEXT SENTENCE
(7)        END-IF
(8)      ELSE
(9)        MOVE P TO Q.
```

Example 1b:

```
(1)  IF A = B THEN
(2)      IF C = D THEN
(3)          IF E = F THEN
(4)              MOVE X TO Y
(5)          ELSE
(6)              NEXT SENTENCE
(7)      END-IF
(8)  ELSE
(9)      MOVE P TO Q.
```

Which of the indentations shown reflects the correct interpretation? Example 1a is probably what a "naive" user would expect. However, syntax rule 3 of the IF statement prohibits use of the END-IF together with the NEXT SENTENCE phrase, so that END-IF apparently must be associated with statement #2, rather than statement #3; this interpretation is shown in example 1b. (By the way, this is no problem for the CODASYL COBOL Journal of Development because the JOD allows the END-IF together with the NEXT SENTENCE phrase.)

It seems that interpretation 1b must be assumed as correct, but the consequences concerning some other rules of the IF statement and elsewhere turn out to be ambiguous, or at least unclear.

1. General Rule 1: The scope of the IF statement may be terminated by any of the following:

   a. An END-IF at the same level of nesting;

   b. A separator period;

   c. If nested, by an ELSE phrase associated with an IF statement at a higher level of nesting.

The rule does not say "only", but if it is not meant to be exhaustive, what is its purpose?

If it does mean "only", then statement #3 is not terminated by the END-IF, but by the ELSE phrase. Does that mean that the END-IF is: (a) invalid, or (b) part of statement #3? If yes, then the ELSE phrase #8 belongs to statement #2. If no, according to interpretation 1b, then how can the END-IF be part of statement #2, but still in the scope of statement #3?

2. General Rule 2: " ... control passes to the end of the IF statement." This is a change from the previous standard, where control always passed to the "next executable sentence". This change was necessary because of the introduction of the scope terminators. But how is "the end of the IF statement" defined?

If the rules for scope termination apply, the END-IF in example 1b, which is supposed to be the scope terminator of statement #2, would be part of statement #3; this is of course a conflict.

It could be attempted to define the "end of the statement" without referring to the scope termination rules, for example by the syntactical end of the statement as defined in the general format. In the case of the IF statement, this could, however, only be determined when the statement ends with the NEXT SENTENCE phrase.

3. General Rule 3: "IF statements within IF statements may be considered as paired IF, ELSE, and END-IF combinations, proceeding from left to right." (Question: also if they do not have the ELSE phrase, or do not have the END-IF?)

"Thus, any ELSE or END-IF is considered to apply to the immediately preceding IF that has not been already paired with an ELSE or END-IF."

Does that mean that the END-IF in example 1 has to be paired with statement #3, because that's the immediately preceding IF statement, and it has not yet been paired with an END-IF?

But how about the following:

Example 2:

```
(1)  IF A = B THEN
(2)      IF C = D THEN
(3)        ADD A TO B
(4)      ELSE
(5)        ADD C TO D
(6)  ELSE
(7)      ADD E TO F
(8)  END-IF.
```

Would the END-IF then have to be associated with statement #2 according to general rule 3? This does not make sense: it cannot be associated with statement #2 because statement #2 is terminated by #6, the next phrase of the containing statement. So general rule 3 should refer to "unterminated" IF statements only, but it doesn't. If we make this assumption, example 3 is okay, but how about example 1? There is nothing between statement #3 and the END-IF that terminates statement #3. Hence the END-IF belongs to statement #3. But this is of course invalid syntax because of syntax rule 3, so the whole example could be considered invalid.

Or should general rule 3 be implicitly expanded further to apply only if this is syntactically allowed? Then the END-IF would have to be paired with statement #2.

4. The third paragraph on page IV-40: "When any statement is contained within another statement, the next phrase of the containing statement following the contained statement terminates the scope of any unterminated contained statement."

Question: Is END-IF a phrase? The glossary definition seems to apply. If yes, is this not a contradiction to general rule 1?

5. The general format of the IF statement shows repetition periods after statement-1 and statement-2. According to syntax rule 1, "statement-1 and statement-2 represent either an imperative statement or a conditional statement optionally preceded by an imperative statement." The periods imply that this may be repeated. (Note that ANSI COBOL X3.23-1974 did not have the repetition periods.) Thus a conditional statement in the THEN and/or ELSE branch may be followed by another statement (which may even be another conditional statement). As far as I can see this can only happen when this series of statements consists of at least one IF (or WHEN) statement ending with the NEXT SENTENCE phrase, followed by an imperative or another conditional statement; this is because all other conditional statements are "open-ended", i.e. any statement following would be regarded as part of the ELSE (or WHEN) phrase.

Example 3:

