SPIRIT Platform Blueprint

SPIRIT COBOL Language Portability Guide

(SPIRIT Issue 3.0)

Network Management Forum

SPIRIT Platform Blueprint

SPIRIT COBOL Language Portability Guide (SPIRIT Issue 3.0)

ISBN: N/A
Document Number: J409

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to:

Network Management Forum
1201 Mount Kemble Avenue
Morristown, NJ 07960
U.S.A.

Tel: +1 201 425 1900

Contents

Chapter 1

Introduction

1.1 Purpose

This document is intended to make application programs based on the SPIRIT COBOL specifications more portable.

SPIRIT COBOL is defined to improve the portability of application programs by eliminating the differences among implementations as far as possible. Therefore, a SPIRIT COBOL application program may be ported with little modification. The application program will necessarily contain some contrived coding in order to make the implementation-defined portion more portable.

The purpose of this document is to improve the portability of the implementation-defined portion.

Implementations may include extensions beyond the range defined by SPIRIT — these are not described here.

1.2 Programming Technique

This section describes the kinds of technique available for a program for which the wording ‘‘it should be rewritten’’ is specified in the guide:

1. Cases where changes are made on the word basis.

   Replace the words with the statement below, and gather occurrences of the statements into a specific area in the source program:

   COBOL: REPLACE statement

2. Cases where changes are made on the line basis.

   Separate the lines to be changed and then include them into the source program.

   COBOL: COPY statement

3. Cases where changes are made on the execution basis.

   Make subroutines from the parts to be changed.

   COBOL: program or PERFORM procedure

4. Cases of exception handling.

   Exception handling should be localised as described below, regardless of dependency on implementations.

   COBOL: use a USE statement and describe the post-process within the subsequent procedure

5. Cases other than the above.

When the methods shown above are not applicable, comments for rewriting information should be written *in situ*. How to write a comment is described below.

COBOL: Lines with ‘‘*’’ or ‘‘/’’ on column 7.

Limit Values

In SPIRIT COBOL, limit values are defined for each specification. An application program which exceeds those limit values may be executed properly, but is not portable. Application programs should be written within the limit value.

1.3 How to Read this Document

Each section of the guide is composed of the following items:

NAME

<type of guide>-<classification number>-<sequence number>

<type of guide> is CBL (COBOL Language).
<classification number> is a number assigned to CLASSIFICATION.
<sequence number> is a number within the CLASSIFICATION.

CLASSIFICATION

Classification based on the content.

TITLE

A title which represents the content.

CLAUSE

The section/chapter number and its title of the corresponding specification.

GUIDANCE

The matters to be followed in order to improve portability.

EXPLANATION

Reason why it should be done in this way.

EXAMPLE

An example which shows an application program without portability, if necessary, and a way to improve portability.

Category for which a Solution is Available

This category includes those programs for which, although implementation-defined, there is a solution, such as a coding method; for example, order of evaluation. The guidance for a case like this is expressed as ‘‘shall’’ or ‘‘should’’ for a requirement, and ‘‘shall not’’ or ‘‘should not’’ for a prohibition.

In EXAMPLE, the example of an application program of low portability is shown in Application Program Without Portability, and the portion which varies depending on implementation is explained. In Application Program With Portability, as a solution which does not depend on the implementation, an example of an application program of high portability is shown.

Category for which Rewriting is Required

An application program in which names, such as file names, or processing method, such as input/output, vary with implementations and which requires rewriting belong to this category.

The guidance for this category is expressed as ‘‘shall be rewritten’’ or ‘‘should be rewritten’’.

In EXAMPLE, the example of an application program of low portability is shown in Application Program Without Portability, and the portion which is not portable is explained.

Others

Reserved words, for example, are listed for reference in EXAMPLE.

Chapter 2

Application Program Portability Guide

NAME

CBL-1-1

CLASSIFICATION

Internal Representation

TITLE

Character set

CLAUSE

Section IV:	4.2.2.2.1	Non-numeric Literals
Section IV:	4.2.2.2.2a	National-Character Literals
Section IV:	7.2.4	Comment Lines

GUIDANCE

For non-numeric literals, national-character literals and comment lines, only the characters within the Common Character Sets should be used.

EXPLANATION

Some implementations cannot handle the characters outside the Common Character Sets, and the use of those characters outside the Common Character Sets may affect the portability of application program.

