L1MOS

COBOL Language Reference Manual

olivetti

UPDATING STATUS

LEVEL	DATE	UPDATED PAGES	PAGES	CODE
4	June 86		536	4004290 H (4)
5	Feb. 87	Pref., vii÷xviii, 2-3÷2-6, 2-40, 2-46÷2-52, 3-11÷3-18, SELECT..ASSIGN-1-2÷SAME AREA-1, 4-2, VALUE OF-2÷4-29, PICTURE-2, 5-7÷5-13, ACCEPT-2-3, ADD-1, EVALUATE-5, OPEN-4÷ PERFORM-10, FULL-1, JUSTIFIED-1, LINE-1, SIGN-1, ACCEPT-3÷DISPLAY-2, EDIT-1-4-5, A-1÷A-3, B-1, B-3, B-6, B-7, B-13, B-19, C-1÷C-6, I-0÷I-17	120	4004295 B

*Pages marked * must be suppressed*

L1 MOS

COBOL Language Reference Manual

olivetti

Copyright © 1987, by Olivetti
All rights reserved

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 IVREA (Italy)

PREFACE

This manual provides a definition of the COBOL language and syntax as implemented on the L1 MOS system. A knowledge of programming in COBOL is required for a complete understanding of this manual.

SUMMARY

The manual is divided into chapters describing level of implementation, the language syntax, the language concepts and program structure. The four divisions of a program are described in separate chapters. The Olivetti extension SCREEN SECTION and the optional modules of the language are described in separate chapters. The indices contain a COBOL reserved word list, a glossary, a table of I/O status values and the analytical index.

REFERENCES

To read first ...

Introduction to MOS - Code 4002130 G (Vol. 2)

Programming Languages Features and Functions - Code 4004210 Z (Vol. 2)

For further information read ...

COBOL Program Preparation and Execution - Code 4004310 T (Vol. 6D)

MOS Programmer Guide - Code 4002570 L (Vol. 6G)

Glossary/Glossario - Code 4002140 H (Vol. 1)

FIRST EDITION: March 1984 - Release 3.0

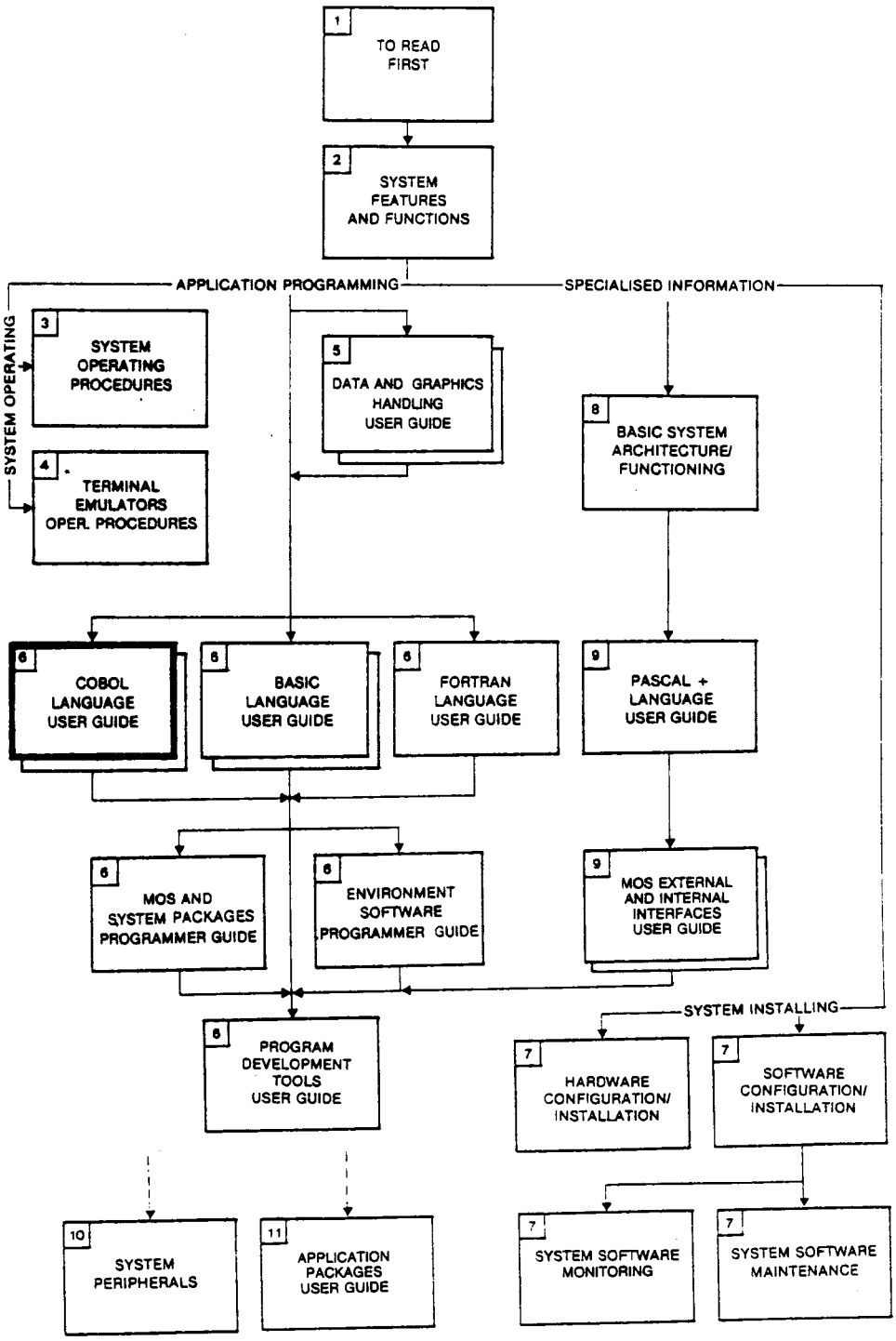
SECOND EDITION: October 1984 - Release 4.0

UPDATES: November 1984 - Release 4.0
May 1985 - Release 5.0
November 1985 - Release 5.1

REPRINTED: June 1986

(Including updates from 2 to 4)

UPDATE: February 1987 - Release 5.2


1
TO READ FIRST
2
SYSTEM FEATURES AND FUNCTIONS
APPLICATION PROGRAMMING
SPECIALISED INFORMATION
SYSTEM OPERATING
3
SYSTEM OPERATING PROCEDURES
4
TERMINAL EMULATORS OPER. PROCEDURES
5
DATA AND GRAPHICS HANDLING USER GUIDE
8
BASIC SYSTEM ARCHITECTURE/ FUNCTIONING
6
COBOL LANGUAGE USER GUIDE
6
BASIC LANGUAGE USER GUIDE
6
FORTRAN LANGUAGE USER GUIDE
9
PASCAL + LANGUAGE USER GUIDE
6
MOS AND SYSTEM PACKAGES PROGRAMMER GUIDE
6
ENVIRONMENT SOFTWARE PROGRAMMER GUIDE
9
MOS EXTERNAL AND INTERNAL INTERFACES USER GUIDE
SYSTEM INSTALLING
6
PROGRAM DEVELOPMENT TOOLS USER GUIDE
7
HARDWARE CONFIGURATION/ INSTALLATION
7
SOFTWARE CONFIGURATION/ INSTALLATION
10
SYSTEM PERIPHERALS
11
APPLICATION PACKAGES USER GUIDE
7
SYSTEM SOFTWARE MONITORING
7
SYSTEM SOFTWARE MAINTENANCE

CONTENTS

PAGE	
1-1	1. INTRODUCTION
1-1	CONTENT OF THIS RELEASE OF THE COBOL COMPILER
1-2	STRUCTURE OF COBOL LANGUAGE REFERENCE MANUAL
1-3	COBOL LANGUAGE LEVEL OF IMPLEMENTATION
1-5	NOTATION USED IN FORMATS
1-5	DEFINITION OF A GENERAL FORMAT
1-5	ELEMENTS
1-5	WORDS
1-5	LEVEL-NUMBERS
1-6	BRACKETS, BRACES, and CHOICE INDICATORS
1-6	THE ELLIPSIS
1-8	RULES
1-8	SYNTAX RULES
1-8	GENERAL RULES
2-1	2. OVERALL LANGUAGE CONSIDERATION
2-1	LANGUAGE ELEMENTS AND CONCEPTS
2-1	CHARACTER SET
2-1	LANGUAGE STRUCTURE
2-1	SEPARATORS
2-2	CHARACTER-STRINGS
2-3	COBOL WORDS
2-3	USER-DEFINED WORDS
2-5	SYSTEM-NAMES

PAGE

2-5 RESERVED WORDS

2-7 LITERALS

2-9 FIGURATIVE CONSTANT VALUES

2-11 PICTURE CHARACTER-STRINGS

2-11 COMMENT-ENTRIES

2-12 CONCEPT OF COMPUTER INDEPENDENT DATA DESCRIPTION

2-12 LOGICAL RECORD CONCEPT

2-13 CONCEPT OF LEVELS

2-14 UNIQUENESS OF REFERENCE

2-14 QUALIFICATION

2-17 SUBSCRIPTING

2-18 IDENTIFIER

2-19 CONDITION-NAME

2-19 TABLE HANDLING

2-24 FILES

2-24 FILE ATTRIBUTES

2-24 SEQUENTIAL ORGANIZATION

2-25 RELATIVE ORGANIZATION

2-25 INDEXED ORGANIZATION

2-26 FILE PROCESSING

2-26 RECORD OPERATIONS ACCESS MODE

2-29 SHARED MEMORY AREA

2-30 EXCEPTION HANDLING

2-30 EXCEPTION DECLARATIVES

2-30 THE INVALID KEY CONDITION

2-31 THE AT END CONDITION

2-31 THE FILE ATTRIBUTE CONFLICT CONDITION

PAGE

2-32 I-O STATUS

2-33 LIBRARY FACILITY

COPY

2-38 A COBOL SOURCE PROGRAM

2-38 ORGANIZATION

2-38 STRUCTURE

2-39 DIVISIONS

2-39 IDENTIFICATION DIVISION

2-40 ENVIRONMENT DIVISION

2-42 DATA DIVISION

2-42 PHYSICAL AND LOGICAL ASPECTS OF DATA DESCRIPTION

2-44 PROCEDURE DIVISION

2-45 PROCEDURE DIVISION STRUCTURE

2-46 STATEMENTS AND SENTENCES

2-50 EXPLICIT AND IMPLICIT SPECIFICATIONS

2-52 EXTERNAL SWITCH

2-53 REFERENCE FORMAT

2-53 GENERAL DESCRIPTION

2-53 REFERENCE FORMAT REPRESENTATION

2-55 DIVISION, SECTION, PARAGRAPH FORMATS

2-56 DATA DIVISION ENTRIES

2-56 DECLARATIVES

2-57 END PROGRAM HEADER

2-57 EXAMPLE PROGRAM

3-1 3. IDENTIFICATION AND ENVIRONMENT DIVISIONS

3-1 IDENTIFICATION DIVISION

3-1 GENERAL DESCRIPTION

PAGE

3-3 THE PROGRAM-ID PARAGRAPH

3-4 THE DATE-COMPILED PARAGRAPH

3-5 ENVIRONMENT DIVISION

3-5 GENERAL DESCRIPTION

3-6 CONFIGURATION SECTION

3-7 THE SOURCE-COMPUTER PARAGRAPH

3-8 THE OBJECT-COMPUTER PARAGRAPH

3-10 THE SPECIAL-NAMES PARAGRAPH

3-15 INPUT-OUTPUT SECTION

3-15 THE FILE-CONTROL PARAGRAPH

3-16 THE FILE CONTROL ENTRY

SELECT .. ASSIGN clause

ACCESS MODE clause

ALTERNATE RECORD KEY clause

FILE STATUS clause

ORGANIZATION clause

RECORD KEY clause

RESERVE clause

3-20 THE I-O-CONTROL PARAGRAPH

SAME AREA clause

4-1 4. DATA DIVISION

4-1 GENERAL DESCRIPTION

4-2 DATA DESCRIPTION

4-2 NONCONTIGUOUS WORKING STORAGE

4-2 WORKING STORAGE RECORDS

4-3 CONCEPT OF LEVELS

4-3 LEVEL-NUMBERS

PAGE

4-4 CONCEPT OF CLASSES OF DATA

4-4 SELECTION OF CHARACTER REPRESENTATION AND RADIX

4-5 ALGEBRAIC SIGNS

4-5 STANDARD ALIGNMENT RULES

4-6 ITEM ALIGNMENT FOR INCREASED OBJECT-CODE EFFICIENCY

4-7 FILE SECTION

4-8 FILE DESCRIPTION ENTRY

4-8 RECORD DESCRIPTION STRUCTURE

4-9 THE FILE DESCRIPTION ENTRY

BLOCK CONTAINS clause

CODE-SET clause

DATA RECORDS clause

LABEL RECORDS clause

LINAGE clause

RECORD CONTAINS clause

VALUE OF clause

4-27 WORKING-STORAGE SECTION

4-28 LINKAGE SECTION

4-29 THE SCREEN SECTION

4-29 THE REPORT SECTION

4-30 THE DATA DESCRIPTION ENTRY

BLANK WHEN ZERO clause

DATA-NAME or FILLER clause

JUSTIFIED clause

LEVEL-NUMBER description

OCCURS clause

PICTURE clause

PAGE

REDEFINES clause

RENAMES clause

SIGN clause

SYNCHRONIZED clause

USAGE clause

VALUE clause

5-1 5. PROCEDURE DIVISION

5-1 GENERAL DESCRIPTION

5-3 ARITHMETIC EXPRESSIONS

5-3 ARITHMETIC OPERATORS

5-3 FORMATION AND EVALUATION RULES

5-5 CONDITIONAL EXPRESSIONS

5-5 SIMPLE CONDITIONS

5-5 RELATION CONDITION

5-6 COMPARISON OF NUMERIC OPERANDS

5-6 COMPARISON OF NONNUMERIC OPERANDS

5-7 COMPARISONS INVOLVING INDEX-NAMES and/or INDEX DATA ITEMS

5-7 CLASS CONDITION

5-9 CONDITION-NAME CONDITION (Conditional Variable)

5-9 SWITCH-STATUS CONDITION

5-9 SIGN CONDITION

5-10 COMPLEX CONDITIONS

5-10 NEGATED CONDITIONS

5-10 COMBINED CONDITIONS

5-12 ABBREVIATED COMBINED RELATION CONDITIONS

5-13 ORDER OF EVALUATION OF CONDITIONS

5-14 COMMON OPTIONS AND RULES FOR STATEMENT FORMATS

PAGE

5-14 THE ROUNDED PHRASE

5-14 THE ON SIZE ERROR PHRASE

5-15 THE CORRESPONDING PHRASE

5-16 THE ARITHMETIC STATEMENTS

5-16 OVERLAPPING OPERANDS

5-17 MULTIPLE RESULTS IN ARITHMETIC STATEMENTS

5-17 INCOMPATIBLE DATA

5-18 PROCEDURE DIVISION STATEMENTS

ACCEPT

ADD

CLOSE

COMPUTE

DELETE

DELETE FILE

DISPLAY

DIVIDE

EVALUATE

EXIT

GO TO

IF

INSPECT

LOCK

MOVE

MULTIPLY

OPEN

PERFORM

READ

PAGE

REWRITE

SEARCH

SET

START

STOP

STRING

SUBTRACT

UNLOCK

UNSTRING

USE

WRITE

6-1 6. SCREEN HANDLING

6-1 SCREEN HANDLING IN THE DATA DIVISION

6-1 SCREEN SECTION

6-1 SCREEN RECORD DESCRIPTION STRUCTURE

6-2 THE SCREEN DESCRIPTION ENTRY

AUTO clause

BELL clause

BLANK clause

BLANK WHEN ZERO clause

BLINK clause

BOX clause

COLUMN clause

FULL clause

HEBREW clause

HIGHLIGHT clause

JUSTIFIED clause

PAGE

LEVEL NUMBER description

LINE clause

PICTURE clause

REQUIRED clause

REVERSE clause

SCREEN-NAME OR FILLER clause

SECURE clause

SIGN clause

UNDERLINE clause

VALUE clause

6-7 SCREEN HANDLING IN THE PROCEDURE DIVISION

6-7 INTRODUCTION

ACCEPT

DISPLAY

EDIT

7-1 7. INTER-PROGRAM COMMUNICATION

7-1 LANGUAGE CONCEPTS

7-1 PROGRAM AND RUN UNIT ORGANIZATION

7-2 GLOBAL NAMES AND LOCAL NAMES

7-3 EXTERNAL OBJECTS AND INTERNAL OBJECTS

7-3 EXAMPLE

7-5 COMMON PROGRAMS AND INITIAL PROGRAMS

7-5 SHARING DATA

7-5 SHARING FILES

7-6 SCOPE OF NAMES

7-8 LINK UNITS AND RUN UNITS

7-9 SPECIAL REGISTER MESSAGE

PAGE

7-10 TRANSFER OF CONTROL

7-11 PASSING PARAMETERS TO PROGRAMS

7-13 NESTED SOURCE PROGRAMS

7-13 ORGANIZATION

7-13 STRUCTURE

7-15 INITIAL STATE OF A PROGRAM

7-16 END PROGRAM HEADER

7-17 INTER-PROGRAM COMMUNICATION IN THE IDENTIFICATION DIVISION

7-17 THE PROGRAM-ID PARAGRAPH AND NESTED SOURCE PROGRAMS

7-18 INTER-PROGRAM COMMUNICATION IN THE DATA DIVISION

7-18 LINKAGE SECTION

7-20 THE FILE DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

7-26 THE DATA DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

7-29 THE SCREEN DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

7-34 THE REPORT DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

EXTERNAL clause

GLOBAL clause

7-35 INTER-PROGRAM COMMUNICATION IN THE PROCEDURE DIVISION

7-36 THE PROCEDURE DIVISION HEADER

CALL

CANCEL

EXIT PROGRAM

USE

8-1 8. SEGMENTATION

8-1 SCOPE

8-1 ORGANIZATION

8-2 SEGMENT CLASSIFICATION

PAGE

8-2 SEGMENTATION CONTROL

8-3 SEGMENTATION IN THE ENVIRONMENT DIVISION

8-3 CONFIGURATION SECTION

8-3 THE OBJECT-COMPUTER PARAGRAPH

8-4 SEGMENT-LIMIT clause

8-5 SEGMENTATION IN THE PROCEDURE DIVISION

8-5 GENERAL DESCRIPTION

8-6 SEGMENT-NUMBERS

9-1 9. SORT-MERGE

9-1 LANGUAGE CONCEPTS

9-1 SORT FILE

9-1 MERGE FILE

9-2 RELATIONSHIP WITH SEQUENTIAL I-O MODULE

9-3 SORT-MERGE IN THE ENVIRONMENT DIVISION

9-3 INPUT-OUTPUT SECTION

9-3 THE FILE-CONTROL PARAGRAPH

9-3 FILE-CONTROL ENTRY

9-5 THE I-O-CONTROL PARAGRAPH

SAME AREA clause

9-7 SORT-MERGE IN THE DATA DIVISION

9-7 FILE SECTION

9-8 THE SORT-MERGE FILE DESCRIPTION ENTRY

9-9 SORT-MERGE IN THE PROCEDURE DIVISION

MERGE

RELEASE

RETURN

SORT

PAGE

10-1 10. DEBUG

10-1 LANGUAGE CONCEPTS

10-1 DEBUG FEATURES

10-1 SPECIAL REGISTER DEBUG-ITEM

10-2 COMPILE TIME SWITCH

10-2 AN OBJECT TIME SWITCH

10-2 DEBUG IN THE ENVIRONMENT DIVISION

WITH DEBUGGING MODE clause

10-4 DEBUG IN THE PROCEDURE DIVISION

10-4 GENERAL DESCRIPTION

USE FOR DEBUGGING

10-12 DEBUGGING LINES

11-1 11. THE REPORT WRITER

11-1 LANGUAGE CONCEPTS

11-2 REPORT STRUCTURE

11-3 REPORT FILE

11-3 SPECIAL REGISTER PAGE-COUNTER

11-4 SPECIAL REGISTER LINE-COUNTER

11-4 SUBSCRIPTING

11-5 REPORT WRITER IN THE ENVIRONMENT DIVISION

11-5 INPUT-OUTPUT SECTION

11-5 THE FILE-CONTROL PARAGRAPH

11-5 THE FILE CONTROL ENTRY

11-8 THE I-O-CONTROL PARAGRAPH

11-9 REPORT WRITER IN THE DATA DIVISION

11-9 FILE SECTION

11-10 THE FILE DESCRIPTION ENTRY

PAGE

REPORT clause

11-13 REPORT SECTION

11-14 THE REPORT DESCRIPTION ENTRY

CODE clause

CONTROL clause

PAGE clause

11-23 THE REPORT GROUP DESCRIPTION ENTRY

11-28 PRESENTATION RULES TABLES

11-31 REPORT HEADING GROUP PRESENTATION RULES

11-34 PAGE HEADING GROUP PRESENTATION RULES

11-36 BODY GROUP PRESENTATION RULES

11-41 PAGE FOOTING PRESENTATION RULES

11-43 REPORT FOOTING PRESENTATION RULES

COLUMN NUMBER clause

DATA-NAME clause

GROUP INDICATE clause

LEVEL-NUMBER description

LINE NUMBER clause

NEXT GROUP clause

SIGN clause

SOURCE clause

SUM clause

TYPE clause

USAGE clause

VALUE clause

11-65 REPORT WRITER IN THE PROCEDURE DIVISION

11-65 GENERAL DESCRIPTION

PAGE

CLOSE

GENERATE

INITIATE

OPEN

SUPPRESS

TERMINATE

USE BEFORE REPORTING

A-1 A. COBOL RESERVED WORDS

A-4 SYSTEM NAMES

A-4 COMPUTER-NAMES

A-4 ENVIRONMENT DIVISION IMPLEMENTOR-NAMES

A-4 DATA DIVISION IMPLEMENTOR-NAMES

A-4 PROCEDURE DIVISION IMPLEMENTOR-NAMES

B-1 B. GLOSSARY

B-1 DEFINITIONS

C-1 C. I-O STATUS VALUES

C-1 THE I-O STATUS CONDITIONS

C-1 I-O STATUS VALUES

I-1 I. INDEX

ACKNOWLEDGEMENT

This publication is based on the American National Standards (ANSI) publication X3.23-1974, American National Standard Programming Language COBOL, and refers also to the Draft Proposed Revision of X3.23-1974.

The following paragraphs are reproduced from the former document as an acknowledgement of the source.

"COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

The authors and copyright holders of the copyrighted material used herein FLOW-MATIC (trademark of Sperry Rand Corporation) Programming for the UNIVAC 1 and 11, Data Automation Systems copyrighted 1958, 1959 by Sperry Rand Corporation; IBM Commercial Translator Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by MINNEAPOLIS-Honeywell have specifically authorized the use of this material, in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications."

1. INTRODUCTION

COBOL is a high-level computer language. Such languages are relatively machine independent, freeing the programmer from many of the restrictions of assembler language and allowing him to transport his business problem solution from one computer to another.

The COBOL language processor is composed of three elements:

. The source language in which programs are written;

. The COBOL compiler: a program that accepts COBOL source programs as input and produces object modules as output; and

. The generated object module which, with its supporting execution-time subsystems, performs the application task.

This document defines the COBOL source language, beginning with concepts which are generally true for all COBOL statements and proceeding to descriptions of all available COBOL verbs.

CONTENT OF THIS RELEASE OF THE COBOL COMPILER

This release of the COBOL compiler comprises all the language elements described in this reference manual except for the following:

- The External Switch facility

- Support for MTU devices.

STRUCTURE OF COBOL LANGUAGE REFERENCE MANUAL

The organization of the COBOL reference manual is in chapters according to the program structure and the language specification concept. The manual comprises eleven chapters :
Introduction, language Specification, Identification and Environment Division, Data Division, Procedure Division, Screen Handling, Sort-Merge, Debug, Inter-Program Communication, Report Writer, and Segmentation.

The Language Specification chapter describes the language concepts, and the syntax used in writing a program. A desription of the structure of a program showing the four divisions and an example program are presented.

The Identification and Environment chapter describes the language elements for the definition of the program identity and the machine dependent aspects.

The Data Division chapter describes the language elements for the definition of the data used in the program.

The Procedure Division chapter describes the language elements for the definition of the data statements for the processing of data.

The Screen Handling chapter describes the language elements for the formatting and the handling of data representing screen images, this feature is an Olivetti extension to the ANSI 74 standard.

The Inter-Program Communication chapter describes the language elements which enable a program to communicate with one or more other programs. It provides for the access of certain common data items by both programs and for the nesting of programs within other programs.

The Segmentation chapter describes the overlaying at object time of Procedure Division sections.

The Sort-Merge chapter contains the language elements for the ordering of one or more files, or for the combining of two or more identically ordered files. Optionally, input or output procedures may be specified to process each of the records.

The Debug chapter describes a means by which the user can specify his debugging algorithm and describes the conditions under which data or procedure items are monitored during execution of the program.

The Report Writer chapter describes the language elements for the semi-automatic production of printed reports.

The appendices contain the COBOL reserved words list, the I/O Status Value list and a COBOL glossary.

COBOL LANGUAGE LEVEL OF IMPLEMENTATION

The ANSI COBOL standard divides COBOL into processing modules with levels of implementation. The following summary specifies the content of the modules supported by COBOL with respect to the standard.

MODULE	LEVEL	STATEMENT
NUCLEUS	- Omission	The ALTER statement.
	- ANSI '74	All statements to level 2 where applicable.
	- Revised '83	The COMMON and INITIAL PROGRAM-ID options. The EVALUATE statement. The INSPECT statement with the CONVERTING clause. The PERFORM statement with an imperative statement in line. The SET statement with mneumonic or condition name. The END PROGRAM switch.
	- Extensions	The data types COMP-1, COMP-2, COMP-3, COMP-4 and BINARY in the USAGE clause.
TABLE HANDLING	- ANSI '74	All statements to level 2 where applicable.
	- Revised '83	The mixing of Subscripts and Indexes is supported.
SEQUENTIAL I/O	- ANSI '74	All statements to level 2 where applicable. except for the SAME AREA, RESERVE, TAPE, REEL, UNIT, MULTIPLE FILE, and FOR REMOVAL clauses.
	- Extensions	The OPEN statement with LOCK clause. The DELETE FILE statement.
RELATIVE I/O	- ANSI '74	All statements to level 2 where applicable except the SAME AREA and RESERVE clauses.
	- Revised '83	The USE and OPEN statements with ON EXTEND clause.
	- Extensions	The READ and WRITE statements with the WITH LOCK clause. The READ PREVIOUS statement. The LOCK and UNLOCK statements with the AT KEY clause. The DELETE FILE statement.

INDEXED I/O	- ANSI '74	All other statements to level 2 where applicable except for the SAME AREA and RESERVE clauses.
	- Revised '83	The USE and OPEN statements with ON EXTEND clause.
	- Extensions	The READ and WRITE statements with the WITH LOCK clause. The READ PREVIOUS statement. The LOCK and UNLOCK statements with the PRIMARY or ALTERNATE clause. The DELETE FILE statement.
LIBRARY FACILITY	- ANSI '74	All statements to level 2 where applicable.
INTER PROGRAM COMMUNI-CATION	- ANSI '74	All statements to level 2 where applicable.
	- Revised '83	Nested programs are supported. The GLOBAL and EXTERNAL clauses. The ON EXCEPTION clause.

SEGMENTATION is permitted but is treated as a comment.

The Olivetti extension to the ANSI '74 standard.

SCREEN SECTION	- Extensions	The LINE, COLUMN, AUTO, BELL, BLANK, BOX, BLINK, REQUIRED and SECURE clauses. The ACCEPT statement with the TIME-OUT clause. The DISPLAY Screen Name statement. The PICTURE clause with the FROM, TO options.

NOTATION USED IN FORMATS

DEFINITION OF A GENERAL FORMAT

A general format is the specific arrangement of the elements of a clause or a statement. A clause or a statement consists of elements as defined below. Throughout this document a format is shown adjacent to information defining the clause or statement. When more than one specific arrangement is permitted, the general format is separated into numbered formats. Clauses must be written in the sequence given in the general formats. (Clauses that are optional must appear in the sequence shown if they are used.) In certain cases, stated explicitly in the rules associated with a given format, the clauses may appear in sequences other than that shown. Applications, requirements, or restrictions are shown as rules.

ELEMENTS

Elements which make up a clause or a statement consist of uppercase words, lowercase words, level-numbers, brackets, braces, connectives, and special characters.

WORDS

All underlined uppercase words are called keywords and are required when the functions of which they are a part are used. Uppercase words which are not underlined, are optional to the user and need not be written in the source program. Uppercase words, whether underlined or not, must be spelled correctly.

Lowercase words, in a general format, are generic terms used to represent COBOL words, literals, PICTURE character-strings, comment-entries, or a complete syntactical entry that must be supplied by the user. Where generic terms are repeated in a general format, a number or letter appended to the term serves to identify that term for explanation or discussion.

LEVEL-NUMBERS

When specific level-numbers appear in data description entry formats, those specific level-numbers are required when such entries are used in a COBOL program. In this document, the form 01, 02, ... , 09 is used to indicate level-numbers 1 through 9.

BRACKETS, BRACES, and CHOICE INDICATORS

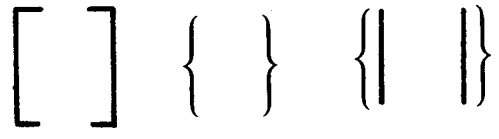
When brackets, [ ], enclose a portion of a general format, one of the options contained within the brackets may be explicitly specified or that portion of the general format may be omitted.

When braces, { }, enclose a portion of a general format, one of the options contained within the braces must be either explicitly specified or implicitly selected. If one of the options contains only reserved words which are not key words, that option is the default option and is implicitly selected unless one of the options is explicitly specified.

When choice indicators, {| |}, enclose a portion of a general format, one or more of the unique options contained within the choice indicators must be specified, but a single option may be specified only once.

Options are indicated in a general format or a portion of a general format by vertically stacking alternative possibilities, by a series of brackets, braces, or choice indicators or by a combination of both. An option is selected by specifying one of the possibilities from a stack of alternative possibilities or by specifying a unique combination of possibilities from a series of brackets, braces, or choice indicators.

In the format, brackets, braces and choice indicators can also be indicated by



THE ELLIPSIS

In text, other than general formats, the ellipsis (...) shows omission of a word or words when such omission does not impair comprehension. This is the conventional meaning of the ellipsis, and this use becomes apparent in context.

In the general format, the ellipsis (...) represents the position at which the user elects repetition of a portion of a format. The portion of the format that may be repeated is determined as follows:

Given the ... (ellipsis) in a format, scanning right to left, determine the] (right bracket) or } (right brace) delimiter immediately to the left of the ... (ellipsis); continue scanning right to left and determine the logically matching [(left bracket) or { (left brace) delimiter; the ... (ellipsis) applies to the portion of the format between the determined pair of delimiters.

Format Punctuation

The separators comma and semicolon may be used anywhere the separator space is used in the formats (see Separators). In the source program, these separators are interchangeable.

The separator period, when used in the formats, has the status of a required word.

Use of Special Character Words in Formats

The special character words '+', '-', '>', '<', '=', when appearing in formats, although not underlined, are required when such portions of the formats are used.

SYNTAX RULES

Syntax rules are those rules that define or clarify the order in which words or elements are arranged to form larger elements such as phrases, clauses, or statements. Syntax rules may also either impose restrictions on individual words or elements or relax restrictions implied by words or elements.

These rules are used to define or clarify how the statement must be written, i.e., the order of the elements of the statement and the restrictions or amplifications of what each element may represent.

GENERAL RULES

A general rule is a rule that defines or clarifies the meaning or relationship of meanings of an element or set of elements. It is used to define or clarify the semantics of the statement and the effect that it has on either execution or compilation.

2. OVERALL LANGUAGE CONSIDERATION

LANGUAGE ELEMENTS AND CONCEPTS

CHARACTER SET

The most basic and indivisible unit of the language is the character. The set of characters used to form COBOL character-strings and separators includes the letters of the alphabet, digits and special characters. The character set consists of the characters as defined under COBOL Character Set in the glossary. In the case of nonnumeric literals, comment-entries, and comment lines, the character set is expanded to include the computer's entire character set. The characters allowable in each type of character-string and as separators are defined below.

LANGUAGE STRUCTURE

The individual characters of the language are concatenated to form character-strings and separators. A separator may be concatenated with another separator or with a character-string. A character-string may only be concatenated with a separator. The concatenation of character-strings and separators forms the text of a source program.

SEPARATORS

A separator is a character or two contiguous characters formed according to the following rules:

1. The punctuation character space is a separator. Anywhere a space is used as a separator, or a part of a separator, more than one space may be used. All spaces immediately following the separators comma, semicolon, or period are considered part of that separator and are not considered to be the separator space.

2. Except when the comma is used in a PICTURE character-string, the punctuation characters comma and semicolon, immediately followed by a space, are separators that may be used anywhere the separator space is used. They may be used to improve program readability.

3. The punctuation character period, when followed by a space is a separator. It must be used only to indicate the end of the sentence, or as shown in formats.

4. The punctuation characters right and left parenthesis are separators. Parentheses may appear only in balanced pairs of left and right parentheses delimiting subscripts, indices, arithmetic expressions, or conditions.

5. The punctuation character quotation mark is a separator. An opening quotation mark must be immediately preceded by a space or left parenthesis; a closing quotation mark must be immediately followed by one of the separators space, comma, semicolon, period, or right parenthesis.
   Quotation marks may appear only in balanced pairs delimiting nonnumeric literals except when the literal is continued.

6. Pseudo-text delimiters are separators. An opening pseudo-text delimiter must be immediately preceded by a space; a closing pseudo-text delimiter must be immediately followed by one of the separators space, comma, semicolon, or period.

7. Pseudo-text delimiters may appear only in balanced pairs delimiting pseudo-text.

8. The separator space may optionally immediately precede all separators except:

   a) As specified by reference format rules (see The Reference Format.)

   b) The separator closing quotation mark. In this case, a preceding space is considered as part of the nonnumeric literal and not as a separator.

   c) The opening pseudo-text delimiter, where the preceding space is required.

9. The separator space may optionally immediately follow any separator except the opening quotation mark. In this case, a following space is considered as part of the nonnumeric literal and not as a separator.

Any punctuation character which appears as part of the specification of a PICTURE character-string or numeric literal is not considered as a punctuation character, but rather as a symbol used in the specification of that PICTURE character-string or numeric literal. PICTURE character-strings are delimited only by the separators space, comma, semicolon, or period.

The rules established for the formation of separators do not apply to the characters which comprise the contents of nonnumeric literals, comment-entries, or comment lines.

CHARACTER-STRINGS

A character-string is a character or a sequence of contiguous characters which forms a COBOL word, a literal, a PICTURE character-string, or a comment-entry. A character-string is delimited by separators.

COBOL WORDS

A COBOL word is a character-string of not more than 30 characters which forms a user-defined word, a system-name, or a reserved word. Within a source program, reserved words and user-defined words form disjoint sets; reserved words and system-names form disjoint sets; system-names and user-defined words form intersecting sets. The same COBOL word may be used as a system-name and as user-defined word within a source program; and the class of a specific occurrence of this COBOL word is determined by the context of the clause or phrase in which it occurs.

USER-DEFINED WORDS

A user-defined word is a COBOL word that must be supplied by the user to satisfy the format of a clause or statement. Each character of a user-defined word is selected from the set of characters 'A', 'B', 'C', ...'Z', '0', ... '9', and '-', except that the '-' may not appear as the first or last character.

The types of user-defined words are:

1. alphabet-name
2. class-name
3. condition-name
4. data-name
5. file-name
6. index-name
7. level-number
8. mnemonic-name
9. paragraph-name
10. program-name
11. record-name
12. report-name
13. screen-name
14. section-name
15. segment-number
16. text-name

Within a given source program, but excluding any contained program, the user-defined words are grouped into the following disjoint sets:

1. alphabet-names
2. class-names
3. condition-names, data-names, record-names, and screen-names
4. file-names
5. index-names
6. mnemonic-names
7. paragraph-names
8. program-names
9. report-names
10. section-names
11. text-names

All user-defined words, except segment-numbers and level-numbers, can belong to one and only one of these disjoint sets. Further, all user-defined words within a disjoint set must be unique, except as specified in the rules for uniqueness of reference (see Uniqueness of Reference).

With the exception of section-names, paragraph-names, segment-numbers, and level-numbers, all user-defined words must contain at least one alphabetic character. Segment-numbers and level-numbers need not be unique; a given specification of a segment-number or level-number may be identical to any other segment-number or level-number.

Condition-Name

A condition-name is a name which is assigned to a specific value, set of values, or range of values, within a complete set of values that a data item may assume. The data item itself is called a conditional variable.

Condition-names may be defined in the Data Division or in the SPECIAL-NAMES paragraph within the Environment Division where a condition-name must be assigned to the ON STATUS or OFF STATUS, or both, of implementor-defined switches.

A condition-name is used in conditions as an abbreviation for the relation condition; this relation condition posits that the associated conditional variable is equal to one of the set of values to which that condition-name is assigned. A condition-name is also used in a SET statement, indicating that the associated value is to be moved to the conditional variable.

Mnemonic-Name

A mnemonic-name assigns a user-defined word to an implementor-name. These associations are established in the SPECIAL-NAMES paragraph of the Environment Division. (See The SPECIAL-NAMES Paragraph.)

Paragraph-Name

A paragraph-name is a word which names a paragraph in the Procedure Division. Paragraph-names are equivalent if, and only if, they are composed of the same sequence of the same number of digits and/or characters.

Section-Name

A section-name is a word which names a section in the Procedure Division. Section-names are equivalent if, and only if, they are composed of the same sequence of the same number of digits and/or characters.

Other User-Defined Names

All other types of user-defined words are defined in the glossary.

SYSTEM-NAMES

A system-name is a COBOL word which is used to communicate with the operating environment. Each character used in the formation of a system-name must be selected from the set of characters 'A', 'B', 'C', ... 'Z', '0', ... '9', and '-', except that the '-' may not appear as the first or last character.

There are two types of system-names:

1. computer-name
2. implementor-name

Within a given implementation these two types of system-names form disjoint sets; a given system-name may belong to one and only one of them. (See Appendix 1 COBOL reserved words.)

RESERVED WORDS

A reserved word is a COBOL word that is one of a specified list of words which may be used in COBOL source programs, but which must not appear in the programs as user-defined words or system-names. Reserved words can only be used as specified in the general formats. (See Appendix 1 COBOL reserved Words.)

Reserved words satisfy the following conditions:

1. Reserved words do not begin with the characters '0', ... , '9', 'X', 'Y', or 'Z' except for the reserved words ZERO, ZEROES, and ZEROS.

2. Reserved words do not contain only one alphabetic character.

3. Reserved words do not start with 1 or 2 characters followed by '-' except for the reserved words I-O, I-O-CONTROL, and reserved words which begin with 'B-' or 'DB-'.