```
(1)  IF A = B THEN
(2)      MOVE X TO Y
(3)  ELSE
(4)      IF C = D THEN
(5)        MOVE U TO V
(6)      ELSE
(7)        NEXT SENTENCE
(8)      DISPLAY "MESSAGE 123".
```

The indentation shows how this might be interpreted: #8 cannot be part of the ELSE phrase #6, according to the general format; but it can be part of the ELSE phrase #3, which consists of the IF statement #4, and the DISPLAY statement #8.

Here we have again the same question as in the previous case: where is the end of statement #4?

But apart from that, is #8 really part of the ELSE branch #3? The rules seem to leave no other choice. However, I strongly doubt that this was anybody's intent. This would be a substantive change of the ANSI X3.23-1974 language that is not listed in the appendix; it certainly does not improve the clarity of the language.

The list of questions of this kind could go on and on. Apparently it was felt that the NEXT SENTENCE phrase does not make sense in the structured programming environment, that's why syntax rule 3 was introduced. However, the case of nesting the two kinds of IF statements (those with NEXT SENTENCE and those with scope terminators) was not taken into account, and some of the rules don't fit together anymore.

These are the key questions to be answered:

1. Does the syntactical exhaustion of the statement format implicitly terminate the scope of a statement?

2. What is the "end of a statement"?

3. Does general rule 3 only apply to statements that are:

   a. not yet terminated? (depending on the previous question)

   b. whose syntax permits the pairing?

4. Concerning example 3, is there an incompatibility with ANSI X3.23-1974 not listed in the appendix?

Unfortunately, most of the possible answers to these questions will create incompatibilities to the previous standard, to extensions of the previous standard, or among different interpretations used in implementations.

There may be one (rather pragmatic) way out of this dilemma: general rule 1 is interpreted in a strict sense; exhaustion of the statement format does not terminate a statement. This implies for the examples 1 and 3, that anything between the NEXT SENTENCE phrases and the terminating periods or ELSE phrases is considered part of the

IF statements immediately preceding them, regardless of syntactical correctness; i. e. the END-IF #7 in example 1 is part of statement #3, as indicated in 1a; and the DISPLAY statement in example 3 part of statement #4. As a result of this, the IF statements are in conflict with the general format of the IF statement; thus the examples are non-standard.

With this interpretation, the above questions can be answered as follows:

1. Statement exhaustion does not terminate the scope of a statement.

2. The end of the IF statement is determined by the end of its scope, as defined by general rule 1.

3. General rule 3 applies only to statements that are not yet terminated (whether syntactical correctness is required is irrelevant because of general rule 1).

   The repetition periods do not create an incompatibility but are redundant and misleading, because any statement following after the NEXT SENTENCE phrase will be considered part of the ELSE phrase and thus will make it invalid.

Finally, this interpretation protects users from the traps shown in the above examples. It seems that all other possible interpretations can be supported as extensions to this interpretation, if necessary. Also, the CODASYL COBOL Journal of Development rule could be implemented as an extension.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

Example 1 is nonstandard because:

1. the END-IF #7 cannot belong to statement #3, according to the general format of the IF statement,

2. the closest valid terminator for statement #3 would be the ELSE phrase #8, according to general rule #1, and

3. the END-IF #7 thus is in the scope of statement #3, but syntactically invalid.

The "key questions" in the inquiry are answered as follows:

1. Syntactic exhaustion does not implicitly terminate the scope of a statement; however, this is not relevant in the example because the contained statement #3 is terminated by the next phrase ELSE #8 of the containing statement #2.

2. The end of the IF statement is determined by the end of its scope, as defined by general rule 1.

3. General rule 3 applies only to statements that are not yet terminated (whether syntactical correctness is required is irrelevant because of general rule 1).

4. Example 3 is invalid because the closest valid terminator for the IF statement #4 is the separator period; the DISPLAY statement therefore is syntactically invalid. There is no incompatibility with the previous standard.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-9. Page 95 of this document contains the proposed correction to pages VI-90 and VI-91 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-16

SUBJECT: MERGE USING Phrase and Relative Files

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XI-10 and 11, paragraph 4.1.4, general rule 7, MERGE statement
2. Page XI-19, paragraph 4.4.4, general rule 9, SORT statement

DATE: April 10, 1987

QUESTION:

Reference 1 for the USING phrase of a MERGE statement does not indicate for a relative file whether the content of the relative key data item is defined after the execution of the MERGE statement.

Reference 2 states that, for the USING phrase of a SORT statement, "For a relative file, the content of the relative key data item is undefined after execution of the SORT statement if file-name-2 is not referenced in the GIVING phrase". Since the content of the relative key data item for a relative file specified in the USING phrase of a SORT statement is explicitly undefined after the execution of the SORT statement, ANSI COBOL X3.23-1985 appears to imply, by omission of a corresponding rule for the MERGE statement that the content of the relative key data item is defined.

Is the content of the relative key data item for a relative file specified in the USING phrase of a MERGE statement defined after the execution of the MERGE statement?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that at the end of the MERGE statement, the content of the relative key data item is implicitly undefined.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-16. Page 101 of this document contains the proposed correction to page XI-10 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-21

SUBJECT: NOT AT END Phrase Not Specified

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-46, paragraph 4.4.4, general rule 11, READ statement in Sequential I-O module
2. Page VIII-28, paragraph 4.5.4, general rule 11, READ statement in Relative I-O module
3. Page IX-31, paragraph 4.5.4, general rule 11, READ statement in Indexed I-O module

DATE: December 1, 1987

QUESTION:

It is not specified in American National Standard COBOL X3.23-1985 when to execute the NOT AT END phrase. Also, the relationship of the NOT AT END phrase with any pertinent USE procedures needs to be specified. Is the NOT AT END phrase executed when no error exists, or when there was no at end condition?

This problem occurs in the sequential, relative, and indexed modules. We feel that it should be defined in a manner consistent with NOT INVALID KEY. That is, it should only be executed when no error condition exists.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the NOT AT END phrase and any pertinent USE procedures are adequately defined and no additional clarification is required.

The following applies to a READ statement in the Sequential I-O module (see reference 1):

> If an at end condition does not occur during the execution of a READ statement, the AT END phrase is ignored, if specified, and

a. The file position indicator is set and the I-O status associated with the file is updated.

b. If an exception condition that is not an at end condition exists, control is transferred according to the rules of the USE statement following the execution of any USE AFTER EXCEPTION procedure applicable to the file.

c. If no exception condition exists, the record is made available in the record area and any implicit move resulting from the presence of an INTO phrase is executed. Control is transferred to the end of the READ statement or to the NOT AT END phrase, if specified. In the latter case, execution continues according to the rules for each statement specified in the NOT AT END phrase. If a procedure branching or conditional statement which causes explicit transfer of control is executed, control is transferred in accordance with the rules of that statement; otherwise, upon completion of the execution of the NOT AT END phrase, control is transferred to the end of the READ statement.

Similar wording for the READ statement in the Relative I-O module (see reference 2) and in the Indexed I-O module (see reference 3) defines the rules for the NOT AT END phrase and pertinent USE procedures.

X3J4 DOCUMENT A-24

SUBJECT: I-O Status Codes

REFERENCES:

American National Standard COBOL X3.23-1985

1. Pages IX-2 and IX-3, paragraph 1.3.4, I-O status, Indexed I-O module
2. Page VII-2, paragraph 1.3.5, I-O status, Sequential I-O module
3. Page VIII-2, paragraph 1.3.4, I-O status, Relative I-O module
4. Page VII-40, paragraph 4.3.4, general rule 3, OPEN statement
5. Page VII-41, paragraph 4.3.4, general rule 8, OPEN statement
6. Page VII-5, paragraph 1.3.7, rule 1, file attribute conflict conditions

DATE: March 22, 1987

QUESTION:

The following is a list of I-O status codes that have overlapping definitions. We would like to know which of the two codes should be returned in the following cases:

1. When writing a sequentially accessed indexed file, the following keys were written 2, 4, 6, and then 4. Should the program get an I-O status code 21 (sequence error) or 22 (duplicate key)? Both codes seem to apply.

2. An OPEN statement attempts to open a file where both: (a) the fixed file attributes conflict, and (b) the file was opened using an incorrect mode. Should an I-O status 37 or 39 be returned?

3. How can an I-O status 04 (record read is the wrong length) be returned? The program should not be able to do the READ because the OPEN got a permanent error 39 (fixed file attributes conflict).

In the cases where more than one I-O status code seems to apply, because there is no rule determining the priority of the status codes, we feel that it is implicitly undefined which code is returned.

In the cases where more than one I-O status code applies, some priorities need to be set to determine which error cases to check for first, second, etc. Without this ordering,

programs will not execute in the same way on different systems. Until such time these will continue to be implicitly undefined.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that Standard COBOL does define means of handling overlapping status code values in cases 1 and 2. The implementor determines which of the applicable values to place in the I-O status. (See reference 1.) In the third case, the Standard does not define the circumstances under which the 04 status code is returned.

X3J4 DOCUMENT A-28

SUBJECT: Code Set as a Fixed File Attribute

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page II-1, paragraph 2.1, file attributes
2. Page VII-5, paragraph 1.3.7, file attribute conflict condition

CODASYL COBOL Journal of Development April 1980

1. Page II-2-1, paragraph 2.1.1, file attributes
2. Page III-6-18, paragraph 6.4.1.4, general rule 13, FILE-CONTROL paragraph

DATE: August 2, 1987

QUESTION:

Reference 1 in American National Standard COBOL X3.23-1985 states that code set is a fixed file attribute. CODASYL COBOL has not made code set a fixed file attribute at least not since April 1980. In fact, the code set may be modified during execution via the SET statement capability in CODASYL COBOL. Making code set a fixed file attribute greatly restricts its usefulness, particularly when enumerated code sets are allowed (CODE-SET IS alphabet-name where alphabet-name is defined with the alphabet-name IS literal phrase).

Must an implementation enforce code set as a fixed file attribute in order to be a conforming implementation? If so, ANSI COBOL X3.23-1985 requires implementors to be in direct conflict with CODASYL COBOL, rather than allowing them to conform to CODASYL COBOL as an extension to ANSI COBOL X3.23-1985.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the implementor defines whether a code set mismatch causes an I-O status value of 39 to be returned.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-28. Pages 91 through 100 of this document contain the proposed corrections to pages II-1, III-10, VII-5, VII-41, VII-43, VIII-6, VIII-24, VIII-25, IX-7, IX-26, and IX-27 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-41

SUBJECT: COPY REPLACING Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IV-5, paragraph 4.2.2.1, COBOL words
2. Page III-25, definition of text-word, item 3
3. Page III-23, definition of source program
4. Page XII-1, paragraph 1.1, second paragraph, function of Source Text Manipulation module
5. Page III-4, note 2 in definition of COBOL character set
6. Page XII-4, paragraph 2.4, general rule 8, COPY statement

DATE: December 4, 1987

QUESTION:

Are text words COBOL words? Are text word characters treated like their uppercase equivalents?

According to reference 1, each lowercase letter of a COBOL word is "considered to be equivalent to its uppercase correspondent letter." A similar rule exists for certain PICTURE symbols. (Has anyone found out which are not?)

Apparently, no such rule exists for text words. But possibly no such rule is needed, because they are COBOL words anyhow. Or what else are they?

1. According to paragraph 4.2 on page IV-4, the text of a source program consists of character-strings and separators.

According to paragraph 4.2.2 on page IV-5, character-strings are: COBOL words, literals, PICTURE character-strings, or comment-entries in the Identification Division.

According to paragraph 4.2.1 on page IV-4, separators are: space, comma/space, semicolon/space, period/space, parentheses, quote, pseudo-text delimiters (= =), or colons.

(Note that this implies that comment lines and blank lines are not part of the text of the source program.)

Open question: What are text words? If they are part of the source text, they can only be COBOL words; but they are not listed as such. If they are not, then why are pseudo-text delimiters listed?

2. A source program is defined as a "syntactically correct set of COBOL statements" in the glossary. Syntactical correctness of the program can however, except for the COPY and REPLACE statements, be only determined after all COPY and REPLACE statements are resolved (general rule 8 of the COPY statement). This implies that a program containing COPY or REPLACE statements cannot be regarded as a source program. However, the glossary definition of source program explicitly allows that it begins with a COPY or a REPLACE statement. This seems contradictory.

What is the X3J4 interpretation? Are text words part of the source program or not? If yes, are they COBOL words or not?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

Since the text word may contain COBOL characters (see reference 2), then according to reference 5 each lowercase letter in the COBOL character set is equivalent to the corresponding uppercase letter. Text words are part of the source program (see reference 2) and a given occurrence of a text word might also be a COBOL word.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-41. Page 103 of this document contains the proposed correction to pages XII-3 and XII-7 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-46

SUBJECT: SORT/MERGE and RECORD VARYING Clause with DEPENDING

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XI-7, paragraph 3.2, sort-merge file description entry
2. Page XI-19, paragraph 4.4.4, general rule 9b, SORT statement
3. Page XI-20, paragraph 4.4.4, general rule 12b, SORT statement

DATE: January 21, 1988

QUESTION:

If a sort file has in its sort-merge file description entry a RECORD VARYING clause with the DEPENDING phrase, what should be the effect of the DEPENDING phrase when the sort file is used in a SORT or MERGE statement with a USING or GIVING phrase? Also, if the files in the USING or GIVING phrases are variable length, is the length of each record from a USING file transmitted to the corresponding record in each GIVING file?

Example:

```
DATA DIVISION.
SD SFILE  RECORD VARYING DEPENDING ON SSIZ.
01 SREC.
   02  SKEY  PIC XXXX.
   02  SDATA  PIC X(10).

FD IFILE  RECORD VARYING DEPENDING ON ISIZ.
01 IREC  PIC X(14).

FD OFILE  RECORD VARYING DEPENDING ON OSIZ.
01 OREC  PIC X(14).

WORKING-STORAGE SECTION.
01 SSIZ  PIC 99.
01 ISIZ  PIC 99.
01 OSIZ  PIC 99.
```

```
PROCEDURE DIVISION.
   MOVE 7 TO SSIZ.
   MOVE 8 TO OSIZ.
   SORT SFILE  ON ASCENDING KEY SKEY, USING IFILE, GIVING OFILE.
```

As there is no opportunity to change SSIZ or OSIZ during the sort, the records in SFILE and OFILE cannot be specified by this means to vary in size. Therefore we believe that DEPENDING ON phrases of the RECORD VARYING clauses for SFILE and OFILE can have no effect, unless it be to truncate all the records being sorted to some fixed length, which does not seem a useful facility.

IFILE in the example is a variable length file. Will each record be the same length when written to OFILE as it was in IFILE, i.e. unaffected by the values in SSIZ and OSIZ; or will it have been truncated to the length in SSIZ or OSIZ?

We feel that the DEPENDING ON phrases of the RECORD VARYING clauses in the sort file description entry for SFILE and the file description entry for OFILE should have no effect.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the effect of DEPENDING ON items with the RECORD VARYING clause on sort operations is inadequately defined. The results of the operation in the example provided will be dependent on the implementation.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-46. Pages 102 and 103 of this document contain the proposed corrections to pages XI-19 and XI-20 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-47

SUBJECT: OPEN EXTEND of a Relative or Indexed File

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VIII-21, paragraph 4.4.3, syntax rule 1, OPEN statement
2. Page VIII-24, paragraph 4.4.4, general rule 11, OPEN statement
3. Page VIII-39, paragraph 4.9.4, general rule 12, WRITE statement
4. Page IX-23, paragraph 4.4.3, syntax rule 1, OPEN statement
5. Page IX-26, paragraph 4.4.4, general rule 11, OPEN statement
6. Page IX-42, paragraph 4.9.4, general rule 14, WRITE statement

DATE: January 8, 1988

QUESTION:

If the following program were executed, what relative key value would be displayed, 001 or 101?

```
FILE-CONTROL.
    SELECT REL-FILE  ORGANIZATION IS RELATIVE
        ACCESS MODE IS DYNAMIC
        RELATIVE KEY IS REL-KEY.

DATA DIVISION.
FD  REL-FILE
01  REL-REC  PIC X(120).