EXAMPLE

Application Program Without Portability

An example of the non-numeric literal:

```
77  MOJI  PIC  X(20)  .
         :
         MOVE  "$$$$$$$$$$$$$$$$$$$$"  TO  MOJI
```

An example of the national-character literal:

```
      :                                                  :
MOVE   N"①①①①①①①①①①"  TO  NIHONGO
```

An example of the comment line:

```
*    [ [ [  SAMPLE  PROGRAM  ] ] ]
```

NAME

CBL-1-2

CLASSIFICATION

Internal Representation

TITLE

Internal representation of data

CLAUSE

Section IV:	4.3.4	Selection of Character and National-Character Representation Radix
Section IV:	4.3.5	Algebraic Signs
Section VI:	5.10	The REDEFINES Clause
Section VI:	5.12	The SIGN Clause
Section VI:	5.14	The USAGE Clause

GUIDANCE

The internal representation of data should not be assumed.

EXPLANATION

The internal representation of data depends on the implementation. The representation of national-characters, including the size, depends on the implementation. The representation of numeric values, including the representation of signs, depends on the implementation. When the REDEFINES clause is used, the corresponding data item statement should be rewritten. The size of numeric items in the number of character positions for items with USAGE IS BINARY and USAGE IS PACKED-DECIMAL is defined as follows:

USAGE IS BINARY

The character positions are 2 when the number of digits specified by the PIC clause is 1 to 4, and 4 when 5 to 9, and 8 when 10 to 18.

USAGE IS PACKED-DECIMAL

The character positions are worked out as follows: the PIC clause is extracted and the numeric value is divided by 2; 1 is added to the result. Any remainder is ignored.

EXAMPLE

In an example of dependence on internal representation, a numeric item in PACKED-DECIMAL, and a character item are assumed represented equivalently by using a REDEFINES clause.

Application Program Without Portability

```
01      RECORD-1
        04  NUM  PIC  S999  PACKED-DECIMAL  .
01      FILLER  REDEFINES  RECORD-1  .
        04  CH1  PIC  X  .
        04  CH2  PIC  X  .

        MOVE  410  TO  NUM  .
        IF  CH1  EQUAL  TO  "A"  THEN   . . .
```

This example assumes that, in a given implementation, a numeric item is represented equivalently to the letter A in PACKED-DECIMAL (’41’ form the numeric value 410 inputs the initial character position (redefined in character item CH1) as x’41’).

NAME

CBL-1-3

CLASSIFICATION

Internal Representation

TITLE

Synchronization of data

CLAUSE

Section IV: 4.3.7 Item Alignment for Increased Object-Code Efficiency

GUIDANCE

SYNCHRONIZED clause should not be specified.

EXPLANATION

When the SYNCHRONIZED clause is used, the data item is stored so as to be aligned on boundaries determined by the implementation. Therefore, if the data items are used within a group, it may affect the results of the statements in which the group is used as an operand.

EXAMPLE

Application Program Without Portability

ITEM-1 in the example below is stored so as to be aligned on either one-byte or four-byte boundaries depending on the implementation.

```
WORKING-STORAGE SECTION  .
01        RECORD-1  .
          05  ITEM-0  PIC  X  VALUE  "A"  .
          05  ITEM-1  PIC  S9(5)  BINARY  SYNC  VALUE  123 .    . . . (1)
77        REC-2  PIC  X(5)  .
PROCEDURE DIVISION  .
          MOVE  RECORD-1  TO  REC-2 .    . . . (2)
```

In (1) above, as the SYNCHRONIZED clause is used, the ITEM-1 will be aligned on boundaries depending on the implementation. The RECORD-1 in (2) above will be aligned without truncation, if it is on one-byte boundaries, but with truncation, if it is aligned on four-byte boundaries.

Application Program With Portability

```
WORKING-STORAGE SECTION  .
01         RECORD-1  .
           05  ITEM-0  PIC  X  VALUE  "A"  .
           05  ITEM-1  S9(5)  BINARY  VALUE  123  .     . . . (3)
77         REC-2  PIC  C(5)  .
PROCEDURE DIVISION  .
           MOVE  RECORD-1  TO  REC-2  .     . . . (4)
```

As the SYNCHRONIZED clause is not used in (3) above, the RECORD-1 will use five character positions without truncation in (4) in any implementation.

NAME

CBL-1-4

CLASSIFICATION

Internal Representation

TITLE

Collating sequence of characters

CLAUSE

Section VI: 4.4 The OBJECT-COMPUTER Paragraph

GUIDANCE

The relation of characters should be determined by equal or not-equal condition. Specify STANDARD-1 or STANDARD-2 for a comparison of collating sequence of one-byte characters.