4. Reserved words do not contain two or more contiguous hyphens.

There are three types of reserved words:

1. required words
2. optional words
3. special purpose words

Required Words

A required word is a word whose presence is required when the format in which the word appears is used in a source program.

Required words are of two types:

1. Key words. Within each format, such words are uppercase and underlined.

2. Special character words. These are the arithmetic operators and relation characters.

Optional Words

Within each format, uppercase words that are not underlined are called optional words and may appear at the user's option with no effect on the semantics of the format.

Special Purpose Words

There are two types of special purpose words:

1. special registers
2. figurative constants

Special Registers

Certain reserved words are used to name and reference special registers, compiler-generated storage areas whose primary use is to store information produced in conjunction with the use of specific COBOL features. Unless specified otherwise in these specifications, one special register of each type is allocated for each program. In the general formats of this specification, a special register may be used, unless otherwise restricted, wherever a data-name or identifier is specified provided that the special register is the same category as the data-name or identifier. If qualification is allowed, special registers may be qualified as necessary to provide uniqueness. (See Qualification.)

There are five special registers:

1. DEBUG-ITEM
2. LINAGE-COUNTER
3. LINE-COUNTER
4. MESSAGE
5. PAGE-COUNTER

Figurative Constants

Certain reserved words are used to name and reference specific constant values. These reserved words are specified in paragraph on Figurative Constant Values.

LITERALS

A literal is a character-string whose value is implied by an ordered set of characters of which the literal is composed or by specification of a reserved word which references a figurative constant. Every literal belongs to one of two types, nonnumeric or numeric.

Nonnumeric Literals

A nonnumeric literal is a character-string delimited at the beginning and at the end by the separator quotation mark. The length of a nonnumeric literal applies to its representation in the object program.

General Format

Format 1

"{character-1} ... "

Format 2

""{character-2} ... ""

Syntax Rules

1. Character-1 may be any character in the computer character set.

2. If character-1 is to represent the quotation mark, two contiguous quotation mark characters must be used to represent a single occurrence of that character.

3. Format 1 literals may be of 1 through 160 characters in length.

4. Character-2 may be any character chosen from the following: '0', ... '9', and 'A', 'B', 'C', ...'F'.

5. Format 2 literals may be from 2 through 320 characters in length. The number of hexadecimal characters must always be even.

General Rules

1. Format 1 describes a regular nonnumeric literal. Format 2 is that of a hexadecimal literal.

2. In format 1, the value of the nonnumeric literal in the object program is the value represented by character-1.

3. In format 2, the value of the nonnumeric literal in the object program is the value represented by pairs of character-2. The size of the literal is equal to one-half of the number of characters in the string specified by the user.

4. The separator quotation mark that delimits the nonnumeric literal is not part of the value of the nonnumeric literal.

5. All nonnumeric literals are of category alphanumeric. (See The PICTURE Clause.)

Numeric Literals

A numeric literal is a character-string whose characters are selected from the digits '0' through '9', the plus sign, the minus sign, the decimal point and the character 'E'.

General Format

Format 1

```
[±] {digit-1}...
```

Format 2

```
[±] [digit-2]... '.' {digit-2}...
```

Format 3

```
[±] [digit-3]... '.' {digit-3}... 'E' [±] {digit-4}...
```

Syntax Rules

1. Format 1 and format 2 literals may contain from 1 to 18 digits.

2. In format 3, digit-3 defines the mantissa of the floating point literal and may consist of 1 to 15 digits. However, in single-precision context (when a data item is declared with USAGE COMPUTATIONAL-1), digit-3 may consist of 1 to 7 digits.

3. In format 3, digit-4 defines the exponent of the floating point literal and may consist of 1 to 4 digits. However, in single-precision context (when a data item is declared with USAGE COMPUTATIONAL-1), digit-4 may consist of 1 to 3 digits.

4. A literal must not contain more than one decimal point. The decimal point is treated as an assumed decimal point, and may appear anywhere within the literal except as the rightmost character.

General Rules

1. Format 1 describes an unscaled numeric literal or integer. Format 2 and format 3 describe two representations for a scaled numeric literal.

2. If the literal is unsigned, the literal is positive. In format 3, the literal is unsigned if the leftmost character is not the plus sign or minus sign.

3. If a literal conforms to the rules for the formation of numeric literals, but is enclosed in quotation marks, it is a nonnumeric literal and it is treated as such by the compiler.

4. The value of a numeric literal is the algebraic quantity represented by the characters in the numeric literal. Every numeric literal is category numeric. (See The PICTURE Clause.)

5. The size of a numeric literal described by format 1 and format 2 in standard data format characters is equal to the number of digits specified by the user. In format 3, the size is four bytes or eight bytes depending upon the context. In the context of a VALUE clause for an item defined with USAGE COMPUTATIONAL-1, the size is four bytes. In all other contexts, the size is eight bytes.

FIGURATIVE CONSTANT VALUES

Figurative constant values are generated by the compiler and referenced through the use of the reserved words given below. These words must not be bounded by quotation marks when used as figurative constants. The singular and plural forms of figurative constants are equivalent and may be used interchangeably.

The figurative constant values and the reserved words used to reference them are as follows:

1. [ALL] ZERO, [ALL] ZEROS, [ALL] ZEROES Represents the numeric value '0', or one or more of the character '0' from the computer's character set.

2. [ALL] SPACE, [ALL] SPACES Represents one or more of the character space from the computer's character set.

3. [ALL] HIGH-VALUE, [ALL] HIGH-VALUES Except in the ALPHABET clause, represents one or more of the character that has the highest ordinal position in the program collating sequence.

4. [ALL] LOW-VALUE, [ALL] LOW-VALUES Except in the ALPHABET clause, represents one or more of the character that has the lowest ordinal position in the program collating sequence.

5. [ALL] QUOTE, [ALL] QUOTES Represents one or more of the character '"'. The word QUOTE or QUOTES cannot be used in place of a quotation mark in a source program to bound a nonnumeric literal. Thus, QUOTE ABD QUOTE is incorrect as a way of stating the nonnumeric

literal "ABD".

6. ALL literal Represents all or part of the string generated by successive concatenations of the characters comprising the literal. The literal must be a nonnumeric literal. The literal must not be a figurative constant containing the word ALL.

When a figurative constant represents a string of one or more characters, the length of the string is determined by the compiler from context according to the following rules:

1. When a figurative constant is specified in a VALUE clause, or when a figurative constant is associated with another data item (e.g., when the figurative constant is moved to or compared with another data item), the string of characters specified by the figurative constant is repeated character by character on the right until the size of the resultant string is greater than or equal to the size in characters of the associated data item. This resultant string is then truncated from the right until it is equal to the number of character positions in the associated data item. This is done prior to and independent of the application of any JUSTIFIED clause that may be associated with the data item.

2. When a figurative constant other than ALL literal is not associated with another data item, as when the figurative constant appears in a DISPLAY, STOP, STRING or UNSTRING statement, the length of the string is one character.

3. When the figurative constant ALL literal is not associated with another data item, the length of the string is the length of the literal.

A figurative constant may be used whenever 'literal' appears in a format with the following exceptions:

1. If the literal is restricted to a numeric literal, the only figurative constant permitted is ZERO (ZEROS, ZEROES).

2. The figurative constant ALL literal cannot be associated with a numeric or numeric edited data item.

3. When a figurative constant other than ALL literal is used, the word ALL is redundant and is used for readability only.

Except in the ALPHABET clause, when the figurative constants HIGH-VALUE(S) or LOW-VALUE(S) are used in the source program, the actual character associated with each figurative constant depends upon the program collating sequence specified. (See The OBJECT-COMPUTER Paragraph, and The SPECIAL-NAMES Paragraph.) The following table shows the values used for the predefined collating sequences.

Collating sequence specified	LOW-VALUE (hex. value)	HIGH-VALUE (hex. value)
NATIVE	00	FF
STANDARD-1	00	7F

Each reserved word which is used to reference a figurative constant value is a distinct character-string with the exception of the constructs using the word ALL such as ALL literal, ALL SPACES, etc., which are composed of two distinct character-strings.

PICTURE CHARACTER-STRINGS

A PICTURE character-string consists of certain symbols which are composed of the currency symbol and certain combinations of characters in the COBOL character set. An explanation of the PICTURE character-string and the rules that govern its use are given under the appropriate paragraph. (See The PICTURE Clause.)

Any punctuation character which appears as part of the specification of a PICTURE character-string is not considered as a punctuation character, but rather as a symbol used in the specification of that PICTURE character-string.

COMMENT-ENTRIES

A comment-entry is an entry in the Identification Division that may be any combination of characters from the computer's character set.

CONCEPT OF COMPUTER INDEPENDENT DATA DESCRIPTION

To make data as computer-independent as possible, the characteristics or properties of the data are described in relation to a standard data format rather than an equipment-oriented format. This standard data format is oriented to general data processing applications and uses the decimal system to represent numbers (regardless of the radix used by the computer) and all characters in the COBOL character set to describe nonnumeric data items.

LOGICAL RECORD CONCEPT

In order to separate the logical characteristics of data from the physical characteristics of the data storage media, separate clauses or phrases are used. The following paragraphs discuss the characteristics of files.

Physical Aspects of a File

The physical aspects of a file describe the data as it appears on the input or output media and include such features as:

1. The grouping of logical records within the physical limitations of the file medium.

2. The means by which the file can be identified.

Conceptual Characteristics of a File

The conceptual characteristics of a file are the explicit definition of each logical entity within the file itself. In a COBOL program, the input or output statements refer to one logical record.

It is important to distinguish between a physical record and a logical record. A COBOL logical record is a group of related information, uniquely identifiable, and treated as a unit.

A physical record is a physical unit of information whose size and recording mode depend on each input or output device. The size of a physical record is hardware dependent and bears no direct relationship to the size of the file of information contained on a device.

A logical record may be contained within a single physical unit; or several logical records may be contained within a single physical unit; or, a logical record may require more than one physical unit to contain it. In this document, references to records means to logical records, unless the term 'physical record' is specifically used.

The concept of a logical record is not restricted to file data but is carried over into the definition of working storage. Thus, working storage is grouped into logical records and defined by a series of record description entries.

When a logical record is transferred to or from a physical unit, any translation required by the presence of a CODE-SET clause is accomplished. Padding characters are added or deleted as necessary. None of the clauses used to describe the data in the logical record have any effect on this transfer.

Record Concepts

The record description consists of a set of data description entries which describe the characteristics of a particular record. Each data description entry consists of a level-number followed by a data-name, if required, followed by a series of independent clauses, as required.

CONCEPT OF LEVELS

A level concept is inherent in the structure of a logical record. This concept arises from the need to specify subdivisions of a record for the purpose of data reference. Once a subdivision has been specified, it may be further subdivided to permit more detailed data referral.

The most basic subdivisions of a record, that is, those not further subdivided, are called elementary items; consequently, a record is said to consist of a sequence of elementary items, or the record itself may be an elementary item.

In order to refer to a set of elementary items, the elementary items are combined into groups. Each group consists of a named sequence of one or more elementary items. Groups, in turn, may be combined into groups of two or more groups, etc. Thus, an elementary item may belong to more than one group.

UNIQUENESS OF REFERENCE

Every user-defined name in a COBOL program is assigned, by the user, to name a resource which is to be used in solving a data processing problem. (See User-Defined Words.) In order to use a resource, a statement in a COBOL program must contain a reference which uniquely identifies that resource. In order to ensure uniqueness of reference, a user-defined name may be qualified or subscripted as described in the following paragraphs.

When the same name has been assigned in separate programs to two or more occurrences of a resource of a given type, and when qualification by itself does not allow the reference in one of those programs to differentiate between two identically named resources, then certain conventions which limit the scope of names apply. These conventions ensure that the resource identified is that described in the program containing the reference. (See Scope of Names.)

Unless otherwise specified by the rules for a statement, any subscripting is evaluated only once as the first operation of the execution of that statement.

QUALIFICATION

Every user-defined name explicitly referenced in a COBOL source program must be uniquely referenced because either:

1. No other name has the identical spelling and hyphenation.

2. It is unique within the context of a REDEFINES clause. (See The REDEFINES Clause.)

3. The name exists within a hierarchy of names such that reference to the name can be made unique by mentioning one or more of the higher level names of the hierarchy.
   These higher levels are called qualifiers and this process that specifies uniqueness is called qualification. Identical user-defined names may appear in a source program; however, uniqueness must then be established through qualification of each user-defined name explicitly referenced, except in the case of redefinition. All available qualifiers need not be specified so long as uniqueness is established. Reserved words naming special registers require qualification to provide uniqueness of reference whenever a source program would result in more than one occurrence of any of these special registers. A paragraph-name or section-name appearing in a program may not be referenced from any other program.

4. A program is contained within a program or contains another program (see Scope of Names).

Regardless of the above, the same data-name must not be used as the name of an external record and as the name of any other external data item described in any program contained within or containing the program which describes that external data record. The same data-name must not be used as the name of an item possessing the global attribute and as the name of

any other data item described in the program which describes that global data item.

The general formats for qualification are:

Format 1

```
{ data-name-1    }   { { IN } }                                  [ { IN } ]
{ condition-name }   { { OF } } data-name-2 ... [ { OF } file-name ]
                     {
                     { { IN } }
                     { { OF } } file-name
```

Format 2

```
                        { IN }
paragraph-name          { OF }  section-name
```

Format 3

```
                        { IN }
LINAGE-COUNTER          { OF }  file-name
```

Format 4

```
{ PAGE-COUNTER }   { IN }
{ LINE-COUNTER }   { OF }  report-name
```

Format 5

```
                   { { IN } }                                  [ { IN } ]
data-name-3        { { OF } } data-name-4 [ { OF } report-name ]
                   {
                   { { IN } }
                   { { OF } } report-name
```

The rules for qualification are as follows:

1. For each nonunique user-defined name that is explicitly referenced, uniqueness must be established through a sequence of qualifiers which precludes any ambiguity of reference.

2. A name can be qualified even though it does not need qualifications; if there is more than one combination of qualifiers that ensures uniqueness, then any such set can be used.

3. IN and OF are logically equivalent.

4. In format 1, each qualifier must be the name associated with a level indicator, the name of a group item to which the item being qualified is subordinate, or the name of the conditional variable with which the condition-name being qualified is associated. Qualifiers are specified in the order of successively more inclusive levels in the hierarchy.

5. In format 1, data-name-1 or data-name-2 may be a record-name.

6. If explicitly referenced, a paragraph-name must not be duplicated within a section. When a paragraph-name is qualified by a section-name, the word SECTION must not appear. A paragraph-name need not be qualified when referred to from within the same section. A paragraph-name or section-name appearing in a program may not be referenced from any other program.

7. LINAGE-COUNTER must be qualified each time it is referenced if more than one file description entry containing a LINAGE clause has been specified in the source program.

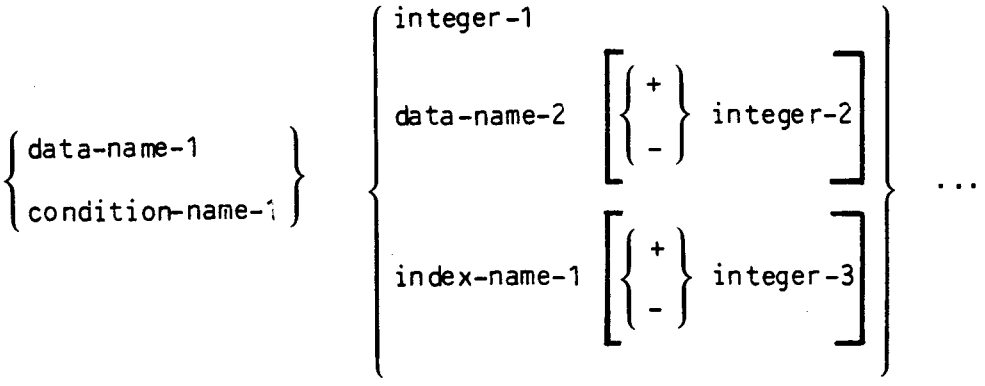
8. LINE-COUNTER must be qualified each time it is referenced in the Procedure Division if more than one report description entry is specified in the source program. In the Report Section, an unqualified reference to LINE-COUNTER is qualified implicitly by the name of the report in whose report description entry the reference is made. Whenever the LINE-COUNTER of a different report is referenced, LINE-COUNTER must be qualified explicitly by the report-name associated with the different report.

9. PAGE-COUNTER must be qualified each time it is referenced in the Procedure Division if more than one report description entry is specified in the source program. In the Report Section, an unqualified reference to the PAGE-COUNTER is qualified implicitly by the name of the report in whose report description entry the reference is made. Whenever the PAGE-COUNTER of a different report is referenced, PAGE-COUNTER must be qualified explicitly by the report-name associated with the different report.

SUBSCRIPTING

Subscripts are used when reference is made to an individual element within a table of like elements that have not been assigned individual data-names. (See The OCCURS clause.)

General Format



Syntax Rules

1. The data description-entry containing data-name-1 or the data-name associated with condition-name-1 must contain an OCCURS clause or must be subordinate to a data description entry which contains an OCCURS clause.

2. Except as defined in syntax rule 4, when a reference is made to a table element, the number of subscripts must equal the number of OCCURS clauses in the description of the table element being referenced. When more than one subscript is required, the subscripts are written in the order of successively less inclusive dimensions of the table.

3. Index-name-1 must correspond to a data description entry in the hierarchy of the table being referenced which contains an INDEXED BY phrase specifying that index-name.

4. Each table element reference must be subscripted except when such reference appears:

   a) In a USE FOR DEBUGGING statement.

   b) As the subject of a SEARCH statement.

   c) In a REDEFINES clause.

   d) In the KEY IS phrase of an OCCURS clause.

5. Data-name-2 may be qualified and must be a numeric elementary item representing an integer.

6. Integer-1 may be signed and, if signed, it must be positive.

General Rules

1. The value of the subscript must be a positive integer. The lowest possible occurrence number represented by a subscript is 1. The first element of any given dimension of a table is referenced by an occurrence number of 1. Each successive element within that dimension of the table is referenced by occurrence numbers of 2, 3, ... The highest permissible occurrence number for any given dimension of the table is the maximum number of occurrences of the item as specified in the associated OCCURS clause.

2. The value of the index referenced by index-name-1 corresponds to the occurrence number of an element in the associated table. The value of the index is defined as the occurrence number decreased by one and multiplied by the length of the corresponding element.

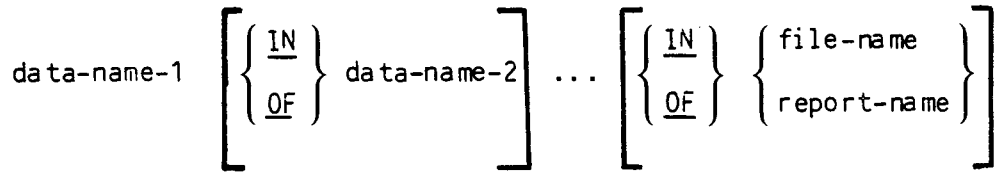
3. The value of the index referenced by index-name-1 must be initialized before it is used as a subscript. An index may be given an initial value by either a PERFORM statement with the VARYING phrase, a SEARCH statement with the ALL phrase, or a SET statement. An index may be modified only by the PERFORM, SEARCH and SET statements.

4. If integer-2 or integer-3 is specified, the value of the subscript is determined by incrementing by the value of integer-2 or integer-3 (when the operator + is used) or by decrementing by the value of integer-2 or integer-3 (when the operator - is used) either the occurrence number represented by the value of the index referenced by index-name-1 or the value of the data item referenced by data-name-2.

IDENTIFIER

An identifier is a term used to reflect that a data-name, if not unique in a program, must be followed by a syntactically correct combination of qualifiers, or subscripts necessary for uniqueness of reference. (See Scope of Names.)

The general format for identifier is:



[subscript-1 [subscript-2 [subscript-3]]]

CONDITION-NAME

If explicitly referenced, a condition-name must be unique or be made unique through qualification and/or subscripting except when the scope of names conventions by themselves ensure uniqueness of reference. (See Scope of Names.)

If qualification is used to make a condition-name unique, the associated conditional variable may be used as the first qualifier. If qualification is used, the hierarchy of names associated with the conditional variable or the conditional variable itself must be used to make the condition-name unique.

If references to a conditional variable require subscripting, reference to any of its condition-names also requires the same combination of subscripting.

The format and restrictions on the combined use of qualification and subscripting of condition-names is exactly that of 'identifier' except that data-name-1 is replaced by condition-name-1.

In the general formats of the chapters that follow, 'condition-name' refers to a condition-name qualified or subscripted, as necessary.

TABLE HANDLING

Tables of data are common components of business data processing problems. Although the repeating items that make up a table could be otherwise described by a series of separate data description entries all having the same level-number and all subordinate to the same group item, there are two reasons why this approach is not satisfactory. First, from a documentation standpoint, the underlying homogeneity of the items would not be readily apparent; and second, the problem of making available an individual element of such a table would be severe when there is a decision as to which element is to be made available at object time.

Tables of data items are defined in COBOL by including the OCCURS clause in their data description entries. This clause specifies that the item is to be repeated as many times as stated. The item is considered to be a table element and its name and description apply to each repetition or occurrence. Since each occurrence of a table element does not have assigned to it a unique data-name, reference to a desired occurrence may be made only by specifying the data-name of the table element together with the occurrence number of the desired table element. The occurrence number is known as a subscript.

The number of occurrences of a table element may be specified to be fixed or variable.

Table Definition

To define a one-dimensional table, the programmer uses an OCCURS clause as part of the data description of the table element, but the OCCURS clause must not appear in the description of group items which contain the table element. Example 1 shows a one-dimensional table defined by the item TABLE-ELEMENT.

Example 1:

```
01  TABLE-1.
    02  TABLE-ELEMENT  OCCURS 20 TIMES.
        03  DOG ...
        03  FOX ...
```

In example 2, TABLE-ELEMENT defines a one-dimensional table, and DOG defines a two-dimensional table, since there is an OCCURS clause in the description of the group item (TABLE-ELEMENT) which contains DOG.

Example 2:

```
02  TABLE-1.
    03  TABLE-ELEMENT  OCCURS 20 TIMES.
        04  DOG  OCCURS 5 TIMES.
            05  EASY ...
            05  FOX ...
```

In both examples, the complete set of occurrences of TABLE-ELEMENT has been assigned the name TABLE-1. However, it is not necessary to give a group name to the table unless it is desired to refer to the complete table as a group item.

None of the three one-dimensional tables which appear in the following two examples has a group name.

Example 3:

```
01  TABLE.
    02  BAKER ...
    02  CHARLIE  OCCURS 20 TIMES ...
    02  DOG ...
```

Example 4:

```
01  TABLE.
    02  BAKER  OCCURS 20 TIMES ...
    02  CHARLIE ...
    02  DOG  OCCURS 5 TIMES ...
```

Defining a one-dimensional table within each occurrence of an element of another one-dimensional table gives rise to a two-dimensional table. To define a two-dimensional table, then, an OCCURS clause must appear in the data description of the element of the table, and in the description of only one group item which contains that table element. Thus, in example 5, DOG is an element of a two-dimensional table; it occurs 5 times within each element of the item BAKER which itself occurs 20 times. BAKER is an

element of a one-dimensional table.

Example 5:

```
02  BAKER  OCCURS 20 TIMES ...
    03  CHARLIE ...
    03  DOG  OCCURS 5 TIMES ...
```

In the general case, to define an n-dimensional table, the OCCURS clause should appear in the data description of the element of the table and in the descriptions of (n - 1) group items which contain the element.

References to Table Items

Whenever the user references a table element or a condition-name associated with a table element, the reference must indicate which occurrence of the element is intended, except in a USE FOR DEBUGGING statement and SEARCH statement. For access to a one-dimensional table the occurrence number of the desired element provides complete information. For tables of more than one dimension, an occurrence number must be supplied for each dimension of the table. In example 5, then, a reference to the fourth BAKER or the fourth CHARLIE would be complete, whereas a reference to the fourth DOG would not. To reference DOG, which is an element of a two-dimensional table, the user must reference, for example, the fourth DOG in the fifth BAKER.

Subscripting

Occurrence numbers are specified by appending one or more subscripts to the data-name.

The subscript can be represented either by an integer, a data-name which references an integer numeric elementary item, or an index-name associated with the table. A data-name or index-name may be followed by either the operator + or the operator - and an integer, which is used as an increment or decrement, respectively. It is permissible to mix integers, data-names and index-names.

The subscripts, enclosed in parentheses, are written immediately following any qualification for the name of the table element. The number of subscripts in such a reference must equal the number of dimensions in the table whose element is being referenced. That is, there must be a subscript for each OCCURS clause in the hierarchy containing the data-name including the data-name itself.

When more than one subscript is required, they are written in the order of successively less inclusive dimensions of the data organization. If a multi-dimensional table is thought of as a series of nested tables and the most inclusive or outermost table in the nest is considered to be the major table with the innermost or least inclusive table being the minor table, the subscripts are written from left to right in the order major, intermediate, and minor.

A reference to an item must not be subscripted if the item is not a table element or an item or condition-name within a table element.

The lowest permissible occurrence number is 1. The highest permissible occurrence number in any particular case is the maximum number of occurrences of the item as specified in the OCCURS clause.

Subscripting Using Integers or Data-Names

When an integer or data-name is used to represent a subscript, it may be used to reference items within different tables. These tables need not have elements of the same size. The same integer or data-name may appear as the only subscript with one item and as one of two or more subscripts with another item.

Subscripting Using Index-Names

In order to facilitate such operations as table searching and manipulating specific items, a technique called indexing is available. To use this technique, the programmer assigns one or more index-names to an item whose data description entry contains an OCCURS clause. An index associated with an index-name acts as a subscript, and its value corresponds to an occurrence number for the item to which the index-name is associated.

The INDEXED BY phrase, by which the index-name is identified and associated with its table, is an optional part of the OCCURS clause. There is no separate entry to describe the index associated with index-name since its definition is completely hardware oriented. At object time the contents of the index correspond to an occurrence number for that specific dimension of the table with which the index is associated; however, the manner of correspondence is determined by the implementor. The initial value of an index at object time is undefined, and the index must be initialized before use. The initial value of an index is assigned with the PERFORM statement with the VARYING phrase, the SEARCH statement with the ALL phrase, or the SET statement.

The use of an integer or data-name as a subscript referencing a table element or an item within a table element does not cause the alteration of any index associated with that table.

An index-name can be used to reference only the table to which it is associated using the INDEXED BY phrase.

Data that is arranged in the form of a table is often searched. The SEARCH statement provides facilities for producing serial and nonserial (for example binary) searches. It is used to search a table for a table element that satisfies a specific condition and to adjust the value of the associated index to indicate that table element.

Relative indexing is an additional option for making references to a table element or to an item within a table element. When the name of a table element is followed by a subscript of the form (index-name + or - integer), the occurrence number required to complete the reference is the same as if index-name were set up or down by integer via the SET

statement before the reference. The use of relative indexing does not cause the object program to alter the value of the index.

The value of an index can be made accessible to an object program by storing the value in an index data item. Index data items are described in the program by a data description entry containing a USAGE IS INDEX clause. The index value is moved to the index data item by the execution of a SET statement.

SUBSCRIPTING EXAMPLE

Assuming the following data definition:

```
02  XCOUNTER ...

02  BAKER  OCCURS 20 TIMES  INDEXED BY BAKER-INDEX ...
    03  CHARLIE ...
    03  DOG  OCCURS 5 TIMES ...
        04  EASY ...
        88  MAX VALUE IS ...
        04  FOX ...
            05  GEORGE  OCCURS 10 TIMES ...
                06  HARRY ...
                06  JIM ...
```

references to BAKER and CHARLIE require only one subscript, references to DOG, EASY, MAX, and FOX require two, and references to GEORGE, HARRY, and JIM require three.

To illustrate the requirement of order from major to minor, HARRY (18, 2, 7) means the HARRY in the seventh GEORGE, in the second DOG, in the the eighteenth BAKER.

Mixing integers, data-names, and index-names is illustrated by HARRY (BAKER-INDEX, 4, XCOUNTER + 5).

FILES

A file is a collection of records which may be placed into or retrieved from a storage medium. The user not only chooses the file organization, but also chooses the file processing method and sequence. Although the file organization and processing method are restricted for sequential media, no such restrictions exist for mass storage media.

When describing the capabilities of COBOL programs to manipulate files, the following conventions are used. The term 'file-name' means the user-defined word used in the COBOL source program to reference a file. The terms 'file referenced by file-name' and 'file' mean the physical file regardless of the file-name used in the COBOL program. The term file. All accesses to physical files occur through file control blocks.

FILE ATTRIBUTES

A file has several attributes which apply to the file at the time it is created and cannot be changed throughout the lifetime of the file. The primary attribute is the organization of the file, which describes its logical structure. There are three organizations: sequential, relative, and indexed. Other fixed attributes of the file provided by the COBOL program are prime record key, alternate record keys, code set, the collating sequence of the keys for indexed files, and the blocking factor.

SEQUENTIAL ORGANIZATION

Sequential files are organized so that each record, except the last, has a unique successor record; each record, except the first, has a unique predecessor record. The successor relationships are established by the order of execution of WRITE statements when the file is created. Once established, successor relationships do not change except in the case where records are added to the end of a file.

A sequentially organized mass storage file has the same logical structure as a file on any sequential medium; however, a sequential mass storage file may be updated in place. When this technique is used, new records cannot be added to the file and each replaced record must be the same size as the original record.

There are two types of sequential files; those with fixed-length records; those with variable-length records (text-files). All printer-oriented files are text-files. This includes:

- Files declared with LINAGE parameters.
- Report Files.
- Files to be printed or edited in the future.
- Files assigned to WS-PRINTER (implicit).
- Files assigned to a system printer (implicit).

Text-files may not be updated in place.

RELATIVE ORGANIZATION

A file with relative organization is a mass storage file from which any record may be stored or retrieved by providing the value of its relative record number.

Conceptually, a file with relative organization comprises a serial string of areas, each capable of holding a logical record. Each of these areas is denominated by a relative record number. Each logical record in a relative file is identified by the relative record number of its storage area. For example, the tenth record is the one addressed by relative record number 10 and is in the tenth record area, whether or not records have been written in any of the first through the ninth record areas.

In order to achieve more efficient access to records in a relative file, the number of character positions reserved on the medium to store a particular logical record may be different from the number of character positions in the description of that record in the program.

INDEXED ORGANIZATION

A file with indexed organization is a mass storage file from which any record may be accessed by giving the value of a specified key in that record. For each key data item defined for the records of a file, an index is maintained. Each such index represents the set of values from the corresponding key data item in each record. Each index, therefore, is a mechanism which can provide access to any record in the file.

Each indexed file has a primary index which represents the prime record key of each record in the file. Each record is inserted in the file, changed, or deleted from the file based solely upon the value of its prime record key. The prime record key of each record in the file must be unique, and it must not be changed when updating a record. The prime record key is declared in the RECORD KEY clause of the file control entry for the file.

Alternate record keys provide alternative means of retrieval for the records of a file. Such keys are named in the ALTERNATE RECORD KEY clauses of the file control entry. The value of a particular alternate record key in each record need not be unique. When these values may not be unique, the DUPLICATE phrase is specified in the ALTERNATE RECORD KEY clause.

Logical Records

A logical record is the unit of data which is retrieved from or stored into a file. The number of records that may exist in a file is limited only by the capacity of the storage media. Records are fixed length.

Fixed length records must contain the same number of character positions for all the records in the file. All input-output operations on the file

can process only this one record size.

The content of the record area does not reflect any information that may be added to the record on the physical storage medium (such as record length headers), nor does the length of the record used by the COBOL programmer reflect these additions.

FILE PROCESSING

A file can be processed by performing operations upon individual records or upon the file as a unit. Unusual conditions that occur during processing are communicated back to the program.

RECORD OPERATIONS ACCESS MODE

The ACCESS MODE clause of the file description entry specifies the manner in which the object program operates upon records within a file. The access mode be sequential, random, or dynamic.

The organization, format, and contents of an output report may be specified using the report writer feature. (See Report Writer.)

Sequential Access Mode

A file can be accessed sequentially irrespective of the file organization.

For sequential organization, the order of sequential access is the order in which the records were originally written.

For relative organization, the order of sequential access is ascending based on the value of the relative record numbers. Only records which currently exist in the file are made available. The START statement may be used to establish a starting point for a series of subsequent sequential retrievals.

For indexed organization, the order of sequential access is ascending based on the value of the key of reference according to the collating sequence associated with the native character set. Any of the keys associated with the file may be established as the key of reference during the processing of the file. The order of retrieval from a set of records which have duplicate key of reference values is the original order of arrival of those records into the set. The START statement may be used to establish a starting point within an indexed file for a series of subsequent sequential retrievals.

Random Access Mode

When a file is accessed in random mode, input-output statements are used to access the records in a programmer-specified order. The random access mode may only be used with relative or indexed file organizations.

For a file with relative organization, the programmer specifies the desired record by placing its relative record number in a relative key data item. With the indexed organization, the programmer specifies the desired record by placing the value of one of its record keys in a record key or an alternate record key data item.

Dynamic Access Mode

With dynamic access mode, the programmer may change at will from sequential accessing to random accessing, using appropriate forms of input-output statements. The dynamic access mode may only be used on files with relative or indexed organizations.

Open Mode

The open mode of the file is related to the actions to be performed upon records in the file. The open modes and purposes are: input, to retrieve records; output, to place records into a file; extend, to append records to an existing file; I-O, to retrieve and update records. The open mode is specified in the OPEN statement.

When the open mode is input, a file may be accessed by the READ statement. The START statement may also be used for files organized as indexed or relative which are in sequential or dynamic access modes.

When the open mode is output, the records are placed into the file by issuing WRITE, GENERATE, or TERMINATE statements.

Only mass storage files may be referenced in the open I-O mode. The additional capabilities of mass storage devices permit updating in place, thus READ and REWRITE statements may always be used. A mass storage file may be updated in the same manner as a file on a sequential medium, by transcribing the entire file into another file (perhaps in a separate area of mass storage) using READ and WRITE statements. However, it is sometimes more efficient to update a mass storage file in place. This mass storage file maintenance technique uses the REWRITE statement to return to their previous locations on the storage medium only those records which have changed. READ and REWRITE statements are the only operations allowed while updating in place sequentially organized files. However, for indexed or relative organized files, the following additional functions may be applied: the START statement may be used in sequential or dynamic access mode to alter the sequence of record retrieval; the DELETE statement may be used with any access mode to remove a record logically from a file; the WRITE statement may be used in random or dynamic access mode to insert a new record into the file.

File Operations

Several COBOL statements operate upon files as entities or as collections of records. These are the CLOSE, DELETE FILE, MERGE, OPEN, and SORT statements.

Current Volume Pointer

The current volume pointer is a conceptual entity used in this document to facilitate exact specification of the current physical volume of a sequential file. The status of the current volume pointer is affected by the CLOSE, OPEN, READ, and WRITE statements.

Sorting and Merging

Sorting and Merging provide the capability to order one or more files of records, or to combine two or more identically ordered files of records, according to a set of user-specified keys contained within each record. Optionally, each record may be processed by input or output procedures. This processing may be applied before and/or after the records are ordered by the SORT or after the records have been combined by the MERGE.

The sort or merge files have no label procedures which the programmer can control and the rules for for blocking and for allocation of internal storage are peculiar to the SORT or MERGE statements. The RELEASE and RETURN statements imply nothing with respect to buffer areas, blocks, or reals. A sort or merge file, then may be considered as an internal file which is created (RELEASE statement) from the input file, processed (SORT or MERGE statement), and then made available (RETURN statement) to the output file. The sort file itself is referred and accessed only by the SORT statement. The merge file itself is referred and accessed only by the MERGE statement. A sort-merge file description can be considered to be a particular type of file description.

File Position Indicator

The file position indicator is a conceptual entity used in this document to facilitate exact specification of the next record to be accessed within a given file during certain sequences of input-output operations. The setting of the file position indicator is affected only by the OPEN, READ, and START statements. The concept of a file position indicator has no meaning for a file opened in the output or extend mode.

Linage Concepts

The LINAGE clause may be used when specifying an output report. It facilitates definition of a logical page, and the positioning within that logical page of top and bottom margins and a footing area. Use of the LINAGE clause implicitly defines an associated special register, the LINAGE-COUNTER, which acts as a pointer to a line within the page body.

Special Register LINAGE-COUNTER

The reserved word LINAGE-COUNTER is a name for a line counter generated by the presence of a LINAGE clause in a file description entry. (See The LINAGE Clause.) The implicit description is that of an unsigned integer data item: PICTURE 9999 COMP. LINAGE-COUNTER may be referenced only in

Procedure Division statements; however, only the input-output control system may change the value of LINAGE-COUNTER.

SHARED MEMORY AREA

This feature is oriented toward saving memory space in the object program as it allows more than one file to share the same file area and alternate areas, but often gives other benefits.

When the RECORD option of the SAME clause is used, (see The SAME clause in the ENVIRONMENT DIVISION chapter) the record area is shared and the alternate areas for each file remain independent. In this case any number of the files sharing the same record area may be active at one time. This factor can give rise to an increase in the speed of the object program.

To illustrate this point, consider file maintenance. If the programmer assigns the same record area to both the old and new files, he not only saves memory in the object program, but because this technique eliminates a move of each record from the input to theoutput area, significant time savings result. An additional benefit of this technique is that the programmer need not define the record in detail as a part of both the old and new files. Rather, he defines the record completely in one case and simply includes the level 01 entry in the other. Because these record areas are in fact the same area, one set of names suffices for all processing requirements without requiring qualification.

EXCEPTION HANDLING

During the execution of any input or output operation, unusual conditions may arise which preclude normal completion of the operation. There are three methods by which these conditions are communicated to the object program; status keys, exception declaratives, and optional phrases associated with the imperative statement.

EXCEPTION DECLARATIVES

A USE AFTER EXCEPTION PROCEDURE, when one is specified for the file, is executed whenever an input or output condition arises which results in an unsuccessful input-output operation. However, the exception declarative is not executed if the condition is invalid key and the INVALID KEY phrase is specified, or if the condition is at end and the AT END phrase is specified.

THE INVALID KEY CONDITION

The invalid key condition can occur as a result of the execution of a DELETE, READ, REWRITE, START, or WRITE statement.

Some of the conditions that give rise to an invalid key condition are when a requested key does not exist in the file (DELETE, READ, or START statements), when a key is already in a file and duplicates are not allowed (WRITE statement), and when a key does not exist in the file or when it was not the last key read (REWRITE statement). If the invalid key condition occurs during the execution of a statement for which the INVALID KEY phrases has been specified, the statement identified by that INVALID KEY phrase is executed.

When the invalid key condition occurs, execution of the input-output statement which recognized the condition is unsuccessful and the file is not affected. (See The DELETE Statement; The READ Statement; The REWRITE Statement; The START Statement; and The WRITE Statement.)

If the invalid key condition exists after the execution of the input-output operation specified in an input-output statement, the following actions occur in the order shown:

1. The I-O status of the file control block associated with the statement is set to a value indicating the invalid key condition. (See I-O Status.)