PROCEDURE DIVISION.
BEGIN.
    OPEN OUTPUT REL-FILE.
    MOVE 100 TO REL-KEY.
    WRITE REL-REC  INVALID KEY  STOP RUN.
    CLOSE REL-FILE.
DELETE-100.
    OPEN I-O REL-FILE.
    MOVE 100 TO REL-KEY.
    DELETE REL-FILE RECORD  INVALID KEY  STOP RUN.
```

```
EXTEND-FILE.
    OPEN EXTEND REL-FILE.
    WRITE REL-REC.
    DISPLAY REL-KEY
    CLOSE REL-FILE.
    STOP RUN.
```

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that ANSI COBOL X3.23-1985 adequately describes the handling of a relative or indexed file opened in the extend mode.

The sample COBOL program included in the question is syntactically incorrect. As stated in the syntax rules of the OPEN statement of a relative file and for an indexed file, the file to be opened in the extend mode must have sequential access (see references 1 and 4).

Even if the file were opened in the extend mode in a separate program after it had been output and updated as a dynamic access file, the general rule for the OPEN statement states how the file is positioned after it is opened in the extend mode: it is positioned immediately after the last logical record for that file and the last logical record is the currently existing record with the highest relative record number for relative files or the highest prime key value for indexed files (references 2 and 5 respectively).

The general rule for the WRITE statement states what the key value is for the first record written after it is opened in the extend mode: the first record released to the file has a relative record number one greater than the highest relative number existing in the file for relative files and the prime record key of the first record released must be greater than the highest prime record key existing in the file for indexed files (references 3 and 6 respectively).

In response to your specific question, the value of the relative key displayed in the sample program would be 001.

X3J4 DOCUMENT A-48

SUBJECT: MULTIPLE FILE TAPE Clause and Leveling

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page I-47, summary of elements in Environment Division
2. Page XI-17, paragraph 4.4.3, syntax rule 7, SORT statement
3. Page XI-7, paragraph 3.2.3, syntax rule 4, sort-merge file description entry
4. Page XI-3, paragraph 2.4.3, syntax rule 1, I-O-CONTROL paragraph in Sort-Merge module
5. Page XIII-5, paragraph 2.4.3, syntax rule 3, I-O-CONTROL paragraph in Report Writer module
6. Page XI-2, paragraph 2.1, Input-Output Section
7. Page I-6, paragraph 1.5.1, definition of subsets

DATE: January 21, 1988

QUESTION:

The MULTIPLE FILE TAPE clause in the Sequential I-O module is a level 2 feature (see reference 1). The implicit support of this functionality is assumed in the Sort-Merge module (see reference 2), which is a level 1 module. This appears to be a leveling conflict.

Is an intermediate level conforming implementation expected to support multiple files on a single reel in the Sort-Merge module? In our opinion, it is not and the I-O-CONTROL description in the Sort-Merge module is incorrect in its omission of the MULTIPLE FILE TAPE clause, along with a leveling statement qualifier similar to reference 3 for the VARYING phrase in the RECORD clause, reference 4 for the RECORD option of the SAME clause, and reference 5 for the clauses in the I-O-CONTROL paragraph in the Report Writer module. Also the summary of elements for the I-O-CONTROL paragraph should be changed to reflect the Sort-Merge module for the MULTIPLE FILE TAPE clause.

It appears that the approach taken when all of the Sort-Merge module was changed to level 1 in this standard is inconsistent with other leveling. The concept of having features whose levels are not clear and consistent and are dependent on other factors, is a deviation

from the rest of the standard and very misleading. For example, in the level 1 of the Sort-Merge module. However, this is not true for implementations that only support level 1 of the Sequential I-O module. We would request that in any future revisions or addenda, that this practice be eliminated.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the leveling of the Sort-Merge module is consistent in regards to the MULTIPLE FILE TAPE clause. Reference 6 directs the user to page VII-6, the Input-Output Section for the Sequential I-O module. Within the Sequential I-O module, the MULTIPLE FILE TAPE clause is defined as a level 2 element. Additionally, reference 7 states that an intermediate subset of Standard COBOL contains level 1 elements from the Sequential I-O module and level 1 elements from the Sort-Merge module.

X3J4 DOCUMENT A-49

SUBJECT: REWRITE to Indexed File

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IX-4, paragraph 1.3.4, item 3a, I-O status 21
2. Page IX-4, paragraph 1.3.4, item 3b, I-O status 22
3. Page IX-34, paragraph 4.6.4, general rules 14 and 16, REWRITE statement

DATE: October 8, 1987

QUESTION:

How can a REWRITE statement create a duplicate prime record key?

1. In sequential access mode, the prime record key must not be changed since the last record read.

2. In random access mode, the prime record key must refer to an existing record (whether or not the record had previously been read).

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that a REWRITE statement cannot create a duplicate prime record key in an indexed file.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-49. Page 99 of this document contains the proposed correction to page IX-4 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-50

SUBJECT: REPLACE Statement and Comment Lines

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XII-4, paragraph 2.4, general rule 6, COPY statement
2. Page XII-7, paragraph 3.4, general rule 7, REPLACE statement
3. Page XII-6, paragraph 3.4, general rule 1, REPLACE statement

DATE: January 29, 1988

QUESTION:

Reference 1 establishes the reasonable rule that comment lines and blank lines appearing in library text are not copied if they appear within the sequence of text words that match pseudo-text-1 of a COPY statement.

Reference 2 appears to have picked up that rule for the REPLACE statement and simply changed "not copied" to "not replaced". This rule for the REPLACE statement seems unreasonable. It is also meaningless without additional explanation of where such blank lines and comment lines should appear in the resultant source program.

If a comment line or blank line in source program text that appears within the sequence of text words that match pseudo-text-1 of a REPLACE statement is not replaced, where does it appear in the source program?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that comment lines and blank lines that are part of a matched pseudo-text-1 should not be placed in the resultant source program.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-50. Page 104 of this document contains the proposed correction to page XII-7 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-51

SUBJECT: INSPECT Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-28, paragraph 5.8.4, general rule 3b, OCCURS clause
2. Page VI-95, paragraph 6.18.3, syntax rule 1, INSPECT statement
3. Page VI-96, paragraph 6.18.4, general rule 2, INSPECT statement
4. Pages VI-96 and VI-97, paragraph 6.18.4, general rules 6a and 6d, INSPECT statement

DATE: February 1, 1988

QUESTION:

Reference 1 specifies a semantic difference that is dependent on whether a group data item is referenced as a sending item or as a receiving item.

Reference 2 allows identifier-1 in an INSPECT statement to reference a group item, but no rule specifies whether identifier-1 is considered to be a sending or receiving item. Three choices are possible candidates:

1. identifier-1 is always a sending item; or
2. identifier-1 is always a receiving item; or
3. identifier-1 is a sending item in format 1 and a receiving item in format 2 and format 3.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that for the purpose of determining its length, identifier-1 is treated as if it were a sending data item.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-51. Page 95 of this document contains the proposed correction to page VI-96 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-52

SUBJECT: COPY Statement Continuation

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XII-3, paragraph 2.4, general rule 1, COPY statement
2. Page XII-4, paragraph 2.4, general rule 8, COPY statement
3. Page IV-42, paragraph 7.2.2, continuation of lines

DATE: December 4, 1987

QUESTION:

What is the resultant of the following code?

```
Columns
7       12

        C
-       O
-       P
-       Y text-1.
```

According to reference 1, this is not a COPY statement since the "processing of the resultant source program" is done "after the processing of all COPY statements". So, the hyphens will not be processed and the word COPY will not be recognized.

What is the resultant of the code shown at the top of the next page?

```
Columns
7       12

        COPY
        t
-       e
-       x
-       t
-       -
-       1
-       .
```

This is indeed a COPY statement. However, for the same reason as in the first question, the hyphens will not be processed. Will the COPY statement look for text-name "t"?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that, applying the continuation rules of reference format, the two examples shown are both syntactically correct COPY statements and are both read as:

COPY text-1.

In reference 1, "the resultant source program" means the one produced from the original source and the processing of COPY and REPLACE statements; this has no bearing on the processing of hyphens that exist in the original source program.

References 2 and 3 require that all syntactically correct COPY statements be recognized and processed.

X3J4 DOCUMENT A-53

SUBJECT: COPY Text Words

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page III-25, definition of text word
2. Page IV-9, paragraph 4.2.2.2.1, nonnumeric literals
3. Page IV-5, paragraph 4.2.1, rule 5 for separator
4. Page XII-3, paragraph 2.4, general rule 4, COPY statement
5. Page XII-2, paragraph 2.3, syntax rule 9, COPY statement
6. Page IV-4, paragraph 4.2.1, rule 2 for separator
7. Page III-18, definition of pseudo-text
8. Page III-18, definition of pseudo-text delimiter

DATE: January 29, 1988

QUESTION #1:

Is `--"abc"--` a valid pseudo-text?

According to the definition of text word in the glossary, a text word may be "a literal including, in the case of nonnumeric literals, the opening quotation mark and the closing quotation mark which bound the literal" (see reference 1).

A nonnumeric literal is defined as "a character-string delimited at the beginning and at the end by the separator quotation mark" (see reference 2).

The separator quotation mark is defined such that the "opening quotation mark must be immediately preceded by a space or left parenthesis; the closing quotation mark, when paired with an opening quotation mark, must be immediately followed by one of the separators space, comma, semicolon, period, or right parenthesis" (see reference 3).

In this case the quotation marks are preceded and followed by the separator pseudo-text delimiter, which is not in the above list. Hence the `"abc"` in the above example is not a valid literal. It could be argued that it can be regarded as "any other sequence of

contiguous COBOL characters except comment lines and the word COPY, bounded by separators, which is neither a separator nor a literal" (see reference 1). This cannot, however, be considered as a valid interpretation since it would lead to ambiguities in cases involving literals containing blanks, like --"␣-- where the character ␣ indicates a blank or space character. Is the second -- part of the literal, or is the second -- the ending pseudo-text delimiter?

It seems that in a strict interpretation of Standard COBOL, --"abc"-- is not a valid pseudo-text. However, "abc" is a valid operand of the REPLACING phrase of the COPY statement and reference 4 states that it is treated for matching purposes like pseudo-text containing only the literal, i.e., like --"abc"-- . If this were invalid, all nonnumeric literals are invalid as operands, in contradiction to the general format of the COPY statement.

X3J4 RESPONSE TO QUESTION #1:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

The intent of Standard COBOL is that by applying item 2 of reference 1, the opening quotation mark and closing quotation mark do not require preceding and following separators in the pseudo-text. Therefore --"abc"-- is a valid nonnumeric literal.

In the example --"␣-- , the quote is the start of a literal (see rule 2 of reference 1) and the second -- is part of the literal (see syntax rule 2 of reference 2). The example is an invalid pseudo-text because there is not a closing quotation mark.

QUESTION #2:

Are the following valid pseudo-text?

1. --.--
2. --;--
3. --,--

The special period, comma, and semicolon in the examples cannot be regarded as separators because they are not immediately followed by a space (see reference 3).

This implies, however, that they are valid as "any other sequence of contiguous COBOL characters except comment lines and the word COPY, bounded by separators, which is neither a separator nor a literal" (see reference 1).

Or should be assumed, also in light of the previous question, that separator spaces are assumed immediately following the opening pseudo-text delimiter and preceding the closing pseudo-text delimiter? This would make the --.-- equivalent to --.␣-- ; the others would be invalid according to reference 5.

X3J4 RESPONSE TO QUESTION #2:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

According to item 3 of reference 1, a text word is a "sequence of contiguous COBOL characters except comment lines and the word COPY, bounded by separators, which is neither a separator nor a literal". The pseudo-text delimiter is recognized as a separator (see reference 8) that bounds the contiguous COBOL characters. Therefore, all three examples are valid pseudo-text, however the semantics of using these examples is not defined in Standard COBOL.

X3J4 DOCUMENT A-54

SUBJECT: COPY Conformance Rules

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XII-4, general rule 8, COPY statement
2. Page XII-3, general rule 4, COPY statement
3. Page I-8, extension language elements
4. Page XII-3, general rule 1, COPY statement

DATE: September 24, 1987

QUESTION:

Do operands of the REPLACING phrase that are only valid outside the selected subset (e.g. an identifier with qualification or reference modification) have to be flagged as nonconforming?

This is at least doubtful. The syntactic correctness of the program except for the COPY and REPLACE statements cannot be determined before the end of COPY processing (reference 1). Furthermore, the operands of COPY REPLACING are treated "for matching purposes" like pseudo-text containing only this operand (reference 2); i.e. an identifier is treated like an arbitrary string of COBOL characters. But matching is the only purpose of such an operand, i.e. for the execution of the COPY statement it does not really matter whether they are anything more than such a string of characters.

To verify the "syntactic correctness" of these operands beyond the purposes of matching should not be required prior to the end of COPY processing, hence no conformity flagging should be required either.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

ANSI COBOL X3.23-1985 does not require a conforming implementation of Standard COBOL to provide a warning mechanism to indicate that a program contains standard extensions. Reference 3 distinguishes between standard and nonstandard extensions, and requires a warning mechanism only for the latter.

If a nonstandard extension is used in the REPLACING phrase, then it is treated simply as a string of characters for matching, and so need not be flagged (reference 2). If a nonstandard extension appears in the text after substitution it must be flagged, just as a nonstandard extension written in the source must be flagged (reference 4).

X3J4 DOCUMENT A-55

SUBJECT: REPLACE Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XII-6, paragraph 3.3, syntax rules of REPLACE statement
2. Page XII-6, paragraph 3.4, general rule 3, REPLACE statement
3. Page XII-7, paragraph 3.4, general rule 5, REPLACE statement
4. Page III-25, definition of text word, item 3
5. Page XII-8, paragraph 3.4, general rule 9, REPLACE statement
6. Page I-9, paragraph 1.6, definition of a conforming source program
7. Page III-24, definition of statement

DATE: December 3, 1987

QUESTION:

Are the following examples valid and what is their meaning?

Example 1:

```
REPLACE --REPLACE-- BY --SET SWITCH-X TO--.
...
REPLACE OFF.
```

Example 2:

```
REPLACE --REPLACE-- BY --SET SWITCH-X TO--.
...
REPLACE ON.
```

Example 3:

```
REPLACE --OFF-- BY --ON--.
...
REPLACE OFF.
```

Example 4:

```
REPLACE --ON-- BY --OFF--.
...
REPLACE ON.
```

According to reference 3, the REPLACE statement cannot introduce a REPLACE statement. However, there is no rule with the effect that a REPLACE statement cannot remove or modify a REPLACE statement. While the word COPY is explicitly forbidden as a text word (reference 4), the word REPLACE seems to be allowed. In pseudo-text-2, this cannot happen, however, because otherwise general rule 5 (reference 3) would be violated. But no such restriction seems to exist for pseudo-text-1, and words like GARBAGE or OFF are certainly valid text words.

For the interpretation of the above examples, the following two questions have to be answered:

1. According to reference 2, the next occurrence of a REPLACE statement terminates the effect of the previous one. But is reference 2 applicable before or after execution of the REPLACE statement, i.e. is the REPLACE operation performed on a text word before that text word is recognized as another REPLACE statement or afterwards?

2. Furthermore, what are the conditions for recognizing the next REPLACE statement? Is any occurrence of the word REPLACE sufficient? Or does it require a syntactically correct REPLACE statement, including its operands?

The following cases are possible:

1. The effect of a REPLACE statement is terminated by the next occurrence of the words REPLACE before translation. In none of the above examples a replacement takes place; it is always terminated. (However, examples 2 and 4 would be invalid COBOL language.) This seems to be the clearest and least ambiguous interpretation.

2. The effect of a REPLACE statement is terminated by the next valid REPLACE statement before translation. Replacement takes place in examples 2 and 4, it is terminated in examples 2 and 4. However, the determination of a syntactically valid REPLACE statement at this point could put an extreme burden on the COPY phase of a compiler, because it would have to parse the complete next REPLACE statement in the middle of processing the previous one. Note also the peculiarity that in example 4 a valid REPLACE OFF is created by the execution of the first REPLACE; what is its semantic?

3. The effect of a REPLACE statement is terminated by the next occurrence of the word REPLACE, after translation. Replacement takes place in examples 1 and 2; it is terminated in examples 3 and 4.

4. The effect of a REPLACE statement is terminated by the next valid REPLACE statement after translation. Replacement takes place in all four examples; it is not terminated in examples 1, 2, and 3. It seems that the REPLACE OFF being created in example 4 terminates the replacement process; but this is not quite clear, because the replacement process could be considered to be already past the REPLACE OFF statement just created. This interpretation would be as cumbersome as the second one.

What is the X3J4 interpretation in this matter?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

The syntactic correctness of all REPLACE statements is determined during REPLACE processing (reference 5). The effect of a REPLACE statement is terminated by the next occurrence of a syntactically correct REPLACE statement (references 2, 5, and 7). The second REPLACE statement is not changed by the first. The effect of a syntactically incorrect statement is explicitly undefined (reference 6). Since examples 1 and 3 are syntactically correct, no replacement occurs in the second REPLACE statement. The results of examples 2 and 4 are undefined since the second REPLACE statement in each example is syntactically incorrect.

X3J4 DOCUMENT A-56

SUBJECT: USE Procedure for CLOSE Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-51, paragraph 4.6.4, general rule 5, USE statement
2. Page VII-2, paragraph 1.3.5, second and third paragraphs, I-O status
3. Page VII-4, paragraph 1.3.5, item 4b, I-O status
4. Page VII-38, paragraph 4.2.4, general rule 4, CLOSE statement
5. Page VII-38, paragraph 4.2.4, general rule 7, CLOSE statement
6. Page VII-35, paragraph 4.2.4, general rule 1, CLOSE statement
7. Page VII-40, paragraph 4.3.4, general rule 3, OPEN statement

DATE: December 22, 1987

QUESTION:

It is not clear what USE PROCEDURE should be executed after the I-O status code is updated to reflect status code 42 (a CLOSE statement is attempted for a file not in the open mode).

According to references 3 and 4, an unsuccessful CLOSE statement can occur.

According to reference 2, an unsuccessful CLOSE statement is a critical error condition and execution of the applicable USE procedure takes place.

According to reference 1, a USE procedure for (INPUT, OUTPUT, I-O, EXTEND) is executed for any file open, or in the process of being opened, in the corresponding mode.

According to reference 5, a successful CLOSE statement causes the file to be removed from the open mode.

Therefore, for the sequence of events that lead to I-O status code 42, the file is not in the open mode when the condition is detected, and there is no defined USE procedure to execute that fits the criteria of the above references. We conclude that the only USE procedure that can be executed is one that may be specified for file-name. If a USE

procedure for file-name is not specified, the standard does not specify what action should take place.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the only applicable USE procedure will be the one that lists the file-name referenced in the CLOSE statement (reference 1). The implementor determines what action is taken after the execution of any applicable USE AFTER STANDARD EXCEPTION procedure, or if none applies, after completion of the normal input-output control system error processing (reference 2).

X3J4 DOCUMENT A-57

SUBJECT: Ordering of Alternate Keys After REWRITE

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IX-34, paragraph 4.6.4, general rule 7, REWRITE statement
2. Page IX-35, paragraph 4.6.4, general rule 15b, REWRITE statement
3. Page IX-29, paragraph 4.5.4, general rule 4, READ statement

DATE: November 13, 1987

QUESTION:

Consider an indexed file. A program reads a record from it, changes the value of an alternate key, and writes the record back to the file using a REWRITE statement. It then executes a READ NEXT statement on the file, using that alternate key. Which record will be read by this READ statement?

The rule that comes closest to resolving this is reference 2 whose first sentence states: "When the value of an alternate key is changed, the subsequent order of retrieval of that record may be changed when that specific alternate key is the key of reference." But although it is clear from this in what order records will subsequently be retrieved, it is not clear where subsequent retrievals will begin.

For example, say the file contains five records, with alternate keys 1, 2, 3, 4, and 6; the program reads the record with alternate key 2, changes the alternate key to 5, and rewrites the record. Will the next READ NEXT statement read this new record 5, or the record following it (record 6), or record 3?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the file position indicator is unaffected by a REWRITE statement (reference 1). The file position indicator prior to the REWRITE statement will be used to determine the results of a subsequent READ NEXT statement (reference 3). In the example given, this would be record 3.

X3J4 DOCUMENT A-58

SUBJECT: Index-Names and EXTERNAL Clause

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-27, paragraph 5.8.3, syntax rule 13, OCCURS clause
2. Page III-11, definition of index-name
3. Page III-11, definition of index
4. Page II-15, paragraph 4.4.2, subscripting using index-names
5. Page III-9, definition of external data item
6. Page III-9, definition of external data record
7. Page X-7, paragraph 1.3.8.3, conventions for index-names
8. Page X-23, paragraph 4.5.4, general rule 1, EXTERNAL clause
9. Pages X-2 and X-3, paragraph 1.3.4, external objects and internal objects
10. Pages II-19 through 21, paragraph 6.2.2, objects, through paragraph 6.2.2.2.4, other objects

DATE: January 15, 1988

QUESTION:

References 1 through 4 clearly define the object of an index-name as something that is not a data item. References 5, 6, and 8 describe the external attribute as referring only to data items. Reference 7 attempts to apply the external attribute to index-names. This appears to be a contradiction.

How is the correspondence between index-names associated with external data items established? In particular, consider a run unit containing the following programs:

Program A:

```
01  A  IS EXTERNAL.
    02  A2  PIC X  OCCURS 3 TIMES  INDEXED BY IX-A, IX-B.
```

Program B:

```
01  B  IS EXTERNAL.
    02  B2  PIC X  OCCURS 3 TIMES  INDEXED BY IX-A.
```

Program C:

```
01  A  IS EXTERNAL.
    02  A2  PIC X  OCCURS 3 TIMES  INDEXED BY IX-B, IX-A.
```

Program D:

```
01  IX-A  IS EXTERNAL  USAGE IS INDEX.
```

Are IX-A of program A and IX-A of program C the same object?

Are IX-A of program A and IX-A of program B the same object?

Is IX-A of program D the same object as any other IX-A?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that indexes are not external. Indexes are not data and are not associated with any data hierarchy (see reference 1). The value of an index can be made accessible to an object program by storing the value in an index data item (see reference 4). Index data items may attain the external attribute by the rules pertaining to the EXTERNAL clause. Reference 7 should not have referenced the external attribute since its subject is scope of names. Scope of names refers to the recognition of names within directly or indirectly containing programs and is unrelated to the external attribute. The external attribute applies only to data items and file connectors (see reference 9).

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-58. Page 101 of this document contains the proposed correction to page X-7 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-59

SUBJECT: File Attributes

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page II-1, paragraph 2.1, file attributes
2. Page VII-23, paragraph 3.3.3, general rule 4, BLOCK CONTAINS clause
3. Page III-10, definition of fixed file attributes
4. Page VII-23, paragraph 3.3.2, general format, BLOCK CONTAINS clause
5. Page II-3, paragraph 2.2.1, record operations

DATE: February 1, 1988

QUESTION:

1. It is not clear what specific attributes of the prime record key and alternate record keys are intended as file attributes. Is it type, size, or, in the case of alternate record keys, the number of, or all of the above?

2. Are similar specific attributes of a relative record data item also considered a fixed file attribute?

3. In the case of blocking factor, is the specification of RECORDS or CHARACTERS considered in the reference to integer-1 and integer-2 in reference 2?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that:

1. All the attributes of the prime record key and the alternate record keys are file attributes.

2. Similar specific attributes of a relative record data item are not fixed file attributes, and

3. The same blocking factor can be specified in two different ways and is equivalent whether it is derived from CHARACTERS or RECORDS.

X3J4 DOCUMENT A-60

SUBJECT: PADDING CHARACTER Clause

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page X-21, paragraph 4.3.3, syntax rule 2, data description entry in Inter-Communication module
2. Page VII-12, paragraph 2.7.3, syntax rule 3, PADDING CHARACTER clause
3. Page VII-12, paragraph 2.7.4, general rule 6, PADDING CHARACTER clause

DATE: January 8, 1988

QUESTION:

Reference 1 states that the external attribute can only be specified for data description entries in the Working-Storage Section.

Reference 2 implies that a data-name used to define the padding character can be specified in the Working-Storage or Linkage Section.

Reference 3 states that the data-name must be defined as external if the file is external.

These rules are contradictory. In particular, reference 2 should be qualified to disallow the padding character data-name to be specified in the Linkage Section when the file connector is external.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the rules for the padding character are not contradictory. Reference 3 states that if a data-name is specified for a padding character, it must reference an external data item. Since external data items can only be specified in the File or Working-Storage Section and padding character data-names can only be specified in the Working-Storage or Linkage Section, the only section where a padding character data-name could be specified as an external data item is the Working-Storage Section.

X3J4 DOCUMENT A-62

SUBJECT: Communication Status Key Value 80

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XIV-15, table 1, communication status key conditions

DATE: January 31, 1988

QUESTION:

Table 1 for communication status key conditions (see reference 1) states that the value of 80 is returned when a "combination of at least two status key conditions 10, 15, and 20 have occurred". Yet table 1 shows that only condition 15 and neither condition 10 or 20 can occur on an ENABLE INPUT or DISABLE INPUT.

RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that in level 2 of the Communication module, for ENABLE INPUT and DISABLE INPUT, both status key 15 and 20 conditions can occur; thus a status key condition 80 is a valid possibility.

X3J4 DOCUMENT A-63

SUBJECT: REPLACING LINE Phrase of SEND Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XIV-29, paragraph 3.6.4, general rule 16, SEND statement
2. Page XIV-26, paragraph 3.6.4, general rule 1, SEND statement
3. Page XIV-27, paragraph 3.6.4, general rule 2, SEND statement
4. Page XIV-30, paragraph 3.6.4, general rule 17, SEND statement
5. Page II-28, paragraph 7.1, item 3, message control system

DATE: December 8, 1987

QUESTION:

If the following message is sent:

FEED THE ANIMALS

and the next SEND to the same output queue(s) contains the REPLACING LINE phrase and sends the message:

STOP

What appears at the destination(s)?

STOP

or

STOP THE ANIMALS

General rule 16a on page XIV-29 states that "If the REPLACING phrase is specified, the characters transmitted by the SEND statement replace all characters which may have previously been transmitted to the same line beginning with the leftmost character position of the line." This implies that the sending program, the receiving program or terminal or the MCS must know either of the following for each destination:

1. The length of the last message sent/received

2. The maximum line length of each destination so that the new message may be padded with sufficient spaces.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that following completion of the sequence of operations described in the example, the line appearing on the output device will contain STOP as shown below assuming the device supports replacement of characters.

STOP

This interpretation is based on the following:

1. Reference 1, which refers to replacement of all the characters, and

2. Reference 2, which specifies that, for devices oriented to a fixed line size (printer, display screen, card punch, etc.), a message or message segment begins at the leftmost character position of the line and, if smaller than the physical line size, is released so as to appear space filled to the right.

In order for an implementation to support communication with devices oriented toward fixed line sizes, it is necessary that a mechanism exist for releasing a message or message segment so as to appear space filled to the right. Since the message control system (MCS) is responsible for performing device dependent tasks (see reference 5), X3.23-1985 implies that the mechanism exists in the MCS, or in the interface between the COBOL object program and the MCS, or in the interface between the MCS and the communication device.

X3J4 DOCUMENT A-64

SUBJECT: PADDING CHARACTER Clause

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-12, paragraph 2.7.4, general rule 1, PADDING CHARACTER clause
2. Page VII-49, paragraph 4.5.4, general rule 6, REWRITE statement
3. Pages VII-36 and VII-37, paragraph 4.2.4, general rule 3A, Effect of CLOSE on input files and input-output files
4. Page VII-1, paragraph 1.3.1, first paragraph, organization of sequential files
5. Page VII-12, paragraph 2.7.4, general rule 2, PADDING CHARACTER clause

DATE: December 9, 1988

QUESTION:

Does the PADDING CHARACTER clause give one the ability to effectively delete records in a sequential file?

Suppose a sequential file, SEQ-FILE, exists consisting of exactly 5 records. Consider the execution of the following COBOL program:

```
FILE-CONTROL.
    SELECT SEQ-FILE  ORGANIZATION IS SEQUENTIAL
        ACCESS MODE IS SEQUENTIAL
        PADDING CHARACTER IS "P".