EXPLANATION

The collating sequence for the characters is not defined, except the character collating sequence which is defined in the Common Character Sets. They depend on their code values in the implementation.

EXAMPLE

Application Program Without Portability

```
ENVIRONMENT  DIVISION  .

CONFIGURATION  SECTION  .

OBJECT-COMPUTER .  SPIRIT-COMPUTER

        PROGRAM  COLLATING  SEQUENCE  SPIRIT-CODE  .

SPECIAL-NAMES  .

        ALPHABET  SPIRIT-CODE  IS  NATIVE  .    . . . (1)

PROCEDURE  DIVISION  .

        IF  CHAR-1  GREATER  THAN  "0" . . .
```

As the NATIVE phrase is specified for SPIRIT-CODE in (1), the native collating sequence is used.

Application Program With Portability

```
ENVIRONMENT  DIVISION  .
CONFIGURATION  SECTION  .
OBJECT-COMPUTER .  SPIRIT-COMPUTER
          PROGRAM  COLLATING  SEQUENCE  SPIRIT-CODE  .
ALPHABET  SPIRIT-CODE  IS  STANDARD-1  .     . . . (2)
PROCEDURE  DIVISION  .
          IF  CHAR-1  GREATER  THAN  "0" . . .
```

As the STANDARD-1 phrase is specified for SPIRIT-CODE in (2), the STANDARD-1 collating sequence is used.

NAME

CBL-2-1

CLASSIFICATION

Precision

TITLE

Precision of arithmetic expressions

CLAUSE

Section VI: 6.2 Arithmetic Expressions

GUIDANCE

The use of complex arithmetic expressions should be avoided, where the portability of precision of operation is important.

EXPLANATION

As the precision of the intermediate result depends on the implementation, application programs using the complex arithmetic expressions may not be portable.

EXAMPLE

The complex arithmetic expression means an arithmetic expression which involves divisions or which requires an intermediate result of more than or equal to 19 digits.

Application Program Without Portability

```
01  I  PIC  9(2)  VALUE  10  .
01  J  PIC  9  VALUE  4  .
           :
  COMPUTE  I  =  I/J*  J  .
```

The implementation which generates an intermediate result without a decimal point may result in a size error in a division operation and the result is not accurate.

Application Program With Portability

```
01  I  PIC  9(2)  VALUE  10  .
01  J  PIC  9  VALUE  4  .
01  K  PIC  9V9  .
           :
DIVIDE  I  BY  J  GIVING  K .  COMPUTE  I = K*  J  .
           or
  COMPUTE  I  =  I*  J/J  .
```

NAME

CBL-3-1

CLASSIFICATION

Input-Output

TITLE

Notes for the RECORD clause

CLAUSE

Section II:	2.1.4.3	Inplementor-defined Record Types
Section VII:	3.8	The RECORD Clause
Section VIII:	3.2	The File Description Entry
Section IX:	3.2	The File Description Entry

GUIDANCE

The RECORD clause should be specified only if the record type of file could be specified explicitly. If a fixed-length record were specified in this case, the following must be used:

RECORD CONTAINS INTEGER-1 CHARACTERS

At this time, it may be necessary to rewrite integer-1. If a variable-length record were specified, the following should be used:

RECORD IS VARYING [DEPENDING ON DATA-NAME-1]

If a range of character positions in the record is specified, it may be necessary to rewrite the integer — specifying a range is not recommended.

EXPLANATION

If no RECORD clause is specified, or if a range of character positions is specified in the RECORD clause (Format 3), it is implementor-defined whether fixed-length records or variable-length records are obtained. However, if the file is converted into the record type suitable for the implementation, the application program is not affected. If such conversion is not desirable, that is, if the RECORD type would be specified explicitly, it is necessary to specify the record clause. At this time, it may be necessary to rewrite the integer specifying the size of character positions because the relation between national-character positions and character positions is different depending on the implementation.

NAME

CBL-3-2

CLASSIFICATION

Input-Output

TITLE

Actions to be taken when I-O Status indicates error conditions

CLAUSE

Section VII:	1.3.5	I-O Status
Section VIII:	1.3.4	I-O Status
Section IX:	1.3.4	I-O Status

GUIDANCE

If I-O status indicates an error condition, handle the case by specifying a USE statement in the PROCEDURE DIVISION.