2. If the INVALID KEY phrase is specified in the input-output statement, any USE AFTER STANDARD EXCEPTION procedure associated with the file control block is not executed and control is transferred to the imperative-statement specified in the INVALID KEY phrase. Execution then continues according to the rules for each statement specified in that imperative-statement. If a procedure branching or conditional statement which caused explicit transfer of control is executed, control is transferred in accordance with the rules for that statement; otherwise, upon completion of the execution of the imperative-statement specified in the INVALID KEY phrase, control is transferred

to the end of the input-output statement.

3. If the INVALID KEY phrase is not specified in the input-output statement, a USE AFTER STANDARD EXCEPTION procedure must be associated with the file control block and that procedure is executed. Upon completion of the execution of that procedure, control is transferred to the end of the input-output statement.

If the invalid key condition does not exist after the execution of the input-output operation specified by an input-output statement, the INVALID KEY phrase, if specified, is ignored and the following actions occur in the order shown:

1. The I-O status of the file control block associated with the statement is updated and, if an exception condition which is not an invalid key condition exists, any USE AFTER STANDARD EXCEPTION procedures associated with the file control block is executed. (See I-O Status.)

2. Control is then transferred to the end of the input-output statement.

THE AT END CONDITION

The AT END condition can occur as a result of the execution of a READ statement (See The READ Statement).

The AT END condition occurs in a sequentially accessed file when no next (previous) logical record exists in the file, when the number of significant digits in the relative record number is larger than the size of the relative key data item, when an optional file is not present, or when a READ statement is attempted and the at end condition already exists. If the at end condition occurs during the execution of a statement for which the AT END phrase has been specified, the statement identified by that AT END phrase is executed.

THE FILE ATTRIBUTE CONFLICT CONDITION

The file attribute conflict condition can result from the execution of an OPEN statement. When the file attribute conflict condition occurs, execution of the input-output statement that recognized the condition is unsuccessful and the file is not affected. (See The OPEN Statement.)

When the file attribute conflict condition is recognized, these actions take place in the following order:

1. A value is placed in the I-O status associated with the file-name to indicate the file attribute conflict condition. (See I-O Status.)

2. A USE AFTER EXCEPTION procedure, if any, associated with the file-name is executed.

I-O STATUS

The I-O status is a two-character conceptual entity whose value is set to indicate the status of an input-output operation during the execution of a CLOSE, DELETE, DELETE FILE, LOCK, OPEN, READ, REWRITE, START, UNLOCK, or WRITE statement and prior to the execution of any imperative statement associated with that input-output statement or prior to the execution of any applicable USE AFTER STANDARD EXCEPTION procedure. The value of the I-O status is made available to the COBOL program through the use of the FILE STATUS clause in the file control entry for the file.
A list of I-O status values with explanatory texts is given in Appendix C.

LIBRARY FACILITY

The Source Text Library Facility is provided by the COPY statement. This statement provides an extensive capability to insert and replace source program text as part of the compilation of a source program.

The effect of the interpretation of the COPY statement is to generate, from a library text available to the compiler at compile time, text which is treated by the compiler as part of the source program.

COPY

Function

The COPY statement incorporates text into a COBOL source program.

General Format

```
COPY literal-1 [ { OF } literal-2 ]
               [ { IN }           ]

[ REPLACING { { ==pseudo-text-1== } BY { ==pseudo-text-2== } } ... ]
            { { literal-3         }    { literal-4         } }
            { { word-1            }    { word-2            } }
```

Syntax Rules

1. Literal-1, literal-2 must be a non-numeric literal other than a figurative constant. Literal-1 may not begin with the character '/'. Literal-2 must begin with the character '/'. If literal-2 does not begin with the character '/', it is treated as a local name for the directory which contains a copy file named by literal-1.

2. The COPY statement must be preceded by a space and terminated by the separator period.

3. Pseudo-text-1 must contain one or more text words.

4. Pseudo-text-2 may contain zero, one, or more text words.

5. Character-strings within pseudo-text-1 and pseudo-text-2 may be continued. (See Pseudo-Text.) However, both characters of a pseudo-text delimiter must be on the same line.

6. Literals within pseudo-text-1 and pseudo-text-2 may not contain a pseudo-text delimiter.

7. Word-1 or word-2 may be any single COBOL word except 'COPY'.

8. A COPY statement may be specified in the source program anywhere a character-string or a separator, other than the closing quotation mark, may occur except that a COPY statement must not occur within a COPY statement.

9. A text-word within pseudo-text and within library text may be 1 to 322 characters in length.

10. Pseudo-text-1 must not consist entirely of a separator comma or a separator semi-colon.

11. If the word COPY appears in a comment-entry or in the place where a comment-entry may appear, it is considered part of the comment-entry.

General Rules

1. The compilation of a source program containing COPY statements is logically equivalent to processing all COPY statements prior to the processing of the resulting source program.

2. Literal-1 contains the name of the copy file. Literal-2, if present, contains the path name of the copy file. If literal-2 is not present, the current working directory is assumed.

3. The effect of processing a COPY statement is that the library text associated with literal-1 is copied into the source program, logically replacing the entire COPY statement, beginning with the reserved word COPY and ending with the punctuation character period, inclusive.

4. If the REPLACING phrase is not specified, the library text is copied unchanged.

   If the REPLACING phrase is specified, the library text is copied and each properly matched occurrence of pseudo-text-1, word-1, and literal-3 in the library text is replaced by the corresponding pseudo-text-2, word-2, or literal-4.

5. For purposes of matching, word-1 and literal-3 are treated as pseudo-text containing only word-1 or literal-3, respectively.

6. The comparison operation to determine text replacement occurs in the following manner:

   The leftmost library text word which is not a separator comma or separator semicolon is the first text word used for comparison. Any text word or space preceding this text word is copied into the source program. Starting with the first text word for comparison and the first pseudo-text-1, word-1, or literal-3 that was specified in the REPLACING phrase, the entire REPLACING phrase operand that precedes the reserved word BY is compared to an equivalent number of contiguous library text words.

   Pseudo-text-1, word-1, or literal-3 match the library text if, and only if, the ordered sequence of text words that forms pseudo-text-1, word-1, or literal-3 is equal, character for character, to the ordered sequence of library text words. For purposes of matching, each occurrence of a separator, comma, semicolon, or space in pseudo-text-1 or in the library text is considered to be a single space. Each sequence of one or more space separators is considered to be a single space.

   If no match occurs, the comparison is repeated with each next successive pseudo-text-1, word-1, or literal-3, if any, in the REPLACING phrase until either a match is found or there is no next successive REPLACING operand.

When all the REPLACING phrase operands have been compared and no match has occurred, the leftmost library text word is copied into the source program. The next successive library text word is then considered as the leftmost library text word, and the comparison cycle starts again with the first pseudo-text-1, word-1, or literal-3 specified in the REPLACING phrase.

Whenever a match occurs between pseudo-text-1, word-1, or literal-3 and the library text, the corresponding pseudo-text-2, word-2, or literal-4 is placed into the source program. The library text word immediately following the rightmost text word that participated in the match is then considered as the leftmost library text word. The comparison cycle starts again with the first pseudo-text-1, word-1, or literal-3 specified in the REPLACING phrase.

The comparison operation continues until the rightmost text word in the library text has either participated in a match or been considered as a leftmost library text word and participated in a complete comparison cycle.

7. Comment lines or blank lines occurring in the library text and in pseudo-text-1 are ignored for purposes of matching; and the sequence of text words in the library text, if any, and in pseudo-text-1 is determined by the rules for reference format. (See Reference Format Representation.) Comment lines or blank lines appearing in pseudo-text-2 are copied into the resultant program unchanged whenever pseudo-text-2 is placed into the source program as a result of text replacement. Comment lines or blank lines appearing in library text are copied into the resultant source program unchanged with the following exception: a comment line or blank line in library text is not copied if that comment line or blank line appears within the sequence of text words that match pseudo-text-1.

8. Debugging lines are permitted within library text and pseudo-text. Text words within a debugging line participate in the matching rules as if the 'D' did not appear in the indicator area. A debugging line is specified within pseudo-text if the debugging line begins in the source program after the opening pseudo-text-delimiter but before the matching closing pseudo-text-delimiter.

9. The syntactic correctness of the library text cannot be independently determined. Except for COPY statements, the syntactic correctness of the entire COBOL source program cannot be determined until all COPY statements have been completely processed.

10. Each text word copied from the library but not replaced is copied so as to start in the same area of the line in the resultant program as it begins in the line within the library. However, if a text word copied from the library begins in area A but follows another text word, which also begins in area A of the same line, and if replacement of a preceding text word in the line by replacement text of greater length occurs, the following text word begins in area B if it cannot begin in area A. Each text word in pseudo-text-2 that is to be placed into the resultant program begins in the same area of the resultant program as it appears in pseudo-text-2. Each literal-4 and word-2 that is to be placed into the resultant program begins in the

same area of the resultant program as the leftmost library text word that participated in the match would appear if it had not been replaced.

If additional lines are introduced into the source program as a result of a COPY statement, each text word introduced appears on a debugging line if the COPY statement begins on a debugging line. When a text word specified in the BY phrase is introduced, it appears on a debugging line if the first library text word being replaced is specified on a debugging line. Except in the preceding cases, only those text words which are specified on debugging lines where the debugging line is within pseudo-text-2 appear on debugging lines in the resultant program. If any literal specified as literal-4 or within pseudo-text-2 or library text is of too great length to be accommodated on a single line without continuation to another line in the resultant program and the literal is not being placed on a debugging line, additional continuation lines are introduced which contain the remainder of the literal. If replacement requires that the continued literal be continued on a debugging line, the program is in error.

11. For purposes of compilation, text words after replacement are placed in the source program according to the rules for reference format. (See Reference Format Representation.) When copying text words of pseudo-text-2 into the source program, additional spaces may be introduced only between text words where there already exists a space (including the assumed space between source lines).

12. If additional lines are introduced into the source program as a result of the processing of COPY statements, the indicator area of the introduced lines contains the same character as the line on which the text being replaced begins unless that line contains a hyphen, in which case the introduced line contains a space. In the case where a literal is continued onto an introduced line which is not a debugging line, a hyphen is placed in the indicator area.

A COBOL SOURCE PROGRAM

A COBOL source program is a syntactically correct set of COBOL statements.

ORGANIZATION

With the exception of the COPY statement and the end program header, the statements, entries, paragraphs, and sections of a COBOL source program are grouped into four divisions which are sequenced in the following order:

The Identification Division
The Environment Division
The Data Division
The Procedure Division

The end of a COBOL source program is indicated either by the end program header or by the end of that program's Procedure Division.

STRUCTURE

The following gives the general format and order of presentation of the entries and statements which constitute a COBOL source program.

General Format

IDENTIFICATION-DIVISION

[ENVIRONMENT-DIVISION]

[DATA-DIVISION]

PROCEDURE-DIVISION

[END-PROGRAM-HEADER]

IDENTIFICATION DIVISION

General Description

The Identification Division must be included in every COBOL source program. This division identifies both the source program and the resultant output listing. In addition, the user may include the date the program is written, the date the compilation of the source program is accomplished and such other information as desired under the paragraphs in the general format shown below.

Organization

Paragraph headers identify the type of information contained in the paragraph. The name of the program must be given in the first paragraph, which is the PROGRAM-ID paragraph. The other paragraphs are optional and may be included in this division at the user's choice, in order of presentation shown by the format below.

Structure

The following is the general format of the paragraphs in the Identification Division and it defines the order of presentation in the source program.

General Format

IDENTIFICATION DIVISION.

PROGRAM-ID. program-name.

[AUTHOR. [comment-entry] ...]

[INSTALLATION. [comment-entry] ...]

[DATE-WRITTEN. [comment-entry] ...]

[DATE-COMPILED. [comment-entry] ...]

[SECURITY. [comment-entry] ...]

ENVIRONMENT DIVISION

General Description

The Environment Division specifies a standard method of expressing those aspects of a data processing problem that are dependent upon the physical characteristics of a specific computer. This division allows specification of the configuration of the compiling computer and the object computer. In addition, information relating to input-output control, special hardware characteristics and control techniques can be given.

Organization

Two sections make up the Environment Division: the Configuration Section and the Input-Output Section.

The Configuration Section deals with the characteristics of the source computer and the object computer. This section is divided into three paragraphs: The SOURCE-COMPUTER paragraph, which describes the computer configuration on which the source program is compiled; the OBJECT-COMPUTER paragraph, which describes the computer configuration on which the object program produced by the compiler is to be run; and the SPECIAL-NAMES paragraph, which provides a mean for specifying the currency sign, choosing the decimal point, relating implementor-name to user-specified mnemonic-names, relating alphabet-names to character sets or collating sequences, and relating class-name to sets of characters.

The Input-Output Section deals with the information needed to control transmission and handling of data between external media and the object program. This section is divided into two paragraphs: the FILE-CONTROL paragraph which names and associates the files with external media; and the I-O-CONTROL paragraph which defines special control techniques to be used in the object program.

Structure

The following is the general format of the sections and paragraphs in the Environment Division, and defines the order of presentation in the source program.

General Format

ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

[SOURCE-COMPUTER. [source-computer-entry]]

[OBJECT-COMPUTER. [object-computer-entry]]

[SPECIAL-NAMES. [special-names-entry]]]

[INPUT-OUTPUT SECTION.

FILE-CONTROL. {file-control-entry} ...

[I-O-CONTROL. [input-output-control-entry]]]

DATA DIVISION

Overall Approach

The Data Division describes the data that the object program is to accept as input, to manipulate, to create, or to produce as output.

PHYSICAL AND LOGICAL ASPECTS OF DATA DESCRIPTION

Organization

The Data Division is subdivided into sections. These are the File, Working-Storage, Linkage, Screen and Report Sections.

The File Section defines the structure of data files. Each file is defined by a file description entry and one or more record description entries. Record descriptions are written immediately following the file description entry. When the file description specifies a file to be used as a Report Writer output file, no record description entries are permitted for that file. Report description entries appear in a separate section of the Data Division, the Report Section.

The Working-Storage Section describes records and noncontiguous data items which are not part of external data files but are developed and processed internally. It also describes data items whose values are assigned in the source program and do not change during the execution of the object program.

The Linkage Section appears in the called program and describes data items that are to be referred to by the calling program and the called program. Its structure is the same as the Working-Storage Section.

The Screen Section describes the content and format of screens that are to be generated by the Screen Handling function.

The Report Section describes the content and format of reports that are to be generated.

Structure

The following gives the general format of the sections in the Data Division, and defines the order of their presentation in the source program.

General Format

```
DATA DIVISION.

[FILE SECTION.

[ file-description-entry [record-description-entry] ...
  sort-merge-file-description-entry
                        {record-description-entry} ... ] ... ]

[WORKING-STORAGE SECTION.

[ 77-level-description-entry ] ... ]
[ record-description-entry   ]

[LINKAGE SECTION.

[ 77-level-description-entry ] ... ]
[ record-description-entry   ]

[SCREEN SECTION.

[screen-element-description-entry] ...]

[REPORT SECTION.

[report-description-entry {report-group-description-entry}...]...]
```

PROCEDURE DIVISION

General Description

The Procedure Division may contain declaratives and nondeclarative procedures.

DECLARATIVES

Declarative sections must be grouped at the beginning of the Procedure Division preceded by the key word DECLARATIVES and followed by the key words END DECLARATIVES.

Procedures

A procedure is composed of a paragraph, or group of successive paragraphs, or a section, or a group of successive sections within the Procedure Division. If one paragraph is in a section, then all paragraphs must be in sections. A procedure-name is a word used to refer to a paragraph or section in the source program in which it occurs. It consists of a paragraph-name (which may be qualified), or a section-name.

A section consists of a section header followed by zero, one, or more successive paragraphs. A section ends immediately before the next section or at the end of the Procedure Division or, in the declaratives portion of the Procedure Division, at the key words END DECLARATIVES.

A paragraph consists of a paragraph-name followed by the separator period and by zero, one, or more successive sentences. A paragraph ends immediately before the next paragraph-name or section-name or at the end of the Procedure Division or, in the declaratives portion of the Procedure Division, at the key words END DECLARATIVES. A sentence consists of one or more statements and is terminated by the separator period.

A statement is a syntactically valid combination of words and symbols beginning with a COBOL verb.

The term 'identifier' is defined as the word or words necessary to make unique reference to a data item.

Execution

Execution begins with the first statement of the Procedure Division, excluding declaratives. Statements are then executed in the order in which they are presented for compilation, except where the rules indicate some other order.

PROCEDURE DIVISION STRUCTURE

Procedure Division Header

The Procedure Division is identified by and must begin with the following header:

```
PROCEDURE DIVISION [USING {data-name-1} ... ] .
```

Procedure Division Body

The body of the Procedure Division must conform to one of the following formats:

Format 1

```
[DECLARATIVES.

{section-name SECTION [segment-number].

    USE statement.

[paragraph-name.  [sentence] ... ] ... } ...

 END DECLARATIVES.]

{section-name SECTION  [segment-number].

[paragraph-name.  [sentence] ... ] ... } ...
```

Format 2

```
{paragraph-name.  [sentence] ... } ...
```

STATEMENTS AND SENTENCES

There are four types of statements: imperative statements, conditional statements, compiler directing statements, and delimited scope statements.

There are three types of sentences: imperative sentences, conditional sentences, and compiler directing sentences.

Definition of Conditional Statement

A conditional statement specifies that the truth value of a condition is to be determined and that the subsequent action of the object program is dependent on this truth value.

A conditional statement is one of the following:

1. An EVALUATE, IF, SEARCH or RETURN statement.
2. A READ statement that specifies the AT END or INVALID KEY phrase.
3. A WRITE statement that specifies the INVALID KEY or END-OF-PAGE phrase.
4. A START, REWRITE or DELETE statement that specifies the INVALID KEY phrase.
5. An arithmetic statement (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT) that specifies the ON SIZE ERROR phrase.
6. A STRING or UNSTRING that specifies the ON OVERFLOW phrase.
7. A CALL statement that specifies the ON OVERFLOW or ON EXCEPTION phrase.
8. An ACCEPT or EDIT statement that specifies the ON ESCAPE or ON EXCEPTION phrase.

Definition of Conditional Sentence

A conditional sentence is a conditional statement, optionally preceded by an imperative statement, terminated by the separator period.

Definition of Compiler Directing Statement

A compiler directing statement consists of a compiler directing verb and its operands. The compiler directing verbs are COPY and USE. A compiler directing statement causes the compiler to take specific action during compilation.

Definition of Compiler Directing Sentence

A compiler directing sentence is a single compiler directing statement terminated by the separator period.

Definition of Imperative Statement

An imperative statement begins with an imperative verb and specifies an unconditional action to be taken by the object program or is a conditional statement that is delimited by its explicit scope terminator (delimited scope statement). An imperative statement may consist of a sequence of imperative statements, each possibly separated from the next by a separator. The imperative verbs are:

ACCEPT (7)	GO TO	SET
ADD (1)	INITIATE	SORT
CALL (6)	INSPECT	START (2)
CANCEL	LOCK	STOP
CLOSE	MERGE	STRING (3)
COMPUTE (1)	MOVE	SUBTRACT (1)
DELETE (2)	MULTIPLY (1)	SUPPRESS
DISPLAY	OPEN	TERMINATE
DIVIDE (1)	PERFORM	
EDIT (7)	READ (4)	UNLOCK
EXIT	RELEASE	UNSTRING (3)
GENERATE	REWRITE (2)	WRITE (5)

1 Without the optional ON SIZE ERROR phrase.
2 Without the optional INVALID KEY phrase.
3 Without the optional ON OVERFLOW phrase.
4 Without the optional AT END phrase or INVALID KEY phrase.
5 Without the optional INVALID KEY phrase or END-OF-PAGE phrase.
6 Without the optional ON OVERFLOW or ON EXCEPTION phrase.
7 Without the optional ON ESCAPE or ON EXCEPTION phrase.

Whenever 'imperative-statement' appears in the general format of statements, 'imperative-statement' refers to that sequence of consecutive imperative statements that must be ended by a period or by any phrase associated with a statement containing that 'imperative-statement'.

Definition of Imperative Sentence

An imperative sentence is an imperative statement terminated by the separator period.

Delimited Scope Statements

A delimited scope statement is any statement which includes its explicit scope terminator. (See Explicit and Implicit Scope Terminators.)

Categories of Statements

Arithmetic

- ADD
- COMPUTE
- DIVIDE
- INSPECT (TALLYING)
- MULTIPLY
- SUBTRACT

Compiler Directing

- COPY
- USE

Conditional

- ACCEPT (ON ESCAPE or ON EXCEPTION)
- ADD (ON SIZE ERROR)
- CALL (ON OVERFLOW or ON EXCEPTION)
- COMPUTE (ON SIZE ERROR)
- DELETE (INVALID KEY)
- DIVIDE (ON SIZE ERROR)
- EDIT (ON ESCAPE or ON EXCEPTION)
- EVALUATE
- IF
- MULTIPLY (ON SIZE ERROR)
- READ (AT END or INVALID KEY)
- RETURN
- REWRITE (INVALID KEY)
- SEARCH
- START (INVALID KEY)
- STRING (ON OVERFLOW)
- SUBTRACT (ON SIZE ERROR)
- UNSTRING (ON OVERFLOW)
- WRITE (INVALID KEY or END-OF-PAGE)

Data Movement

- ACCEPT (DATE,DAY,TIME or LINE-PARAM)
- INSPECT (CONVERTING or REPLACING)
- MOVE
- SET (TO TRUE)
- STRING
- UNSTRING

Ending

- STOP

Input-output

- ACCEPT (identifier or screen-item)
- CLOSE
- DELETE
- DISPLAY
- LOCK
- OPEN
- READ
- REWRITE
- SET (TO ON or TO OFF)
- START
- STOP (literal)
- UNLOCK
- WRITE

Inter-program Communication

- CALL
- CANCEL

No Operation

- EXIT

Ordering

- MERGE
- RELEASE
- RETURN
- SORT

Procedure Branching

CALL	MERGE (OUTPUT PROCEDURE)
EXIT PROGRAM	PERFORM
GO TO	SORT (INPUT or OUTPUT PROCEDURE)

Screen Handling

ACCEPT (screen-name or FROM ESCAPE KEY)
DISPLAY (screen-name)
EDIT

Report Writing

GENERATE	SUPPRESS
INITIATE	TERMINATE

Scope Delimiting

ACCEPT (END-ACCEPT)	READ (END-READ)
ADD (END-ADD)	RETURN (END-RETURN)
CALL (END-CALL)	REWRITE (END-REWRITE)
COMPUTE (END-COMPUTE)	SEARCH (END-SEARCH)
DELETE (END-DELETE)	START (END-START)
DIVIDE (END-DIVIDE)	STRING (END-STRING)
EDIT (END-EDIT)	
EVALUATE (END-EVALUATE)	SUBTRACT (END-SUBTRACT)
IF (END-IF)	UNSTRING (END-UNSTRING)
MULTIPLY (END-MULTIPLY)	WRITE (END-WRITE)
PERFORM (END-PERFORM)	

Table Handling

SEARCH	SET (TO, UP BY, or DOWN BY)

IF is a verb in the COBOL sense; it is recognized that it is not a verb in English.

Specific Statement Formats

The specific statement formats, together with a detailed discussion of the restrictions and limitations associated with each, appear in alphabetic sequence in the appropriate chapter of this document.

Scope of Statements

Scope terminators delimit the scope of certain Procedure Division statements. Statements which include their explicit scope terminators are termed delimited scope statements. (See Explicit and Implicit Scope Terminators.) The scope of statements which are contained within statements (nested) may also be implicitly terminated.

When statements are nested within other statements, a separator period which terminates the sentence also implicitly terminates all nested statements.

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 a delimited scope statement is nested within another delimited scope statement with the same verb, each explicit scope terminator terminates the statement begun by the most recently preceding, and as yet unpaired, occurrence of that verb.

EXPLICIT AND IMPLICIT SPECIFICATIONS

There are four types of explicit and implicit specifications that occur in COBOL source programs:

- Explicit and implicit Procedure Division references
- Explicit and implicit transfers of control
- Explicit and implicit attributes
- Explicit and implicit scope terminators.

Explicit and Implicit Procedure Division References

A COBOL source program can reference data items either explicitly or implicitly in Procedure Division statements. An explicit reference occurs when the name of the referenced item is written in a Procedure Division statement or when the name of the referenced item is copied into the Procedure Division by the processing of a COPY statement. An implicit reference occurs when the item is referenced by a Procedure Division statement without the name of the referenced item being written in the source statement. An implicit reference also occurs, during the execution of a PERFORM statement, when the index or data item referenced by the index-name or identifier specified in the VARYING, AFTER or UNTIL phrase is initialized, modified, or evaluated by the control mechanism associated with that PERFORM statement. Such an implicit reference occurs if and only if the data item contributes to the execution of the statement.

Explicit and Implicit Transfers of Control

The mechanism that controls program flow transfers control from statement to statement in the sequence in which they were written in the source program unless an explicit transfer of control overrides this sequence or there is no next executable statement to which control can be passed. The transfer of control from statement to statement occurs without the writing of an explicit Procedure Division statement, and therefore, is an implicit transfer of control.

COBOL provides both explicit and implicit means of altering the implicit control transfer mechanism.

In addition to the implicit transfer of control between consecutive statements, implicit transfer of control also occurs when the normal flow is altered without the execution of a procedure branching statement. COBOL provides the following types of implicit control flow alterations

which override the statement-to-statement transfers of control:

1. If a paragraph is being executed under control of another COBOL statement (for example, PERFORM, USE, SORT and MERGE) and the paragraph is the last paragraph in the range of the controlling statement, then an implied transfer of control occurs from the last statement in the paragraph to the control mechanism of the last executed controlling statement. Further, if a paragraph is being executed under the control of a PERFORM statement which causes iterative execution and that paragraph is the first paragraph in the range of that PERFORM statement, an implicit transfer of control occurs between the control mechanism associated with that PERFORM statement and the first statement in that paragraph for each iterative execution of the paragraph.

2. When a SORT or MERGE statement is executed, an implicit transfer of control occurs to any associated input or output procedures.

3. When any COBOL statement is executed which results in the execution of a declarative section, an implicit transfer of control to the declarative section occurs. Note that another implicit transfer of control occurs after execution of the declarative section, as described in rule 1 above.

An explicit transfer of control consists of an alteration of the implicit control transfer mechanism by the execution of a procedure branching or conditional statement. (See Categories of Statements.) An explicit transfer of control can be caused only by the execution of a procedure branching or conditional statement. The procedure branching statement EXIT PROGRAM causes an explicit transfer of control when the statement is executed in a called program.

In this document, the term 'next executable statement' is used to refer to the next COBOL statement to which control is transferred according to the rules above and the rules associated with each language element.

There is no next executable statement following:

1. The last statement in a declarative section when the paragraph in which it appears is not being executed under the control of some other COBOL statement.

2. The last statement in a program when the paragraph in which it appears is not being executed under the control of some other COBOL statement.

3. A STOP RUN statement or EXIT PROGRAM statement that transfers control outside the COBOL program.

4. The end program header.

When there is no next executable statement and control is not transferred outside the COBOL program, the program flow of control is undefined unless the program is under control of a CALL statement, in which case an implicit EXIT PROGRAM statement is executed.

Explicit and Implicit Attributes

Attributes may be implicitly or explicitly specified. Any attribute which has been explicitly specified is called an explicit attribute. If an attribute has not been specified explicitly, then the attribute takes on the default specification. Such an attribute is known as an implicit attribute.

For example, the usage of a data item need not be specified, in which case a data item's usage is DISPLAY.

Explicit and Implicit SCOPE Terminators

Scope terminators serve to delimit the scope of certain Procedure Division statements. Scope terminators are of two types: explicit and implicit.

The explicit scope terminators are the following:

END-ACCEPT	END-EVALUATE	END-SEARCH
END-ADD	END-IF	END-START
END-CALL	END-MULTIPLY	END-STRING
END-COMPUTE	END-PERFORM	END-SUBTRACT
END-DELETE	END-READ	END-UNSTRING
END-DIVIDE	END-RETURN	END-WRITE
END-EDIT	END-REWRITE	

The implicit scope terminators are the following:

- At the end of any sentence, the separator period which terminates the scope of all previous statements not yet terminated.

- Within any statement containing another statement, the next phrase of the containing statement following the contained statement terminates the scope of any unterminated contained statement. Examples of such phrases are ELSE, WHEN, etc.

EXTERNAL SWITCH

External switches provide communication between the running COBOL program and the workstation and/or other COBOL programs. The operating system provides 16 switches unique for each workstation. In other words, this facility cannot be used to communicate with other workstation or programs not associated with that workstation. These switches may be turned on and off and tested in the COBOL program or in the environment in which the COBOL program is running.

REFERENCE FORMAT

GENERAL DESCRIPTION

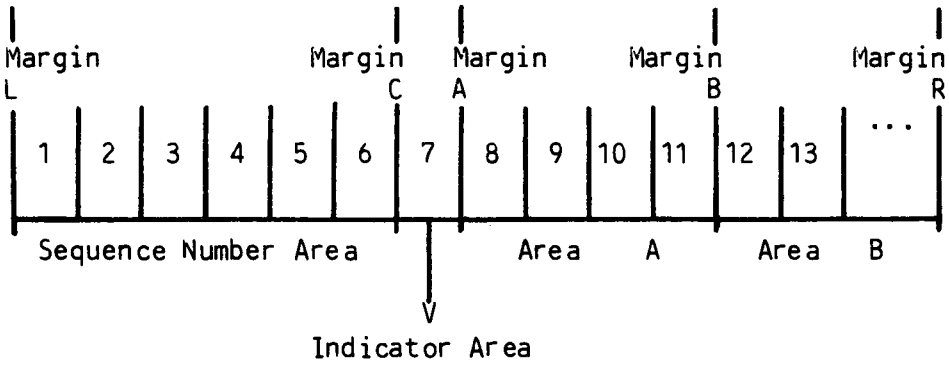
The reference format, which provides a standard method for describing COBOL source programs, is described in terms of character positions in a line on an input-output medium. Within these definitions, the COBOL compiler accepts source programs written in reference format and produces an output listing of the source program input in reference format.

The rules for spacing given in the discussion of the reference format take precedence over all other rules for spacing.

The divisions of a COBOL source program must be ordered as follows: the Identification Division, then the Environment Division, then the Data Division, then the Procedure Division. Each division must be written according to the rules for the reference format.

REFERENCE FORMAT REPRESENTATION

The reference format for a line is represented as follows:



Margin L is immediately to the left of the leftmost character position of a line.

Margin C is between the 6th and 7th character positions of a line.

Margin A is between the 7th and 8th character positions of a line.

Margin B is between the 11th and 12th character positions of a line.

Margin R is immediately to the right of the rightmost character position of a line.

The sequence number area occupies six character positions (1-6), and is between margin L and margin C.

The indicator area is the 7th character position of a line.

Area A occupies character positions 8, 9, 10, and 11, and is between margin A and margin B.

Area B occupies a finite number of character positions: it begins immediately to the right of margin B and terminates immediately to the left of margin R.

Sequence Numbers

A sequence number, consisting of six digits in the sequence area, may be used to label a source program line.

Continuation of Lines

Any sentence, entry, phrase, or clause may be continued by starting subsequent line(s) in area B. These subsequent lines are called the continuation line(s). The line being continued is called the continued line. Any word, literal or PICTURE character-string may be broken in such a way that part of it appears on a continuation line.

A hyphen in the indicator area of a line indicates that the first nonblank character in area B of the current line is the successor of the last non-blank character of the preceding line, excluding intervening comment lines or blank lines, without any intervening space. However, if the continued line contains a nonnumeric literal without closing quotation mark, the first nonblank character in area B of the continuation line must be a quotation mark, and the continuation starts with the character immediately after that quotation mark. All spaces at the end of the continued line are considered part of the literal. Area A of a continuation line must be blank. Note that if the nonnumeric literal is a hexadecimal literal represented by format 2 (see Nonnumeric Literals), spaces at the end of the continued line are illegal.

If there is no hyphen in the indicator area of a line, it is assumed that the first nonblank character in the line is preceded by a space.

Both characters composing the separator '==' must be on the same line.

Blank Lines

A blank line is one that is blank from margin C to margin R, inclusive. A blank line can appear anywhere in the source program. (See above.)

Comment Lines

A comment line is any line with an asterisk or slant in the indicator area of the line. A comment line can appear as any line in a source program after the Identification Division header and as any line in library text. Any combination of the characters from the computer's character set may be included in area A and area B of that line. The asterisk or

slant and the characters in area A and area B will be produced on the listing but serve as documentation only and will not be checked syntactically. The slant in the indicator area causes page ejection prior to printing the comment line in the listing of the source program; an asterisk in the indicator area causes printing of the line at the next available line position in the listing.

Pseudo-Text

The character-strings and separators comprising pseudo-text may start in either area A or area B. If, however, there is a hyphen in the indicator area of a line which follows the opening pseudo-text delimiter, area A of the line must be blank; and the normal rules for continuation of lines apply to the formation of text-word. (See Continuation of Lines.)

DIVISION, SECTION, PARAGRAPH FORMATS

Division Header

The division header must start in area A.

Section Header

The section header must start in area A.

A section consists of zero, one, or more paragraphs in the Environment Division or Procedure Division or zero, one, or more entries in the Data Division.

Paragraph Header, Paragraph-Name, and Paragraph

A paragraph consists of a paragraph-name followed by the separator period and by zero, one, or more sentences, or a paragraph header followed by one or more entries.

The paragraph header or paragraph-name starts in area A of any line following the first line of a division or a section.

The first sentence or entry in a paragraph begins either on the same line as the paragraph header or paragraph-name or in area B of the next nonblank line that is not a comment line. Successive sentences or entries either begin in area B of the same line as the preceding sentence or entry or in area B of the next nonblank line that is not a comment line.

When the sentences or entries of a paragraph require more than one line they may be continued on a subsequent line or lines. (See Continuation of Lines.)

DATA DIVISION ENTRIES

Each Data Division entry begins with a level indicator or a level-number, followed by a space, followed by its associated name (except in the Report Section), followed by a sequence of independent descriptive clauses describing the item. The last clause is always terminated by a separator period.

There are two types of such entries: those which begin with a level indicator and those which begin with a level-number.

In the Data Division, a level indicator is any of the following: FD, SD, RD.

In those entries that begin with a level indicator, the level indicator begins in area A followed by a space and followed in area B with its associated name and appropriate descriptive information.

Those entries that begin with level-numbers are called data description entries.

A level-number has a value taken from the set of values 01, 02, ..., 49, 66, 77, 88. Level-numbers in the range 01, 02, ..., 09 may be written either as a single digit or as a zero followed by a significant digit. At least one space must separate a level-number from the word following the level-number.

In those data description entries that begin with a level-number 01 or 77, the level-number begins in area A followed by a space and followed in area B by its associated record-name or item-name and appropriate descriptive information.

Data description entries may be indented. Any indentation is with respect to margin A. Each new data description entry may begin any number of positions to the right of margin A, except data description entries that begin with level-number 01 or 77 must begin in area A. The extent of indentation is determined only by the width of the physical medium. The entries on the output listing meed be indented only if the input is indented. Indentation does not affect the magnitude of a level-number.

DECLARATIVES

The key word DECLARATIVES and the pair of key words END DECLARATIVES that precede and follow, respectively, the declaratives portion of the Procedure Division must appear on a line by itself. Each must begin in area A and be followed by the separator period.

END PROGRAM HEADER

The end program header must start in area A.

EXAMPLE PROGRAM

The example shows the form of a COBOL program, the definiton of data items and the coding of statements.

The program contains a declarative section, a subroutine call and the Input Output statements referring to an indexed file named WAGEFILE assumed to exist in the present working directory. The subroutine shares part of the data area by the use of the USING clause.

```
   Format Reference Line.
L-----CA---B-------------------------------------------------------------R

       IDENTIFICATION DIVISION.
       PROGRAM-ID.             EZEXMPLE.
       AUTHOR.                 ANON.
       DATE-WRITTEN.
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.        M-40.
       OBJECT-COMPUTER.        M-40.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
      *
      * The file organization and access method is specified, the COBOL
      * file name PERSDATA is associated to the file WAGEFILE.
      *
       SELECT   PERSDATA   ASSIGN TO  "WAGEFILE"
           ORGANIZATION IS  INDEXED    ACCESS MODE IS  RANDOM
           RECORD KEY IS   RECNUMBER
           ALTERNATE KEY IS   NAME   WITH   DUPLICATES
           FILE STATUS IS PERSDATA-STAT.
      *
       DATA DIVISION.
       FILE SECTION.
      *
      * The files record structure is described.
      *
       FD   PERSDATA
            RECORD CONTAINS 128 CHARACTERS  LABEL RECORD IS STANDARD.
      *
       01  PERSDAT.
           02  RECNUMBER        PICTURE   X(4).
           02  NAME             PICTURE   X(20).
           02  ADDRESS          PICTURE   X(30).
           02  DIVI             PICTURE   9(4).
      *
```

```
* The following part of the record area is word aligned and is
* shared with the subroutine by use of the USING clause.
*
     02  TAXDATA.
       03  RATE          PICTURE   S9(7)  COMP.
       03  HOURS         PICTURE   S9(5)  COMP.
       03  TAXID         PICTURE    9(3).
       03  WAGE          PICTURE   S9(12).
*
* Redefinition makes the area an elementary item for the Call
*
     02  TAXDATAR  REDEFINES  TAXDATA  PIC X(22).
*
 WORKING-STORAGE SECTION.
* The constants and data items used are specified.
*
 01   PERSDATA-STAT     PICTURE    X(2).
*
 01   ANSWER            PICTURE    X(3).
     88   YES               VALUE     "YES".
     88   YE                VALUE     "YE".
     88   Y                 VALUE     "Y".
*
 PROCEDURE DIVISION.
*
 DECLARATIVES.
* The error routine entered when an invalid key is encountered
* on execution of the REWRITE statement.
*
 BADKEY SECTION.
     USE AFTER EXCEPTION PROCEDURE ON I-O.
 WRONGKEY.
     DISPLAY  "I/O ERROR  -  STATUS = "  PERSDATA-STAT  "  FOR RECO
-     RD -    NAME
     STOP RUN.
 END DECLARATIVES.
*
 PROCONE SECTION.
*  Open the file and check the status.
*
 LABL1.
     OPEN  I-O  PERSDATA.
     IF PERSDATA-STAT NOT EQUAL TO ZERO
     DISPLAY "OPEN ERROR -" PERSDATA-STAT "ON FILE PERSDATA"
     GO TO FINITO.
*
 PROCTWO SECTION.
* Input a key value and READ the record locking it.
*
 LABL2.
     DISPLAY  "ENTER NUMBER".
     ACCEPT  RECNUMBER.
     READ  PERSDATA  WITH LOCK  KEY IS  RECNUMBER
     INVALID KEY  DISPLAY "INVALID KEY PLEASE RE-"  GO TO PROCTWO.
     DISPLAY "ENTER NUMBER OF HOURS ".
     ACCEPT  HOURS.
```

```
      * CALL a subroutine passing some parameters.
      *
           CALL "TAXCALC" USING  TAXDATAR.
      *
      * Rewrite the data to the file if invalid key occurs then the USE
      * routine will be executed.
      *
           REWRITE  PERSDAT.
           DISPLAY  NAME  "UPDATED ".
      *
       PROCTHREE SECTION.
      * Ask the use for his requirements and check answer.
      *
       LABL3.
           DISPLAY  "DO YOU WISH TO CONTINUE"
           ACCEPT  ANSWER.
           IF  YES  GO TO   PROCTWO
           ELSE  GO TO   PROGEND.
      *
       PROGEND.
      * Close the file implicitly releasing all records.
      *
           CLOSE  PERSDATA.
      *
      * End of program execution.
      *
       FINITO.
           STOP RUN.

  Format Reference Line
L-----CA---B-------------------------------------------------------------R

       IDENTIFICATION DIVISION.
       PROGRAM-ID.            TAXCALC.
      *
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *
      * The data internal to the subroutine.
      *
       01 TAXING               PICTURE   9(3).
           88  TAX1              VALUE   250.
           88  TAX2              VALUE   350.
           88  TAX3              VALUE   450.
      *
       01  TAXRATE.
           02  TAXR1             PICTURE  S9(4)  VALUE  2500.
           02  TAXR2             PICTURE  S9(4)  VALUE  2000.
           02  TAXR3             PICTURE  S9(4)  VALUE  1500.
           02  TAXFIL            PICTURE  S9(12)  COMP.
           02  TAXCOMP           PICTURE  S9(4)   COMP.
```

```
      *
       LINKAGE SECTION.
      *
      *  The Common data area shared by use of the USING clause.
      *
       01  TDATA.
           02  RATE              PICTURE  S9(7)   COMP.
           02  HOURS             PICTURE  S9(5)   COMP.
           02  TAXNR             PICTURE   9(3).
           02  PAY               PICTURE  S9(12)   COMP.
      *
       PROCEDURE DIVISION USING TDATA.
       TP0.
           MOVE  TAXNR  TO  TAXING.
      *
           IF  TAX1  MOVE  TAXR1  TO  TAXCOMP
           PERFORM  PAYCOMP  GO TO  SUBEX.
      *
           IF  TAX2  MOVE  TAXR2  TO  TAXCOMP
           PERFORM  PAYCOMP  GO TO  SUBEX.
      *
           IF  TAX3  MOVE  TAXR3  TO  TAXCOMP
           PERFORM  PAYCOMP  GO TO  SUBEX.
      *
           DISPLAY  "-**- TAXID  NUMBER IN ERROR _***_ TAXCALC PROGRAM RU
      -     N ENDED .
           STOP RUN.
      *
       PAYCOMP.
      *
      * The Pay and Tax amount are calculated.
      *
           COMPUTE  PAY  ROUNDED = HOURS * RATE
            ON  SIZE  ERROR  DISPLAY "OVERFLOW IGNORED .
      *
           COMPUTE  TAXFIL  ROUNDED = PAY * TAXCOMP / 10000
            ON  SIZE  ERROR  DISPLAY "OVERFLOW IGNORED .
           SUBTRACT  TAXFIL  FROM  PAY.
      *
       SUBEX.
      *
      * Return to the calling program.
      *
           EXIT PROGRAM.
      *
      * Inform the compiler that the nested program
      * compilation is complete
      *
       END PROGRAM TAXCALC.
       END PROGRAM EZEXMPLE.
      *
L-----CA---B-----------------------------------------------------------R
  Format reference line.
```

3. IDENTIFICATION AND ENVIRONMENT DIVISIONS

IDENTIFICATION DIVISION

GENERAL DESCRIPTION

The Identification Division identifies the program. The Identification Division is required in a COBOL source program. In addition, the user may include the date the program is written and such other information as desired under the paragraphs in the general format shown below.