DATA DIVISION.
FD  SEQ-FILE.
01  SEQ-REC  PIC X(80).
```

```
PROCEDURE DIVISION.
BEGIN.
    OPEN I-O SEQ-FILE.
    PERFORM 3 TIMES
        READ SEQ-FILE
    END-PERFORM.
    MOVE ALL "P" TO SEQ-REC.
    REWRITE SEQ-REC.
    CLOSE SEQ-FILE.
    OPEN INPUT SEQ-FILE.
    PERFORM 5 TIMES
        READ SEQ-FILE
            AT END  DISPLAY "RECORD 3 DELETED"
        END-READ
        STOP RUN
    END-PERFORM.
```

Will the message **RECORD 3 DELETED** be displayed?

The supporting argument is as follows: General rule 1 of the PADDING CHARACTER clause states, in part: "During input operations, a logical record which consists solely of padding characters will be ignored."

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that for the example program the message **RECORD 3 DELETED** will be displayed when both the REWRITE statement is legal and the padding character is applicable to the device, because the third record is ignored (see reference 5).

X3J4 DOCUMENT A-65

SUBJECT: File Attribute Conflict Condition

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-4, paragraph 1.3.5, item 3f, I-O status 39
2. Page VII-4, paragraph 1.3.5, item 4d, I-O status 44
3. Page VII-5, paragraph 1.3.7, file attribute conflict condition
4. Page II-1, paragraph 2.1, file attributes
5. Page VII-2, paragraph 1.3.5, I-O status

DATE: June 17, 1988

QUESTION:

How can a REWRITE or WRITE statement cause a file attribute conflict condition?

Reference 3 states that the file attribute conflict condition can result from the execution of an OPEN, REWRITE, or REWRITE statement.

However, I-O status code 39 is the only status code that can result from detection of a file attribute conflict condition (reference 1 and reference 5) and is only associated with an OPEN statement. The only somewhat-related error for a WRITE or REWRITE is status code 44 (reference 2) which is not a file attribute conflict condition, but a boundary violation logic error.

It appears this may be a semantic problem. Is a "file attribute conflict condition" a superset of the other "exception conditions"? If would make more sense if it were a subset of possible "exception conditions". Therefore, reference 3 should be modified to remove "REWRITE or WRITE".

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that ANSI COBOL X3.23-1985 does not define how the file attribute conflict condition can result from the execution of a REWRITE or WRITE statement.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-65. Pages 96 through 100 of this document contain the proposed corrections to pages VII-5, VII-41, VII-43, VIII-6, VIII-24, VIII-25, IX-7, IX-26, and IX-27 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-69

SUBJECT: ACCEPT Conversion

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-71, paragraph 6.5.4, general rule 1, ACCEPT statement

DATE: June 10, 1988

QUESTION:

Reference 1 says "Any conversion of data required between the hardware device and the data item referenced by identifier-1 is defined by the implementor."

ANSI COBOL X3.23-1974 does not explicitly require conversion. This is a substantive change not called out in ANSI COBOL X3.23-1985, although "ACCEPT identifier" is listed in the Summary of Differences in Procedure Division on page XVII-31.

Does reference 1:

A. allow an implementor to define whether conversion occurs between the hardware device and the data item referenced by identifier-1, in the case where the class and category of identifier-1 requires different representation than presented by the device?

or

B. require the implementor to provide conversion, if any is needed, and allow the implementor to define how the conversion occurs?

Case A can result in incompatible data and would seem to be counter to the intent of the Standard. Yet the wording could be stretched to have this meaning, especially since a requirement for conversion is not listed as a substantive change.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that reference 1 allows but does not require an implementor to provide conversion in the transfer of data from a particular hardware device.

X3J4 DOCUMENT A-70

SUBJECT: REPLACE Statement Containing COPY Statement

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page XII-2, paragraph 2.3, syntax rule 7, COPY statement
2. Page XII-7, paragraph 3.4, general rule 4, REPLACE statement
3. Page III-25, definition of text word
4. Page III-18, definition of pseudo-text
5. Page XII-3, paragraph 2.4, general rule 1, COPY statement
6. Page XII-3, paragraph 2.4, general rule 2, COPY statement
7. Page XII-4, paragraph 2.4, general rule 8, COPY statement

DATE: December 9, 1988

QUESTION #1:

REPLACE ==COPY textname.== BY ==ABC==.

Is COPY allowed in pseudo-text in a REPLACE statement?

Since COPY is not a text word (see reference 3) and pseudo-text can only have text words, comment lines and the separator space (see reference 4), COPY is not allowed in pseudo-text.

In conflict, reference 1 states that COPY statements may appear anywhere a character-string or a separator, may occur, and reference 2 states that REPLACE statements are processed after all COPY statements.

These rules are contradictory.

X3J4 RESPONSE TO QUESTION #1:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the word COPY, bounded by separators, is allowed between pseudo-text delimiters in a REPLACE statement, if it begins a valid COPY statement. The REPLACE statement, including any of its pseudo-text operands, is processed after processing COPY statements (see reference 2). Processing the COPY statement results in its logical replacement by the referenced library text (see reference 6). The requirements of references 3 and 4, for a REPLACE statement, may only be applied to the source program text resulting from the processing of any COPY statements.

QUESTION #2:

REPLACE COPY textname1. COPY textname2. .

More generally, is a COPY statement allowed anywhere in a REPLACE statement?

If COPY statements are not allowed in a COPY statement (see reference 1), then why should COPY statements be allowed in a REPLACE statement? If allowed, then the syntactic inconsistencies that are removed from COPY, via reference 1, would exist for REPLACE.

We suggest that COPY statements should not be allowed in REPLACE statements by amending reference 1. This would make both examples given above and on the previous page illegal.

X3J4 RESPONSE TO QUESTION #2:

The X3J4 interpretation of ANSI X3.23-1985 is that COPY statements are allowed in REPLACE statements. Since COPY statements are processed before REPLACE statements (see reference 2) and are logically replaced by the referenced library text (see reference 6), the problems posed by COPY statements contained in COPY statements are not relevant.

QUESTION #3:

However, if COPY statements are not allowed in REPLACE statements and COPY statements are processed before REPLACE statements, when is the syntactic correctness of a REPLACE statement determined?

X3J4 RESPONSE TO QUESTION #3:

The X3J4 interpretation of ANSI X3.23-1985 is that the Standard does not specify when the syntactic correctness of a REPLACE statement is determined (see reference 7). The only requirement is that REPLACE statements be processed after the processing of all COPY statements in a source program (see reference 2).

X3J4 DOCUMENT A-71

SUBJECT: Floating Insertion Editing

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-34, paragraph 5.9.5, editing rule 7, PICTURE clause
2. Page VI-37, paragraph 5.9.6, PICTURE precedence chart

DATE: October 7, 1988

QUESTION:

In the second paragraph of reference 1, it is stated that the fixed insertion characters CR and DB may be used immediately to the right of a string of floating insertion currency symbols.

Does this mean that the other editing sign control symbols + and – are not allowed in this situation?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that fixed insertion characters + and – are allowed immediately to the right of a string of floating insertion currency symbols, as indicated in the PICTURE precedence table (see reference 2).

X3J4 DOCUMENT A-72

SUBJECT: Precedence of USE Procedures in Nested Programs

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IX-40, paragraph 4.8.4, general rule 4, USE statement
2. Page X-34, paragraph 5.5.4, general rule 1, USE statement

DATE: December 2, 1988

QUESTION:

Reference 1 says: "When file-name-1 is specified explicitly, no other USE statement applies to file-name-1." Reference 2 specifies special precedence rules when programs are contained within other programs, and says that " ... only the first qualifying declarative will be selected ... . The declarative which is selected for execution must satisfy the rules for execution of that declarative."

Does reference 1 have any meaning across program boundaries when selecting the "first qualifying" declarative to be invoked during the execution of a contained program? For example, in the following illustration, when an error occurs during execution of the READ statement in PROGRAMC, is the USE statement for file-name-1 executed because the USE statement for input does not satisfy the rules ("When file-name-1 ... no other USE statement applies ... ") according to reference 1?

Or for contained programs, does reference 1 apply only to precedence of declaratives within the single program within which it is specified? In this case, the input declarative associated with the USE statement for input would be invoked in the example on the next page.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  PROGRAMA.
  ...
PROCEDURE DIVISION.
DECLARATIVES.
DECLARATIVE-A SECTION.
    USE GLOBAL AFTER STANDARD EXCEPTION PROCEDURE ON file-name-1.
  ...
  ...
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  PROGRAMB.
         ...
       PROCEDURE DIVISION.
       DECLARATIVE-B SECTION.
           USE GLOBAL AFTER STANDARD EXCEPTION PROCEDURE ON INPUT.
         ...
         ...
              IDENTIFICATION DIVISION.
              PROGRAM-ID.  PROGRAMC.
                ...
              PROCEDURE DIVISION.
                ...
              READ file-name-1 ...
                ...
              END PROGRAM PROGRAMC.
        END PROGRAM PROGRAMB.
END PROGRAM PROGRAMA.
```

X3J4 RESPONSE:

The X3J4 interpretation of ANSI X3.23-1985 is the following:

Reference 1 only applies when more than one applicable USE procedure is specified in the same program or subprogram and at the same level of scope. Reference 2 specifies the order of precedence used to select a declarative when programs are contained in other programs.

In the given example, the following steps would be taken to select a declarative:

1. Is there an applicable declarative within PROGRAMC? If no, go to step 2.

2. Is there an applicable GLOBAL declarative within PROGRAMB? If yes, execute the declarative in PROGRAMB.

X3J4 DOCUMENT A-73

SUBJECT: OCCURS and REDEFINES

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-27, paragraph 5.8.4, general rule 2a, OCCURS clause
2. Page VI-30, paragraph 5.9.4, general rule 4, PICTURE clause.
3. Page VI-39, paragraph 5.10.4, general rule 1, REDEFINES clause
4. Page VI-44, paragraph 5.13.4, general rule 2, SYNCHRONIZED clause
5. Page II-12, paragraph 4.1, table definition

DATE: October 7, 1988

QUESTION:

I can find no rule preventing the REDEFINES and OCCURS clauses from appearing in the same data definition. If both do appear in the same data definition, does the first occurrence redefine the redefined item, and each subsequent occurrence redefine the previous one, or does the whole table redefine the redefined item?

Example:

```
01  W05-STATE-AREA.
    03  W05-STATE-CODES.
        05  FILLER  PIC XX  VALUE IS "AK".
        05  FILLER  PIC XX  VALUE IS "AL".
         ...
         ...  (repeat 05 level to get 52 FILLER items)
         ...
    03  W05-STATE-TABLE  REDEFINES W-05-STATE-CODES  OCCURS 52 TIMES.
        05  STATE-ABRV  PIC XX.
```

Will the definition of W05-STATE-TABLE produce 52 occurrences of a 104-byte item or 52 occurrences of a two-byte item?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the whole table redefines the redefined item and the definition of W05-STATE-TABLE produces 52 occurrences of a two-character item. The size of the table defined by W05-STATE-TABLE (see reference 5) is completely determined by the rules in references 1, 2, and 4 to be 104 characters. The storage for W05-STATE-TABLE starts at W05-STATE-CODES and continues for 104 characters (see reference 3).

X3J4 DOCUMENT A-74

SUBJECT: NOT INVALID KEY Phrase

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IX-28, paragraph 4.5.3, syntax rule 7, READ statement in Indexed I-O module
2. Page IX-41, paragraph 4.9.3, syntax rule 3, WRITE statement in Indexed I-O module
3. Page VII-51, paragraph 4.6.4, general rule 5, USE statement in Sequential I-O module
4. Page IX-40, paragraph 4.8.4, general rule 5, ACCESS MODE clause in Indexed I-O module

DATE: December 2, 1988

QUESTION:

Reference 2 states: The INVALID KEY phrase must be specified if an applicable USE AFTER STANDARD EXCEPTION procedure is not specified for the associated file-name. Does the existence of a NOT INVALID KEY phrase satisfy this requirement? Similar questions arise for the AT END phrase and possibly other phrases. Precise definitions for these optional phrases are needed.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is the following:

The existence of a NOT INVALID KEY phrase does not satisfy the requirement for an INVALID KEY phrase when no applicable USE AFTER STANDARD EXCEPTION procedure is specified (see references 1 and 2). In the same way, a NOT AT END phrase does not satisfy the requirement for an AT END phrase when no applicable USE AFTER STANDARD EXCEPTION procedure is specified (see reference 1).

X3J4 DOCUMENT A-75

SUBJECT: Repeated WHEN Phrase of EVALUATE

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-84, paragraph 6.13.2, general format of EVALUATE statement
2. Page VI-86, paragraph 6.13.4, general rule 2a, EVALUATE statement
3. Page VI-86, paragraph 6.13.4, general rule 3a, EVALUATE statement

DATE: December 2, 1988

QUESTION:

According to the general format for the EVALUATE statement, the following sentence must be allowed:

```
EVALUATE data-item
   WHEN 5
   WHEN 6  MOVE ...
   WHEN OTHER  DISPLAY ...
END-EVALUATE.
```

The problem is the first WHEN phrase in the above example. Please see the braces in the general format that allow multiple WHEN phrases to be specified before imperative-statement-1. I think that neither the syntax rules nor the general rules exclude the first WHEN phrase in the above example.

However, I think the first WHEN phrase in the above example should not be allowed because such a phrase without a corresponding imperative-statement will create more confusion than give help. If data-item is equal to 5, execution will continue with the MOVE statement. Is that the idea of X3J4?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the syntax of the EVALUATE statement in the above example is correct (see reference 1). If data-item

contained the value 5, the first WHEN phrase would be selected (see reference 2). Execution would continue with the MOVE statement (see reference 3).