EXPLANATION

When a USE procedure is not specified, the action taken by an implementation for an I-O status expressing an error condition varies with implementations. Therefore, handle the case by specifying a USE procedure. If the I-O status expresses a critical error condition, do not expect that control is transferred to the next sequential statement of the relevant input-output statement.

EXAMPLE

When a USE procedure is not specified, the actions taken by the implementation vary as follows:

Implementation-A	Continues processing after execution of some actions.
Implementation-B	Ends abnormally.

NAME

CBL-3-3

CLASSIFICATION

Input-Output

TITLE

I-O Status = 9x

CLAUSE

Section VII:	1.3.5	I-O Status
Section VIII:	1.3.4	I-O Status
Section IX:	1.3.4	I-O Status

GUIDANCE

Do not use I-O Status = 9x.

EXPLANATION

An application program which uses I-O Status = 9x is not portable, because the I-O Status = 9x varies with implementations.

EXAMPLE

Application Program Without Portability

```
SELECT  A  FILE . . .
        FILE  STATUS  IS  F-ST  .
IF  F-ST  =  "90"   . . .
```

Application Program With Portability

```
SELECT  A  FILE . . .
        FILE  STATUS  IS  F-ST  .
IF  F-ST  NOT  =  "00"   . . .
```

NAME

CBL-3-4

CLASSIFICATION

Input-Output

TITLE

The ASSIGN clause (implementor-name, literal)

CLAUSE

Section VII:	2.3	The File Control Entry
Section VIII:	2.3	The File Control Entry
Section IX:	2.3	The File Control Entry

GUIDANCE

The implementor-name and literal of the ASSIGN clause must be rewritten.

EXPLANATION

The implementor-name and literal of the ASSIGN clause are implementation-dependent.

EXAMPLE

Application Program Without Portability

```
SELECT  MYFILE  ASSIGN  "¥SPIRIT¥COBOL.DAT" .
```

NAME

CBL-3-5

CLASSIFICATION

Input-Output

TITLE

The ASSIGN clause (consistency of implementor-name and literal)

CLAUSE

Section VII:	2.3	The File Control Entry
Section VIII:	2.3	The File Control Entry
Section IX:	2.3	The File Control Entry

GUIDANCE

The same specification should be used for the same file.

EXPLANATION

The consistency rules for implementor-name and literal are implementor-defined, and they depend on file systems.

EXAMPLE

In the examples below, it is assumed that the consistency rule is that in the literal ‘‘xx-yyyy’’, if yyyy are the same, they reference the same file, regardless of the character of xx.

Application Program Without Portability

Program-A:

```
SELECT  MYFILE  ASSIGN  "AS-FILE1"  .

FD  MYFILE  IS  EXTERNAL
```

Program-B:

```
SELECT  MYFILE  ASSIGN  "FILE1"  .

FD  MYFILE  IS  EXTERNAL
```

Application Program With Portability

Program-A:

```
SELECT  MYFILE  ASSIGN  "AS-FILE1"  .

FD  MYFILE  IS  EXTERNAL
```

Program-B:

```
SELECT  MYFILE  ASSIGN  "AS-FILE1"  .

FD  MYFILE  IS  EXTERNAL
```

NAME

CBL-3-6

CLASSIFICATION

Input-Output

TITLE

Omission of the RESERVE clause

CLAUSE

Section VII: 2.9 The RESERVE Clause

GUIDANCE

The RESERVE clause should not be specified.

EXPLANATION

The maximum number of allowed input-output areas varies with the implementation. Moreover, the number of input-output areas is usually specified outside the COBOL program.

EXAMPLE

Application Program Without Portability

```
SELECT  MYFILE  ASSIGN  "FILE1"  RESERVE  10  AREAS  .
```

Application Program With Portability

```
SELECT  MYFILE  ASSIGN  "FILE1"  .
```

NAME

CBL-3-7

CLASSIFICATION

Input-Output

TITLE

Mnemonic-name in a WRITE statement

CLAUSE

Section VII: 4.7 The WRITE Statement

GUIDANCE

A mnemonic-name should not be used in a WRITE statement.

EXPLANATION

A mnemonic-name means a specific function defined by the implementor. Its meaning varies with systems.

EXAMPLE

In a certain implementation, a C05 may be a mnemonic-name which means "advancing one line", but it is not always the mnemonic-name of the same meaning in other implementations.