Organization

Paragraph headers identify the type of information contained in the paragraph. The name of the program must be given in the first paragraph, which is the PROGRAM-ID paragraph. The other paragraphs are optional and may be included in this division at the user's choice, in the order of presentation shown by the general format below.

Structure

The following is the general format of the paragraphs in the Identification Division and it defines the order of presentation in the source program. The PROGRAM-ID paragraph and the DATE-COMPILED paragraph are on the following pages, while the other paragraphs are not defined, each general format is formed in the same manner.

General Format

IDENTIFICATION DIVISION.

PROGRAM-ID. program-name.

[AUTHOR. [comment-entry] ...]

[INSTALLATION. [comment-entry] ...]

[DATE-WRITTEN. [comment-entry] ...]

[DATE-COMPILED. [comment-entry] ...]

[SECURITY. [comment-entry] ...]

Syntax Rules

1. The comment-entry may be any combination of the characters from the computer's character set. The continuation of the comment-entry by the use of the hyphen in the indicator area is not permitted; however, the comment-entry may be contained on one or more lines.

THE PROGRAM-ID PARAGRAPH

Function

The PROGRAM-ID paragraph specifies the name by which a program is identified.

General Format

PROGRAM-ID. program-name.

Syntax Rules

1. The program-name must conform to the rules for formation of a user-defined word.

General Rules

1. The program-name identifies the source program and all listings pertaining to a particular program.

THE DATE-COMPILED PARAGRAPH

Function

The DATE-COMPILED paragraph provides the compilation date in the Identification Division source program listing.

General Format

DATE-COMPILED. [comment-entry] ...

Syntax Rules

1. The comment-entry may be any combination of the characters from the computer's character set. The continuation of the comment-entry by the use of the hyphen in the indicator area is not permitted; however, the comment-entry may be contained on one or more lines.

General Rules

1. The paragraph-name DATE-COMPILED causes the current date to be inserted during program compilation. If a DATE-COMPILED paragraph is present, it is replaced during compilation with a paragraph of the form:

    DATE-COMPILED. current date.

ENVIRONMENT DIVISION

GENERAL DESCRIPTION

The Environment Division specifies a standard method of expressing those aspects of a data processing problem that are dependent upon the physical characteristics of a specific computer. The Environment Division is optional in a COBOL source program.

The general format of the Environment Division is shown below.

```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

[SOURCE-COMPUTER.  [source-computer-entry] ]

[OBJECT-COMPUTER.  [object-computer-entry] ]

[SPECIAL-NAMES.  [special-names-entry] ]

INPUT-OUTPUT SECTION.

FILE-CONTROL. {file-control-entry} ...

[I-O-CONTROL. [input-output-control-entry]]
```

CONFIGURATION SECTION

The Configuration Section is located in the Environment Division of a source program. The Configuration Section deals with the characteristics of the source computer and the object computer. This section also provides a means for specifying the currency sign; choosing the decimal point; relating implementor-names to user-specified mnemonic-names; and relating alphabet-names to character sets or collating sequences. The Configuration Section is optional in the Environment Division of a COBOL source program.

The general format of the Configuration Section is shown below.

CONFIGURATION SECTION.

[SOURCE-COMPUTER. [source-computer-entry]]

[OBJECT-COMPUTER. [object-computer-entry]]

[SPECIAL-NAMES. [special-names-entry]]

The Configuration Section must not be stated in a program which is contained directly or indirectly within another program.

The entries explicitly or implicitly stated in the Configuration Section of a program which contains other programs apply to each contained program.

THE SOURCE-COMPUTER PARAGRAPH

Function

The SOURCE-COMPUTER paragraph provides a means of describing the computer upon which the program is to be compiled.

General Format

SOURCE-COMPUTER. [computer-name.]

Syntax Rules

1. Computer-name must be one of the following system-names:

   M-30
   M-40
   M-60

General Rules

1. All clauses of the SOURCE-COMPUTER paragraph apply to the program in which they are explicitly or implicitly specified and to any program contained within that program.

2. When the SOURCE-COMPUTER paragraph is not specified and the program is not contained within a program including a SOURCE-COMPUTER paragraph, the computer on which the source program is being compiled is the source computer.

3. When the SOURCE-COMPUTER paragraph is specified, but the source-computer-entry is not specified, the computer upon which the source program is being compiled is the source computer.

THE OBJECT-COMPUTER PARAGRAPH

Function

The OBJECT-COMPUTER paragraph provides a means of describing the computer on which the program is to be executed.

General Format

```
OBJECT-COMPUTER.  [computer-name

    [PROGRAM COLLATING SEQUENCE IS alphabet-name-1] . ]
```

Syntax Rules

1. Computer-name must be one of the following system-names:

   M-30
   M-40
   M-60

General Rules

1. All clauses of the OBJECT-COMPUTER paragraph apply to the program in which they are explicitly or implicitly specified and to any program contained within that program.

2. When the OBJECT-COMPUTER paragraph is not specified and the program is not contained within a program including an OBJECT-COMPUTER paragraph, the computer on which the source program is being compiled is the object computer.

3. When the OBJECT-COMPUTER paragraph is specified, but the object-computer entry is not specified, the computer on which the source program is being compiled is the object computer.

4. The computer-name combined with compiler control information provides a means for identifying equipment configuration.

5. If the PROGRAM COLLATING SEQUENCE clause is specified, the program collating sequence is the collating sequence associated with alphabet-name-1 specified in that clause.

6. If the PROGRAM COLLATING SEQUENCE clause is not specified, the program collating sequence is the native collating sequence.

7. The program collating sequence established in the OBJECT-COMPUTER paragraph is used to determine the truth value of any nonnumeric comparisons that are:

a) Explicitly specified in relation conditions. (See Relation Condition.)

b) Explicitly specified in condition-name conditions. (See Condition-Name Condition.)

c) Implicitly specified by the presence of a CONTROL clause in a report description entry. (See The CONTROL Clause.)

8. The program collating sequence established in the OBJECT-COMPUTER paragraph is applied to any nonnumeric merge or sort keys unless the COLLATING SEQUENCE phrase of the respective MERGE or SORT statement is specified. (See The MERGE Statement, and The SORT Statement.)

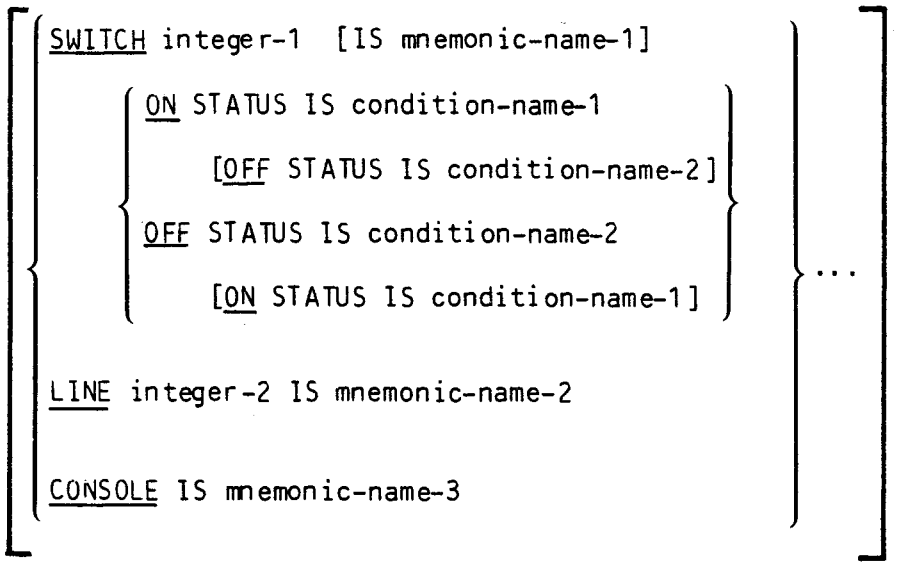
THE SPECIAL-NAMES PARAGRAPH

Function

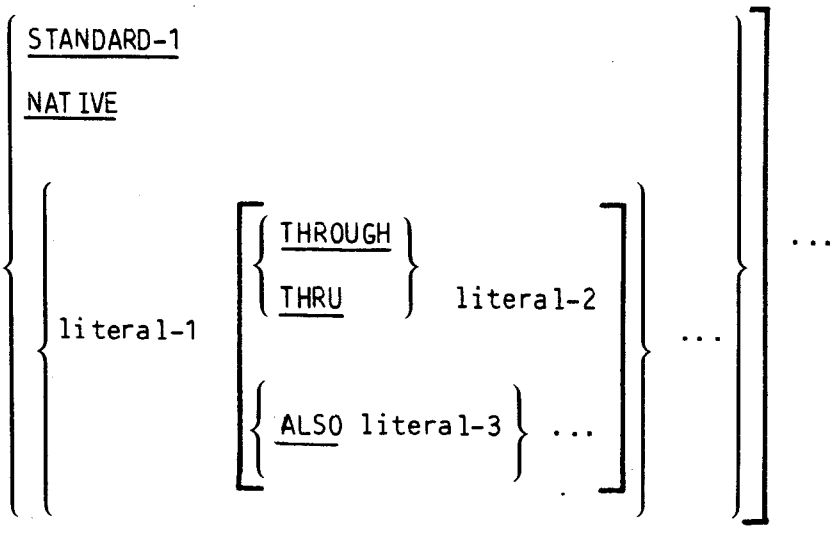
The SPECIAL-NAMES paragraph provides a means for specifying the currency sign; choosing the decimal point; relating implementor-names to user-specified mnemonic-names; and for relating alphabet-names to character sets or collating sequences.

General Format

SPECIAL-NAMES. [



[ALPHABET alphabet-name-1 IS



```
CLASS class-name is

{ literal-4 [ { THROUGH } literal-5 ] } ... ] ...
              { THRU    }

[CURRENCY SIGN IS literal-6]

[DECIMAL-POINT IS COMMA] . ]
```

Syntax Rules

1. If the SWITCH clause is specified:

   a) Integer-1 may have a value between 1 and 16.

   b) Mnemonic-name-1 may be specified only in the SET statement.

2. If the LINE clause is specified:

   a) Integer-2 may have a value between 1 and 999.

   b) Mnemonic-name-2 may be specified only in the WRITE statement.

3. If CONSOLE is specified, mnemonic-name-3 may be specified only in the ACCEPT and DISPLAY statements.

4. If the literal phrase of the ALPHABET clause is specified, a given character must not be specified more than once in an ALPHABET clause.

5. The literals specified in the literal phrase of the ALPHABET clause:

   a) If numeric, must be unsigned integers and must have a value within the range of 1 through the maximum number of characters in the native character set (256).

   b) If nonnumeric and associated with a THROUGH phrase, must each be one character in length.

6. The words THRU and THROUGH are equivalent.

7. The literal specified in the literal-4 phrase:

   a) If numeric, must be unsigned integers and must have a value within the range of one through the maximum number of characters in the native character set (256).

b) If nonnumeric and associated with a THROUGH phrase, must each be one character in length.

8. Literal-6 must not be a figurative constant.

General Rules

1. All clauses specified in the SPECIAL-NAMES paragraph for a program also apply to programs contained within that program. The condition-names specified in the containing program's SPECIAL-NAMES paragraph may be referenced from any contained program.

2. If the SWITCH clause is specified, the on status and/or off status of each switch may be associated with condition-names. The status of that switch may be interrogated by testing these condition-names (see Switch-Status Condition).

3. If the SWITCH clause is specified, the status of the switch may be altered by execution of a format 3 SET statement which specifies as its operands the mnemonic-name associated with that switch (see The SET Statement).

4. If the LINE clause is specified, mnemonic-name-2 may be specified in a WRITE statement. In this case integer-2 represents an absolute line number. Execution of the WRITE statement causes an advance to this line number (or the next logical page if the page position is past the indicated line number) before or after the record is written.

5. The ALPHABET clause provides a means for relating a name to a specified character code set and/or collating sequence. When alphabet-name-1 is referenced in the PROGRAM COLLATING SEQUENCE clause (see The OBJECT-COMPUTER Paragraph) or the COLLATING SEQUENCE phrase of a SORT or MERGE statement, (see The MERGE Statement, and The SORT Statement), the ALPHABET clause specifies a collating sequence. When alphabet-name-1 is referenced in a CODE-SET clause in a file description entry (see The CODE-SET Clause), the ALPHABET clause specifies a character code set.

   a) If the STANDARD-1 phrase is specified, the character code set or collating sequence identified is that defined in American National Standard X3.4-1977, Code for Information Interchange. Each character of the standard character set is associated with and identical to its corresponding character of the native character set.

   b) If the NATIVE phrase is specified, the native character code set or native collating sequence is used.

   c) If the literal phrase is specified, the alphabet-name may not be referenced in a CODE-SET clause (see The CODE-SET Clause). The collating sequence identified is that defined according to the following rules:

- The value of each literal specifies:

  (a) The ordinal number of a character within the native character set, if the literal is numeric. This value must not exceed the value which represents the number of characters in the native character set (256).

  (b) The actual character within the native character set, if the literal is nonnumeric. If the value of the nonnumeric literal contains multiple characters, each character in the literal, starting with the leftmost character, is assigned successive ascending positions in the collating sequence being specified.

- The order in which the literals appear in the alphabet-name clause specifies, in ascending sequence, the ordinal number of the character within the collating sequence being specified.

- Any characters within the native collating sequence, which are not explicitly specified in the literal phrase, assume a position, in the collating sequence being specified, greater than any of the explicitly specified characters. The relative order within the set of these unspecified characters is unchanged from the native collating sequence.

- If the THROUGH phrase is specified, the set of contiguous characters in the native character set beginning with the character specified by the value of literal-1, and ending with the character specified by the value of literal-2, is assigned a successive ascending position in the collating sequence being specified. In addition, the set of contiguous characters specified by a given THROUGH phrase may specify characters of the native character set in either ascending or descending sequence.

- If the ALSO phrase is specified, the characters of the native character set specified by the value of literal-1 and literal-3 are assigned to the same position in the collating sequence being specified.

6. The character that has the highest ordinal position in the program collating sequence specified is associated with the figurative constant HIGH-VALUE. If more than one character has the highest position in the program collating sequence, the last character specified is associated with the figurative constant HIGH-VALUE.

7. The character that has the lowest ordinal position in the program collating sequence specified is associated with the figurative constant LOW-VALUE. If more than one character has the lowest position in the program collating sequence, the first character specified is associated with the figurative constant LOW-VALUE.

8. When specified as literals in the SPECIAL-NAMES paragraph, the figurative constants HIGH-VALUE and LOW-VALUE are associated with those characters having the highest and lowest positions, respectively, in the native collating sequence.

9. The CLASS clause provides a means for relating a name to the specified set of characters listed in that clause. Class-name can be referenced only in a class condition. The characters specified by the values of the literals in this clause define the exclusive set of characters of which this class-name consists. The value of each literal specifies:

   a) The ordinal number of a character within the native character set, if the literal is numeric. This value must not exceed the value which represents the number of characters in the native character set (256).

   b) The actual character within the native character set, if the literal is nonnumeric. If the value of the nonnumeric literal contains multiple characters, each character in the literal is included in the set of characters identified by class-name.

      If the THROUGH phrase is specified, the contiguous characters in the native character set beginning with the character specified by the value of literal-4, and ending with the character specified by the value of literal-5, are included in the set of characters identified by class-name. In addition, the contiguous characters specified by a given THROUGH phrase may specify characters of the native character set in either ascending or descending sequence.

10. Literal-6 which appears in the CURRENCY SIGN clause is used in the PICTURE clause to represent the currency symbol. The literal must be nonnumeric and is limited to a single character. It may be one of the following:

    a) alphabetic characters E, F, G, H, I, K, L, M, N, O, Q, T, U, W, or Y;

    b) special characters '#', '$', '%', '&', ':', '<', '>', '=', '?', '@', '|', '±'.

    If this clause is not present, only the currency sign defined in the COBOL character set may be used in the PICTURE clause.

11. The clause DECIMAL-POINT IS COMMA means that the functions of comma and period are exchanged in the character-string of the PICTURE clause and in numeric literals.

INPUT-OUTPUT SECTION

The Input-Output Section is located in the Environment Division of a source program. The Input-Output Section deals with the information needed to control transmission and handling of data between external media and the object program.

The general format of the Input-Output Section is shown below.

```
INPUT-OUTPUT SECTION.

FILE-CONTROL. {file-control-entry} ...

[I-O-CONTROL. [input-output-control-entry]]
```

THE FILE-CONTROL PARAGRAPH

Function

The FILE-CONTROL paragraph allows specification of file-related information.

General Format

```
FILE-CONTROL. {file-control-entry} ...
```

THE FILE CONTROL ENTRY

Function

The file control entry declares the relevant physical attributes of a file.

General Format

Format 1 - Sequential Organization

```
SELECT [OPTIONAL] file-name-1

                    ┌                                                    ┐
                    │ [TEXT-FILE] {literal-1  }  [SYNC         ]         │
                    │             {data-name-1}  [SYNCHRONIZED ]         │
                    │ SYSPRINT1                                          │
                    │ SYSPRINT2                                          │
  ASSIGN TO         │ SYSPRINT3                                          │
                    │ SYSPRINT4                                          │
                    │ WS-PRINTER                                         │
                    └                                                    ┘

[RESERVE integer-1 [AREA ]]
                   [AREAS]

[ORGANIZATION IS SEQUENTIAL]

[ACCESS MODE IS SEQUENTIAL]

[FILE STATUS IS data-name-2].
```

Format 2 - Relative Organization

```
SELECT [OPTIONAL] file-name-1 ASSIGN TO {literal-1  }  [SYNC         ]
                                        {data-name-1}  [SYNCHRONIZED ]
```

```
[RESERVE integer-1 [AREA ]]
                   [AREAS]

ORGANIZATION IS RELATIVE

[                  {SEQUENTIAL [RELATIVE KEY IS data-name-2]}
ACCESS MODE IS     {{RANDOM   }                              }
                   {{DYNAMIC  } RELATIVE KEY IS data-name-2  }
[FILE STATUS IS data-name-3].
```

Format 3 - Indexed Organization

```
SELECT [OPTIONAL] file-name-1

ASSIGN TO {literal-1 } [SYNC        ]
          {data-name-1}[SYNCHRONIZED]

[RESERVE integer-1 [AREA ]]
                   [AREAS]

ORGANIZATION IS INDEXED

[                  {SEQUENTIAL}
ACCESS MODE IS     {RANDOM    }
                   {DYNAMIC   }

RECORD KEY IS data-name-2

[ALTERNATE RECORD KEY IS data-name-3 [WITH DUPLICATES]] ...

[FILE STATUS IS data-name-4].
```

Syntax Rules

1. The SELECT clause must be specified first in the file control entry. The clauses which follow the SELECT clause may appear in any order.

2. Each file-name in the Data Division must be specified only once in the FILE-CONTROL paragraph. Each file-name specified in the SELECT clause must have a file description entry in the Data Division of the same program.

General Rules

1. The native character set is assumed for data on the external media.

2. For an indexed file, the collating sequence associated with the native character set is assumed. This is the sequence of values of a given key of reference used to process the file sequentially.

3. If the file control block referenced by file-name-1 is an external file control block all file control entries in the run unit which reference this file control block must have:

   a) The same specification for the OPTIONAL and SYNCHRONIZED phrases.

   b) A consistent specification for literal-1, that is a non-numeric literal containing the external file-name.

   c) The same external data item for data-name-1 in the ASSIGN clause.

   d) The same value for integer-1 in the RESERVE clause.

   e) The same organization.

   f) The same access mode.

   g) In format 2 the same external data item for data-name-2 in the RELATIVE KEY phrase.

   h) In format 3 the same data description entry for data-name-2 with the same relative location within the associated record.

   i) In format 3 the same data description entry for data-name-3, the same relative location within the associated record, the same number of alternate record keys, and the same DUPLICATES phrase.

4. In format 3, a maximum of five ALTERNATE RECORD KEY clauses may be specified.

5. The presence of the SYNC or SYNCHRONIZED phrase in the ASSIGN clause indicates that the output to the indicated file is synchronized, that is, the user program receives control when the output operation has terminated. The default is asynchronous output which means that the output operation is requested immediately, but control is returned to the COBOL program before the operation is completed.

Function

The SELECT specifies the name of a file as referred to within a COBOL program and the ASSIGN specifies the association of the file referenced by the SELECT to a storage medium.

General Format

Format 1 Sequential Organization

```
SELECT [OPTIONAL] file-name-1

                    ┌ [TEXT-FILE] { literal-1   } ┌ SYNC         ┐
                    │             { data-name-1 } │ SYNCHRONIZED │
                    │ SYSPRINT1                   └              ┘
                    │ SYSPRINT2
ASSIGN TO           │ SYSPRINT3
                    │ SYSPRINT4
                    │ WS-PRINTER
                    └
```

Format 2 Relative, Indexed Organization

```
SELECT [OPTIONAL] file-name-1

                    { literal-1   } ┌ SYNC         ┐
ASSIGN TO           { data-name-1 } │ SYNCHRONIZED │
                                    └              ┘
```

Syntax Rules

1. Literal-1 must be a non-numeric literal other than a figurative constant.

2. Data-name-1 may be qualified.

3. Data-name-1 must reference an alphanumeric data item.

4. Literal-1 or the content of the data item referenced by data-name-1 at the time of execution of an OPEN statement or DELETE FILE statement referencing file-name-1 specifies the name of the physical device. If the name contains a space character, only the characters up until the first blank are considered to be part of the name. If the

physical file name starts with the character '/', then it is assumed to be a complete path name with an allowed maximum length of fifty-nine characters. Otherwise, literal-1 or the content of the data item referenced by data-name-1 is assumed to be a local name or the name of a file in the working directory, and may not contain the character '/'. In this case, the maximum length allowed is twelve characters.

5. When data-name-1 is specified, the above length restriction applies to the number of characters before the first blank character, if any, or the length of the data item referenced by data-name-1.

General Rules

1. The OPTIONAL phrase applies only to files opened in the INPUT, I-O, or EXTEND mode. Its specification is required for files that are not necessarily present each time the object program is executed.

2. The ASSIGN clause specifies the association of the file referenced by file-name-1 to a storage medium referenced by literal-1, or the content of the data item referenced by data-name-1, SYSPRINT1, SYSPRINT2, SYSPRINT3, SYSPRINT4 or WS-PRINTER.

3. If literal-1 or data-name-1 is specified in the ASSIGN clause, the presence of the SYNC or SYNCHRONIZED phrase indicates that the output to the indicated file is synchronized, that is, the user program receives control when the output operation has terminated. The default is asynchronous output which means that the output operation is requested immediately, but control is returned to the COBOL program before the operation is completed.

Function

The ACCESS MODE clause specifies the order in which records are to be accessed in the file.

General Format

Format 1: - Sequential Organization

```
ACCESS MODE IS  SEQUENTIAL
```

Format 2: - Relative Organization

```
                                  SEQUENTIAL [RELATIVE KEY IS data-name-1]
ACCESS MODE IS  {                 {RANDOM  }
                {                 {        } RELATIVE KEY IS data-name-1
                {                 {DYNAMIC }
                }
```

Format 3: - Indexed Organization

```
                 SEQUENTIAL
ACCESS MODE IS  {RANDOM    }
                {DYNAMIC   }
```

Syntax Rules

Format 2

1. Data-name-1 may be qualified.

2. Data-name-1 must reference an unsigned integer data item whose description does not contain the PICTURE symbol 'P'.

3. Data-name-1 must not be defined in a record description entry associated with that file-name.

4. If a relative file is referenced by a START statement, the RELATIVE KEY phrase within the ACCESS MODE clause must be specified for that file.

General Rules

1. If the ACCESS MODE clause is not specified, sequential access is assumed.

2. If the access mode is sequential, records in the file are accessed in the sequence dictated by the file organization.

   a) For sequential files this sequence is specified by predecessor-successor record relationships established by the execution of WRITE statements when the file is created or extended.

   b) For relative files this sequence is the order of ascending relative record numbers of existing records in the file.

   c) For indexed files this sequence is ascending within a given key of reference according to the collating sequence of the file.

3. If the access mode is random,

   a) In format 2 the value of the relative key data item for relative files indicates the record to be accessed.

   b) In format 3 the value of a record key data item for indexed files indicates the record to be accessed.

4. If the access mode is dynamic, records in the file may be accessed sequentially and/or randomly.

5. All records stored in a relative file are uniquely identified by relative record numbers. This relative record number of a given record specifies the record's logical ordinal position in the file. The first logical record has a relative record number of 1, and subsequent logical records have relative record numbers of 2, 3, 4, .. .

6. In format 2 the data item specified by data-name-1 is used to communicate a relative record number between the user and the L1 MOS file system.

7. In format 2 the relative key data item associated with the execution of an input-output statement is the data item referenced by data-name-1 in the file description entry associated with that input-output statement.

8. If the associated file control block is an external file control block, every file control entry in the run unit which is associated with that file control block must specify the same access mode. In addition for relative access, data-name-1 must reference an external data item and the RELATIVE KEY phrase in each associated file control entry must reference that same external data item in each case.

Function

The ALTERNATE RECORD KEY clause specifies an alternate record key access path to the records in an indexed file.

General Format

ALTERNATE RECORD KEY IS data-name-1 [WITH DUPLICATES]

Syntax Rules

1. Data-name-1 may be qualified.

2. Data-name-1 must be defined as a data item of the category alphanumeric within a record description entry associated with the file-name to which the ALTERNATE RECORD KEY clause is subordinate.

3. Data-name-1 must not reference a group item that contains a variable occurrence data item.

4. Data-name-1 must not reference an item whose leftmost character position corresponds to the leftmost character position of the prime record key or of any alternate record key associated with this file.

5. The size of Data-name-1 must not exceed 100 characters.

General Rules

1. An ALTERNATE RECORD KEY clause specifies an alternate record key for the file with which this clause is associated.

2. The data description of data-name-1 as well as its relative location within a record must be the same as that used when the file was created. The number of alternate record keys for the file must also be the same as that used when the file was created.

3. The DUPLICATES phrase specifies that the value of the associated alternate record key may be duplicated within any of the records in the file. If the DUPLICATES phrase is not specified, the value of the associated alternate record key must not be duplicated among any of the records in the file.

4. If the file has more than one record description entry, data-name-1 need only be described in one of these record description entries. The identical character positions referenced by data-name-1 in any one record description entry are implicitly referenced as keys for all other record description entries of that file.

5. If the associated file control block is an external one, every file description entry in the run unit associated with that file control block must specify the same data description entry for data-name-1, the same relative location within the associated record, the same number of alternate record keys, and the same DUPLICATES phrase.

FILE STATUS clause

Function

The FILE STATUS clause specifies a data item which contains the status of an input-output operation.

General Format

FILE STATUS IS data-name-1

Syntax Rules

1. Data-name-1 may be qualified.

2. Data-name-1 must be defined in the Data Division as a two-character data item of the category alphanumeric and must not be defined in the File Section or Report Section.

General Rules

1. If the FILE STATUS clause is specified, the data item referenced by data-name-1 is always updated to contain the value of the I-O status whenever the I-O status is updated. This value indicates the status of execution of the statement. (See I-O Status.)

2. The data item referenced by data-name-1 which is updated during the execution of an input-output statement is the one specified in the file control entry associated with that statement.

Function

The ORGANIZATION clause specifies the logical structure of a file.

General Format

```
ORGANIZATION IS { SEQUENTIAL }
                { RELATIVE   }
                { INDEXED    }
```

General Rules

1. The file organization is established at the time a file is created and cannot subsequently be changed.

   a) Sequential organization is a permanent logical file structure in which a record is identified by a predecessor-successor relationship established when the record is placed into the file.

   b) Relative organization is a permanent logical file structure in which each record is uniquely identified by an integer value greater than zero, which specifies the record's logical ordinal position in the file.

   c) Indexed organization is a permanent logical file structure in which each record is identified by the value of one or more keys within that record.

2. When the ORGANIZATION clause is not specified, sequential organization is implied.

RECORD KEY clause

Function

The RECORD KEY clause specifies the prime record key access path to the records in an indexed file.

General Format

RECORD KEY IS data-name-1

Syntax Rules

1. Data-name-1 may be qualified.

2. Data-name-1 must reference a data item of the category alphanumeric within a record description entry associated with the file-name to which the RECORD KEY clause is subordinate.

3. Data-name-1 must not reference a group item that contains a variable occurrence data item.

4. The size of data-name-1 may not exceed 100 characters.

General Rules

1. The RECORD KEY clause specifies the prime record key for the file with which this clause is associated. The values of the prime record key must be unique among records of the file.

2. The data description of data-name-1 as well as its relative location within a record must be the same as that used when the file was created.

3. If the file has more than one record description entry, data-name-1 need only be described in one of these record description entries. The identical character positions referenced by data-name-1 in any one record description entry are implicitly referenced as keys for all other record description entries of that file.

4. If the associated file control block is an external file control block, all file description entries in the run unit which are associated with that file control block must specify the same data description entry for data-name-1 with the same relative location within the associated record.

Function

The RESERVE clause allows the user to specify the number of input-output areas allocated. The RESERVE clause is treated as a comment.

General Format

RESERVE integer-1 [AREA / AREAS]

Syntax Rules

1. Integer-1 must be less than 32768.

2. The number of input-output areas used by the MOS Operating System at any one time during execution is dependent upon system configuration and the number of files being used.

THE I-O-CONTROL PARAGRAPH

Function

The I-O-CONTROL paragraph specifies the memory area which is to be shared by different files and the location of files on a multiple file reel.

General Format

```
I-O-CONTROL.

[[SAME [RECORD] AREA FOR file-name-1 {file-name-2} ...] ...

 [MULTIPLE FILE TAPE CONTAINS {file-name-3

       [POSITION IS integer-1]} ...] ... .]
```

Syntax Rules

1. The I-O-CONTROL paragraph is optional.
2. File-name-3 must have sequential organization.
3. The MULTIPLE FILE clause is treated as a comment.

Function

The SAME AREA clause specifies the memory area which is to be shared by different files. This feature is oriented toward saving memory space in the object program as it allows more than one file to share the same file area and alternate areas.

General Format

SAME [RECORD] AREA FOR file-name-1 {file-name-2} ...

Syntax Rules

1. If the SAME AREA clause is used without the RECORD option, not only the file areas but also the alternate areas are shared. This form is treated as a comment.

2. File-name-1 and file-name-2 must be specified in the FILE-CONTROL paragraph of the same program.

3. File-name-1 and file-name-2 may not reference an external file control block.

4. More than one SAME AREA clause may be included in the program; however, file-name-1 must not appear in more than one SAME RECORD AREA clause.

5. The files referenced in the SAME RECORD AREA clause need not all have the same organization or access.

General Rules

1. The SAME RECORD AREA clause specifies that two or more files are to use the same memory area for processing of the current logical record. All of the files may be in the open mode at the same time. A logical record in the SAME RECORD AREA is considered as a logical record of each file open in the output mode whose file-name appears in this SAME RECORD AREA clause and of the most recently read file open in the input mode whose file-name appears in this SAME RECORD AREA clause. This is equivalent to an implicit redefinition of the area, i.e., records are aligned on the leftmost character position.

4. DATA DIVISION

GENERAL DESCRIPTION

The Data Division describes the data that is to be processed by the object program. The Data Division is optional in a COBOL source program.

The general format of the DATA DIVISION is shown below.

```
DATA DIVISION.

FILE SECTION.

[file-description-entry

{record-description-entry} ... ] ...

WORKING-STORAGE SECTION.

[77-level-description-entry
 record-description-entry ] ...

LINKAGE SECTION.

[77-level-description-entry
 record-description-entry ] ...

SCREEN SECTION.

[screen-record-description-entry] ...

REPORT SECTION.

[report-description-entry {report-group-description-entry} ... ] ...
```

NONCONTIGUOUS WORKING STORAGE

Items and constants in working storage which bear no hierarchical relationship to one another need not be grouped into records, provided they do not need to be further subdivided. Instead, they are classified and defined as noncontiguous elementary items. Each of these items is defined in a separate data description entry which begins with the special level-number, 77.

The following data clauses are required in each data description entry:

a. level-number 77
b. data-name
c. the PICTURE clause, the USAGE IS INDEX clause, the USAGE IS COMPUTATIONAL-1 clause, the USAGE IS COMPUTATIONAL-2 clause, the USAGE IS SINGLE-REAL clause or the USAGE IS DOUBLE-REAL clause

Other data description clauses are optional and can be used to complete the description of the item if necessary.

WORKING STORAGE RECORDS

Data elements in working storage which bear a definite hierarchical relationship to one another must be grouped into records according to the rules for formation of record descriptions. Data elements in the Working-Storage Section which bear no hierarchical relationship to any other data item may be described as records which are single elementary items. All clauses which are used in record descriptions in the File Section can be used in record descriptions in the Working-Storage Section.

Record Description Structure

A record description consists of a set of data description entries which describe the characteristics of a particular record. Each data description entry consists of a level-number followed by the data-name or FILLER clause, if specified, followed by a series of independent clauses as required. A record description may have a hierarchical structure and therefore the clauses used with an entry may vary considerably, depending upon whether or not it is followed by subordinate entries. The structure of a record description and the elements allowed in a record description entry are explained in Concept of Levels and The Data Description Entry. The size of the record may not be bigger than 32,766 bytes.

Initial Values

The initial value of any data item in the Working-Storage Section except an index data item is specified by associating the VALUE clause with the data item. The initial value of any index data item or any data item not associated with a VALUE clause is undefined.

CONCEPT OF LEVELS

A level concept is inherent in the structure of a logical record. This concept arises from the need to specify subdivisions of a record for the purpose of data reference. Once a subdivision has been specified, it may be further subdivided to permit more detailed data referral.

The most basic subdivisions of a record, that is, those not further subdivided, are called elementary items; consequently, a record is said to consist of a sequence of elementary items, or the record itself may be an elementary item.

In order to refer to a set of elementary items, the elementary items are combined into groups. Each group consists of a named sequence of one or more elementary items. Groups, in turn, may be combined into groups of two or more groups, etc. Thus, an elementary item may belong to more than one group.

LEVEL-NUMBERS

A system of level-numbers shows the organization of elementary items and group items. Since records are the most inclusive data items, level-numbers for records start at 01. Less inclusive data items are assigned higher (not necessarily successive) level-numbers not greater in value than 49. There are special level-numbers 66, 77, and 88, which are exceptions to this rule (see below). Separate entries are written in the source program for each level-number used.

A group includes all group and elementary items following it until a level-number less than or equal to the level-number of that group is encountered. All items which are immediately subordinate to a given group item must be described using identical level-numbers greater than the level-number used to describe that group item.

Three types of entries exist for which there is no true concept of level. These are:

1. Entries that specify elementary items or groups introduced by a RENAMES clause,

2. Entries that specify noncontiguous working storage and linkage data items,

3. Entries that specify condition-names.

Entries describing items by means of RENAMES clauses for the purpose of regrouping data items have been assigned the special level-number 66.

Entries that specify noncontiguous data items, which are not subdivisions of other items, and are not themselves subdivided, have been assigned the special level-number 77.

Entries that specify condition-names, to be associated with particular values of a conditional variable, have been assigned the special level-number 88.

CONCEPT OF CLASSES OF DATA

There are five categories of data items. (See The PICTURE Clause.) These are grouped into three classes: alphabetic, numeric, and alphanumeric. For alphabetic and numeric, the classes and categories are synonymous. The alphanumeric class includes the categories of alphanumeric edited, numeric edited and alphanumeric (without editing). Every elementary item except for an index data item belongs to one of the classes and further to one of the categories. The class of a group item is treated at object time as alphanumeric regardless of the class of elementary items subordinate to that group item. The following chart depicts the relationship of the class and categories of data items.