If the programmer had to repeat the imperative statement for each WHEN phrase there is a greater likelihood for programmer error. Also, the meaning of the statement would be less clear. If the imperative statement were repeated, it would require detailed inspection to determine that the actions of the two WHEN phrases were the same.

X3J4 DOCUMENT A-76

SUBJECT: Uppercase or Lowercase Letters in Program-Name in CALL or CANCEL

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IV-5, paragraph 4.2.2.1, COBOL words
2. Page IV-6, paragraph 4.2.2.1.1, user-defined words
3. Page VI-7, paragraph 3.3, syntax rule 1, PROGRAM-ID paragraph
4. Page IV-9, paragraph 4.2.2.2.1, nonnumeric literal
5. Page X-12, paragraph 3.1, PROGRAM-ID and nested source programs

DATE: October 7, 1988

QUESTION:

Reference 1 says that in a COBOL word each lowercase letter is equivalent to its corresponding uppercase letter. Reference 2 says a program-name is a COBOL word. Reference 3 is the only place where the term program-name appears in a general format.

In reference 4, it is clear that uppercase and lowercase letters are not equivalent in nonnumeric literals. Obviously, they are not regarded by the COBOL language as equivalent in data stored in a data item.

Reference 5 says that in a CALL of a COBOL program the literal following the verb, or the content of the data item reference by the identifier following the verb, is a program-name. Does this mean that in searching for a COBOL program named in a CALL statement a COBOL system must treat uppercase and lowercase letters as equivalent?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI X3.23-1985 is that, in comparison of program names during the search of a program named in a CALL or CANCEL statement, if the called program is a COBOL program, uppercase letters are considered to be equal to their corresponding lowercase letters. If the called program is not a COBOL program, the rules for these comparisons are defined by the implementor.

X3J4 DOCUMENT A-77

SUBJECT: Reference Modification on Variable Length Groups

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IV-23, paragraph 4.3.8.3.4, general rule 4b, reference modification
2. Page VI-28, paragraph 5.8.4, general rule 3a, OCCURS clause
3. Page VI-28, paragraph 5.8.4, general rule 3b, OCCURS clause
4. Page VI-26, paragraph 5.8.2, general format 2, OCCURS DEPENDING ON clause

DATE: January 16, 1989

QUESTION:

Reference 1 states that, if the length is omitted from a reference modifier, the data item referenced extends up to and includes the rightmost character of the data item being reference modified.

Reference 2 says (for most situations) that, when a group item having a subordinate item with a DEPENDING ON clause is referenced, only the part of the group item indicated by the current value of the item referenced in the DEPENDING ON clause is used.

In determining the rightmost character of such a group for the purposes of reference modification, is the maximum length of the group item or its current length as indicated by the DEPENDING ON item used?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI X3.23-1985 is that reference modification creates a unique data item from the referenced data item. The length of this referenced data item is determined by first applying the rules of references 2 and 3. Subsequently, the rules of reference modification are applied to determine the length of the unique data item created through reference modification.

X3J4 DOCUMENT A-78

SUBJECT: Collating Sequences When Sorting Indexed Files

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-12, paragraph 4.4.4, general rules 7 and 8, OBJECT-COMPUTER
2. Page IX-9, paragraph 2.3.4, general rule 3, file control entry for indexed file
3. Page IX-42, paragraph 4.9.4, general rule 14, WRITE statement
4. Page XI-10, paragraph 4.1.4, general rule 5b, MERGE statement
5. Page XI-18, paragraph 4.4.4, general rule 5b, SORT statement
6. Page XVII-47, item 77
7. Page XVII-49, item 96

DATE: October 10, 1988

QUESTION:

Question 1:

Reference 1 states, in general rules 7 and 8, the places where the program collating sequence is used. None of these general rules mentions the ordering of an indexed file. Are indexed files therefore always ordered according to the native collating sequence?

Question 2:

If so, a second question arises. References 6 and 7 say that indexed files can appear in MERGE and SORT statements. References 4 and 5 say that, if no collating sequence is established in the MERGE or SORT statement, the merge or sort operation will order the file according to the program collating sequence. If the GIVING file is indexed, will this not create a file that conflicts with the principle established in question 1?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is as follows:

Question 1: Indexed files are always ordered according to the native collating sequence as indicated by reference 2.

Question 2: The collating sequence used by a sort or merge operation determines the order in which the records are returned from the operation. If a GIVING file has indexed organization, and a sort or merge operation uses the program collating sequence, and for some values of the primary key (which is also the sort or merge key) the ordering of the keys differs in the two collating sequences, the results depend on the file access mode. If the access is sequential, general rule 14 of the WRITE statement is violated (see reference 3). If access is dynamic, the records will be returned and written in program collating sequence order, but the ordering of records in the GIVING file will be by native collating sequence. Thus there is no conflict.

X3J4 DOCUMENT A-80

SUBJECT: Transfer of Control and Exception Condition

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-46, paragraph 4.4.4, general rule 11b, READ statement
2. Page VII-51, paragraph 4.6.4, general rule 6, USE statement

DATE: December 2, 1988

QUESTION:

Reference 1 says: "If an exception condition which is not an at end condition exists, control is transferred according to rules of the execution of the USE statement following the execution of any USE AFTER EXCEPTION PROCEDURE applicable to file-name-1. (See page VII-50, The USE Statement.)"

The USE statement then describes only the transfer of control for the case where a USE AFTER EXCEPTION procedure is present (see reference 2).

I am interpreting "according to" as used in reference 1 and similar text through input-output, to mean that reference 2 describes the transfer of control regardless of whether or not there is an applicable USE AFTER EXCEPTION procedure.

Is this correct?

If not, then what is the transfer of control when there is no USE AFTER EXCEPTION procedure?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that it is correct that reference 1 refers to the rule stated in reference 2. Reference 1 states that the transfer of control takes place following the execution of "any USE AFTER EXCEPTION procedure applicable". This implies that there may not be an applicable USE AFTER EXCEPTION procedure. The rules in reference 2 are to be applied in any case.

X3J4 DOCUMENT A-81

SUBJECT: EVALUATE Statement Scope Rules

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IV-37, paragraph 6.4.2.1.1, definition of conditional statement
2. Page IV-39, paragraph 6.4.2.3.1, definition of imperative statement
3. Page VI-84, paragraph 6.13.2, general rule of EVALUATE statement
4. Page I-9, paragraph 1.6, definition of a conforming source program

DATE: October 7, 1988

QUESTION:

I am interested in a clarification of ANSI COBOL X3.23-1985. The two programs (PROGRAMA and PROGRAMB) on the next two pages would appear to be identical, based on the reading of the Standard.

The scope termination rules in paragraph 6.4.3 on page IV-40 state:

"Whenever any statement is contained within another statement, the next phrase of the containing statement following the contained statement terminates the scope of any unterminated contained statement."

"When statements are nested within other statements which allow optional conditional phrases, any optional conditional phrase encountered is considered to be the next phrase of the nearest preceding unterminated statement with which that phrase is permitted to be associated according to the general format and the syntax rules for that statement, but with which no such phrase has already been associated."

Because the WHEN OTHER is the last clause of the contained EVALUATE statement, I believe that the WHEN phrase at line 004200 is properly part of the containing EVALUATE statement (beginning at line 003000). That is, PROGRAMA is equivalent to PROGRAMB; thus both should produce the following output:

```
01: 1, 3-7 (ODD)
02: 2, 8
03: 1, 3-7 (ODD)
04: 1, 3-7 (NOT ODD)
05: 1, 3-7 (ODD)
06: 1, 3-7 (NOT ODD)
07: 1, 3-7 (ODD)
08: 2, 8
09: ANY
10: ANY
```

Program A:

```
000100 IDENTIFICATION DIVISION.
000500 PROGRAM-ID.  PROGRAMA.
000600
001700 DATA DIVISION.
001800 WORKING-STORAGE SECTION.
001900 01  WORK-VALUE  PIC 99.
001902
002000 PROCEDURE DIVISION.
002100 A-MAIN.
002200     PERFORM A-EVALUATES
002300         VARYING WORK-VALUE FROM 1 BY 1 UNTIL WORK-VALUE > 10
002700     STOP RUN.
002800
002900 A-EVALUATES.
003000     EVALUATE WORK-VALUE
003100         WHEN 1
003200         WHEN 3 THRU 7
003300             EVALUATE WORK-VALUE
003400                 WHEN 1
003500                 WHEN 3
003600                 WHEN 5
003700                 WHEN 7
003800                     DISPLAY WORK-VALUE ": 1, 3-7 (ODD)"
003900                 WHEN OTHER
004000                     DISPLAY WORK-VALUE ": 1, 3-7 (NOT ODD)"
004100*************END-EVALUATE
004200         WHEN 2
004300         WHEN 8
004400              DISPLAY WORK-VALUE ":2, 8"
004500         WHEN ANY
004600              DISPLAY WORK-VALUE ": ANY"
004700         WHEN OTHER
004800              DISPLAY WORK-VALUE ": OTHER"
004900     END-EVALUATE.
```

Program B:

```
000100 IDENTIFICATION DIVISION.
000500 PROGRAM-ID.  PROGRAMB.
000600
001700 DATA DIVISION.
001800 WORKING-STORAGE SECTION.
001900 01  WORK-VALUE  PIC 99.
001902
002000 PROCEDURE DIVISION.
002100 A-MAIN.
002200     PERFORM A-EVALUATES
002300         VARYING WORK-VALUE FROM 1 BY 1 UNTIL WORK-VALUE > 10
002700     STOP RUN.
002800
002900 A-EVALUATES.
003000     EVALUATE WORK-VALUE
003100         WHEN 1
003200         WHEN 3 THRU 7
003205             PERFORM A-NESTED
004200         WHEN 2
004300         WHEN 8
004400             DISPLAY WORK-VALUE  ": 2, 8"
004500         WHEN ANY
004600             DISPLAY WORK-VALUE ": ANY"
004700         WHEN OTHER
004800             DISPLAY WORK-VALUE ": OTHER"
004900     END-EVALUATE.
005000
005100 A-NESTED.
005200     EVALUATE WORK-VALUE
005300         WHEN 1
005400         WHEN 3
005500         WHEN 5
005600         WHEN 7
005700             DISPLAY WORK-VALUE ": 1, 3-7 (ODD)"
005800         WHEN OTHER
005900             DISPLAY WORK-VALUE ": 1, 3-7 (NOT ODD)"
006000     END-EVALUATE.
```

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the program having the program name PROGRAMA does not conform to Standard COBOL and thus the output produced by its execution is not predictable based on Standard COBOL (see reference 4). The source text in PROGRAMA contains a conditional statement at lines 003300 through 004100 where Standard COBOL requires an imperative statement (see references 1, 2, and 3). The reference cited in the request is not relevant to this example.