Application Program Without Portability

```
SPECIAL-NAMES  .
        C05  IS  NEXT-LINE  .
WRITE  REC-1  ADVANCING  NEXT-LINE  .
```

Application Program With Portability

```
WRITE  REC-1  ADVANCING  1  LINE  .
```

NAME

CBL-3-8

CLASSIFICATION

Input-Output

TITLE

PAGE phrase without the LINAGE clause

CLAUSE

Section VII: 4.7 The WRITE Statement

GUIDANCE

Do not specify a PAGE phrase without the LINAGE clause.

EXPLANATION

If the LINAGE clause is not specified, the printing position on the next page depends on the printer or implementation.

EXAMPLE

Application Program Without Portability

```
FD  MYFILE  .
        WRITE  REC-1  ADVANCING  PAGE  .
```

Application Program With Portability

```
FD  MYFILE  LINAGE  IS  30  LINES  .
        WRITE  REC-1  ADVANCING  PAGE  .
```

NAME

CBL-3-9

CLASSIFICATION

Input-Output

TITLE

The PADDING CHARACTER clause

CLAUSE

Section VII: 2.7 The PADDING CHARACTER Clause

GUIDANCE

The PADDING CHARACTER clause should not be specified.

EXPLANATION

The process specified by this clause is outside of the COBOL program — the system is responsible.

NAME

CBL-3-10

CLASSIFICATION

Input-Output

TITLE

The RECORD DELIMITER clause

CLAUSE

Section VII: 2.8 The RECORD DELIMITER Clause

GUIDANCE

The RECORD DELIMITER clause should not be specified.

EXPLANATION

Standard-1 defines external storage as magnetic tape unit only. Implementor-name varies with implementor.

NAME

CBL-4-1

CLASSIFICATION

Inter-Program Communication

TITLE

The CALL statement which calls program written in language other than COBOL

CLAUSE

Section X: 5.2 The CALL Statement

GUIDANCE

In order to call a program written in languages other than the COBOL language, all the CALL statements must be rewritten.

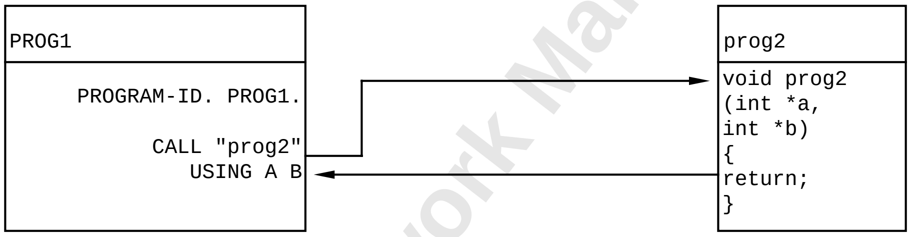
EXPLANATION

As the method of referencing names in programs written in languages other than COBOL varies with implementations, it may be necessary to rewrite the CALL statements.

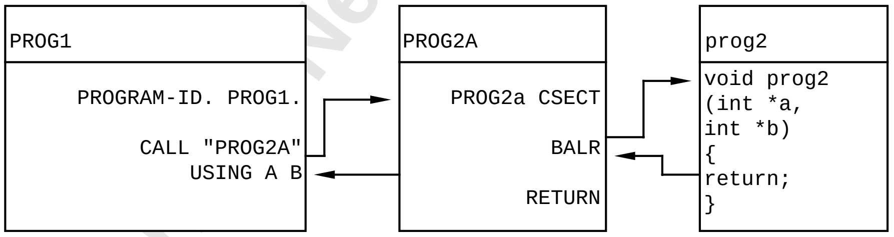
As the method of transferring parameters to the program written in languages other than COBOL varies with implementations, it may be required to rewrite the CALL statements.

EXAMPLE

Implementation-1:



Implementation-2:



NAME

CBL-4-2

CLASSIFICATION

Inter-Program Communication

TITLE

Literal in CALL statement

CLAUSE

Section X: 5.2.4 General Rules

GUIDANCE

The literal in the CALL statement should be written in upper-case.

EXPLANATION

If lower-case letters are used for the program-name in PROGRAM-ID paragraph, the equivalence rule applies. However, as the equivalence rule for the case where lower-case letters are used for the literal in a CALL statement is not defined, its interpretation varies with implementations. As the expected program may not be called, the application program which uses lower-case letters for the literal in a CALL statement is not portable.

EXAMPLE

Application Program Without Portability

```
CALL "abc"
```

The program called varies with implementations as follows:

1. Implementations where the equivalence rule is applied.

   Program "ABC" will be called.

2. Implementations where the equivalence rule is not applied.

   Program "ABC" will not be called.

Application Program With Portability

```
CALL "ABC"
```

Program "ABC" will always be called.