LEVEL OF ITEM	CLASS	CATEGORY
Elementary	Alphabetic	Alphabetic
	Numeric	Numeric
	Alphanumeric	Numeric Edited Alphanumeric Edited Alphanumeric
Nonelementary (Group)	Alphanumeric	Alphabetic Numeric Numeric Edited Alphanumeric Edited Alphanumeric

SELECTION OF CHARACTER REPRESENTATION AND RADIX

The value of a numeric item may be represented in either binary or decimal form depending on the equipment. In addition, there are several ways of expressing decimal. Since these representations are combinations of bits, they are commonly called binary-coded decimal forms. The selection of radix is generally dependent upon the arithmetic capability of the computer. Olivetti COBOL language supports four representations of values of numeric items. The selection among these representations is dependent upon the USAGE clause of the associated data description entry.

Representation	Usage
Binary-coded decimal	DISPLAY
Binary	BINARY, WORD-BINARY, COMP, COMP-4
Packed decimal	PACKED-DECIMAL, COMP-3
Floating point	SINGLE-REAL, DOUBLE-REAL, COMP-1, COMP-2

When there is more than one USAGE type for a representation, the various usages indicate differences in sign representation and/or the actual number of character positions required for the internal representation.

The binary-coded decimal form is also used to represent characters and symbols that are alphanumeric items. The selection of the proper binary-coded alphanumeric or binary-coded decimal form is dependent upon the capability of the computer and its external media.

Since the computer provides more than one means of representing data, the binary-coded decimal form is used if not otherwise specified by the data description. If both the external medium and the computer are capable of handling more than one form of data representation, or if there is no external medium associated with the data, the selection is dependent on factors included in USAGE clauses, PICTURE clauses, etc. The method used in selecting the proper data form is also provided to allow the user to anticipate and/or control the selection.

The size of an elementary data item or a group item is the number of characters in standard data format of the item. Synchronization and usage may cause a difference between this size and the actual number of characters required for the internal representation.

ALGEBRAIC SIGNS

Algebraic signs fall into two categories:

1. Operational signs, which are associated with signed numeric data items and signed numeric literals to indicate their algebraic properties.

2. Editing signs, which appear on edited reports to identify the sign of the item.

The SIGN clause permits the programmer to state explicitly the location of the operational sign. The clause is optional; if it is not used, operational signs will be represented depending on the USAGE specified for the data item. If the USAGE is DISPLAY, the operational sign is represented as trailing zoned signed.

Editing signs are inserted into a data item through the use of the sign control symbols of the PICTURE clause.

STANDARD ALIGNMENT RULES

The standard rules for positioning data within an elementary item depend on the category of the receiving item. These rules are:

1. If the receiving data item is described as numeric:

   a) The data is aligned by decimal point and is moved to the receiving character positions with zero fill or truncation on either end as required.

   b) When an assumed decimal point is not explicitly specified, the data item is treated as if it had an assumed decimal point immediately following its rightmost character and is aligned as in paragraph 1a above.

2. If the receiving data item is a numeric edited data item, the data moved to the edited data item is aligned by decimal point with zero fill or truncation at either end as required within the receiving character positions of the data item, except where editing requirements cause replacement of the leading zeros.

3. If the receiving data item is alphanumeric (other than a numeric edited data item), alphanumeric edited or alphabetic, the sending data is moved to the receiving character positions and aligned at the leftmost character position in the data item with space fill or truncation to the right, as required.

If the JUSTIFIED clause is specified for the receiving item, these standard rules are modified (see The JUSTIFIED Clause).

ITEM ALIGNMENT FOR INCREASED OBJECT-CODE EFFICIENCY

Some computer memories are organized in such a way that there are natural addressing boundaries in the computer memory (e.g. word , half-word or byte boundaries). The way in which data is stored is determined by the object program, and need not respect these natural boundaries.

However, certain uses of data (e.g. in arithmetic operations or in subscripting) may be facilitated if the data is stored so as to be aligned on these natural boundaries. Specifically, additional machine operations in the object program may be required for the accessing and storage of data if portions of two or more data items appear between adjacent natural boundaries, or if certain natural boundaries bifurcate a single data item.

Data items which are aligned on these natural boundaries in such a way as to avoid such additional machine operations are defined to be synchronized. A synchronized item is assumed to be introduced and carried in that form; conversion to synchronized form occurs only during the execution of a statement (other than READ or WRITE) which stores data in the item.

On Olivetti M-30 and M-40 computers, the word is the natural boundary. All data is aligned on a byte boundary unless synchronization is requested. When synchronization is requested, the word boundary is used.

Synchronization can be accomplished in two ways:

1. By use of the SYNCHRONIZED clause.

2. By recognizing the appropriate natural boundaries and organizing the data suitably without the use of the SYNCHRONIZED clause. (See The SYNCHRONIZED Clause, General Rule 9).

FILE SECTION

The File Section is located in the Data Division of a source program. The File Section defines the structure of data files. Each file is defined by a file description entry and one or more record description entries. Record description entries are written immediately following the file description entry.

The general format of the File Section is shown below.

```
FILE SECTION.

[file-description-entry

{record-description-entry} ... ] ...
```

FILE DESCRIPTION ENTRY

In a COBOL program the file description entry (FD entry) represents the highest level of organization in the File Section. The File Section header is followed by a file description entry consisting of a level indicator (FD), a file-name, and a series of independent clauses. The clauses of a file description entry (FD entry) specify the size of the logical and physical records, the presence or absence of label records, the names of the records which comprise the file, and finally the number of lines to be written on a logical printer page. The entry itself is terminated by a period.

RECORD DESCRIPTION STRUCTURE

A record description consists of a set of data description entries which describe the characteristics of a particular record. Each data description entry consists of a level-number followed by the data-name or FILLER clause, if specified, followed by a series of independent clauses as required. A record description may have a hierarchical structure and therefore the clauses used with an entry may vary considerably, depending upon whether or not it is followed by subordinate entries. The structure of a record description and the elements allowed in a record description entry are explained in Concept of Levels, and in The Data Description Entry. The size of the record may not be larger than 32,767 bytes.

Initial Values

The initial value of data items in the File Section is undefined.

THE FILE DESCRIPTION ENTRY

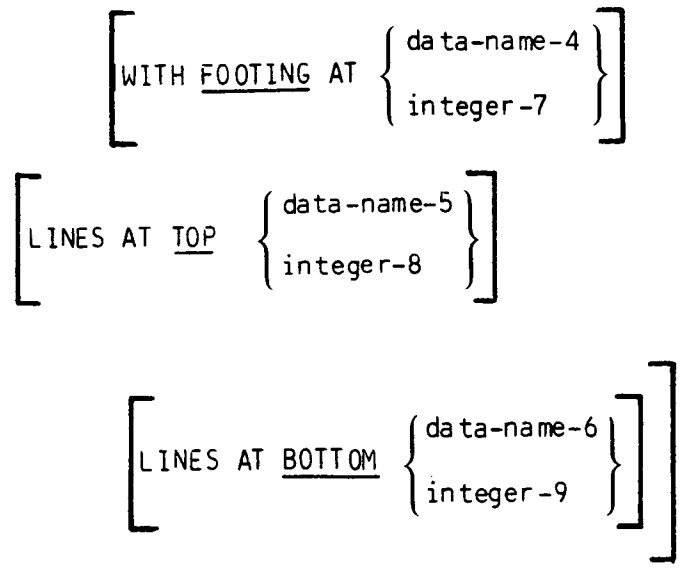
Function

The file description entry furnishes information concerning the physical structure, identification, and record-names pertaining to a file.

General Format

Format 1 - Sequential Organization

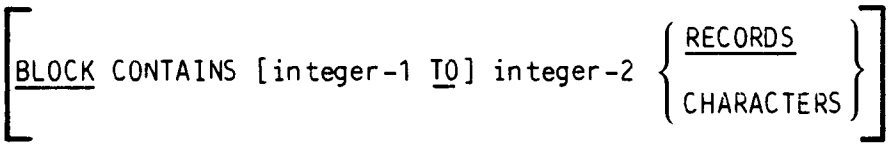
```
FD file-name-1

[BLOCK CONTAINS [integer-1 TO] integer-2 { RECORDS    }]
                                         { CHARACTERS }

[RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

[LABEL { RECORD IS  } { STANDARD }]
       { RECORDS ARE} { OMITTED  }

[VALUE OF {{ ALLOC-UNIT } IS { integer-5   }} ...]
          {{ FILE-SIZE  }    { data-name-1 }}

[DATA { RECORD IS  } {data-name-2} ...]
      { RECORDS ARE}

[LINAGE IS { data-name-3 } LINES]
           { integer-6   }
```



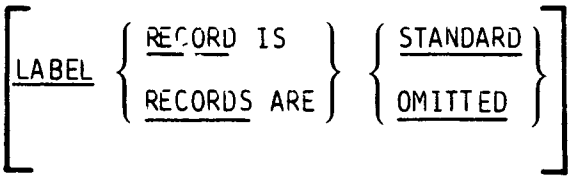
[CODE-SET IS alphabet-name-1].

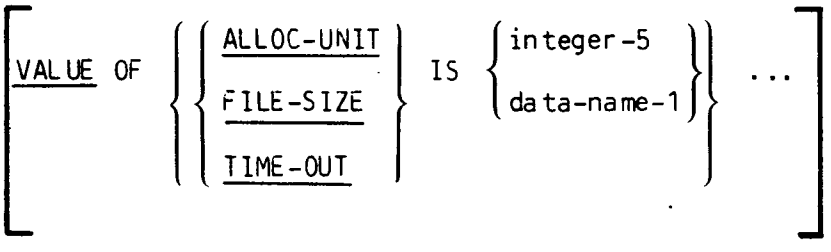
Format 2 - Relative Organization

FD file-name-1



[RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]





```
[DATA {RECORD IS  } {data-name-2} ... ] .
      {RECORDS ARE}
```

Format 3 - Indexed Organization

```
FD file-name-1

[BLOCK CONTAINS [integer-1 TO] integer-2 {RECORDS   }]
                                         {CHARACTERS}

.[RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

[LABEL {RECORD IS  } {STANDARD}]
       {RECORDS ARE} {OMITTED }

[VALUE OF { {ALLOC-UNIT  } IS {integer-5 } } ... ]
          { {FILE-SIZE   }    {data-name-1} }
          { {TIME-OUT    }                  }
          { {PAGE-SIZE   }                  }
          { {SPLIT-STRATEGY}                }

[DATA {RECORD IS  } {data-name-2} ... ] .
      {RECORDS ARE}
```

Syntax Rules

1. The level indicator FD identifies the beginning of a file description entry and must precede file-name-1.

2. The clauses which follow file-name-1 may appear in any order.

3. One or more record description entries must follow the file description entry.

General Rules

1. A file description entry associates file-name-1 with a file control block.

Function

The BLOCK CONTAINS clause specifies the size of a physical record. This clause is treated as a comment.

General Format

BLOCK CONTAINS [integer-1 TO] integer-2 { RECORDS / CHARACTERS }

General Rules

1. If the CHARACTERS phrase is specified, the physical record size is specified in terms of the number of character positions required to store the physical record, regardless of the types of characters used to represent the items within the physical record.

2. If integer-1 is not specified, integer-2 represents the exact size of the physical record. If integer-1 and integer-2 are both specified, they refer to the minimum and maximum size of the physical record, respectively.

3. If the associated file control block is an external file control block, all BLOCK CONTAINS clauses in the run unit which are associated with that file control block must have the same value for integer-1 and integer-2.

CODE-SET clause

Function

The CODE-SET clause specifies the character code convention used to represent data on the external media.

General Format

CODE-SET IS alphabet-name-1

Syntax Rules

1. If the CODE-SET clause is specified for a file, all data in that file must be described as USAGE IS DISPLAY and any signed numeric data must be described with the SIGN IS SEPARATE clause.

2. The alphabet-name clause referenced by the CODE-SET clause must not specify the literal phrase.

General Rules

1. If the CODE-SET clause is specified:

   a) Upon successful execution of an OPEN statement, the character set used to represent the data on the external media is the one referenced by alphabet-name-1 in the file control entry or the file description entry associated with the file-name specified in the OPEN statement. (See The SPECIAL-NAMES Paragraph.)

   b) It specifies the algorithm for converting the character set on the external media from/to the native character set during the execution of an input or output operation.

2. If the CODE-SET clause is not specified, the native character set is assumed for data on the external media.

3. If the associated file control block is an external file control block, all CODE-SET clauses in the run unit which are associated with that file control block must be the same character set and/or collating sequence.

Function

The DATA RECORDS clause serves only as documentation for the names of data records with their associated file. The DATA RECORDS clause is an obsolete element because it is to be deleted from the next revision of American National Standard COBOL.

General Format

```
        { RECORD IS  }
DATA    { RECORDS ARE }  {data-name-1} ...
```

Syntax Rules

1. Data-name-1 is the name of a data record and must have an 01 level-number record description, with the same name, associated with it.

General Rules

1. The presence of more than one data-name indicates that the file contains more than one type of data record. These records may be of differing sizes, different formats, etc. The order in which they are listed is not significant.

2. Conceptually, all data records within a file share the same area. This is in no way altered by the presence of more than one type of data record within the file.

LABEL RECORDS clause

Function

The LABEL RECORDS clause specifies whether labels are present. The LABEL RECORDS clause is an obsolete element because it is to be deleted from the next revision of American National Standard COBOL.

General Format

```
        { RECORD IS  }  { STANDARD }
LABEL   { RECORDS ARE}  { OMITTED  }
```

General Rules

1. OMITTED specifies that no explicit labels exist for the file or the device to which the file is assigned.

2. STANDARD specifies that labels exist for the file or the device to which the file is assigned. For mass storage files, no explicit labels exist. For all other medium, the format of labels is to be defined.

3. If the LABEL RECORDS clause is not specified for a file, STANDARD is the default.

4. If the file control block associated with this file description entry is an external file control block, all LABEL RECORD clauses in the run unit which are associated with this file control block must be the same.

Function

The LINAGE clause provides a means for specifying the depth of a logical page in terms of number of lines. It also provides for specifying the size of the top and bottom margins on the logical page, and the line number, within the page body, at which the footing area begins.

General Format

```
LINAGE IS { data-name-1 } LINES [ WITH FOOTING AT { data-name-2 } ]
          { integer-1   }       [                 { integer-2 } ]

    [ LINES AT TOP    { data-name-3 } ]
    [                 { integer-3   } ]

        [ LINES AT BOTTOM { data-name-4 } ]
        [                 { integer-4   } ]
```

Syntax Rules

1. Data-name-1, data-name-2, data-name-3, data-name-4 must reference unsigned numeric integer data items.

2. Data-name-1, data-name-2, data-name-3, data-name-4 may be qualified.

3. Integer-1 must be less than 32768.

4. Integer-2 must not be greater than integer-1.

5. Integer-3, integer-4 may be zero and must be less than 32768.

6. The LINAGE clause may be specified only if the associated file control entry specified TEXT-FILE, SYSPRINT1, SYSPRINT2, SYSPRINT3, SYSPRINT4, or WS-PRINTER in the ASSIGN clause.

General Rules

1. The LINAGE clause provides a means for specifying the size of a logical page in terms of number of lines. The logical page size is the sum of the values referenced by each phrase except the FOOTING phrase. If the LINES AT TOP or LINES AT BOTTOM phrases are not specified, the values of these items are zero. If the FOOTING phrase is not specified, no end-of-page condition independent of the page overflow condition exists.
   There is not necessarily any relationship between the size of the

logical page and the size of a physical page.

2. Integer-1 or the value of the data item referenced by data-name-1 specifies the number of lines that can be written and/or spaced on the logical page. The value must be greater than zero. That part of the logical page in which these lines can be written and/or spaced is called the page body.

3. Integer-2 or the value of the data item referenced by data-name-2 specifies the line number within the page body at which the footing area begins. The value must be greater than zero and not greater than integer-1 or the value of the data item referenced by data-name-1.

   The footing area comprises the area of the logical page between the line represented by integer-2 or the value of the data item referenced by data-name-2 and the line represented by integer-1 or the value of the data item referenced by data-name-1, inclusive.

4. Integer-3 or the value of the data item referenced by data-name-3 specifies the number of lines that comprise the top margin on the logical page. The value may be zero.

5. Integer-4 or the value of the data item referenced by data-name-4 specifies the number of lines that comprise the bottom margin on the logical page. The value may be zero.

6. Integer-1, integer-3, and integer-4, if specified, are used at the time the file is opened by the execution of an OPEN statement with the OUTPUT phrase, to specify the number of lines that comprise each of the indicated sections of a logical page. Integer-2, if specified, is used at that time to define the footing area. These values are used for all logical pages written for the file during a given execution of the program.

7. The values of the data items referenced by data-name-1, data-name-3, and data-name-4, if specified, are used as follows:

   a) The values of the data items, at the time an OPEN statement with the OUTPUT phrase is executed for the file, are used to specify the number of lines that are to comprise each of the indicated sections for the first logical page.

   b) The values of the data items, at the time a WRITE statement with the ADVANCING PAGE phrase is executed or a page overflow condition occurs are used to specify the number of lines that are to comprise each of the indicated sections for the next logical page. (See The WRITE Statement.)

8. The value of the data item referenced by data-name-2, if specified, at the time an OPEN statement with the OUTPUT phrase is executed for the file, is used to define the footing area for the first logical page. At the time a WRITE statement with the ADVANCING PAGE phrase is executed or a page overflow condition occurs, it is used to define the footing area for the next logical page.

9. A LINAGE-COUNTER is generated by the presence of a LINAGE clause. The value in the LINAGE-COUNTER at any given time represents the line number at which the device is positioned within the current page body. The rules governing the LINAGE-COUNTER are as follows:

   a) A separate LINAGE-COUNTER is supplied for each file described in the File Section whose file description entry contains a LINAGE clause.

   b) LINAGE-COUNTER may be referenced only in Procedure Division statements: however, only the input-output control system may change the value of LINAGE-COUNTER. Since more than one LINAGE-COUNTER may exist in a program, the user must qualify LINAGE-COUNTER by file-name when necessary.

   c) LINAGE-COUNTER is modified, according to the following rules, during the execution of a WRITE statement to an associated file:

      - When the ADVANCING PAGE phrase of the WRITE statement is specified, the LINAGE-COUNTER is reset to the value one. During the resetting of LINAGE-COUNTER it's value is implicitly incremented to exceed the value specified by integer-1 or the data-item referenced by data-name-1.

      - When the ADVANCING identifier-2 or integer-1 phrase of the WRITE statement is specified, the LINAGE-COUNTER is incremented by integer-1 or the value of the data item referenced by identifier-2.

      - When the ADVANCING phrase of the WRITE statement is not specified, the LINAGE-COUNTER is incremented by the value one. (See The WRITE Statement.)

      - The value of LINAGE-COUNTER is reset to one when the device is repositioned to the first line that can be written on for each of the succeeding logical pages. (See The WRITE Statement.)

   d) The value of LINAGE-COUNTER is set to one at the time an OPEN statement is executed for the associated file.

10. Each logical page is contiguous to the next with no additional spacing provided.

11. If the file control block associated with this file description entry is an external file control block, all file description entries in the run unit which are associated with this file control block must have:

   a) A LINAGE clause, if any file description entry has a LINAGE clause.

   b) The same corresponding values for integer-1, integer-2, integer-3, and integer-4, if specified.

c) The same corresponding external data items referenced by data-name-1, data-name-2, data-name-3, and data-name-4.

It is the user's responsibility to check the above consistencies.

Function

The RECORD CONTAINS clause specifies the size of data records.

General Format

Format 1

RECORD CONTAINS integer-1 CHARACTERS

Format 2

RECORD CONTAINS [integer-2 TO] integer-3 CHARACTERS

Syntax Rules

1. Integer-1, integer-2, and integer-3 may not be greater than 32767.

2. In format 1 no record description entry for the file may specify a number of character positions greater than integer-1.

General Rules

1. If the RECORD clause is not specified, the size of each data record is completely defined in the record description entry.

2. If the associated file control block is an external file control block, all file description entries in the run unit which are associated with that file control block must specify the same values for integer-1. If the RECORD clause is not specified, all record description entries associated with this file control block must be the same length.

3. Format 1 is used to specify fixed length records. Integer-1 specifies the number of character positions contained in each record in the file.

4. When format 2 of the RECORD clause is used, integer-2 and integer-3 refer to the minimum number of characters in the smallest size data record and the maximum number of characters in the largest size data record, respectively. However, in this case, the size of each data record is completely defined in the record description entry.

5. In format 2 the size of each data record is specified in terms of the number of character positions required to store the logical record, regardless of the types of characters used to represent the items within the logical record. The size of a record is determined by the sum of the number of characters in all fixed length elementary items plus the sum of the maximum number of characters in any variable length item subordinate to the record. This sum may be different

from the actual size of the record; see Selection of Character Representation and Radix; The SYNCHRONIZED Clause; The USAGE Clause. The actual size of the record written is always the maximum number of characters in the data record with the largest size.

Function

The VALUE OF clause is used to specify the initial allocation size of a file and the size of any extensions made to the file.

For relative and indexed files it is also used to specify the amount of time to wait for a locked file.

Furthermore, for indexed files, it may be used to specify various attributes of the key files.

General Format

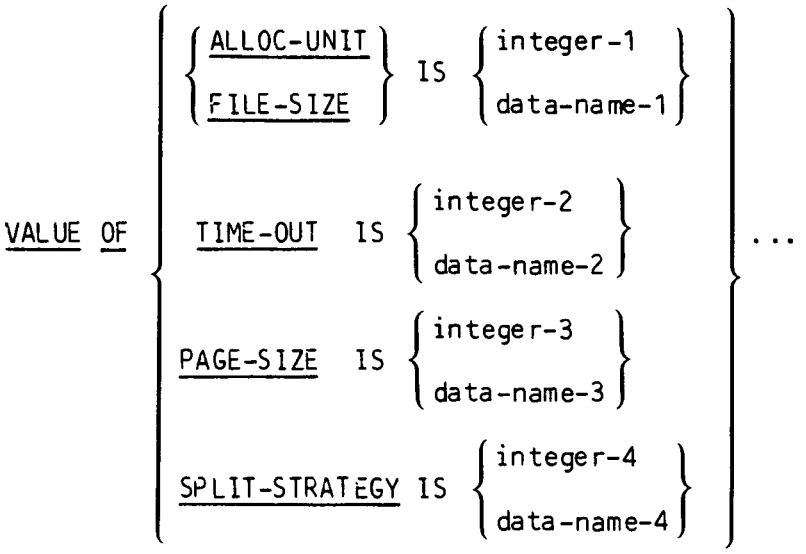
Format 1 - Sequential Organization

```
VALUE OF { { ALLOC-UNIT } IS { integer-1     } } ...
         { { FILE-SIZE  }    { data-name-1   } }
```

Format 2 - Relative Organization

```
                { { ALLOC-UNIT } IS { integer-1   } }
                { { FILE-SIZE  }    { data-name-1 } }
VALUE OF        {                                   } ...
                { TIME-OUT IS { integer-2   }       }
                {             { data-name-2 }       }
```

Format 3 - Indexed Organization



Syntax Rules

1. No phrase may appear more than once in a VALUE OF clause.

2. Data-name-1, data-name-2, data-name-3 and data-name-4 must reference unsigned numeric integer data items.

3. Data-name-1, data-name-2, data-name-3 and data-name-4 may be qualified.

4. Integer-1 or the value of the data item referenced by data-name-1 must not be greater than 2,147,483,647.

5. Integer-1 or the value of the data item referenced by data-name-1 may not be less than the length of the record for the file.

6. Integer-2 or the value of the data item referenced by data-name-2 must not by greater than 15.

7. Integer-3 or the value of the data item referenced by data-name-3 may be 1, 2, 4 or 8.

8. Integer-4 or the value of the data item referenced by data-name-4 may be 50, 90 or 100.

General Rules

Formats 1, 2 and 3

1. The FILE-SIZE phrase is used at open time to determine the number of bytes initially allocated for the data record portion of the file if the file is to be created by the execution of the OPEN statement. The

user should take into consideration that the allocation for the primary and alternate key portion are not included in this value. The phrase meaningless if the file already exists.

2. If the FILE-SIZE phrase is not specified, ten blocks (5120 bytes) is used except when the record size of the file is greater than 5120 bytes in which case the record size is the default value.

3. If the requested amount is not available, and the file is a text-file, the maximum available is allocated.

4. The ALLOC-UNIT phrase is used during the execution of a WRITE statement when the file's current allocation is used up. Integer-1 or the value at open time of the data item referenced by data-name-1 is the number of bytes by which the allocation of the data record portion is extended, if available.

   It is recommended that the ALLOC-UNIT integer be less than FILE-SIZE. However it should be noted that since the execution of an OPEN OUTPUT statement for a file already existing truncates the file, initial allocation of the file in this case is done using ALLOC-UNIT upon the execution of the first WRITE statement.

5. If the ALLOC-UNIT phrase is not specified, the specified or default value for FILE-SIZE is used.

Formats 2 and 3

6. The TIME-OUT phrase is used during the execution of an OPEN statement for a file which has been placed in lock mode by another system program or process. Integer-2 or the value at open time of the data item referenced by data-name-2 is used to determine how long to wait for the resource to become available before resulting in an I-O error. Zero implies no wait. A specification of 15 causes a wait for an indefinite period. Otherwise, the value represents the number of seconds to wait.

7. If the TIME-OUT phrase is not specified, zero is used.

Format 3

8. The PAGE-SIZE phrase is used at open time to determine the number of blocks (512 bytes per block) per page for the file containing the primary keys and the files containing the alternate keys, if any, if the file or files are to be created by the execution of the OPEN statement. The phrase is meaningless if the file or files already exists.

9. If the PAGE-SIZE phrase is not specified, 2 is used.

10. The SPLIT-STRATEGY phrase is used at open time to define the page splitting strategy for pages of the primary key and alternate key files when they become full. Integer-4 or the value at open time of the data item referenced by data-name-4 may be 50 (for a 50-50 split), 90 (for a 90-10 split), or 100 (for a 100-0) split.

11. If the SPLIT-STRATEGY phrase is not specified, 50 is used.

WORKING-STORAGE SECTION

The Working-Storage Section is located in the Data Division of a source program. The Working-Storage Section describes records and subordinate data items which are not part of data files.

The Working-Storage Section is composed of the section header, followed by record description entries and/or data description entries for noncontiguous data items.

The general format of the Working-Storage Section is shown below.

```
WORKING-STORAGE SECTION.

[77-level-description-entry  ]
[record-description-entry    ] ...
```

General Rules

1. Each record name and noncontiguous item name must be unique since it may not be qualified.

LINKAGE SECTION

The Linkage Section is located in the Data Division of a source program. The Linkage Section appears in the called program and describes data items that are to be referred to by the calling program and the called program.

The Linkage Section in a program is meaningful if and only if the object program is to function under the control of a CALL statement, and the CALL statement in the calling program contains a USING phrase.

The Linkage Section is used for describing data that is available through the calling program but is to be referred to in both the calling and the called program. A complete description is given in the Inter-Program Communication chapter.

The structure of the Linkage Section is the same as that of the Working-Storage Section, beginning with a section header, followed by noncontiguous data items and/or record description entries.

The general format of the Linkage Section is shown below.

```
LINKAGE SECTION.

[77-level-description-entry  ]
[record-description-entry    ] ...
```

General Rules

1. Each record name and noncontiguous item name must be unique since it may not be qualified.

THE SCREEN SECTION

The Screen Section is located in the Data Division of a source program. The Screen Section describes records and subordinate data items which are part of Screen record.

The Screen Section is composed of the section header, followed by record description entries and/or data description entries for the screen records.

The general format of the Screen Section is shown below.

```
SCREEN SECTION.

[screen-record-description-entry] ...
```

For a description of the Screen Section refer to chapter 6 Screen Handling.

THE REPORT SECTION

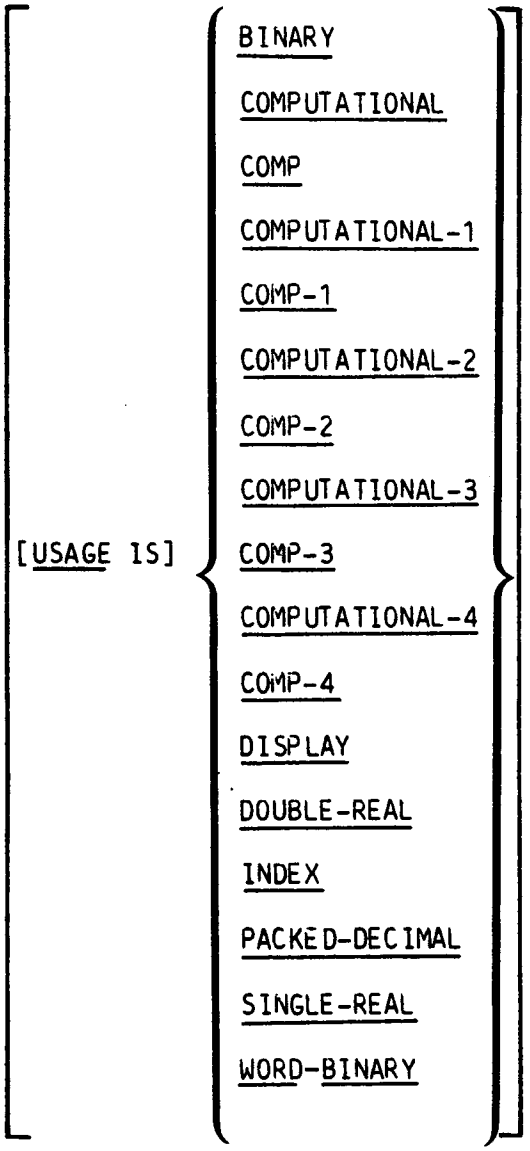
The Report Section is located in the Data Division of a source program. The Report Section describes records and subordinate data items which are part of data files.

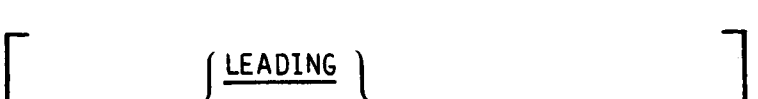
The Report Section is composed of the section header, followed by record description entries.

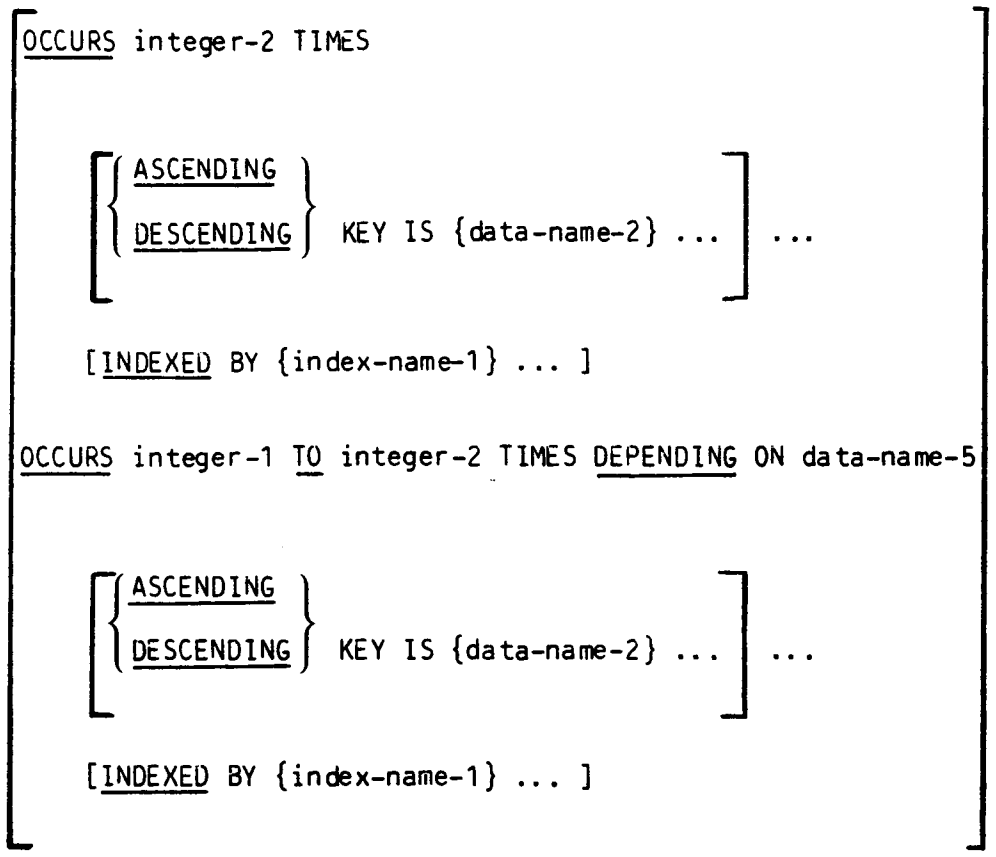
The general format of the Report Section is shown below.

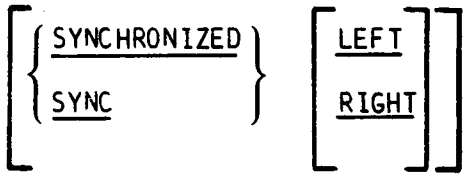
```
REPORT SECTION.

[report-description-entry {report-group-description-entry} ... ] ...
```

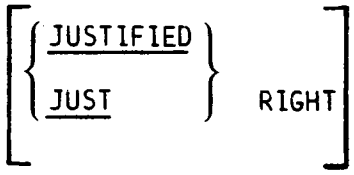
For a complete description of the Report Section refer to chapter 11 Report Writer.


[USAGE IS]
BINARY
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
COMPUTATIONAL-3
COMP-3
COMPUTATIONAL-4
COMP-4
DISPLAY
DOUBLE-REAL
INDEX
PACKED-DECIMAL
SINGLE-REAL
WORD-BINARY


[SIGN IS]
LEADING
TRAILING
[SEPARATE CHARACTER]







[BLANK WHEN ZERO]

[VALUE IS literal-1].

Format 2:

```
66 data-name-1 RENAMES data-name-2 [ { THROUGH } data-name-3 ] .
                                   { THRU    }
```

Format 3:

```
88 condition-name-1

{ VALUE IS   } { literal-1 [ { THROUGH } literal-2 ] } ... .
{ VALUES ARE }             { THRU    }
```

Syntax Rules

1. Level-number in format 1 may be any number from 01 through 49 or 77.

2. In format 1, the data-name-1 or FILLER clause, if specified, must immediately follow the level-number. The REDEFINES clause, if specified, must immediately follow the data-name-1 or FILLER clause if either is specified; otherwise, it must immediately follow the level-number. The remaining clauses may be written in any order.

3. The PICTURE clause must be specified for every elementary item except an index data item and the subject of the RENAMES clause, in which case use of this clause is prohibited.

4. The words THRU and THROUGH are equivalent.

General Rules

1. The clauses SYNCHRONIZED, PICTURE, JUSTIFIED, and BLANK WHEN ZERO must not be specified except for an elementary data item.

2. Format 3 is used for each condition-name. Each condition-name requires a separate entry with level-number 88. Format 3 contains the name of the condition and the value, values, or range of values associated with the condition-name. The condition-name entries for a particular conditional variable must immediately follow the entry describing the item with which the condition-name is associated. A condition-name can be associated with any data description entry which contains a level-number except the following:

a) Another condition-name.

b) A level 66 item.

c) A group containing items with descriptions including JUSTIFIED, SYCHRONIZED, or USAGE (other than USAGE IS DISPLAY).

d) An index data item.

3. Multiple level 01 entries subordinate to any given level indicator other than the level indicator RD for a report description entry, represent implicit redefinitions of the same area.

Function

The BLANK WHEN ZERO clause permits the blanking of an item when its value is zero.

General Format

BLANK WHEN ZERO

Syntax Rules

1. The BLANK WHEN ZERO clause can be used only for an elementary item whose PICTURE is specified as numeric or numeric edited. (See The PICTURE Clause.)

2. The numeric or numeric edited data description entry to which the BLANK WHEN ZERO clause applies must be described, whether implicitly or explicitly, as USAGE IS DISPLAY.

General Rules

1. When the BLANK WHEN ZERO clause is used, the item will contain nothing but spaces if the value of the item is zero.

2. When the BLANK WHEN ZERO clause is used for an item whose PICTURE is numeric, the category of the item is considered to be numeric edited.

DATA-NAME or FILLER clause

Function

A data-name specifies the name of the data being described. The key word FILLER may be used to specify a data item which is not referenced explicitly.

General Format

```
[ data-name-1 ]
[ FILLER      ]
```

Syntax Rules

1. In the File, Working-Storage, Linkage and Screen Sections, data-name-1 or the key word FILLER, if either is specified, must be the first word following the level-number in each data description entry.

General Rules

1. If this clause is omitted, the data item being described it treated as though FILLER had been specified.

2. The key word FILLER may be used to name a data item in a record. Under no circumstances can a FILLER item be referred to explicitly. However, the key word FILLER may be used to name a conditional variable because such use does not require explicit reference to the data item itself, but only to its value.

Function

The JUSTIFIED clause permits alternative positioning of data within a receiving data item.

General Format

```
{JUSTIFIED}
{JUST     }  RIGHT
```

Syntax Rules

1. The JUSTIFIED clause can be specified only at the elementary item level.

2. JUST is an abbreviation for JUSTIFIED.

3. The JUSTIFIED clause cannot be specified for any data item described as numeric or for which editing is specified.

4. The JUSTIFIED clause must not be specified for an index data item.

General Rules

1. When a receiving data item is described with the JUSTIFIED clause and the sending data item is larger than the receiving data item, the leftmost characters are truncated. When the receiving data item is described with the JUSTIFIED clause and it is larger than the sending data item, the data is aligned at the rightmost character position in the data item with space fill for the leftmost character positions.

2. When the JUSTIFIED clause is omitted, the standard rules for aligning data within an elementary item apply (see Standard Alignment Rules).

LEVEL-NUMBER description

Function

The level-number shows the hierarchy of data within a logical record. In addition, it is used to identify entries for working storage items, linkage items, condition-names, and the RENAMES clause.

General Format

level-number

Syntax Rules

1. A level-number is required as the first element in each data description entry.

2. Data description entries subordinate to an FD or SD entry must have level-numbers with the values 01 through 49, 66, or 88.

3. Data description entries in the Working-Storage Section and Linkage Section must have level-numbers 01 through 49, 66, 77, or 88.

General Rules

1. The level-number 01 identifies the first entry in each record description.

2. Special level-numbers have been assigned to certain entries where there is no real concept of hierarchy:

   a) Level-number 77 is assigned to identify noncontiguous working storage data items, noncontiguous linkage data items, and can be used only as described by format 1 of the data description entry. (See The Data Description Entry.)

   b) Level-number 66 is assigned to identify RENAMES entries and can be used only as described in format 2 of the data description entry. (See The Data Description Entry.)

   c) Level-number 88 is assigned to entries which define condition-names associated with a conditional variable and can be used only as described in format 3 of the data description entry. (See The Data Description Entry.)

3. Multiple level 01 entries subordinate to any given level indicator other than the level indicator RD for a report description entry, represent implicit redefinitions of the same area.