X3J4 DOCUMENT A-82

SUBJECT: Pseudo-Text Replacement Involving Parentheses

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page IV-4, paragraph 4.2.1, rule 4, separators
2. Page IV-12, paragraph 4.2.2.3, PICTURE character-strings
3. Page IV-10, paragraph 4.2.2.2.1.2, syntax rule 1, nonnumeric literals
4. Page III-25, definition of text word, item 1
5. Page III-25, definition of text word, item 2
6. Page VI-30, paragraph 5.9.4, general rule 7, PICTURE Clause

DATE: October 7, 1988

QUESTION:

Question 1:

Given the PICTURE character-string X(2), formed according to reference 6, what is its text word sequence? According to reference 4, the parentheses are text words whence the text word sequence is X,(,2,). On the other hand, reference 1 and reference 2 imply that, within a PICTURE character-string, parentheses are not considered separators. Therefore only the third alternative of the definition of text word on page III-25 applies, i.e. the PICTURE character-string X(2) as a whole must be considered one text word.

Question 2:

Given the nonnumeric literal "A(B" , formed according to reference 3, what is its text word sequence? According to reference 4, the left parenthesis is a text word. Then, one may either conclude that the parenthesis breaks the literal and the text word sequence is ",A(,B," ; or one may nevertheless continue considering the quotation marks as the opening and closing delimiters of the literal: then there is no legal text word sequence including the left parenthesis text word and the literal must be illegal with respect to replacement processing. On the other hand reference 4 implies that the text word sequence consists of one text word formed by the literal as a whole.

X3J4 RESPONSE:

Question 1: The X3J4 interpretation of ANSI COBOL X3.23-1985 is that the PICTURE character-string X(2) results in four text words. Reference 5 states: "The right and left parenthesis characters, regardless of context within the library, source program, or pseudo-text, are always considered text words." There are therefore four text words given the PICTURE character-string X(2). The text word sequence is X,(,2,). The text word replacement process treats PICTURE character-strings in the same way as other source text.

Question 2: The X3J4 interpretation of ANSI COBOL X3.23-1985 is that nonnumeric literals are considered to be one text word. Reference 6 is a further qualification of reference 5 clarifying the intent of the Standard with regards to nonnumeric literals.

X3J4 DOCUMENT A-84

SUBJECT: PADDING CHARACTER Clause

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-12, paragraph 2.7.4, general rule 2, PADDING CHARACTER clause
2. Page VII-12, paragraph 2.7.4, general rule 5, PADDING CHARACTER clause

DATE: December 9, 1988

QUESTION:

Question 1:

Reference 1 says: "If the PADDING CHARACTER clause is not applicable to the device type to which the file is assigned, the creation or recognition of padding characters does not occur."

How is it determined whether the PADDING CHARACTER clause is applicable to a given device type?

Is it whether records can be blocked on that device; whether blocks can be padded; or is it entirely up to the implementor to specify whether the PADDING CHARACTER clause is applicable to the device type?

Question 2:

Reference 2 says: "If the PADDING CHARACTER clause is not specified, the value used for the padding character will be defined by the implementor."

Assuming an implementor supports a device type for which a PADDING CHARACTER clause is applicable, does reference 2 require the implementor to specify a character for creation and recognition of padding?

This would seem to inhibit portability of programs accessing files for which PADDING CHARACTER is not specified. If implementor A chose hex 00 and implementor B chose

a space character, all padding records created by implementor A would be read as data by implementor B, and records containing spaces would be ignored.

It would seem more reasonable to say that if the PADDING CHARACTER clause is not specified, creation or recognition of padding character does not occur. Is it within the bounds of reference 2 to allow this as the implementor definition when the PADDING CHARACTER clause is not specified?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is as follows:

Question 1: It is up to the implementor to specify whether the PADDING CHARACTER clause is applicable to a particular device type (reference 1).

Question 2: Reference 2 requires the implementor to define the default value for the padding character. It does not allow the implementor to bypass padding character processing on devices for which the implementor defines that padding is applicable.

X3J4 DOCUMENT A-85

SUBJECT: I-O Status Value 37

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VII-4, paragraph 1.3.5, item 3d, I-O status 37, Sequential I-O module
2. Page VIII-4, paragraph 1.3.4, item 4c, I-O status 37, Relative I-O module
3. Page IX-4, paragraph 1.3.4, item 4c, I-O status 37, Indexed I-O module
4. Page XVII-75, item 40g, I-O status 37
5. Page II-1, paragraph 2.1, file attributes
6. Page I-3, paragraph 1.3, organization of document
7. Page VII-7, paragraph 2.3.3, syntax rule 3, file control
   Page VIII-8, paragraph 2.3.3, syntax rule 3, file control entry
   Page IX-8, paragraph 2.3.3, syntax rule 3, file control entry
8. Page VII-8, paragraph 2.3.4, general rule 3, file control entry
   Page VIII-9, paragraph 2.3.4, general rule 4, file control entry
   Page IX-9, paragraph 2.3.4, general rule 5, file control entry

DATE: December 9, 1988

QUESTION:

The definition of I-O status code 37 in references 1, 2, and 3 states that " ... an OPEN statement is attempted on a file and that file will not support the open mode specified ... ". The definition of I-O status code 37 in reference 4 interprets the above definition as "an OPEN statement is attempted on a file which is required to be a mass storage file but is not". This definition does not include any reference to "open mode". It also introduces the concept of a mass storage/non-mass storage conflict which is not present (at least explicitly) in the first definition.

The justification goes on to state that status code 37 is returned for an attempt to open a non-mass storage file which is declared in the program as indexed or relative. According to reference 5 (file attributes), such an attempt represents an attribute conflict for the organization attribute.

Question 1:

Is there a conflict in the two definitions and, if so, in regards to conformance, is the substantive changes section an actual part of the Standard or an interpretation of the Standard?

Question 2:

Which of status code 37 or status code 39 (file attribute conflict condition) is returned for an attempt to open a non-mass storage file declared as indexed or relative?

Question 3:

Are the three instances described in the justification the only instances for which status code 37 can be returned? That is, if an implementation supports a particular type of mass storage file which does not permit updates, does status code 37 apply?

Question 4:

If an implementation can detect an inconsistency relative to status code 37 during compilation and generates an error preventing object code generation, is the implementation in conformance with the Standard?

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is as follows:

Question 1: Yes, there is a conflict. The substantive changes section regarding I-O status value 37 was not updated to match the final definition contained in the body of the Standard and is incorrect. The substantive changes section is not a part of the Standard for conformance purposes (see reference 6).

Question 2: It would depend on which open mode was specified, and possibly on the order in which open mode and fixed file attributes are checked. If the open mode is supported by the file, I-O status value 37 does not apply.

Question 3: The instances described in the substantive changes section for I-O status value 37 are incorrect since they are based on an outdated definition of the value. References 1, 2, and 3 specify the conditions which cause I-O status values 37 to be returned. If an implementation has a storage medium that does not permit updates, I-O status value 37 would apply in the case of an OPEN I-O statement.

Question 4: An implementation is allowed to detect an inconsistency between the storage medium for the file established by the ASSIGN clause and a reference to that file in an OPEN statement (see references 7 and 8) during compilation. When no inconsistency is detected, an I-O status value 37 must be returned upon execution of an OPEN statement if the associated physical file does not support the open mode specified in that OPEN statement. If I-O status value 37 is not returned, the associated physical file must support all the operations allowed in the specified open mode.

PROPOSED CORRECTION TO X3.23-1985:

X3J4 has proposed a correction to ANSI COBOL X3.23-1985 as a result of the processing of the question in document A-85. Page 106 of this document contains the proposed correction to pages XVII-75 in ANSI COBOL X3.23-1985.

X3J4 DOCUMENT A-86

SUBJECT: SYMBOLIC CHARACTERS Clause

REFERENCES:

American National Standard COBOL X3.23-1985

1. Page VI-14, paragraph 4.5.3, syntax rules 8, 9, and 10, SPECIAL-NAMES paragraph
2. Page III-12, definition of integer

DATE: December 9, 1988

QUESTION:

Reference 1 says in effect that each symbolic character defined in a SYMBOLIC CHARACTERS clause represents the character at a given ordinal position in a corresponding integer in the clause.

For this purpose, do ordinal numbers start at zero, 1 or what? For example, in ASCII, where the lowest character in the collating sequence has code hexadecimal 00, decimal 0, would this character be considered the zeroth or the first? Would the letter "A", which has code hexadecimal 41, decimal 65, be considered the 65th or 66th character?

Since ASCII is such a widely used character set, I feel it would be a source of many programming errors if the number used in the SYMBOLIC CHARACTERS clause for an ASCII character has to be one more than the decimal value of the character's code, rather than equal to it.

X3J4 RESPONSE:

The X3J4 interpretation of ANSI COBOL X3.23-1985 is that a number used in the SYMBOLIC CHARACTERS clause for an ASCII character is one more than the decimal value of the character code. In this Standard, ordinal is used in its common English sense. Ordinal numbers start at 1.

Pages 90 - 116, Draft Correction Amendment have been deleted.

They have been superseded by the published correction amendment.

SECTION 4: INDEX OF INTERPRETATIONS

4.1 INTRODUCTION

This section of the COBOL Information Bulletin contains a cross reference index to the interpretations made by the X3J4 COBOL Technical Committee relative to ANSI COBOL X3.23-1985. This index is arranged in order by subject matter within American National Standard COBOL X3.23-1985.

4.2 DEFINITION OF AN IMPLEMENTATION OF STANDARD COBOL

4.3 NUCLEUS MODULE

Language Concepts

Environment Division

Data Division

Data Division in Nucleus Module (Continued)

Procedure Division

4.4 SEQUENTIAL I-O MODULE

Language Concepts

Environment Division

Data Division in Sequential I-O Module

Procedure Division

4.5 RELATIVE I-O MODULE

Language Concepts

Environment Division

Data Division

Procedure Division

Procedure Division of Relative I-O Module (Continued)

4.6 INDEXED I-O MODULE

Language Concepts

Environment Division

Data Division

Procedure Division

Procedure Division of Indexed I-O Module (Continued)

4.7 INTER-PROGRAM COMMUNICATION MODULE

Language Concepts

Data Division

Procedure Division

4.8 SORT-MERGE MODULE

Environment Division

Data Division

Procedure Division

4.9 SOURCE TEXT MANIPULATION MODULE

4.10 REPORT WRITER MODULE

Procedure Division

4.11 COMMUNICATION MODULE

Data Division

Procedure Division