NAME

CBL-5-1

CLASSIFICATION

National Language

TITLE

User-defined words of national-character

CLAUSE

Section IV: 4.2.2.1.1 User-defined Words

GUIDANCE

Do not use the user-defined words which consist only of Roman letter, digit and hyphen of national-character.

EXPLANATION

The equivalence rule for the characters used in the user-defined words and their corresponding national-characters (that is, A and , 8 and) is not defined. There may be an implementation which handles the user-defined words which consist only of national-character romaji, numeric characters and/or hyphens in the same way as it does for those user-defined words after conversion to English-characters. Therefore, when national-characters are used in a user-defined word, at least one character which is not included in English characters such as a Kanji national-character should be included in the user-defined word.

EXAMPLE

```
(1)  01  ABC
(2)  01  abc
(3)  01  A BC
(4)  01  a b c
```

(1) and (3), (2) and (4), and (1) through (4) above may be assumed as the same user-defined words by some implementations.

NAME

CBL-5-2

CLASSIFICATION

National Language

TITLE

How to write a source program which includes national-characters

CLAUSE

Section IV: 7.2 Reference Format Representation

GUIDANCE

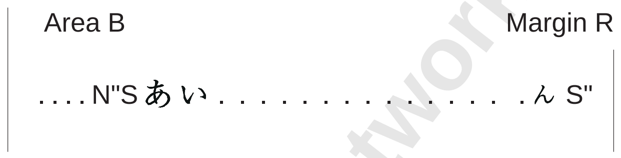
In a source program which includes national-characters (national-character literals and/or national-character user-defined words), the lines which include national-characters should end around ⅔ of Area B.

EXPLANATION

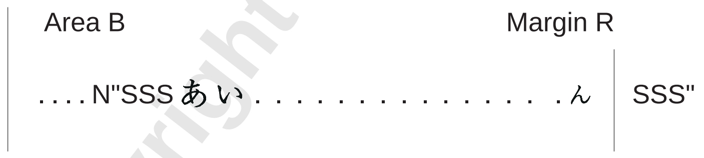
National-characters are represented in various ways in source programs. They are generally distinguished from the characters by shift codes. The shift code is not fixed and is defined by each implementation. Therefore, a line of a source program which uses all the Area B in one implementation may cause an overflow in the line in the Area B when the source program is moved to another implementation, depending on the size of the shift code. So, in a source program which includes national-characters (national-character literals and/or national-character user-defined words), the lines which include national-characters should end around two thirds of the Area B.

EXAMPLE

If the size of the shift code is one byte in Implementation-A and three bytes in Implementation-B, and a line of the source program in the Implementation-A is as follows:



then, the line will exceed the Margin R shown below, when this line is moved to the Implementation B.



‘‘S’’ is a shift code in the figure above.

NAME

CBL-5-3

CLASSIFICATION

National Language

TITLE

USAGE clause for national-character data items

CLAUSE

Section VI: 5.14 The USAGE Clause

GUIDANCE

The USAGE clause should not be specified for national-character data items.

EXPLANATION

Usages for national-character data items are not defined. Therefore, the usage for national-character data item is defined by each implementation. On the other hand, a USAGE clause for national-character data item may be omitted. In this case, the usages for national-character data items are to be defined by the implementation. In consideration of the portability of definitions of national-character data items, the USAGE clause should not be specified for national-character data items.

EXAMPLE

	Definition of Implementation DISPLAY	**Definition of Implementation DISPLAY-1**
USAGE definition omitted	Portable	Portable
USAGE definition DISPLAY	Portable	Unportable
USAGE definition DISPLAY-1	Unportable	Portable

NAME

CBL-5-4

CLASSIFICATION

National Language

TITLE

National operand in the REPLACING phrase of the COPY statement

CLAUSE

Section XII: 2 The COPY Statement

GUIDANCE

National-character should not be specified in the operand of the REPLACING phrase of the COPY statement.

EXPLANATION

It is not defined whether national-characters can be specified as the operand of the REPLACING phrase in the COPY statement. Therefore, national-characters should not be specified in the operand of the REPLACING phrase in the COPY statement.

EXAMPLE

Application Program Without Portability

```
COPY ABC REPLACING == 正 ==誤 ==.
COPY ABC REPLACING 正1 BY 誤1.
COPY ABC REPLACING RIGHT1 BY 誤 .
COPY ABC REPLACING "正1" BY "誤1".
```

NAME

CBL-5-5

CLASSIFICATION

National Language

TITLE

Collating sequence of national-character

CLAUSE

All Sections

GUIDANCE

The relation of national-characters should be determined by the equal or not-equal condition. Do not specify the range for national-characters (THRU paragraph of VALUE clause, and so on).