Function

The OCCURS clause eliminates the need for separate entries for repeated data items and supplies information required for the application of subscripts or indices.

General Format

Format 1:

```
OCCURS integer-2 TIMES

    [ {ASCENDING }                                    ]
    [ {DESCENDING} KEY IS {data-name-2} ... ]  ...

    [INDEXED BY {index-name-1} ... ]
```

Format 2:

```
OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1

    [ {ASCENDING }                                    ]
    [ {DESCENDING} KEY IS {data-name-2} ... ]  ...

    [INDEXED BY {index-name-1} ... ]
```

Syntax Rules

1. The OCCURS clause must not be specified in a data description entry that:

   a) Has a level-number of 01, 66, 77, or 88, or

   b) Has a variable occurrence data item subordinate to it.

2. Data-name-1 and data-name-2 may be qualified.

3. The first specification of data-name-2 must be the name of either the entry containing the OCCURS clause or an entry subordinate to the

entry containing the OCCURS clause. Subsequent specification of data-name-2 name entries subordinate to the entry containing the OCCURS clause.

4. Data-name-2 must be specified without the subscripting normally required.

5. Where both integer-1 and integer-2 are used, integer-1 must be greater than or equal to zero and integer-2 must be greater than integer-1.

6. Data-name-1 must describe an integer.

7. In format 2, the data item defined by data-name-1 must not occupy a character position within the range of the first character position defined by the data description entry containing the OCCURS clause and the last character position defined by the record description entry containing that OCCURS clause.

8. If the OCCURS clause is specified in a data description entry included in a record description entry containing the EXTERNAL clause, data-name-1, if specified, must reference a data item possessing the external attribute, which is described in the same Data Division.

9. If the OCCURS clause is specified in a data description entry subordinate to one containing the GLOBAL clause, data-name-1, if specified, must be a global name and must reference a data item which is described in the same Data Division.

10. A data description entry that contains format 2 of the OCCURS clause may only be followed, within that record description, by data description entries which are subordinate to it.

11. Data items identified by the data-names in the KEY IS phrase must not contain an OCCURS clause except when data-name-2 is the subject of the entry.

12. There must not be an entry that contains an OCCURS clause between the descriptions of the data items identified by the data-names in the KEY IS phrase and the subject of the entry.

13. An INDEXED BY phrase is required if the subject of this entry, or an entry subordinate to this entry, is to be referenced by indexing. The index-name identified by this clause is not defined elsewhere since its allocation and format are dependent on the hardware and, not being data, cannot be associated with any data hierarchy.

14. Index-name-1 must be a unique word within the program.

15. If format 2 is specified in a record description entry associated with a file and the data item referenced by data-name-1 is defined in the same file, it must be in the same record.

General Rules

1. Except for the OCCURS clause itself, all data description clauses associated with an item whose description includes an OCCURS clause apply to each occurrence of the item described.

2. The number of occurrences of the subject entry is defined as follows:

   a) In format 1, the value of integer-2 represents the exact number of occurrences.

   b) In format 2, the current value of the data item referenced by data-name-1 represents the number of occurrences.

      This format specifies that the subject of this entry has a variable number of occurrences. The value of integer-2 represents the maximum number of occurrences and the value of integer-1 represents the minimum number of occurrences. This does not imply that the length of the subject of the entry is variable, but that the number of occurrences is variable.

      The value of the data item referenced by data-name-1 must fall within the range integer-1 through integer-2. Reducing the value of the data item referenced by data-name-1 makes the contents of data items, whose occurrence numbers now exceed the value of the data item referenced by data-name-1, undefined.

3. When a group data item, having subordinate to it an entry that specifies format 2 of the OCCURS clause, is referenced, the part of the table area used in the operation is determined as follows:

   a) If the data item referenced by data-name-1 is outside the group, only that part of the table area that is specified by the value of the data item referenced by data-name-1 at the start of the operation will be used.

   b) If the data item referenced by data-name-1 is included in the same group and the group data item is referenced as a sending item, only that part of the table area that is specified by the value of the data item referenced by data-name-1 at the start of the operation will be used in the operation. If the group is a receiving item, the maximum length of the group will be used.

4. When the KEY phrase is specified, the repeated data must be arranged in ascending or descending order according to the values contained in data-name-2. The ascending or descending order is determined according to the rules for the comparison of operands. (See Comparison of Numeric Operands, and Comparison of Nonnumeric Operands.) The data-names are listed in their descending order of significance.

PICTURE clause

Function

The PICTURE clause describes the general characteristics and editing requirements of an elementary item.

General Format

$$\left\{\begin{array}{l}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{array}\right\} \text{ IS character-string}$$

Syntax Rules

1. A PICTURE clause can be specified only at the elementary item level.

2. A character-string consists of certain allowable combinations of characters in the COBOL character set used as symbols. The allowable combinations determine the category of the elementary item.

3. The maximum number of characters allowed in character-string is 30.

4. The PICTURE clause must be specified for every elementary item except an index data item or the subject of the RENAMES clause. In these cases the use of this clause is prohibited.

5. PIC is an abbreviation for PICTURE.

6. The asterisk when used as the zero suppression symbol and the clause BLANK WHEN ZERO may not appear in the same entry.

General Rules

1. There are five categories of data that can be described with a PICTURE clause: alphabetic, numeric, alphanumeric, alphanumeric edited, and numeric edited.

2. To define an item as alphabetic:

   a) Its PICTURE character-string can only contain the symbols 'A', 'B'; and

   b) Its content when represented in standard data format must be one or more alphabetic characters and the space.

3. To define an item as numeric:

   a) Its PICTURE character-string can only contain the symbols '9', 'P', 'S', and 'V'. The number of digit positions that can be described by the PICTURE character-string must range from 1 to 18 inclusive; and

   b) If unsigned, its content when represented in standard data format must be one or more numeric characters; if signed, the item may

b) If unsigned, its content when represented in standard data format must be one or more numeric characters; if signed, the item may also contain a '+', '-', or other representation of an operational sign. (See The SIGN Clause.)

4. To define an item as alphanumeric:

   a) Its PICTURE character-string is restricted to certain combinations of the symbols 'A', 'X', '9', and the item is treated as if the character-string contained all X's. A PICTURE character-string which contains all A's or all 9's does not define an alphanumeric item. The number of character positions that can be described by the PICTURE character-string must range from 1 to 32,766; and

   b) Its content when represented in standard data format are allowable characters in the computer's character set.

5. To define an item as alphanumeric edited:

   a) Its PICTURE character-string is restricted to certain combinations of the following symbols: 'A', 'X', '9', 'B', '0', and '/'; and must contain at least one 'A' or 'X' and must contain at least one 'B' or '0' (zero) or '/' (slant). The number of character positions that can be described by the PICTURE character-string, including the editing characters, must range from 1 to 32,766; and

   b) Its content when represented in standard data format are allowable characters in the computer's character set.

6. To define an item as numeric edited:

   a) Its PICTURE character-string is restricted to certain combinations of the symbols 'B', '/', 'P', 'V', 'Z', '0', '9', ',', '.', '*', '+', '-', 'CR', The allowable combinations are determined from the order of precedence of symbols and the editing rules; and

      - The number of digit positions that can be represented in the PICTURE character-string must range from 1 to 18 inclusive; and

      - The character-string must contain at least one

   b) The content of each of the character positions must be consistent with the corresponding PICTURE symbol.

7. The size of an elementary item, where size means the number of character positions occupied by the elementary item in standard data format, is determined by the number of allowable symbols that represent character positions. An unsigned nonzero integer which is enclosed in parentheses following the symbols 'A', ',', 'X', '9', 'P', 'Z', '*', 'B', '/', '0', '+', '-', or the currency symbol indicates the number of consecutive occurrences of the symbol. Note that the following symbols may appear only once in a given PICTURE: 'S', 'V',

8. The function of the symbols used to describe an elementary item are explained as follows:

A Each 'A' in the character-string represents a character position which can contain only a letter of the alphabet or a space.

B Each 'B' in the character-string represents a character position into which the space character will be inserted.

P Each 'P' in the character-string indicates an assumed decimal scaling position and is used to specify the location of an assumed decimal point when the point is not within the number that appears in the data item. The scaling position character 'P' is not counted in the size of the data item. Scaling position characters are counted in determining the maximum number of digit positions (18) in numeric edited items or numeric items. The scaling position character 'P' can appear only as a continuous string of 'P's in the leftmost or rightmost digit positions within a PICTURE character-string; since the scaling position character 'P' implies an assumed decimal point (to the left of 'P's if 'P's are leftmost PICTURE characters and to the right if 'P's are rightmost PICTURE characters), the assumed decimal point symbol 'V' is redundant as either the leftmost or rightmost character within such a PICTURE description. The symbol 'P' and the insertion character '.' (period) cannot both occur in the same PICTURE character-string.

If, in any operation involving conversion of data from one form of internal representation to another, the data item being converted is described with the PICTURE character 'P', each digit position described by a 'P' is considered to contain the value zero, and the size of the data item is considered to include the digit positions so described.

S The 'S' is used in a character-string to indicate the presence, but neither the representation nor, necessarily, the position of an operational sign; it must be written as the leftmost character in the PICTURE. The 'S' is not counted in determining the size (in terms of standard data format characters) of the elementary item unless the entry is subject to a SIGN clause which specifies the optional SEPARATE CHARACTER phrase. (See The SIGN Clause.)

V The 'V' is used in a character-string to indicate the location of the assumed decimal point and may only appear once in a character-string. The 'V' does not represent a character position and therefore is not counted in the size of the elementary item. When the assumed decimal point is to the right of the rightmost symbol in the string the 'V' is redundant.

X Each 'X' in the character-string is used to represent a character position which contains any allowable character from the computer's character set.

Z Each 'Z' in a character-string may only be used to represent the leftmost leading numeric character positions which will be replaced by a space character when the content of that character position is a leading zero. Each 'Z' is counted in the size of the item.

9 Each '9' in the character-string represents a character position which contains a numeral and is counted in the size of the item.

0 Each '0' (zero) in the character-string represents a character position into which the numeral zero will be inserted. The '0' is counted in the size of the item.

/ Each '/' (slant) in the character-string represents a character position into which the slant character will be inserted. The '/' is counted in the size of the item.

, Each ',' (comma) in the character-string represents a character position into which the character ',' will be inserted. The ',' is counted in the size of the item. The insertion character ',' must not be the last character in the PICTURE character-string.

. When the symbol '.' (period) appears in the character-string it is an editing symbol which represents the decimal point for alignment purposes and, in addition, represents a character position into which the character '.' will be inserted. The character '.' is counted in the size of the item. For a given program the functions of the period and comma are exchanged if the clause DECIMAL-POINT IS COMMA is stated in the SPECIAL-NAMES paragraph. In this exchange the rules for the period apply to the comma and the rules for the comma apply to the period wherever they appear in a PICTURE clause.

+, -, CR, DB These symbols are used as editing sign control symbols. When used, they represent the character position into which the editing sign control symbol will be placed. The symbols are mutually exclusive in any one character-string and each character used in the symbol is counted in determining the size of the data item.

***** Each '*' (asterisk) in the character-string represents a leading numeric character position into which an asterisk will be placed when the content of that position is zero. Each '*' is counted in the size of the item.

cs The currency symbol in the character-string represents a character position into which a currency symbol is to be placed. The currency symbol in a character-string is represented by either the currency sign or by the single character specified in the CURRENCY SIGN clause in the SPECIAL-NAMES paragraph. The currency symbol is counted in the size of the item.

Editing Rules

1. There are two general methods of performing editing in the PICTURE clause, either by insertion or by suppression and replacement. There are four types of insertion editing available. They are:

    a. Simple insertion
    b. Special insertion
    c. Fixed insertion
    d. Floating insertion

    There are two types of suppression and replacement editing:

a. Zero suppression and replacement with spaces
b. Zero suppression and replacement with asterisks

2. The type of editing which may be performed upon an item is dependent upon the category to which the item belongs. The following table specifies which type of editing may be performed upon a given category:

CATEGORY	TYPE OF EDITING
Alphabetic	Simple insertion 'B' only
Numeric	None
Alphanumeric	None
Alphanumeric Edited	Simple insertion '0', 'B' and '/'
Numeric Edited	All, subject to rules in rule 3 below

3. Floating insertion editing by zero suppression and replacement are mutually exclusive in a PICTURE clause. Only one type of replacement may be used with zero suppression in a PICTURE clause.

4. Simple insertion editing. The ',' (comma), 'B' (space), '0' (zero), and '/' (slant) are used as the insertion characters. The insertion characters are counted in the size of the item and represent the position in the item into which the character will be inserted. If the insertion character ',' (comma) is the last symbol in the PICTURE character-string, the PICTURE clause must be the last clause of the data description entry and must be immediately followed by the separator period. This results in the combination of ',.' appearing in the data description entry, or, if the DECIMAL POINT IS COMMA clause is used, in two consecutive periods.

5. Special insertion editing. The '.' (period) is used as the insertion character. In addition to being an insertion character it also represents the decimal point for alignment purposes. The insertion character used for the actual decimal point is counted in the size of the item. The use of the assumed decimal point, represented by the symbol 'V' and the actual decimal point, represented by the insertion character, in the same PICTURE character-string is disallowed. If the insertion character is the last symbol in the PICTURE character-string, the PICTURE clause must be the last clause of that data description entry and must be immediately followed by the separator period. This results in two consecutive periods appearing in the data description entry, or in the combination of ',.' if the DECIMAL POINT IS COMMA clause is used. The result of special insertion editing is the appearance of the insertion character in the item in the same position as shown in the character-string.

6. Fixed insertion editing. The currency symbol and the editing sign control symbols, '+', '-', 'CR', 'DB', are the insertion characters. Only one currency symbol and only one of the editing sign control

symbols can be used in a given PICTURE character-string. When the symbols 'CR' or 'DB' are used they represent two character positions in determining the size of the item and they must represent the rightmost character positions that are counted in the size of the item. The symbol '+' or '-', when used, must be either the leftmost or rightmost character position to be counted in the size of the item. The currency symbol must be the leftmost character position to be counted in the size of the item except that it can be preceded by either a '+' or a '-' symbol. Fixed insertion editing results in the insertion character occupying the same character position in the edited item as it occupied in the PICTURE character-string. Editing sign control symbols produce the following results depending upon the value of the data item:

EDITING SYMBOL IN PICTURE CHARACTER-STRING	RESULT	
	DATA ITEM POSITIVE OR ZERO	DATA ITEM NEGATIVE
+	+	-
-	space	-
CR	2 spaces	CR
DB	2 spaces	DB

7. Floating insertion editing. The currency symbol and editing sign control symbols '+' or '-' are the floating insertion characters and as such are mutually exclusive in a given PICTURE character-string.

   Floating insertion editing is indicated in a PICTURE character-string by using a string of at least two of the floating insertion characters. This string of floating insertion characters may contain any of the simple insertion characters or have simple insertion characters immediately to the right of this string. These simple insertion characters are part of the floating string.

   The leftmost character of the floating insertion string represents the leftmost limit of the floating symbol in the data item. The rightmost character of the floating string represents the rightmost limit of the floating symbols in the data item.

   The second floating character from the left represents the leftmost limit of the numeric data that can be stored in the data item. Non-zero numeric data may replace all the characters at or to the right of this limit.

   In a PICTURE character-string, there are only two ways of representing floating insertion editing. One way is to represent any or all of the leading numeric character positions on the left of the decimal point by the insertion character. The other way is to represent all of the numeric character positions in the PICTURE character-string by the insertion character.

If the insertion characters are only to the left of the decimal point in the PICTURE character-string, the result is that a single floating insertion character will be placed into the character position immediately preceding either the decimal point or the first nonzero digit in the data represented by the insertion symbol string, whichever is the leftmost in the PICTURE character-string. The character positions preceding the insertion character are replaced with spaces.

If all numeric character positions in the PICTURE character-string are represented by the insertion character, at least two of the insertion characters must be to the left of the decimal point.

When the floating insertion character is the editing control symbol '+' or '-', the character inserted depends upon the value of the data item:

EDITING SYMBOL IN PICTURE CHARACTER-STRING	RESULT	
	DATA ITEM POSITIVE OR ZERO	DATA ITEM NEGATIVE
+	+	-
-	space	-

If all numeric character positions in the PICTURE character-string are represented by the insertion character, the result depends upon the value of the data. If the value is zero the entire data item will contain spaces. If the value is not zero, the result is the same as when the insertion character is only to the left of the decimal point.

To avoid truncation, the minimum size of the PICTURE character-string for the receiving data item must be the number of characters in the sending data item, plus the number of non-floating insertion characters being edited into the receiving data item, plus one for the floating insertion character. If truncation does occur, the value of the data that is used for editing is the value after truncation. (See Standard Alignment Rules.)

8. Zero suppression editing. The suppression of leading zeroes in numeric character positions is indicated by the use of the alphabetic character 'Z' or the character '*' (asterisk) as suppression symbols in a PICTURE character-string. These symbols are mutually exclusive in a given PICTURE character-string. Each suppression symbol is counted in determining the size of the item. If 'Z' is used the replacement character will be the space and if the asterisk is used, the replacement character will be '*'.

   Zero suppression and replacement is indicated in a PICTURE character-string by using a string of one or more of the allowable symbols to represent leading numeric character positions which are to be replaced when the associated character position in the data contains a leading zero. Any of the simple insertion characters

embedded in the string of symbols or to the immediate right of this string are part of the string.

In a PICTURE character-string, there are only two ways of representing zero suppression. One way is to represent any or all of the leading numeric character positions to the left of the decimal point by suppression symbols. The other way is to represent all of the numeric character positions in the PICTURE character-string by suppression symbols.

If the suppression symbols appear only to the left of the decimal point, any leading zero in the data which corresponds to a symbol in the string is replaced by the replacement character. Suppression terminates at the first nonzero digit in the data represented by the suppression symbol string or at the decimal point, whichever is encountered first.

If all numeric character positions in the PICTURE character-string are represented by suppression symbols and the value of the data is not zero the result is the same as if the suppression characters were only to the left of the decimal point. If the value is zero and the suppression symbol is 'Z', the entire data item will be spaces. If the value is zero and the suppression symbol is '*', the entire data item including any insertion editing symbols except for the actual decimal point will by '*'. In this case, the actual decimal point will appear in the data item.

9. The symbols '+', '-', '*', 'Z', and the currency symbol, when used as floating replacement characters, are mutually exclusive within a given character-string.

Precedence Rules

The following chart shows the order of precedence when using characters as symbols in a character-string. An 'X' at an intersection indicates that the symbol(s) at the top of the column may precede, in a given character-string, the symbol(s) at the left of the row. Arguments appearing in braces { } indicate that the symbols are mutually exclusive. The currency symbol is indicated by the symbol 'cs'.

At least one of the symbols 'A', 'X', 'Z', '9' or '*', or at least two of the symbols '+', '-' or 'cs' must be present in a PICTURE string.

Non-floating insertion symbols '+' and '-', floating insertion symbols 'Z', '*', '+', '-', and 'cs', and other symbol 'P' appear twice in the following PICTURE character precedence chart. The leftmost column and uppermost row for each symbol represents its use to the left of the decimal point position. The second appearance of the symbol in the chart represents its use to the right of the decimal point position.

*1 Symbol * 2 * Symbol *		Non-floating Insertion Symbol									Floating Insertion Symbol						Other Symbol					
		B	0	/	,	.	+ -	+ -	CR DB	cs	Z *	Z *	+ -	+ -	cs	cs	-9	A X	S	V	P	P
Non Floating Insertion Symbol	B	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x		x		x
	0	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x		x		x
	/	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x		x		x
	,	x	x	x	x	x	x			x	x	x	x	x	x	x	x			x		x
	.	x	x	x	x		x			x	x		x		x		x					
	+-																					
	+-	x	x	x	x	x				x	x	x			x	x	x			x	x	x
	CR DB	x	x	x	x	x				x	x	x			x	x	x			x	x	x
	cs						x															
Floating Insertion Symbol	Z*	x	x	x	x		x			x	x											
	Z*	x	x	x	x	x	x			x	x	x								x		x
	+-	x	x	x	x					x			x									
	+-	x	x	x	x	x				x			x	x						x		x
	cs	x	x	x	x		x								x							
	cs	x	x	x	x	x	x								x	x				x		x
Other Symbol	9	x	x	x	x	x	x			x	x		x		x		x	x	x	x		x
	AX	x	x	x													x	x				
	S																					
	V	x	x	x	x		x			x	x		x		x		x		x		x	
	P	x	x	x	x		x			x	x		x		x		x		x		x	
	P						x			x									x	x		x

Picture character precedence chart.

Function

The REDEFINES clause allows the same computer storage area to be described by different data description entries.

General Format

```
level-number  [ data-name-1 ]  REDEFINES data-name-2
              [ FILLER      ]
```

NOTE: Level-number, data-name-1, and FILLER are shown in the above format to improve clarity. Level-number, data-name-1, and FILLER are not part of the REDEFINES clause.

Syntax Rules

1. The REDEFINES clause, when specified, must immediately follow the subject of the entry.

2. The level-numbers of data-name-2 and the subject of the entry must be identical, but must not be 66 or 88.

3. This clause must not be used in level 01 entries in the File Section.

4. The data description entry for data-name-2 cannot contain an OCCURS clause. However, data-name-2 may be subordinate to an item whose data description entry contains an OCCURS clause. In this case, the reference to data-name-2 in the REDEFINES clause may not be subscripted. Neither the original definition nor the redefinition can include a variable occurrence data item.

5. If the data item referenced by data-name-2 is either declared to be an external data record or is specified with a level-number other than 01, the number of character positions it contains must be greater than or equal to the number of character positions in the data items referenced by the subject of this entry. If the data-name referenced by data-name-2 is specified with a level-number of 01 and is not declared to be an external data record, there is no such constraint.

6. Data-name-2 must not be qualified even if it is not unique since no ambiguity of reference exists in this case because of the required placement of the REDEFINES clause within the source program.

7. Multiple redefinitions of the same character positions are permitted. Multiple redefinitions of the same character positions must all use the data-name of the entry that originally defined the area.

8. The entries giving the new description of the character positions must not contain any VALUE clauses, except in condition-name entries.

9. No entry having a level-number numerically lower than the level-number of data-name-2 and the subject of the entry may occur between the data description entries of data-name-2 and the subject of the entry.

10. The entries giving the new descriptions of the character positions must follow the entries defining the area of data-name-2, without intervening entries that define new character positions.

General Rules

1. Storage allocation starts at data-name-2 and continues over a storage area sufficient to contain the area of character positions in the data item referenced by the data-name-1 or FILLER clause.

2. When the same character position is defined by more than one data description entry, the data-name associated with any of those data description entries can be used to reference that character position.

Function

The RENAMES clause permits alternative, possibly overlapping, groupings of elementary items.

General Format

```
66 data-name-1 RENAMES data-name-2 [ { THROUGH } data-name-3 ] .
                                   [ { THRU    }             ]
```

NOTE: Level-number 66 and data-name-1 are shown in the above format to improve clarity. Level-number and data-name-1 are not part of the RENAMES clause.

Syntax Rules

1. Any number of RENAMES entries may be written for a logical record.

2. All RENAMES entries referring to data items within a given logical record must immediately follow the last data description entry of the associated record description entry.

3. Data-name-1 cannot be used as a qualifier, and can only be qualified by the names of the associated level 01, FD, or SD entries. Neither data-name-2 nor data-name-3 may have an OCCURS clause in its data description entry nor be subordinate to an item that has an OCCURS clause in its data description entry. (See The OCCURS Clause.)

4. Data-name-2 and data-name-3 must be names of elementary items or groups of elementary items in the same logical record, and cannot be the same data-name. A 66 level entry cannot rename another 66 level entry nor can it rename a 77, 88, or 01 level entry.

5. Data-name-2 and data-name-3 may be qualified.

6. None of the items within the range, including data-name-2 and data-name-3, if specified, can be an item whose size is variable as defined in the OCCURS clause.

7. The words THRU and THROUGH are equivalent.

8. The beginning of the area described by data-name-3 must not be to the left of the beginning of the area described by data-name-2. The end of the area described by data-name-3 must be to the right of the end of the area described by data-name-2. Data-name-3, therefore, cannot be subordinate to data-name-2.

General Rules

1. When data-dame-3 is specified, data-name-1 is a group item which includes all elementary items starting with data-name-2 (if data-name-2 is an elementary item) or the first elementary item in data-name-2 (if data-name-2 is a group item), and concluding with data-name-3 (if data-name-3 is an elementary item) or the last elementary item in data-name-3 (if data-name-3 is a group item).

2. When data-name-3 is not specified, all of the data attributes of data-name-2 become the data attributes for data-name-1.

Function

The SIGN clause specifies the position and the mode of representation of the operational sign when it is necessary to describe these properties explicitly.

General Format

$$[\underline{\text{SIGN}}\ \text{IS}]\ \left\{\begin{matrix}\underline{\text{LEADING}}\\ \underline{\text{TRAILING}}\end{matrix}\right\}\ [\underline{\text{SEPARATE}}\ \text{CHARACTER}]$$

Syntax Rules

1. The SIGN clause may be specified only for a numeric data description entry whose PICTURE contains the character 'S', or a group item containing at least one such numeric data description entry.

2. The numeric data description entries to which the SIGN clause applies must be described as USAGE IS DISPLAY.

3. If the CODE-SET clause is specified in a file description entry, any signed numeric data description entries associated with that file description entry must be described with the SIGN IS SEPARATE clause.

General Rules

1. The optional SIGN clause, if present, specifies the position and the mode of representation of the operational sign for the numeric data description entry to which it applies, or for each numeric data description entry subordinate to the group to which it applies. The SIGN clause applies only to numeric data description entries whose PICTURE contains the character 'S'; the 'S' indicates the presence of, but neither the representation nor, necessarily, the position of the operational sign.

2. If a SIGN clause is specified in a group item subordinate to a group item for which a SIGN clause is specified, the SIGN clause specified in the subordinate group item takes precedence for that subordinate group item.

3. If a SIGN clause is specified in an elementary numeric data description entry subordinate to a group item for which a SIGN clause is specified, the SIGN clause specified in the subordinate elementary numeric data description entry takes precedence for that elementary numeric data item.

4. A numeric data description entry whose PICTURE contains the character 'S', but to which no optional SIGN clause applies, has an operational sign, but neither the representation, nor, necessarily, the position of the operational sign is specified by the character 'S'. In this (default) case, the operational sign is the same as if SIGN IS TRAILING without the SEPARATE phrase had been stated. General rule 5

through 7 do not apply to such signed numeric data items.

5. If the optional SEPARATE CHARACTER phrase is not present, then:

   a) The operational sign will be presumed to be associated with the leading (or, respectively, trailing) digit position of the elementary numeric data item.

   b) The letter 'S' in a PICTURE character-string is not counted in determining the size of the item (in terms of standard data format characters).

   c) The following table shows the valid signs for the SIGN IS LEADING/TRAILING without the SEPARATE clause.

DIGIT	POSITIVE	NEGATIVE
0	{	}
1	A	J
2	B	K
3	C	L
4	D	M
5	E	N
6	F	O
7	G	P
8	H	Q
9	I	R

6. If the optional SEPARATE CHARACTER phrase is present, then:

   a) The operational sign will be presumed to be the leading (or respectively, trailing) character position of the elementary numeric data item; this character position is not a digit position.

   b) The letter 'S' in a PICTURE character-string is counted in determining the size of the item (in terms of standard data format characters).

   c) The operational signs for positive and negative are the standard data format characters '+' and '-', respectively.

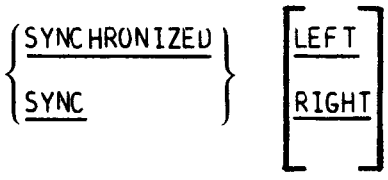
7. Every numeric data description entry whose PICTURE contains the character 'S' is a signed numeric data description entry. If a SIGN clause applies to such an entry and conversion is necessary for purposes of computation or comparisons, conversion takes place.

Function

The SYNCHRONIZED clause specifies the alignment of an elementary item on the natural boundaries of the computer memory.

On the L1 computers, the natural boundary is the word. All data items are, however, automatically aligned to byte boundaries unless the SYNCHRONIZED clause is specified. There are two character positions per word.

General Format



Syntax Rules

1. This clause may only appear with an elementary item.

2. SYNC is an abbreviation for SYNCHRONIZED.

3. An elementary data item declaration, or an elementary data item subordinate to a group item whose declaration contains a USAGE clause specifying BINARY or COMPUTATIONAL cannot be declared with a SYNCHRONIZED LEFT clause.

General Rules

1. This clause specifies that the subject data item is to be aligned in the computer such that no other data item occupies any of the character positions between the leftmost and rightmost natural boundaries delimiting this data item. If the number of character positions required to store this data item is less than the number of character positions between those natural boundaries, the unused character positions (or portions thereof) must not be used for any other data item. Such unused character positions, however, are included in:

   a) The size of any group item(s) to which the elementary item belongs; and

   b) The number of character positions allocated when any such group item is the object of a REDEFINES clause. The unused character positions are not included in the character positions redefined when the elementary item is the object of a REDEFINES clause.

2. SYNCHRONIZED not followed by either RIGHT or LEFT specifies that the elementary item is to be positioned between natural boundaries in such a way as to effect efficient utilization of the elementary data item. LEFT is assumed, except for binary data (USAGE COMP or USAGE BINARY) when RIGHT is assumed.

3. SYNCHRONIZED LEFT specifies that the elementary item is to be positioned such that it will begin at the left character position of the natural boundary in which the elementary item is placed.

4. SYNCHRONIZED RIGHT specifies that the elementary item is to be positioned such that it will terminate on the right character position of the natural boundary in which the elementary item is placed.

5. Whenever a SYNCHRONIZED item is referenced in the source program, the original size of the item is used in determining any action that depends on size, such as justification, truncation or overflow. The original size is determined by:

   - the PICTURE, USAGE and SIGN clauses for USAGE DISPLAY

   - the PICTURE and USAGE clauses for USAGE COMPUTATIONAL, COMPUTATIONAL-3, COMPUTATIONAL-4, BINARY, PACKED-DECIMAL and WORD-BINARY

   - the USAGE clause alone for USAGE COMPUTATIONAL-1, COMPUTATIONAL-2, DOUBLE-REAL, SINGLE-REAL and INDEX.

   Note that the use of the SNCHRONIZED clause in the Working-Storage Section will normally result in more efficient processing of binary items (BINARY, COMPUTATIONAL, WORD-BINARY).

6. If the data description of an item contains the SYNCHRONIZED clause and an operational sign, the sign of the item appears in the normal operational sign position, regardless of whether the item is SYNCHRONIZED LEFT or SYNCHRONIZED RIGHT.

7. When the SYNCHRONIZED clause is specified in a data description entry of a data item that also contains an OCCURS clause, or in a data description entry of a data item subordinate to a data description entry that contains an OCCURS clause, then:

   a) Each occurrence of the data item is SYNCHRONIZED.

   b) Any implicit FILLER generated for other data items within that same table are generated for each occurrence of those data items. (See general rule 8b.)

8. The SYNCHRONIZED clause is hardware dependent and in addition to rules 1 through 7, rules 9 through 13 specify how elementary items associated with this clause are handled regarding:

   a) The format on the external media of records or groups containing elementary items whose data description contains the SYNCHRONIZED clause.

   b) Any necessary generation of implicit FILLER, if the elementary item immediately preceding an item containing the SYNCHRONIZED clause does not terminate at an appropriate natural boundary. Such generated FILLER positions are included in:

- The size of any group to which the FILLER item belongs; and

- The number of character positions allocated when the group item of which the FILLER item is a part appears as the object of a REDEFINES clause.

Note that the appearance of synchronized data items can potentially alter the allocation of elementary data items within a group. Knowledge of these allocation differences is necessary for a user who is specifying group level moves of data into and out of synchronized structures.

9. There is no automatic alignment of elementary items based on usage. However, elementary level 01 data items and level 77 data items are SYNCHRONIZED regardless of usage. Record descriptions (level 01 data items which are not elementary) are always synchronized.

   Note that implicit FILLER data items provided for record synchronization are not counted in the size of either the record being synchronized or any previous level 01 or 77 data item, and are therefore logically equivalent to an elementary level 01 data item with a PICTURE character-string of X.

10. When an elementary item with the SYNCHRONIZED clause follows an item that does not terminate on a word boundary (0 modulo 2-byte address), an implicit FILLER item is generated to align the synchronized item. The size of the FILLER is one character. The FILLER is treated as if an explicit FILLER with the same level number as the synchronized item was inserted immediately preceding the synchronized item. Its size is, therefore, of significance when any of the inclusive groups are the subject or object of a REDEFINES clause or when the implicit FILLER is within the range of a RENAMES clause.

11. FILLER items will be implicitly generated as necessary to ensure that each occurrence of a group table element is the same length and that each begins on the same relative byte address. Such implicit FILLER items will be logically the last elementary item in the occurring groups.

12. Within a record, groups containing synchronized items will be treated the same as those that do not. That is, the first byte of the group will be the next available byte following the last byte allocated to the previous group, or elementary item if there was no previous group.

    Note that the next available byte may not be the next contiguous explicitly defined byte if the previous group or elementary item required implicit FILLER items of the types described in general rules 9, 10 and 11.

13. The structure of the File Section and Linkage Section record descriptions is determined as if the same record description had been allocated in the Working-Storage Section at a 0 modulo 2 starting byte address. This ensures that the relative location of a data item relative to the start of its record will be identical to a replica of the record description appearing in Working-Storage. The fact that a

logical origin of zero is assumed for File and Linkage Section record descriptions has no implication concerning the actual position of a logical record within a block on the external media or the actual physical address in memory of a Linkage Section item.

Function

The USAGE clause specifies the format of a data item in the computer storage.

General Format

```
[USAGE IS] { BINARY
           | COMPUTATIONAL
           | COMP
           | COMPUTATIONAL-1
           | COMP-1
           | COMPUTATIONAL-2
           | COMP-2
           | COMPUTATIONAL-3
           | COMP-3
           | COMPUTATIONAL-4
           | COMP-4
           | DISPLAY
           | DOUBLE-REAL
           | INDEX
           | PACKED-DECIMAL
           | SINGLE-REAL
           | WORD-BINARY }
```

Syntax Rules

1. The USAGE clause may be written in any data description entry with a level-number other than 66 or 88.

2. If the USAGE clause is written in the data description entry for a group item, it may also be written in the data description entry for any subordinate elementary item or group item, but the same usage must be specified in both entries.

3. An elementary data item whose declaration contains, or an elementary data item subordinate to a group item whose declaration contains, a

USAGE clause specifying BINARY, COMPUTATIONAL, COMPUTATIONAL-3, COMPUTATIONAL-4, PACKED-DECIMAL, or WORD-BINARY must be declared with a PICTURE character-string that describes a numeric item, i.e., a PICTURE character-string that contains only the symbols 'P', 'S', 'V', and '9'. (See The PICTURE Clause.)

4. An elementary data item whose declaration contains, or an elementary data item subordinate to a group item whose declaration contains a USAGE clause specifying COMPUTATIONAL-1, COMPUTATIONAL-2, SINGLE-REAL, or DOUBLE-REAL may not be declared with a PICTURE clause.

5. COMP is an abbreviation for COMPUTATIONAL; COMP-n is an abbreviation for COMPUTATIONAL-n.

6. An index data item can be referenced explicitly only in a SEARCH or SET statement, a relation condition, the USING phrase of a Procedure Division header, or the USING phrase of a CALL statement.

7. The BLANK WHEN ZERO, JUSTIFIED, PICTURE, and VALUE clauses must not be specified for data items declared with the USAGE IS INDEX clause.

8. An elementary data item described with a USAGE IS INDEX clause must not be a conditional variable.

General Rules

1. If the USAGE clause is written at a group level, it applies to each elementary item in the group.

2. The USAGE clause specifies the manner in which a data item is represented in the storage of a computer. It does not affect the use of the data item, although the specifications for some statements in the Procedure Division may restrict the USAGE clause of the operands referred to. The USAGE clause may affect the radix or type of character representation of the item.

3. A BINARY, COMPUTATIONAL, COMPUTATIONAL-3, COMPUTATIONAL-4, PACKED-DECIMAL, or WORD-BINARY item is capable of representing a value to be used in computations and must be numeric. If a group item is described with one of the above attributes, the group item itself may not used in computations.

4. The USAGE IS BINARY, USAGE IS COMPUTATIONAL, USAGE IS COMPUTATIONAL-4, and USAGE IS WORD-BINARY clauses specify that a radix of 2 is used to represent a numeric item in the storage of the computer. The words BINARY and COMPUTATIONAL are equivalent; the words WORD-BINARY and COMPUTATIONAL-4 are equivalent.

5. The USAGE IS COMPUTATIONAL-1, USAGE IS COMPUTATIONAL-2, USAGE IS DOUBLE-REAL, and USAGE IS SINGLE-REAL clauses specify internal floating-point items. SINGLE-REAL or COMPUTATIONAL-1 specifies single precision floating point items which are allocated four bytes. DOUBLE-REAL or COMPUTATIONAL-2 specifies double precision floating-point items which are allocated eight bytes.

6. The USAGE IS COMPUTATIONAL-3 and USAGE IS PACKED-DECIMAL clauses specify that a radix of 10 is used to represent a numeric item in the storage of the computer. Furthermore, this clause specifies that each digit position must occupy the minimum possible configuration in computer storage. The same amount of space is allocated for signed and unsigned items. In fact, items which are defined without a sign actually contain a positive sign.

7. The USAGE IS DISPLAY clause (whether specified explicitly or implicitly) indicates that the format of the data is a standard data format.

8. If the USAGE clause is not specified for an elementary item, or for any group to which the item belongs, the usage is implicitly DISPLAY.

9. The USAGE IS INDEX clause specifies that a data item is an index data item and contains a value which must correspond to an occurrence number of a table element. Two bytes are allocated for each index data item. If a group item is described with the USAGE IS INDEX clause the elementary items in the group are all index data items. The group itself is not an index data item and must not be used in the SEARCH or SET statement or in a relation condition, but may be referenced in the USING phrase either of a CALL statement or of the program's Procedure Division header.

10. When a MOVE statement or an input-output statement that references a group item that contains an index data item is executed, no conversion of the index data item takes place.

11. The following table shows the number of bytes allocated for each numeric item (other than floating-point and index) for each of the possible combinations of 9's and S's and usages.

USAGE	9s in PICTURE																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
DISPLAY not sep	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
DISPLAY SEPARATE	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
BINARY signed	1	1	2	2	3	3	4	4	4	5	5	6	6	6	7	7	8	8
BINARY unsigned	1	1	2	2	3	3	3	4	4	5	5	5	6	6	7	7	8	8
PACKED-DECIMAL	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10
WORD-BINARY	2	2	2	2	4	4	4	4	4	8	8	8	8	8	8	8	8	8

VALUE clause

Function

The VALUE clause defines the value of constants, the value of Report Section printable items, the initial value of WORKING STORAGE items and the values associated with a condition-name.