EXPLANATION

The collating sequence of national-characters is not defined. It depends on the code values of the implementation.

EXAMPLE

Application Program Without Portability

```
relation condition N"鵜" IS > N"絵"
88 condition-name 1 VALUE IS N"あ" THRU N"お".
```

Application Program With Portability

```
relation condition N"鵜" IS NOT = N"絵"
88 condition-name 1 VALUE IS N"あ" N"い" N"う" N"え" N"お".
```

NAME

CBL-6-1

CLASSIFICATION

Others

TITLE

Handling of reserved words

CLAUSE

Section I:	1.5.2.5.3	Reserved Words
Section IV:	8	COBOL Reserved Words

GUIDANCE

When user-defined words, except SPIRIT COBOL reserved words, are processed by the implementation as reserved words, they should be changed to other words. Internal names should be written in national-character.

EXPLANATION

The reserved words listed in the Reserved Words list are SPIRIT COBOL reserved words. The implementor is allowed to add words to the list. Therefore, when the user-defined words which are not listed on the Reserved Words list are to be processed by the implementation as the reserved words, they must be changed to other words.

National-characters cannot be used in reserved words, so use the internal names for which national-characters can be used as national-character user-defined words in order to avoid duplicate use with the reserved words.

EXAMPLE

Following are samples of the words which can be reserved words depending on the implementation:

ARITHMETIC	**DB-ACCESS-CONTROL-KEY**	**FUNCTION**
B-AND	**DB-DATA-NAME**	**GET**
B-EXOR	**DB-EXCEPTION**	**KEEP**
B-LESS	**DB-RECORD-NAME**	**LD**
B-NOT	**DB-SET-NAME**	**LOCALLY**
B-OR	**DB-STATUS**	**MODIFY**
BIT	**DEFAULT**	**NONE**
BITS	**DISCONNECT**	**NULL**
BOOLEAN	**DOLLAR**	**ONLY**
COMMIT	**EQUALS**	**OWNER**
COMP-1	**ERASE**	**PRESENT**
COMP-2	**EXCEEDS**	**PRIOR**
COMP-3	**EXCLUSIVE**	**PROTECTED**
COMP-4	**EXOR**	**READY**
COMP-5	**FETCH**	**REALM**
COMPUTATIONAL-1	**FILES**	**RECONNECT**
CONNECT	**FIND**	**RECORD-NAME**
CONTAINED	**FINISH**	**RELATION**

CURRENT	**FORMAT**	**REPEATED**
DB	**FREE**	**RETAINING**
COMPUTATIONAL-2	**COMPUTATIONAL-3**	**COMPUTATIONAL-4**
COMPUTATIONAL-5		

NAME

CBL-6-2

CLASSIFICATION

Others

TITLE

Definition of implementor-name-1 in the SPECIAL-NAMES paragraph

CLAUSE

Section VI: 4.5 The SPECIAL-NAMES Paragraph

GUIDANCE

Implementor-name-1 must be rewritten.

EXPLANATION

What can be specified in the implementor-name-1 depends on the implementation, and the hardware device which corresponds to the implementor-name-1 also depends on the implementation. If the FROM phrase is omitted from the ACCEPT statement, the hardware device relevant to the implementor-name-1 varies with the implementation. Therefore, the FROM phrase should be specified in the ACCEPT statement. In order to do so, the implementor-name-1 which is to be referenced by the mnemonic-name of the FROM phrase must be specified in the SPECIAL-NAMES paragraph when the ACCEPT statement is used.

If the UPON phrase is omitted from the DISPLAY statement, the hardware device relevant to the implementor-name-1 varies with the implementation. Therefore, the UPON phrase should be specified in the DISPLAY statement. In order to do so, the implementor-name-1 which is to be referenced by the mnemonic-name of the UPON phrase must be specified in the SPECIAL-NAMES paragraph when the DISPLAY statement is used.

EXAMPLE

```
SPECIAL-NAMES .

     SYSIN  IS  standard-input-device

     SYSOUT  IS  standard-output-device

     ACCEPT  identifier-1  FROM  standard-input-device

     DISPLAY  identifier-2  UPON  standard-output-device
```

NAME

CBL-6-3

CLASSIFICATION

Others

TITLE

Implementor-name-2 of the ALPHABET clause

CLAUSE

Section VI: 4.5 The SPECIAL-NAMES Paragraph

GUIDANCE

Do not use the implementor-name-2 in the ALPHABET clause.

EXPLANATION

The implementor-name-2 in the ALPHABET clause specifies the character code set and the collating sequence defined by the implementation. As the implementor-name-2 varies with the implementation, the application program which uses the character code set and the collating sequence specified by the implementor-name-2 is not portable.

EXAMPLE

Application Program Without Portability

```
SPECIAL-NAMES .
      ALPHABET  alphabet-name-1  IS  EBCDIC
      ALPHABET  alphabet-name-2  IS  SHIFT-JIS
```

NAME

CBL-6-4

CLASSIFICATION

Others

TITLE

Hardware device associated with the ACCEPT statement without the FROM phrase

CLAUSE

Section VI: 6.5 The ACCEPT Statement

GUIDANCE

In Format 1 of the ACCEPT statement, the FROM phrase must be specified.

EXPLANATION

As the implementor-name-1 which identifies the standard input device varies with the implementation, specify the FROM phrase in the ACCEPT statement of Format 1, and rewrite the implementor-name-1 in the SPECIAL-NAMES paragraph at application program porting time.

EXAMPLE

```
SPECIAL-NAMES .
     SYSIN  IS  standard-input-device
            :
     ACCEPT  identifier-1  FROM  standard-input-device  .
```

Note that the underlined word above is to be rewritten.

NAME

CBL-6-5

CLASSIFICATION

Others

TITLE

The ACCEPT statement (data conversion with hardware device)

CLAUSE

Section VI: 6.5 The ACCEPT Statement

GUIDANCE

The class of the identifier-1 in the ACCEPT statement should be ‘‘alphanumeric’’.

EXPLANATION

As the data conversion between the hardware device and the data item of identifier-1 varies with the implementation, the application program in which the identifier-1 of the ACCEPT statement is a data item which requires data conversion is not portable.

EXAMPLE

Where the identifier-1 specifies a numeric item which is described with the USAGE IS BINARY clause, the corresponding numeric value will be obtained for the input numeric string with the implementation which converts decimal data into binary. However, with the implementation which does not convert decimal into binary, this results in a data error.

If a signed numeric item is specified in the identifier-1, it also results in a data error with the implementation which does not convert the representation of signs.

NAME

CBL-6-6

CLASSIFICATION

Others

TITLE

Definition of zero number of characters in the ACCEPT statement

CLAUSE

Section VI: 6.5 The ACCEPT Statement

GUIDANCE

In Format 1 of the ACCEPT statement, data of one or more characters should be requested to be entered by displaying a message for expediting input.

EXPLANATION

The behaviour for the case when only the ENTER key is pressed varies with the implementation. Therefore, the application program which uses ACCEPT statements of zero number of characters is not portable.

NAME

CBL-6-7

CLASSIFICATION

Others

TITLE

Mnemonic-name of the DISPLAY statement

CLAUSE

Section VI: 6.10 The DISPLAY Statement

GUIDANCE

The UPON phrase must be specified in the DISPLAY Statement.

EXPLANATION

As the implementor-name-1 which references the standard input device varies with the implementation, specify the UPON phrase in the DISPLAY statement, and rewrite the implementor-name-1 in the SPECIAL-NAMES paragraph at application program porting time.

EXAMPLE

The following is an example:

```
SPECIAL-NAMES  .

          SYSOUT  IS  standard-output-device

                    :

          DISPLAY  identifier-1  UPON  standard-output-device  .
```

Note that the underlined word above is to be rewritten.

NAME

CBL-6-8

CLASSIFICATION

Others

TITLE

The DISPLAY statement (data conversion with hardware device)

CLAUSE

Section VI: 6.10 The DISPLAY Statement

GUIDANCE

The class of the identifier-1 in the DISPLAY statement should be alphanumeric.

EXPLANATION

As the data conversion between the hardware device and the data item of identifier-1 varies with the implementation, the application program in which the identifier-1 of the DISPLAY statement is a data item which requires data conversion is not portable.

EXAMPLE

Where the identifier-1 specifies a numeric item which is described with the USAGE IS BINARY clause, the corresponding numeric value will be obtained for the output numeric string with the implementation which converts binary data into decimal. However, with the implementation which does not convert binary into decimal, this results in output of error characters.

If a signed numeric item is specified in the identifier-1, it also results in output of error characters with the implementation which does not convert the representation of signs.