General Format

Format 1:

```
VALUE IS literal-1
```

Format 2:

```
VALUE IS                 { literal-2 [ { THROUGH } literal-3 ] } ...
VALUES ARE               {           [ { THRU    }           ] }
```

Syntax Rules

1. The words THRU and THROUGH are equivalent.

2. A signed numeric literal must have associated with it a signed numeric PICTURE character-string.

3. All numeric literals in a VALUE clause of an item must have a value which is within the range of values indicated by the PICTURE clause, and must not have a value which would require truncation of nonzero digits. Nonnumeric literals in a VALUE clause of an item must not exceed the size indicated by the PICTURE clause.

4. The VALUE clause must not be specified in any entry which is part of the description or redefinition of an external data record. The VALUE clause may be specified for condition-name entries associated with such data description entries.

General Rules

1. The VALUE clause must not conflict with other clauses in the data description of the item or in the data description within the hierarchy of the item. The following rules apply:

   a) If the category of the item is numeric, all literals in the VALUE clause must be numeric. If the literal defines the value of a working storage item, the literal is aligned in the data item according to the standard alignment rules.

b) If the category of the item is alphabetic, alphanumeric, alphanumeric edited or numeric edited, all literals in the VALUE clause must be nonnumeric literals or integer (except for alphabetic and group items). The literal is aligned in the data item as if the data item had been described as alphanumeric. Editing characters in the PICTURE clause are included in determining the size of the data item (See The PICTURE Clause) but have no effect on initialization of the data item. Therefore, the VALUE for an edited item is presented in an edited form.

c) Initialization takes place independent of any BLANK WHEN ZERO or JUSTIFIED clause that may be specified.

Condition-Name Rules

1. In a condition-name entry, the VALUE clause is required. The VALUE clause and the condition-name itself are the only two clauses permitted in the entry. The characteristics of a condition-name are implicitly those of its conditional variable.

2. Format 2 can be used only in connection with condition-names. Wherever the THRU phrase is used, literal-2 must be less than literal-3.

Data Description Entries Other Than Condition-Names

1. Rules governing the use of the VALUE clause differ with the respective sections of the Data Division:

   a) In the File Section, the VALUE clause may be used only in condition-name entries; therefore the initial value of the data items in the File Section is undefined.

   b) In the Linkage Section, the VALUE clause may be used only in condition-name entries.

   c) In the Working-Storage Section, the VALUE clause must be used in condition-name entries. The VALUE clauses in the Working-Storage Section of a program take effect only when the program is placed into its initial state. If the VALUE clause is used in the description of the data item, the data item is initialized to the defined value. If the VALUE clause is not associated with a data item, the initial value of that data item is undefined.

2. The VALUE clause must not be stated in a data description entry that contains an OCCURS clause, or in an entry that is subordinate to an entry containing an OCCURS clause. This rule does not apply to condition-name entries. (See The OCCURS Clause.)

3. The VALUE clause must not be stated in a data description entry that contains a REDEFINES clause, or in an entry that is subordinate to an entry containing a REDEFINES clause. This rule does not apply to condition-name entries.

4. If the VALUE clause is used in an entry at the group level, the literal must be a figurative constant or a nonnumeric literal, and the group area is initialized without consideration for the

individual elementary or group items contained within this group. The VALUE clause cannot be stated at the subordinate levels within this group.

5. The VALUE clause must not be specified for a group item containing items subordinate to it with descriptions including JUSTIFIED, SYNCHRONIZED, or USAGE (other than USAGE IS DISPLAY).

5. PROCEDURE DIVISION

GENERAL DESCRIPTION

The Procedure Division contains procedures to be executed by the object program. The Procedure Division is required in a COBOL source program.

The general formats of the Procedure Division are shown below.

General Format

Format 1:

```
PROCEDURE DIVISION [USING {data-name-1} ... ] .

[DECLARATIVES.

{section-name SECTION [segment-number].

    USE statement.

[paragraph-name.  [sentence] ... ] ... } ...

 END DECLARATIVES.]

{section-name SECTION  [segment-number].

[paragraph-name.  [sentence] ... ] ... } ...
```

Format 2

```
PROCEDURE DIVISION [USING {data-name-1} ... ] .

{paragraph-name.  [sentence] ... } ...
```

Syntax Rules

1. The USING option may only be specified for a called program to specify parameters which are to be used by the calling and called programs. (See The Inter-Program communication chapter).

2. When DECLARATIVE sections are specified, they must be grouped at the beginning of the Procedure Division, and must be preceded by the key word DECLARATIVES and followed by the key words END DECLARATIVES.

3. If DECLARATIVE sections are specified the entire Procedure Division must be divided into sections.

4. A maximum of 100 DECLARATIVE sections are permitted in a program.

5. A section may not be qualified.

ARITHMETIC EXPRESSIONS

An arithmetic expression can be an identifier of a numeric elementary item, a numeric literal, such identifiers and literals separated by arithmetic operators, two arithmetic expressions separated by an arithmetic operator, or an arithmetic expression enclosed in parentheses. Any arithmetic expression may be preceded by a unary operator. The permissible combinations of variables, numeric literals, arithmetic operator and parentheses are given in the table, Combination of Symbols in Arithmetic Expressions.

Those identifiers and literals appearing in an arithmetic expression must represent either numeric elementary items or numeric literals on which arithmetic may be performed.

ARITHMETIC OPERATORS

There are five binary arithmetic operators and two unary arithmetic operators that may be used in arithmetic expressions. They are represented by specific characters that must be preceded by a space and followed by a space.

Binary Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation

Unary Operator	Meaning
+	Multiplication by numeric literal +1
-	Multiplication by numeric literal -1

FORMATION AND EVALUATION RULES

1. Parentheses may be used in arithmetic expressions to specify the order in which elements are to be evaluated. Expressions within parentheses are evaluated first, and within nested parentheses, evaluation proceeds from the least inclusive set to the most inclusive set. When parentheses are not used, or parenthesized expressions are at the same level of inclusiveness, the following hierarchical order of execution is implied:

   1st - Unary plus and minus
   2nd - Exponentiation
   3rd - Multiplication and division
   4th - Addition and subtraction

2. Parentheses are used either to eliminate ambiguities in logic where consecutive operations of the same hierarchical level appear or to modify the normal hierarchical sequence of execution in expressions where it is necessary to have some deviation from the normal

precedence. When the sequence of execution is not specified by parentheses, the order of execution of consecutive operations of the same hierarchical level is from left to right.

3. The ways in which operators, variables, and parentheses may be combined in an arithmetic expression are summarized in the table below. Where 'P' indicates a permissible pair of symbols and '-' an invalid pair.

FIRST SYMBOL	SECOND SYMBOL				
	Identifier or Literal	+ - * / **	Unary + or -	(	)
Identifier or Literal	-	P	-	-	P
+ - * / **	P	-	P	P	-
Unary + or Unary -	P	-	-	P	-
(	P	-	P	P	-
)	-	P	-	-	P

Combination of Symbols in Arithmetic Expressions.

4. An arithmetic expression may only begin with the symbol '(', '+', '-', an identifier, or a literal and may only end with a ')', an identifier, or a literal. There must be a one-to-one correspondence between left and right parentheses of an arithmetic expression such that each left parenthesis is to the left of its corresponding right parenthesis. If the first operator in an arithmetic expression is a unary operator, it must be immediately preceded by a left parenthesis if that arithmetic expression immediately follows an identifier or another arithmetic expression.

5. The following rules apply to evaluation of exponentiation in an arithmetic expression:

   a) If the value of an expression to be raised to a power is zero, the exponent must have a value greater than zero. Otherwise, the size error condition exists. (See The ON SIZE ERROR Phrase.)

   b) If the evaluation yields both a positive and negative real number, the value returned as the result is the positive number.

   c) If no real number exists as the result of the evaluation, the size error condition exists.

6. Arithmetic expressions allow the user to combine arithmetic operations without the restrictions on composite of operands and /or receiving data items. Each implementor will indicate the techniques used in handling arithmetic expressions.

CONDITIONAL EXPRESSIONS

Conditional expressions identify conditions that are tested to enable the object program to select between alternate paths of control depending upon the truth value of the condition. Conditional expressions are specified in the EVALUATE, IF, PERFORM and SEARCH statements. There are two categories of conditions associated with conditional expressions: simple conditions and complex conditions. Each may be enclosed within any number of paired parentheses, in which case its category is not changed.

SIMPLE CONDITIONS

The simple conditions are the relation, class, condition-name, switch-status, and sign conditions. A simple condition has a truth value of true or false. The inclusion in parentheses of simple conditions does not change the simple condition truth value.

RELATION CONDITION

A relation condition causes a comparison of two operands, each of which may be the data item referenced by an identifier, a literal, the value resulting from an arithmetic expression, or an index-name. A relation condition has a truth value of true if the relation exists between the operands. Comparison of two numeric operands is permitted regardless of the formats specified in their respective USAGE clauses. However, for all other comparisons the operands must have the same usage. If either of the operands is a group item, the nonnumeric comparison rules apply. The general format of a relation condition is:

```
{identifier-1           }                {IS [NOT] GREATER THAN}
{literal-1              }                {IS [NOT] >           }
{arithmetic-expression-1} {IS [NOT] LESS THAN                 }
{index-name-1           }                {IS [NOT] <           }
                                         {IS [NOT] EQUAL TO    }
                                         {IS [NOT] =           }

                                  {identifier-2           }
                                  {literal-2              }
                                  {arithmetic-expression-2}
                                  {index-name-2           }
```

The first operand (identifier-1, literal-1, arithmetic-expression-1, or index-name-1) is called the subject of the condition; the second operand (identifier-2, literal-2, arithmetic-expression-2, or index-name-2) is called the object of the condition. The relation condition must contain at least one reference to a variable.

The relational operators specify the type of comparison to be made in a relation condition. A space must precede and follow each reserved word comprising the relational operator. When used, NOT and the next key word or relation character are one relational operator that defines the comparison to be executed for truth value; e.g., NOT EQUAL is a truth test for an unequal comparison; NOT GREATER is a truth test for an equal or less comparison. The meaning of the relational operators is:

Meaning	Relational Operator
Greater than or not greater than	IS [NOT] GREATER THAN
	IS [NOT] >
Less than or not less than	IS [NOT] LESS THAN
	IS [NOT] <
Equal to or not equal to	IS [NOT] EQUAL TO
	IS [NOT] =

COMPARISON OF NUMERIC OPERANDS

For operands whose class is numeric, a comparison is made with respect to the algebraic value of the operands. The length of the literal or arithmetic expression operands, in terms of number of digits represented, is not significant. Zero is considered a unique value regardless of sign.

Comparison of these operands is permitted regardless of the manner in which their usage is described. Unsigned numeric operands are considered positive for purposes of comparison.

COMPARISON OF NONNUMERIC OPERANDS

For nonnumeric operands, or one numeric and one nonnumeric operand, a comparison is made with respect to a specified collating sequence of characters. If one of the operands is specified as numeric, it must be an integer data item or an integer literal and:

1. If the nonnumeric operand is an elementary data item or a nonnumeric literal, the numeric operand is treated as though it were moved to an elementary alphanumeric data item of the same size as the numeric data item (in terms of standard data format characters), and the content of this alphanumeric data item were then compared to the nonnumeric operand. (See The MOVE Statement.)

2. If the nonnumeric operand is a group item, the numeric operand is treated as though it were moved to a group item of the same size as the numeric data item (in terms of standard data format characters), and the content of this group item were then compared to the nonnumeric operand. (See The MOVE Statement.)

3. A noninteger numeric operand cannot be compared to a nonnumeric operand.

The size of an operand is the total number of standard data format characters in the operand. Numeric and nonnumeric operands may be compared only when their usage is the same.

There are two cases to consider: operands of equal size and operands of unequal size.

1. Operands of equal size. If the operands are of equal size, comparison effectively proceeds by comparing characters in corresponding character positions starting from the high order end and continuing until either a pair of unequal characters is encountered or the low order end of the operand is reached, whichever comes first. The operands are determined to be equal if all pairs of corresponding characters are equal.

   The first encountered pair of unequal characters is compared to determine their relative position in the collating sequence. The operand that contains the character that is positioned higher in the collating sequence is considered to be the greater operand.

2. Operands of unequal size. If the operands are of unequal size, comparison proceeds as though the shorter operand were extended on the right by sufficient spaces to make the operands of equal size.

COMPARISONS INVOLVING INDEX-NAMES and/or INDEX DATA ITEMS

Relation tests may be made only between:

1. Two index-names. The result is the same as if the corresponding occurrence numbers were compared.

2. An index-name and a data item (other than an index data item), literal, or arithmetic expression. The occurrence number that corresponds to the value of the index-name is compared to the data item or literal.

3. An index data item and an index-name or another index data item. The actual values are compared without conversion.

CLASS CONDITION

The class condition determines whether the operand is numeric, alphabetic or contain only the characters in the set of characters specified by the CLASS clause as defined in the SPECIAL-NAMES paragraph of the Environment Division. The class of an operand is determined as follows:

1. The operand is numeric if it consist entirely of the characters 0, 1, 2, 3, ..., 9, with or without the operational sign.

2. The operand is alphabetic if it consists entirely of the uppercase letters A, B, C, ..., Z, space, or the lowercase letters a, b, c, ..., z, space, or any combination of the uppercase and lowercase letters and spaces.

3. An operand is in conformance to class-name, if it consist entirely of the characters listed in the definition of class-name in the SPECIAL-NAMES paragraph.

The general format of the class condition is:

```
identifier-1 IS [NOT] { NUMERIC      }
                      { ALPHABETIC   }
                      { class-name   }
```

The usage of the operand being tested must be described as DISPLAY.

When used, NOT and the next key word specify one class condition that defines the class test to be executed for truth value; e.g. NOT NUMERIC is a truth test for determining that an operand is nonnumeric.

The NUMERIC test cannot be used with an item whose data description describes the item as alphabetic or as a group item composed of elementary items whose data description indicates the presence of operational sign(s). If the data description of the item being tested does not indicate the presence of an operational sign, the item being tested is determined to be numeric only if the content are numeric and an operational sign is not present. If the data description of the item does indicate the presence of an operational sign, the item being tested is determined to be numeric only if the content are numeric and a valid operational sign is present. Valid operational signs for data items described with the SIGN IS SEPARATE clause are the standard data format characters, + and -. Valid operational signs for elementary data items described without the SIGN IS SEPARATE clause are defined in The SIGN Clause paragraph.

The ALPHABETIC test cannot be used with an item whose data description describes the item as numeric. The result of the test is true if the content of the data item referenced by identifier-1 consists entirely of alphabetic characters.

The class-name test must not be used with an item whose data description describes the item as numeric.

CONDITION-NAME CONDITION (Conditional Variable)

In a condition-name condition, a conditional variable is tested to determine whether or not its value is equal to one of the values associated with condition-name-1. The general format for the condition-name condition is:

condition-name-1

If condition-name-1 is associated with a range or ranges of values, then the conditional variable is tested to determine whether or not its value falls in this range, including the end values.

The rules for comparing a conditional variable with a condition-name value are the same as those specified for relation conditions.

The result of the test is true if one of the values corresponding to condition-name-1 equals the value of its associated conditional variable.

SWITCH-STATUS CONDITION

A switch-status condition determines the on or off status of a defined switch. The implementor-name and the on or off value associated with the condition must be named in the SPECIAL-NAMES paragraph of the Environment Division. The general format for the switch-status condition is:

condition-name-1

The result of the test is true if the switch is set to the specified position corresponding to condition-name-1.

SIGN CONDITION

The sign condition determines whether or not the algebraic value of an arithmetic expression is less than, greater than, or equal to zero. The general format for a sign condition is:

arithmetic-expression-1 IS [NOT] { POSITIVE / NEGATIVE / ZERO }

When used, NOT and the next key word specify one sign condition that defines the algebraic test to be executed for truth value; e.g., NOT ZERO is a truth test for a nonzero (positive or negative) value. An operand is positive if its value is greater than zero, negative if its value is less than zero, and zero if its value is equal to zero. Arithmetic-expression-1 must contain at least one reference to a variable.

COMPLEX CONDITIONS

A complex condition is formed by combining simple conditions, combined conditions and/or complex conditions with logical connectors (logical operators 'AND' and 'OR') or negating these conditions with logical negation (the logical operator 'NOT'). The truth value of a complex condition, whether parenthesized or not, is that truth value which results from the interaction of the stated logical operators on its constituent conditions. The logical operators and their meanings are:

Logical Operator	Meaning
AND	Logical conjunction; the truth value is true if both of the conjoined conditions are true; false if one or both of the conjoined conditions is false.
OR	Logical inclusive OR; the truth value is true if one or both of the included conditions is true; false if both included conditions are false.
NOT	Logical negation or reversal of truth value; the truth value is true if the condition is false; false if the condition is true.

The logical operators must be preceded and followed by a space.

NEGATED CONDITIONS

A condition is negated by used of the logical operator 'NOT' which reverses the truth value of the condition to which it is applied. Thus, the truth value of a negated condition is true if and only if the truth value of the condition being negated is false; the truth value of a negated condition is false if and only if the truth value of the condition being negated is true. Including a negated condition in parentheses does not change its truth value. The general format for a negated condition is:

NOT condition-1

COMBINED CONDITIONS

A combined condition results from connecting conditions with one of the logical operators 'AND' or 'OR'. The general format of a combined condition is:

condition-1 { { AND / OR } condition-2 } ...

Precedence of Logical Operators and the Use of Parentheses

In the absence of the relevant parentheses in a complex condition, the precedence (i.e., binding power) of the logical operators determines the conditions to which the specified logical operators apply and implies the equivalent parentheses. The order of precedence is 'NOT', 'AND', 'OR'. Thus, specifying 'condition-1 OR NOT condition-2 AND condition-3' implies and is equivalent to specifying 'condition-1 OR ((NOT condition-2) AND condition-3)'. Where parentheses are used in a complex condition, they determine the binding of conditions to logical operators. Parentheses can, therefore, be used to depart from the normal precedence of logical operators as specified above. Thus, the example complex condition above can be given a different meaning by specifying it as '(condition-1 OR (NOT condition-2)) AND condition-3'. (see Order of Evaluation of Conditions.)

The table below indicates the ways in which conditions and logical operators may be combined and parenthesized. There must be a one-to-one correspondence between left and right parentheses such that each left parenthesis is to the left of its corresponding right parenthesis.

Given the following element:	In a conditional expression:		In a left-to-right sequence of elements:	
	May element be first?	May element be last?	Element when not first, may be immediately preceded by only:	Element when not last, may be immediately followed by only:
simple-cond.	Yes	Yes	OR, NOT,AND, (	OR, AND,)
OR or AND	No	No	simple-cond. ,)	simple-cond. ,NOT,(
NOT	Yes	No	OR, AND, (	simple-cond. ,(
(	Yes	No	OR, NOT,AND, (	simple-cond. ,NOT,(
)	No	Yes	simple-cond. ,)	OR, AND,)

Combinations of Conditions, Logical Operators and Parentheses.

Thus, the element pair OR NOT is permissible while the pair NOT OR is not permissible; the pair NOT (is permissible while the pair NOT NOT is not permissible.

ABBREVIATED COMBINED RELATION CONDITIONS

When simple or negated simple relation conditions are combined with logical connectives in a consecutive sequence such that a succeeding relation condition contains a subject or subject and relational operator that is common with the preceding relation condition, and no parentheses are used within such a consecutive sequence, any relation condition except the first may be abbreviated by:

1. The omission of the subject of the relation condition, or

2. The omission of the subject and relational operator of the relation condition.

The format for an abbreviated combined relation condition is:

relation-condition { { AND / OR } [NOT] [relational-operator] object } ...

Within a sequence of relation conditions both of the above forms of abbreviation may be used. The effect of using such abbreviations is as if the last preceding stated subject were inserted in place of the omitted subject, and the last stated relational operator were inserted in place of the omitted relational operator. The result of such implied insertion must comply with the rules of Combinations of Conditions, Logical Operators, and Parentheses, shown above. This insertion of an omitted subject and/or relational operator terminates once a complete simple condition is encountered within a complex condition.

The interpretation applied to the use of the word NOT in an abbreviated combined relation condition is as follows:

1. If the word immediately following NOT is GREATER, >, LESS, <, EQUAL, =, then the NOT participates as part of the relational operator; otherwise

2. The NOT is interpreted as a logical operator and, therefore, the implied insertion of subject or relational operator results in a negated relation condition.

Some examples of abbreviated combined and negated combined relation conditions and expanded equivalents follow.

Abbreviated Combined Relation Condition	Expanded Equivalent
a > b AND NOT < c OR d	((a > b) AND (a NOT < c)) OR (a NOT < d)
a NOT EQUAL b OR c	(a NOT EQUAL b) OR (a NOT EQUAL c)
NOT a = b OR c	(NOT (a = b)) OR (a = c)
NOT (a GREATER b OR < c)	NOT ((a GREATER b) OR (a < c))
NOT (a NOT > b AND c AND NOT d)	NOT ((((a NOT > b) AND (a NOT > c)) AND (NOT (a NOT > d))))

ORDER OF EVALUATION OF CONDITIONS

Parentheses may be used to specify the order in which individual conditions of complex conditions are to be evaluated when it is necessary to depart from the implied evaluation precedence. Conditions within parentheses are evaluated first, and, within nested parentheses, evaluation proceeds from the least inclusive condition to the most inclusive condition. When parentheses are not used, or parenthesized conditions are at the same level of inclusiveness, the following hierarchical order of logical evaluation is implied until the final truth value is determined:

1. Values are established for arithmetic expressions.

2. Truth values for simple conditions and negated simple conditions are established in the following order:
   a. relation (following the expansion of any abbreviated condition)
   b. class
   c. condition-name
   d. switch-status
   e. sign

3. Truth values for combined conditions and negated combined conditions are established in the following order:
   a. AND logical operators
   b. OR logical operators

4. When the sequence of evaluation is not completely specified by parentheses the order of evaluation of consecutive operations of the same hierarchical level is from left to right.

COMMON OPTIONS AND RULES FOR STATEMENT FORMATS

This paragraph and its subordinate paragraphs provide a description of the common options and conditions that pertain to or appear in several different statements.

THE ROUNDED PHRASE

If, after decimal point alignment, the number of places in the fractions of the result of an arithmetic operation is greater than the number of places provided for the fraction of the resultant identifier, truncation is relative to the size provided for the resultant identifier. When rounding is requested, the absolute value of the resultant identifier is increased by one in the low-order position whenever the most significant digit of the excess is greater than or equal to five.

When the low-order integer positions in a resultant identifier are represented by the character P in the picture for that resultant identifier, rounding or truncation occurs relative to the rightmost integer position for which storage is allocated.

The ROUNDED phrase is not allowed for resultant identifiers described with a USAGE IS COMP-1, COMP-2, SINGLE-REAL or DOUBLE-REAL clause.

THE ON SIZE ERROR PHRASE

The size error condition occurs under the following circumstances:

1. Violation of the rules for evaluation of exponentiation always terminates the arithmetic operation and always causes a size error condition. (See Formation and Evaluation Rules.)

2. Division by zero always terminates the arithmetic operation and always causes a size error condition.

3. If, after radix point alignment, the absolute value of a result exceeds the largest value that can be contained in the associated resultant identifier, as specified by the item's PICTURE, a size error condition exists. If the ROUNDED phrase is specified, rounding takes place before checking for size error.

4. If an overflow or underflow condition occurs in operations involving floating point data.

If the ON SIZE ERROR phrase is specified and a size error condition exists after the execution of the arithmetic operations specified by an arithmetic statement, the values of the affected resultant identifiers remain unchanged from the values they had before execution of the arithmetic statement. The values of resultant identifiers for which no size error condition exists are the same as they would have been if the size error condition had not resulted for any of the resultant identifiers. After completion of the arithmetic operations, control is transferred to the imperative-statement specified in the ON SIZE ERROR phrase and execution continues according to the rules for each statement specified in

that imperative-statement. If a procedure branching or conditional statement which causes explicit transfer of control is executed, control is transferred in accordance with the rules for that statement; otherwise, upon completion of the execution of the imperative-statement specified in the ON SIZE ERROR phrase, control is transferred to the end of the arithmetic statement.

If the SIZE ERROR phrase is not specified and a size error condition exists after the execution of the arithmetic operations specified by an arithmetic statement, the values of the affected resultant identifiers are undefined. The values of resultant identifiers for which no size error condition exists are the same as they would have been if the size error condition had not resulted for any of the resultant identifiers. After completion of the arithmetic operations, control is transferred to the end of the arithmetic statement.

If the size error condition does not exist after the execution of the arithmetic operations specified by an arithmetic statement, the ON SIZE ERROR phrase, if specified, is ignored and control is transferred to the end of the arithmetic statement.

For the ADD statement with the CORRESPONDING phrase and the SUBTRACT statement with the CORRESPONDING phrase, if any of the individual operations produces a size error condition, imperative-statement-1 in the SIZE ERROR phrase is not executed until all of the individual additions or subtractions are completed.

THE CORRESPONDING PHRASE

For the purpose of this discussion, identifier-1 and identifier-2 are the identifiers specified in a statement which contains the CORRESPONDING phrase.

1. Rules for valid identifiers are:

    a) All identifiers must refer to group items.

    b) The REDEFINES or OCCURS clause may be specified in the data description entry of any of the identifiers.

    c) Identifiers may be subordinate to a data description entry containing a REDEFINES or OCCURS clause.

    d) No identifier may be defined with level-number 66, with level-number 77, or with the USAGE IS INDEX clause.

2. Data items subordinate to identifier-1 correspond with data items subordinate to identifier-2, if the following rules apply:

    a) Both data items must have the same data-name.

    b) All possible qualifiers for the sending data item, up to but not including identifier-1, must be identical to all possible qualifiers for the receiving data item up to but not including identifier-2.

c) In an ADD or SUBTRACT statement, only elementary numeric data items will be considered.

d) In a MOVE statement, the corresponding sending and/or receiving data items must be elementary. The class of any corresponding pair of data items may differ.

e) A data item with a level-number of 66 or 88 or with a data description entry containing a REDEFINES, OCCURS, or USAGE IS INDEX clause is not considered. Any data item subordinate to a data item which is not eligible for correspondence will also be ignored.

f) FILLER data items and data items subordinate thereto are ignored.

THE ARITHMETIC STATEMENTS

The arithmetic statements are the ADD, COMPUTE, DIVIDE, MULTIPLY, and SUBTRACT statements. They have several common features.

1. The data descriptions of the operands need not be the same; any necessary conversion and decimal point alignment is supplied throughout the calculation.

2. The maximum size of each operand is 18 decimal digits. The composite of operands, which is a hypothetical data item resulting from the superimposition of specified operands in a statement aligned on their decimal points (see The ADD Statement; The DIVIDE Statement; The MULTIPLY Statement; The SUBTRACT Statement) must not contain more than 18 decimal digits.

3. Each arithmetic operation is evaluated using an intermediate data item provided for the result of the arithmetic operation. If the size of the result being developed is larger than this intermediate data item, truncation occurs. The content of the intermediate data item are moved to the resultant identifier according to the rules for the MOVE statement. Rounding is performed and the size error condition is determined only during this move.

OVERLAPPING OPERANDS

When a sending and a receiving item in any statement share a part or all of their storage areas, yet are not defined by the same data description entry, the result of the execution of such a statement is undefined. In addition, the results are undefined for some statements in which sending and receiving items are defined by the same data description entry. These cases are addressed in the general rules associated with those statements.

MULTIPLE RESULTS IN ARITHMETIC STATEMENTS

The ADD, COMPUTE, DIVIDE, MULTIPLY, and SUBTRACT statements may have multiple results. Such statements behave as though they had been written in the following way:

1. A statement whose execution accesses all data items that are part of the initial evaluation of the statement, performs any necessary arithmetic or combining of these data items and stores the result of this operation in a temporary location. See the individual statements for the rules indicating which items are part of the initial evaluation.

2. A sequence of statements whose execution transfers or combines the value in this temporary location with each single resulting data item. These statements are considered to be written in the same left-to-right sequence that the multiple results are specified.

The result of the statement

```
ADD a, b, c TO c, d (c), e
```

is equivalent to

```
ADD a, b, c GIVING temp
ADD temp TO c
ADD temp TO d (c)
ADD temp TO e
```

and the result of the statement

```
MULTIPLY a (i) BY i, a (i)
```

is equivalent to

```
MOVE a (i) TO temp
MULTIPLY temp BY i
MULTIPLY temp BY a (i)
```

in both cases, 'temp' is an intermediate result item.

INCOMPATIBLE DATA

Except for the class condition, when the content of a data item is referenced in the Procedure Division and the content of that data item is not compatible with the class specified for that data item by its PICTURE clause, then the result of such a reference is undefined. (See Class Condition.)

PROCEDURE DIVISION STATEMENTS

The Procedure Division statements are presented in alphabetical order in the following pages.

Function

The ACCEPT statement causes low volume data to be made available to the specified data item.

General Format

Format 1:

```
ACCEPT identifier-1 [ FROM { CONSOLE          } ]
                           { mnemonic-name-1  }
```

Format 2:

```
ACCEPT identifier-2 FROM { DATE }
                         { DAY  }.
                         { TIME }
```

Format 3:

```
ACCEPT identifier-3 FROM LINE-PARAM { literal-1    }
                                    { identifier-4 }
```

Syntax Rules

1. Mnemonic-name-1 in format 1 must also be specified in the SPECIAL-NAMES paragraph of the Environment Division and must be associated with a hardware device.

2. Identifier-3 and identifier-4 must be defined as alphanumeric data items.

3. Identifier-4 must be four characters in length.

4. Literal-1 must be a nonnumeric literal four characters in length and must not be a figurative constant.

General Rules

Format 1

1. The ACCEPT statement causes the transfer of data from the hardware device. This data replaces the content of the data item named by the identifier.

2. The maximum size of the data that can be transferred is 160 characters.

3. Except when the previous ACCEPT/DISPLAY statement executed was a DISPLAY statement specifying the NO ADVANCING phrase, the execution of an ACCEPT statement causes the prompt '?' to be displayed (See the DISPLAY statement).

4. The transferred data is stored in the receiving data item after any necessary alignment, truncation and space or zero filling. (See Standard Alignment Rules).

5. If the FROM phrase is not given, CONSOLE is assumed.

6. If the data entered does not match the attributes of identifier-1, a request is made to reenter the data.

7. If no data has been entered before the enter key is pressed, the value assigned to identifier-1 is spaces if identifier-1 is alphanumeric and zero if identifier-1 is numeric.

8. If identifier-1 is numeric or numeric edited with a sign specified in its associated PICTURE clause, the sign may be entered as leading separate or trailing separate.

Format 2

9. The ACCEPT statement causes the information requested to be transferred to the data item specified by identifier according to the rules of the MOVE statement. DATA, DAY, and TIME are conceptual data items and, therefore, are not described in the COBOL program.

10. DATE is composed of the data elements year of century, month of year, and day of month. The sequence of the data element codes shall be from high order to low order (left to right), year of century, month of year, and day of month. Therefore, February 14, 1984 would be expressed as 840214. DATE, when accessed by a COBOL program, behaves as if it had been described in the COBOL program as an unsigned elementary numeric integer data item six digits in length.

11. DAY is composed of the data elements year of century and day of year. The sequence of the data element codes shall be from high order to low order (left to right) year of century, day of year. Therefore, February 14, 1984 would be expressed as 84045. DAY, when accessed by a COBOL program, behaves as if it had been described in a COBOL program as an unsigned elementary numeric integer data item five digits in length.

12. TIME is composed of the data elements hours, minutes, seconds and hundredths of a second. TIME is based on elapsed time after midnight on a 24-hour clock basis -- thus, 2:41 p.m. would be expressed as 14410000. TIME, when accessed by a COBOL program behaves as if it had been described in a COBOL program as an unsigned elementary numeric integer data item eight digits in length. The minimum value of TIME is 00000000; the maximum value of TIME is 23595999. The value returned from the fractional parts of a second is always zero.

Format 3

13. The execution of a format 3 ACCEPT statement allows a line parameter to be made available to a COBOL program. The parameter key is indicated by literal-1 or the content of the data item referenced by identifier-4.

14. The parameter value, if present, is moved into the data item referenced by identifier-3. If not present, the data item referenced by identifier-3 is set to SPACES.

ADD

Function

The ADD statement causes two or more numeric operands to be summed and the result to be stored.

General Format

Format 1:

```
ADD  { identifier-1 }  ...  TO  { identifier-2 [ROUNDED] }  ...
     { literal-1    }          {                -------- }

     [ON SIZE ERROR imperative-statement-1 [END-ADD]]
```

Format 2:

```
ADD  { identifier-1 }  ...  TO  { identifier-2 }
     { literal-1    }          { literal-2    }

     GIVING {identifier-3 [ROUNDED]} ...

     [ON SIZE ERROR imperative-statement-1 [END-ADD]]
```

Format 3:

```
     { CORRESPONDING }
ADD  { CORR          }  identifier-1 TO identifier-2 [ROUNDED]

     [ON SIZE ERROR imperative-statement-1 [END-ADD]]
```

Syntax Rules

1. In formats 1 and 2, each identifier must refer to an elementary numeric item, except that in format 2 each identifier following the word GIVING must refer to either an elementary numeric item or an elementary numeric edited item. In format 3, each identifier must refer to a group item.

2. If the ROUNDED phrase is specified for identifier-2 in format 1 or identifier-3 in format 2, the identifier cannot be described with a USAGE clause specifying COMPUTATIONAL-1, COMPUTATIONAL-2, COMP-1,

COMP-2, SINGLE-REAL or DOUBLE-REAL.

3. Each literal must be a numeric literal.

4. The composite of operands must not contain more than 18 digits. (See The Arithmetic Statements).

   a) In format 1 the composite of operands is determined by using all of the operands in a given statement.

   b) In format 2 the composite of operands is determined by using all of the operands in a given statement excluding the data items that follow the word GIVING.

   c) In format 3 the composite of operands is determined separately for each pair of corresponding data items.

5. CORR is an abbreviation for CORRESPONDING.

General Rules

1. If format 1 is used, the values of the operands preceding the word TO are added together and the sum is stored in a temporary data item. The value in this temporary data item is added to the value of the data item referenced by identifier-2, storing the result into the data item referenced by identifier-2, and repeating this process for each successive occurrence of identifier-2 in the left-to-right order in which identifier-2 is specified

2. If format 2 is used, the values of the operands preceding the word GIVING are added together, then the sum is stored as the new content of each data item referenced by identifier-3.

3. If format 3 is used, data items in identifier-1 are added to and stored in corresponding data items in identifier-2.

4. The compiler insures that enough places are carried so as not to lose any significant digits during execution.

5. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See Scope of Statements; The ROUNDED Phrase; The Arithmetic Statements; Overlapping Operands; The SIZE ERROR Phrase; The CORRESPONDING Phrase; and Multiple Results in Arithmetic Statements.)

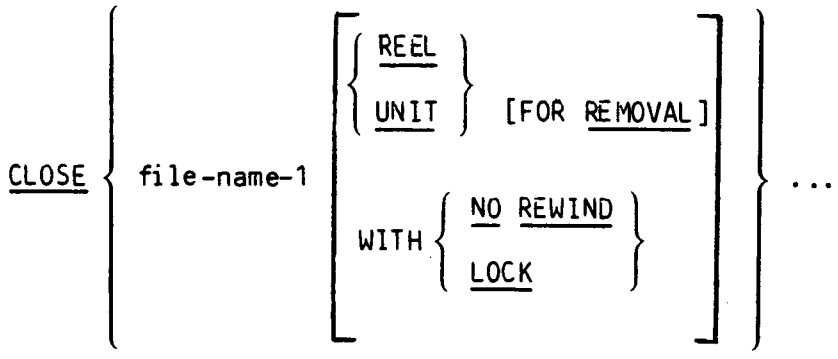
CLOSE

Function

The CLOSE statement terminates the processing of reels/units and files with optional rewind and/or lock or removal where applicable.

General Format

Format 1 - Sequential Organization



Format 2 - Relative and Indexed Organization

```
CLOSE { file-name-1   [ WITH LOCK ] } ...
```

Syntax Rules

1. The files referenced in the CLOSE statement need not all have the same organization or access.

2. A CLOSE clause containing a REEL/UNIT phrase is treated as a comment for that particular file-name-1 only.

General Rules

Except where otherwise stated in the general rules below, the terms 'REEL' and 'UNIT' are synonymous and completely interchangeable in the CLOSE statement. Treatment of sequential mass storage files is logically equivalent to the treatment of a file on tape or analogous sequential media.

1. A CLOSE statement may only be executed for a file in an open mode.

2. For the purpose of showing the effect of various types of CLOSE statements as applied to various storage media, all files are divided into the following categories:

   a) Sequential files whose input or output medium is such that the concepts of rewind and reels/units have no meaning.

b) Sequential files that are entirely contained on one reel/unit.

c) Sequential files that are contained on more than one reel/unit.

d) All Indexed and Relative files.

The results of executing each type of CLOSE for each category of file are summarized in the table below.

CLOSE Statement Type	File Category			
	a)	b)	c)	d)
CLOSE	C	C, E	A, C, E	C
CLOSE WITH LOCK	C, D	C, D, E	A, C, D, E	C,D
CLOSE WITH NO REWIND	F	B, C	A, B, C	

Relationship of Categories of Files and Types of CLOSE statement.

where:
A = Previous Reels/Units are unaffected.
B = No Rewind:- The reel/unit is left in its current position.
C = Close File:- Closing operations are executed.
D = File Lock:- The file cannot be opened again during this execution of the object program.
E = Rewind:- The current reel or analogous device is positioned at its beginning.
F= Optional Phrase Ignored:- The CLOSE statement is executed as if none of the optional phrases is present.

3. The execution of the CLOSE statement causes the value of the I-O status associated with file-name-1 to be updated. (See I-O Status.)

4. If an optional input file is not present, no end-of-file processing is performed for that file and the file position indicator and the current volume pointer are unchanged.

5. Following the successful execution of a CLOSE statement without the REEL or UNIT phrase, the record area associated with file-name-1 is no longer available. The unsuccessful execution of such a CLOSE statement leaves the availability of the record area undefined.

6. Following the successful execution of a CLOSE statement without the REEL or UNIT phrase, the file is removed from the open mode, and the file is no longer associated with the file control block.

7. If more than one file-name-1 is specified in a CLOSE statement, the result of executing this CLOSE statement is the same as if a separate CLOSE statement had been written for each file-name-1 in the same order as specified in the CLOSE statement.

COMPUTE

Function

The COMPUTE statement assigns to one or more data items the value of an arithmetic expression.

General Format

COMPUTE {identifier-1 [ROUNDED]} ... = arithmetic-expression

[ON SIZE ERROR imperative-statement-1 [END-COMPUTE]]

Syntax Rules

1. Identifier-1 must reference either an elementary numeric item or an elementary numeric edited item.

2. If the ROUNDED phrase is specified, identifier-1 cannot be described with a USAGE clause specifying COMPUTATIONAL-1, COMPUTATIONAL-2, COMP-1, COMP-2, SINGLE-REAL or DOUBLE-REAL.

General Rules

1. An arithmetic expression consisting of a single identifier or literal provides a method of setting the value of the data item referenced by identifier-1 equal to the literal or the value of the data item referenced by the single identifier.

2. If more than one identifier is specified for the result of the operation, that is preceding = , the value of the arithmetic expression is computed, and then this value is stored as the new value of each of the data items referenced by identifier-1.

3. The COMPUTE statement allows the user to combine arithmetic operations without the restrictions on composite of operands and/or receiving data items imposed by the arithmetic statements ADD, SUBTRACT, MULTIPLY, and DIVIDE.

4. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See Scope of Statements; The ROUNDED Phrase; The Arithmetic Statements; Overlapping Operands; The SIZE ERROR Phrase; and Multiple Results in Arithmetic Statements.)

Function

The DELETE statement logically removes a record from a mass storage file.

General Format

DELETE file-name-1 RECORD

[INVALID KEY imperative-statement-1 [END-DELETE]]

Syntax Rules

1. The INVALID KEY phrase must not be specified for a DELETE statement which references a file which is in sequential access mode.

2. The INVALID KEY phrase must be specified for a DELETE statement which references a file which is not in sequential access mode and for which an applicable USE AFTER STANDARD EXCEPTION procedure is not specified.

General Rules

1. The file referenced by file-name-1 must be a mass storage file and must be open in the I-O mode at the time of the execution of this statement. (See The OPEN Statement.)

2. For files in the sequential access mode, the last input-output statement executed for file-name-1 prior to the execution of the DELETE statement must have been a successfully executed READ statement. The MOS file system logically removes from the file the record that was accessed by that READ statement.

3. For Relative or Indexed files in random or dynamic access mode, the MOS file system logically removes from the file that record identified by the content of the relative or the prime key, data item associated with file-name-1. If the file does not contain the record specified by the key, the invalid key condition exists.

4. After the successful execution of a DELETE statement, the identified record has been logically removed from the file and can no longer be accessed.

5. The execution of a DELETE statement does not affect the content of the record area.

6. The file position indicator is not affected by the execution of a DELETE statement.

7. The execution of the DELETE statement causes the value of the I-O status associated with file-name-1 to be updated. (See I-O Status.)

8. The END-DELETE phrase delimits the scope of the DELETE statement. (See Scope of Statements.)

9. An attempt to execute a DELETE statement for a record placed in lock mode by another system program or process results in an I-O error indication in the I-O status of the file-name associated with record-name-1. (See I-O Status.)

Function

The DELETE FILE statement removes the specified files physically from the disk.

General Format

DELETE FILE {file-name-1} ...

General Rules

1. File-name-1 must be closed.

2. File-name-1 is immediately marked for removal and is deleted when no processes are connected to it.

3. The execution of the DELETE FILE statement causes the value of the I-O status associated with file-name-1 to be updated. (See I-O Status.)

DISPLAY

Function

The DISPLAY statement causes low volume data to be transferred to an appropriate hardware device.

General Format

```
DISPLAY { identifier-1 } ... [WITH NO ADVANCING]
        { literal-1    }

    [UPON { CONSOLE          }]
          { mnemonic-name-1  }
```

Syntax Rules

1. Mnemonic-name is associated with a hardware device in the SPECIAL-NAMES paragraph in the Environment Division.

2. Literal-1 must not be a figurative constant that begins with the word ALL.

3. If literal-1 is numeric, then it must be integer.

General Rules

1. The DISPLAY statement causes the contents of each operand to be transferred to the hardware device in the order listed.

2. The maximum size of a data transfer during the execution of the DISPLAY statement is 1920 characters. Each identifier-1 which is not defined explicitly as USAGE IS COMPUTATIONAL-1, COMPUTATIONAL-2, SINGLE-REAL or DOUBLE-REAL, or explicitly or implicitly as USAGE IS DISPLAY requires two characters per byte of storage since it is displayed in hexadecimal. Each identifier-1 which is defined as USAGE IS COMPUTATIONAL-1 or SINGLE-REAL rquires fourteen characters. Each identifier-1 which is defined as USAGE IS COMPUTATIONAL-2 or DOUBLE-REAL requires twenty-two characters. Each identifier-1 which is defined explicitly or implicitly as USAGE IS DISPLAY is presented in standard data format characters. If identifier-1 is numeric, it is displayed according to definition, that is, without a sign if unsigned, with a zoned sign if trailing zoned or leading zoned sign, or with a separate sign if defined with a separate sign.

3. If a figurative constant is specified as one of the operands, only a single occurrence of the figurative constant is displayed.

4. If an integer is specified as one of the operands, it is displayed with a trailing zoned sign.

5. If the hardware device is capable of receiving data of the same size as the data item being transferred, then the data item is transferred.

6. If the hardware device is not capable of receiving data of the same size as the data item being transferred, then one of the following applies:

   a) If the size of the data item being transferred exceeds the size of the data that the hardware device is capable of receiving in a single transfer, the data beginning with the leftmost character is stored aligned to the left in the receiving hardware device and additional data is requested.

   b) If the size of the data item that the hardware device is capable of receiving exceeds the size of the data being transferred, the transferred data is stored aligned to the left in the receiving hardware device.

7. When a DISPLAY statement contains more than one operand, the size of the sending item is the sum of the sizes associated with the operands, and the values of the operands are transferred in the sequence in which the operands are encountered.

8. If the UPON phrase is not used, the standard display device is used.

9. If the WITH NO ADVANCING phrase is specified, then the positioning of the hardware device will not be reset to the next line or changed in any other way following the display of the last operand.

10. If the WITH NO ADVANCING phrase is omitted, then after the last operand has been displayed, the positioning of the hardware device will be reset to the leftmost position of the next line of the device.

DIVIDE

Function

The DIVIDE statement divides one numeric data item into others and sets the values of data items equal to the quotient and remainder.

General Format

Format 1:

```
DIVIDE { identifier-1 } INTO  {identifier-2 [ROUNDED] } ...
       { literal-1    }

    [ON SIZE ERROR imperative-statement-1 [END-DIVIDE]]
```

Format 2:

```
DIVIDE { identifier-1 } INTO { identifier-2 }
       { literal-1    }      { literal-2    }

    GIVING {identifier-3 [ROUNDED]} ...

    [ON SIZE ERROR imperative-statement-1 [END-DIVIDE]]
```

Format 3:

```
DIVIDE { identifier-1 } BY { identifier-2 }
       { literal-1    }    { literal-2    }

    GIVING {identifier-3 [ROUNDED]} ...

    [ON SIZE ERROR imperative-statement-1 [END-DIVIDE]]
```

Format 4:

```
DIVIDE  {identifier-1}  INTO  {identifier-2}
        {literal-1   }        {literal-2   }

    GIVING identifier-3 [ROUNDED]

    REMAINDER identifier-4

    [ON SIZE ERROR imperative-statement-1 [END-DIVIDE]]
```

Format 5:

```
DIVIDE  {identifier-1}  BY  {identifier-2}
        {literal-1   }      {literal-2   }

    GIVING identifier-3 [ROUNDED]

    REMAINDER identifier-4

    [ON SIZE ERROR imperative-statement-1 [END-DIVIDE]]
```

Syntax Rules

1. Each identifier must refer to an elementary numeric item, except that any identifier associated with the GIVING or REMAINDER phrase must refer to either an elementary numeric item or an elementary numeric edited item.

2. If the ROUNDED phrase is specified for identifier-2 in format 1 or identifier-3 in the other formats, the identifier cannot be described with a USAGE clause specifying COMPUTATIONAL-1, COMPUTATIONAL-2, COMP-1, COMP-2, SINGLE-REAL or DOUBLE-REAL.

3. Each literal must be a numeric literal.

4. The composite of operands, which is the hypothetical data item resulting from the superimposition of all receiving data items (except the REMAINDER data item) of a given statement aligned on their decimal points, must not contain more than 18 digits.

General Rules

1. When format 1 is used, literal-1 or the value of the data item referenced by identifier-1 is stored in a temporary data item. The value in this temporary data item is then divided into the value of the data item referenced by identifier-2. The value of the dividend (the value of the data item referenced by identifier-2) is replaced by this quotient; similarly the temporary data item is divided into each successive occurrence of identifier-2 in the left-to-right order in which identifier-2 is specified.

2. When format 2 is used, literal-1 or the value of the data item referenced by identifier-1 is divided into literal-2 or the value of the data item referenced by identifier-2 and the result is stored in each data item referenced by identifier-3.

3. When format 3 is used, literal-1 or the value of the data item referenced by identifier-1 is divided by literal-2 or the value of the data item referenced by identifier-2 and the result is stored in each data item referenced by identifier-3.

4. When format 4 is used, literal-1 or the value of the data item referenced by identifier-1 is divided into literal-2 or the value of the data item referenced by identifier-2 and the result is stored in the data item referenced by identifier-3. The remainder is then calculated and the result is stored in the data item referenced by identifier-4. If identifier-4 is subscripted, then the subscripts are evaluated immediately before the remainder is stored in the data item referenced by identifier-4.

5. When format 5 is used, literal-1 or the value of the data item referenced by identifier-1 is divided by literal-2 or the value of the data item referenced by identifier-2 and the division operation continues as specified for format 4 above.

6. Formats 4 and 5 are used when a remainder from the division operation is desired, namely identifier-4. The remainder in COBOL is defined as the result of subtracting the product of the quotient (identifier-3) and the divisor from the dividend. If identifier-3 is defined as a numeric edited item, the quotient used to calculate the remainder is an intermediate field which contains the unedited quotient. If ROUNDED is specified, the quotient used to calculate the remainder is an intermediate field which contains the quotient of the DIVIDE statement, truncated rather than rounded. This intermediate field is defined as a numeric field which contains the same number of digits, the same decimal point location, and the same presence or absence of a sign as the quotient (identifier-3).

7. In formats 4 and 5, the accuracy of the REMAINDER data item (identifier-4) is defined by the calculation described above. Appropriate decimal alignment and truncation (not rounding) will be performed for the value of the data item referenced by identifier-4, as needed.

8. When the ON SIZE ERROR phrase is used in formats 4 and 5, the following rules pertain:

a) If the size error occurs on the quotient, no remainder calculation is meaningful. Thus, the contents of the data items referenced by both identifier-3 and identifier-4 will remain unchanged.

b) If the size error occurs on the remainder, the content of the data item referenced by identifier-4 remains unchanged. However, as with other instances of multiple results of arithmetic statements, the user will have to do his own analysis to recognize which situation has actually occurred.

9. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See Scope of Statements; The Rounded Phrase; The Size Error Phrase; The Arithmetic Statements; Overlapping Operands; Multiple Results in Arithmetic Statements. See also general rules 4 through 6 above for a presentation of the ROUNDED phrase and the ON SIZE ERROR phrase as they pertain to formats 4 and 5.)

EVALUATE

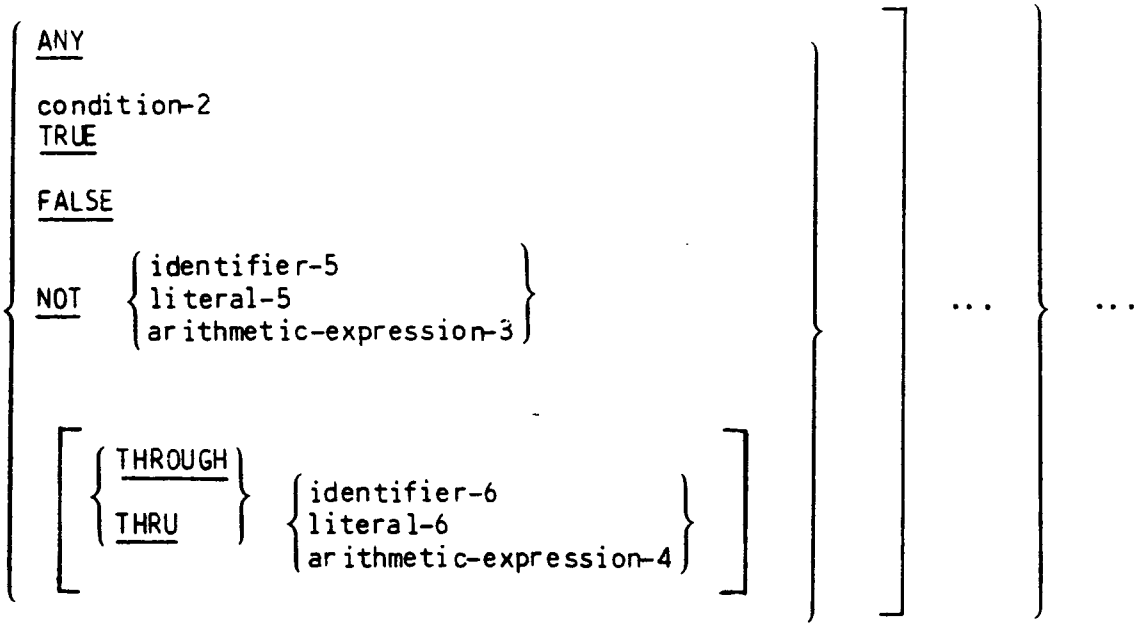
Function

The EVALUATE statement describes a multi-branch, multi-join structure. It can cause multiple conditions to be evaluated. The subsequent action of the object program depends on the results of these evaluations.

General Format

```
EVALUATE { identifier-1 }                [ ALSO { identifier-2 }     ]
         { literal-1    }                [      { literal-2    }     ] ...
         { expression-1 }                [      { expression-2 }     ]
         { TRUE         }                [      { TRUE         }     ]
         { FALSE        }                [      { FALSE        }     ]

  {{WHEN

{ ANY                                                                }
{                                                                    }
{ condition-1                                                        }
{ TRUE                                                               }
{                                                                    }
{ FALSE                                                              }
{                                                                    }
{ [NOT] { identifier-3                  }                            }
{       { literal-3                     }                            }
{       { arithmetic-expression-1       }                            }
{                                                                    }
{     [ { THROUGH } { identifier-4                  } ]              }
{     [ { THRU    } { literal-4                     } ]              }
{     [           } { arithmetic-expression-2       } ]              }
```

```
[ALSO
```




```
imperative-statement-1} ...

[WHEN OTHER imperative-statement-2]

[END-EVALUATE]
```

Syntax Rules

1. The operands or the words TRUE and FALSE which appear before the first WHEN phrase of the EVALUATE statement are referred to individually as selection subjects and collectively, for all those specified, as the set of selection subjects.

2. The operands or the words TRUE, FALSE, and ANY which appear in a WHEN phrase of an EVALUATE statement are referred to individually as selection objects and collectively, for all those specified in a single WHEN phrase, as the set of selection objects.

3. The words THROUGH and THRU are equivalent.

4. Two operands connected by a THROUGH phrase must be of the same class. The two operands thus connected constitute a single selection object.

5. The number of selection objects within each set of selection objects must be equal to the number of selection subjects.

6. Each selection object within a set of selection objects must correspond to the selection subject having the same ordinal position within the set of selection subjects according to the following

rules:

a) Identifiers, literals, or arithmetic expressions appearing within a selection object must be valid operands for comparison to the corresponding operand in the set of selection subjects. (See Relation Condition.)

b) Condition-1, condition-2 or the words TRUE or FALSE appearing as a selection object must correspond to a conditional expression or the words TRUE or FALSE in the set of selection subjects.

c) An data item declared with USAGE IS INDEX clause may not appear as a selection object.

d) The word ANY may correspond to a selection subject of any type.

7. Within the scope of the entire EVALUATE statement (i.e. from the beginning of the outermost EVALUATE statement, if nested, until the present or implicit END-EVALUATE at the end of the outermost); the sum of the number of all of the subjects and the number of objects which are conditions may not be greater than 100.

General Rules

1. The execution of the EVALUATE statement operates as if each selection subject and selection object were evaluated and assigned a numeric or nonnumeric value, a range of numeric or nonnumeric values, or a truth value. These values are determined as follows:

   a) Any selection subject specified by identifier-1, identifier-2 and any selection object specified by identifier-3, identifier-5, without either the NOT or the THROUGH phrases, are assigned the value and class of the data item referenced by the identifier.

   b) Any selection subject specified by literal-1, literal-2 and any selection object specified by literal-3, literal-5 without either the NOT or the THROUGH phrases, are assigned the value and class of the specified literal. If literal-3, literal-5 is the figurative constant ZERO, it is assigned the class of the corresponding selection subject.

   c) Any selection subject in which expression-1, expression-2 is specified as an arithmetic expression and any selection object, without either the NOT or the THROUGH phrases, in which arithmetic-expression-1, arithmetic-expression-3 is specified are assigned a numeric value according to the rules for evaluating an arithmetic expression. (See Arithmetic Expressions.)

   d) Any selection subject in which expression-1, expression-2 is specified as a conditional expression and any selection object in which condition-1, condition-2 is specified are assigned a truth value according to the rules for evaluating conditional expressions. (See Conditional Expressions.)

   e) Any selection subject or any selection object specified by the words TRUE or FALSE is assigned a truth value. The truth value

'true' is assigned to those items specified with the word TRUE, and the truth value 'false' is assigned to those items specified with the word FALSE.

f) Any selection object specified by the word ANY is not further evaluated.

g) If the THROUGH phrase is specified for a selection object, without the NOT phrase, the range of values is all values which, when compared to the selection subject, are greater than or equal to the first operand and less than or equal to the second operand according to the rules for comparison. (See Relation Condition). If the first operand is greater than the second operand, there are no values in the range.

h) If the NOT phrase is specified for a selection object, the values assigned to that item are all values not equal to the value, or range of values, which would have been assigned to the item had the NOT phrase not been specified.

2. The execution of the EVALUATE statement then proceeds as if the values assigned to the selection subjects and selection objects were compared to determine if any WHEN phrase satisfies the set of selection subjects. This comparison proceeds as follows:

a) Each selection object within the set of selection objects for the first WHEN phrase is compared to the selection subject having the same ordinal position within the set of selection subjects. One of the following conditions must be satisfied if the comparison is to be satisfied:

- If the items being compared are assigned numeric or nonnumeric values, or a range of numeric or nonnumeric values, the comparison is satisfied if the value, or one of the range of values, assigned to the selection object is equal to the value assigned to the selection subject according to the rules for comparison. (See Relation Condition.)

- If the items being compared are assigned truth values, the comparison is satisfied if the items are assigned the identical truth value.

- If the selection object being compared is specified by the word ANY, the comparison is always satisfied regardless of the value of the selection subject.

b) If the above comparison is satisfied for every selection object within the set of selection objects being compared, the WHEN phrase containing that set of selection objects is selected as the one satisfying the set of selection subjects.

c) If the above comparison is not satisfied for every selection object within the set of selection objects being compared, that set of selection objects does not satisfy the set of selection

d) This procedure is repeated for subsequent sets of selection objects, in the order of their appearance in the source program, until either a WHEN phrase satisfying the set of selection subjects is selected or until all sets of selection objects are exhausted.

3. After the comparison operation is completed, execution of the EVALUATE statement proceeds as follows:

a) If a WHEN phrase is selected, execution continues with the first imperative-statement-1 following the selected WHEN phrase.

b) If no WHEN phrase is selected and a WHEN OTHER phrase is specified, execution continues with imperative-statement-2.

c) The scope of execution of the EVALUATE statement is terminated when execution reaches the end of the scope of the selected WHEN phrase or WHEN OTHER phrase or when no WHEN phrase is selected and no WHEN OTHER phrase is specified.

Example of the use of EVALUATE statement

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EVAL.
DATA DIVISION.

WORKING-STORAGE SECTION.
01  RESPONSE    PIC X.
01  MONTH       PIC 99.
01  SEASON      PIC X(10).

PROCEDURE DIVISION.
A01.
    DISPLAY "continue ? (Y/N)".
    ACCEPT RESPONSE.
    IF RESPONSE = "N" GO TO A02.
    ACCEPT MONTH.
    EVALUATE MONTH
        WHEN "1" THRU "3"
                        MOVE "WINTER" TO SEASON
        WHEN "4" THRU "6"
                        MOVE "SPRING" TO SEASON
        WHEN "7" THRU "9"
                        MOVE "SUMMER" TO SEASON
        WHEN "10" THRU "12"
                        MOVE "AUTUMN" TO SEASON
        WHEN     OTHER
                        MOVE "ERROR" TO SEASON
    END-EVALUATE.
    DISPLAY "season = " SEASON.
    GO TO A01.
A02.
STOP RUN.
```

Function

The EXIT statement provides a common end point for a series of procedures.

General Format

EXIT

Syntax Rules

1. The EXIT statement must appear in a sentence by itself and comprise the only sentence in the paragraph.

General Rules

1. An EXIT statement serves only to enable the user to assign a procedure-name to a given point in a program. Such an EXIT statement has no other effect on the compilation or execution of the program.

GO TO

Function

The GO TO statement causes control to be transferred from one part of the Procedure Division to another.

General Format

Format 1:

GO TO procedure-name-1

Format 2:

GO TO {procedure-name-1} ... DEPENDING ON identifier-1

Syntax Rules

1. Identifier-1 must reference a numeric elementary item which is an integer.

2. If a GO TO statement represented by format 1 appears in a consecutive sequence of imperative statements within a sentence, it must appear as the last statement in that sequence.

3. The maximum number of procedure-names in a format 2 GO TO statement is 255.

General Rules

1. When a GO TO statement represented by format 1 is executed, control is transferred to procedure-name-1.

2. When a GO TO statement represented by format 2 is executed, control is transferred to procedure-name-1, etc., depending on the value of identifier-1 being 1, 2, ..., n. If the value of identifier-1 is anything other than the positive or unsigned integers 1, 2, ..., n, then no transfer occurs and control passes to the next statement in the normal sequence for execution.

Function

The IF statement causes a condition (see Conditional Expressions) to be evaluated. The subsequent action of the object program depends on whether the value of the condition is true or false.

General Format

```
IF condition-1 THEN { {statement-1} ... }
                    { NEXT SENTENCE     }

  { ELSE {statement-2} ... [END-IF] }
  { ELSE NEXT SENTENCE              }
  { END-IF                          }
```

Syntax Rules

1. Statement-1 and statement-2 represent either an imperative statement or a conditional statement optionally preceded by an imperative statement. A further description of the rules governing statement-1 and statement-2 is given elsewhere. (See Scope of Statements.)

2. The ELSE NEXT SENTENCE phrase may be omitted if it immediately precedes the terminal period of the sentence.

3. If the END-IF phrase is specified, the NEXT SENTENCE phrase must not be specified.

General Rules

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

   a) An END-IF phrase 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.

2. When an IF statement is executed, the following transfers of control occur:

   a) If the condition is true and statement-1 is specified, control is transferred to the first statement of statement-1 and execution continues according to the rules for each statement specified in statement-1. If a procedure branching or conditional statement is executed which causes an explicit transfer of control, control is explicitly transferred in accordance with the rules of that

statement. Upon completion of the execution of statement-1, the ELSE phrase, if specified, is ignored and control passes to the end of the IF statement.

b) If the condition is true and the NEXT SENTENCE phrase is specified instead of statement-1, the ELSE phrase, if specified, is ignored and control passes to the next executable sentence.

c) If the condition is false and statement-2 is specified, statement-1 or its surrogate NEXT SENTENCE is ignored, control is transferred to the first statement of statement-2, and execution continues according to the rules for each statement specified in statement-2. If a procedure branching or conditional statement is executed which causes an explicit transfer of control, control is explicitly transferred in accordance with the rules of that statement. Upon completion of the execution of statement-2, control passes to the end of the IF statement.

d) If the condition is false and the ELSE phrase is not specified, statement-1 is ignored and control passes to the end of the IF statement.

e) If the condition is false and the ELSE NEXT SENTENCE phrase is specified, statement-1 is ignored and control passes to the next executable sentence.

3. Statement-1 and/or statement-2 may contain an IF statement. In this case, the IF statement is said to be nested. More detailed rules on nesting are given in the appropriate paragraph. (See Scope of Statements.)

   IF statements within IF statements may be considered as paired IF, ELSE, and END-IF combinations, proceeding from left to right. Thus, any ELSE or END-IF encountered is considered to apply to the immediately preceding IF that has not been already paired with an ELSE or END-IF.

Function

The INSPECT statement provides the ability to tally or replace occurrences of single characters or groups of characters in a data item.

General Format

Format 1:

```
INSPECT identifier-1 TALLYING {identifier-2 FOR

    { { ALL     }   { identifier-3 }
    { { LEADING }   { literal-1    }
    {
    { CHARACTERS

        [ { BEFORE } INITIAL { identifier-4 }  ]  ...  }  ...  }  ...
        [ { AFTER  }         { literal-2    }  ]       }       }
```

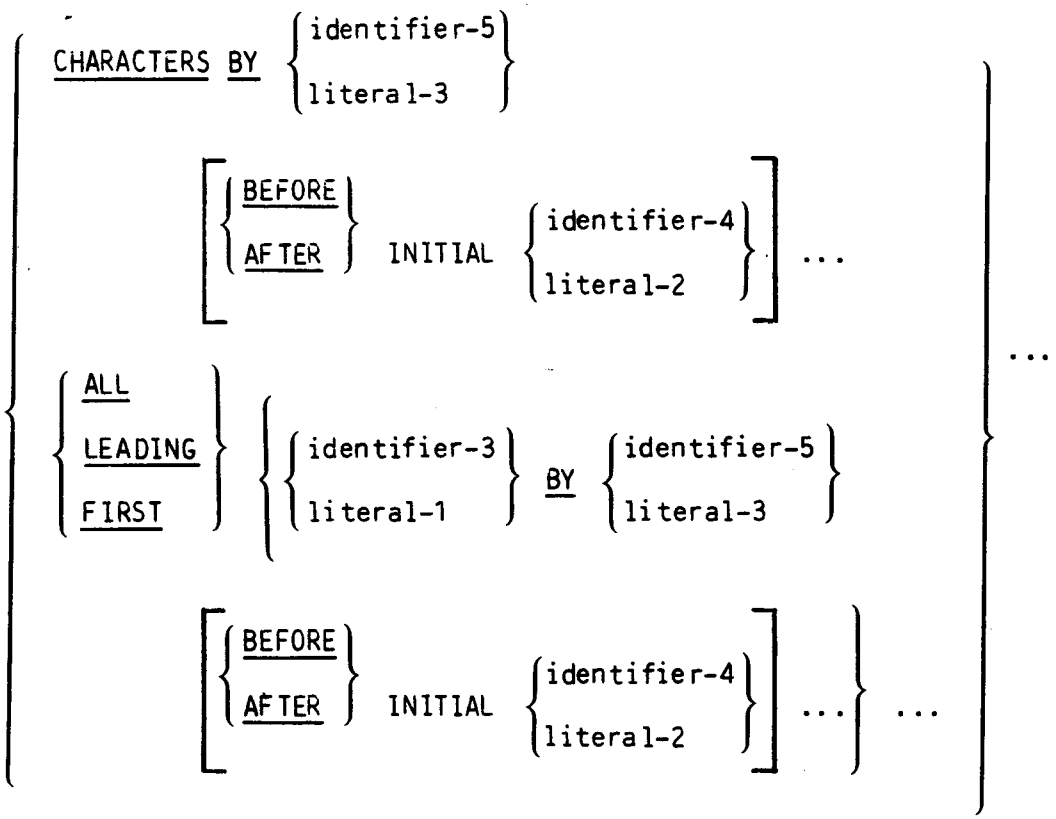
Format 2:

```
INSPECT identifier-1 REPLACING

{ CHARACTERS BY { identifier-5 }
                { literal-3   }

        [ { BEFORE } INITIAL { identifier-4 } ] ...
        [ { AFTER  }         { literal-2   } ]

  { ALL       }   { { identifier-3 } BY { identifier-5 } }
  { LEADING   }   { { literal-1   }    { literal-3   } }
  { FIRST     }

        [ { BEFORE } INITIAL { identifier-4 } ] ... } ...
        [ { AFTER  }         { literal-2   } ]
} ...
```

Format 3:

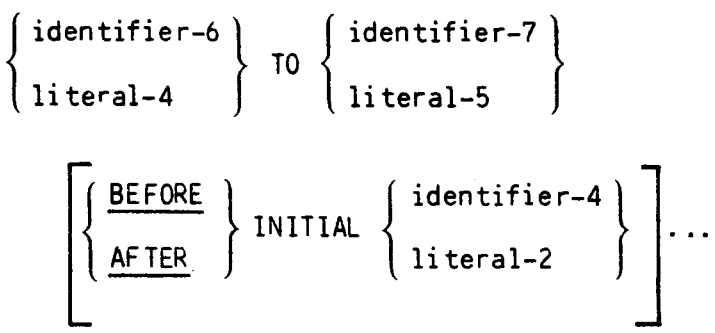
```
INSPECT identifier-1 TALLYING {identifier-2 FOR

  { { ALL       }   { identifier-3 }
  { { LEADING   }   { literal-1   }

  { CHARACTERS

        [ { BEFORE } INITIAL { identifier-4 } ] ... } ... } ...
        [ { AFTER  }         { literal-2   } ]
```

REPLACING



Format 4:

INSPECT identifier-1 CONVERTING



Syntax Rules

All Formats

1. Identifier-1 must reference either a group item or any category of elementary item described, implicitly or explicitly, as USAGE IS DISPLAY.

2. Identifier-3, ..., identifier-n must reference an elementary item described, implicitly or explicitly, as USAGE IS DISPLAY.

3. Each literal must be a nonnumeric literal and must not be a figurative constant that begins with the word ALL. If literal-1, literal-2, or literal-3 is a figurative constant, it refers to an implicit one character data item.

4. No more than one BEFORE phrase and one AFTER phrase can be specified for any one ALL, LEADING, CHARACTERS, FIRST, or CONVERTING phrase.

Formats 1 and 3:

5. Identifier-2 must reference an elementary numeric data item.

Formats 2 and 3:

6. The size of literal-3 or the data item referenced by identifier-5 must be equal to the size of literal-1 or the data item referenced by identifier-3. When a figurative constant is used as literal-3, the size of the figurative constant is equal to the size of literal-1 or the size of the data item referenced by identifier-3.

7. When the CHARACTERS phrase is used, literal-2, literal-3, or the size of the data item referenced by identifier-4, identifier-5 must be one character in length.

Format 4:

8. The size of literal-5 or the data item referenced by identifier-7 must be equal to the size of literal-4 or the data item referenced by identifier-6. When a figurative constant is used as literal-5, the size of the figurative constant is equal to the size of literal-4 or the size of the data item referenced by identifier-6.

9. The same character must not appear more than once either in literal-4 or in the data item referenced by identifier-6.

General Rules

All Formats

1. Inspection (which includes the comparison cycle, the establishment of boundaries for the BEFORE or AFTER phrase, and the mechanism for tallying and/or replacing) begins at the leftmost character position of the data item referenced by identifier-1, regardless of its class, and proceeds from left to right to the rightmost character position as described in general rules 5 through 7.

2. For use in the INSPECT statement, the content of the data item referenced by identifier-1, identifier-3, identifier-4, identifier-5, identifier-6 or identifier-7 will be treated as follows:

   a) If any of identifier-1, identifier-3, identifier-4, identifier-5, identifier-6 or identifier-7 reference an alphabetic or alphanumeric data item, the INSPECT statement treats the contents

of each such identifier as a character-string.

b) If any of identifier-1, identifier-3, identifier-4, identifier-5, identifier-6 or identifier-7 reference alphanumeric edited, numeric edited, or unsigned numeric data items, the data item is inspected as though it had been redefined as alphanumeric (see general rule 2a) and the INSPECT statement had been written to reference the redefined data item.

c) If any of identifier-1, identifier-3, identifier-4, identifier-5, identifier-6 or identifier-7 reference a signed numeric data item, the data item is inspected as though it has been moved to an unsigned numeric data item with length equal to the length of the signed item excluding any separate sign position, and then the rules in general rule 2b had been applied. (See The MOVE Statement.) If identifier-1 is a signed numeric item, the original value of the sign is retained upon completion of the INSPECT statement.

3. In general rules 5 through 15, all references to literal-1, literal-2, literal-3, literal-4 or literal-5 apply equally to the content of the data item referenced by identifier-3, identifier-4, identifier-5, identifier-6 or identifier-7 respectively.

4. Subscripting associated with any identifier is evaluated only once as the first operation in the execution of the INSPECT statement.

Formats 1 and 2

5. During inspection of the content of the data item referenced by identifier-1, each properly matched occurrence of literal-1 is tallied (format 1) or replaced by literal-3 (format 2).

6. The comparison operation to determine the occurrences of literal-1 to be tallied or to be replaced, occurs as follows:

a) The operands of the TALLYING or REPLACING phrase are considered in the order they are specified in the INSPECT statement from left to right. The first literal-1 is compared to an equal number of contiguous characters, starting with the leftmost character position in the data item referenced by identifier-1. Literal-1 matches that portion of the content of the data item referenced by identifier-1 if they are equal, character for character, and:

- If neither LEADING nor FIRST is specified; or

- If the LEADING adjective applies to literal-1 and literal-1 is a leading occurrence as defined in general rules 9 and 12;

- or If the FIRST adjective applies to literal-1 and literal-1 is the first occurrence as defined in general rules 9 and 12.

b) If no match occurs in the comparison of the first literal-1, the comparison is repeated with each successive literal-1, if any, until either a match is found or there is no next successive

literal-1. When there is no next successive literal-1, the character position in the data item referenced by identifier-1, immediately to the right of the leftmost character position considered in the last comparison cycle is considered as the leftmost character position, and the comparison cycle begins again with the first literal-1.

c) Whenever a match occurs, tallying or replacing takes place as described in general rules 9 through 12. The character position in the data item referenced by identifier-1 immediately to the right of the rightmost character position that participated in the match is now considered to be the leftmost character position of the data item referenced by identifier-1, and the comparison cycle starts again with the first literal-1.

d) The comparison operation continues until the rightmost character position of the data item referenced by identifier-1 has participated in a match or has been considered as the leftmost character position. When this occurs, inspection is terminated.

e) If the CHARACTERS phrase is specified, an implied one character operand participates in the cycle described in paragraphs 6a through 6d above as if it has been specified by literal-1, except that no comparison to the content of the data item referenced by identifier-1 takes place. This implied character is considered always to match the leftmost character of the content of the data item referenced by identifier-1 participating in the current comparison cycle.

7. The comparison operation defined in general rule 6 is affected by the BEFORE and AFTER phrase as follows:

a) If neither the BEFORE nor AFTER phrase is specified, literal-1 or the implied operand of the CHARACTERS phrase participates in the comparison operation as described in general rule 6. Literal-1 or the implied operand of the CHARACTERS phrase is first eligible to participate in matching at the leftmost character position of identifier-1.

b) If the BEFORE phrase is specified, the associated literal-1 or the implied operand of the CHARACTERS phrase participates only in those comparison cycles which involve that portion of the content of the data item referenced by identifier-1 from its leftmost character position up to, but not including, the first occurrence of literal-2 within the content of the data item referenced by identifier-1. The position of this first occurrence is determined before the first cycle of the comparison operation described in general rule 6 is begun. If, on any comparison cycle, literal-1 or the implied operand of the CHARACTERS phrase is not eligible to participate, it is considered not to match the content of the data item referenced by identifier-1. If there is no occurrence of literal-2 within the content of the data item referenced by identifier-1, its associated literal-1 or the implied operand of the CHARACTERS phrase participates in the comparison operation as though the BEFORE phrase had not been specified.

c) If the AFTER phrase is specified, the associated literal-1 or the implied operand of the CHARACTERS phrase may participate only in those comparison cycles which involve that portion of the content of the data item referenced by identifier-1 from the character position immediately to the right of the rightmost character position of the first occurrence of literal-2 within the content of the data item referenced by identifier-1 to the rightmost character position of the data item referenced by identifier-1. This is the character position at which literal-1 or the implied operand of the CHARACTERS phrase is first eligible to participate in matching. The position of this first occurrence is determined before the first cycle of the comparison operation described in general rule 6 is begun. If, on any comparison cycle, literal-1 or the implied operand of the CHARACTERS phrase is not eligible to participate, it is considered not to match the content of the data item referenced by identifier-1. If there is no occurrence of literal-2 within the content of the data item referenced by identifier-1, its associated literal-1 or the implied operand of the CHARACTERS phrase is never eligible to participate in the comparison operation.

Format 1

8. The content of the data item referenced by identifier-2 is not initialized by the execution of the INSPECT statement.

9. The rules for tallying are as follows:

   a) If the ALL phrase is specified, the content of the data item referenced by identifier-2 is incremented by one for each occurrence of literal-1 matched within the content of the data item referenced by identifier-1.

   b) If the LEADING phrase is specified, the content of the data item referenced by identifier-2 is incremented by one for the first and each subsequent contiguous occurrence of literal-1 matched within the content of the data item referenced by identifier-1, provided that the leftmost such occurrence is at the point where comparison began in the first comparison cycle in which literal-1 was eligible to participate.

   c) If the CHARACTERS phrase is specified, the content of the data item referenced by identifier-2 is incremented by one for each character matched, in the sense of general rule 6e, within the content of the data item referenced by identifier-1.

10. If identifier-1, identifier-3, or identifier-4 occupies the same storage area as identifier-2, the result of the execution of this statement is undefined, even if they are defined by the same data description entry. (See Overlapping Operands.)

Format 2

11. The required words ALL, LEADING, and FIRST are adjectives that apply to each succeeding BY phrase until the next adjective appears.

12. The rules for replacement are as follows:

    a) When the CHARACTERS phrase is specified, each character matched, in the sense of general rule 6e, in the content of the data item referenced by identifier-1 is replaced by literal-3.

    b) When the adjective ALL is specified, each occurrence of literal-1 matched in the content of the data item referenced by identifier-1 is replaced by literal-3.

    c) When the adjective LEADING is specified, the first and each successive contiguous occurrence of literal-1 matched in the content of the data item referenced by identifier-1 is replaced by literal-3, provided that the leftmost occurrence is at the point where comparison began in the first comparison cycle in which literal-1 was eligible to participate.

    d) When the adjective FIRST is specified, the leftmost occurrence of literal-1 matched within the content of the data item referenced by identifier-1 is replaced by literal-3. This rule applies to each successive specification of the FIRST phrase regardless of the value of literal-1.

13. If identifier-3, identifier-4, or identifier-5 occupies the same storage area as identifier-1, the result of the execution of this statement is undefined, even if they are defined by the same data description entry. (See Overlapping Operands.)

Format 3

14. A format 3 INSPECT statement is interpreted and executed as though two successive INSPECT statements specifying the same identifier-1 had been written with one statement being a format 1 statement with TALLYING phrases identical to those specified in the format 3 statement, and the other statement being a format 2 statement with REPLACING phrases identical to those specified in the format 3 statement. The general rules given for matching and counting apply to the format 1 statement and the general rules given for matching and replacing apply to the format 2 statement.

Format 4

15. A format 4 INSPECT statement is interpreted and executed as though a format 2 INSPECT statement specifying the same identifier-1 had been written with a series of ALL phrases, one for each character of literal-4. The effect is as if each of these ALL phrases referenced, as literal-1, a single character of literal-4 and referenced, as literal-3, the corresponding single character of literal-5. Correspondence between the characters of literal-4 and the characters of literal-5 is by ordinal position within the data item.

16. If identifier-4, identifier-6, or identifier-7 occupies the same storage area as identifier-1, the result of the execution of this statement is undefined, even if they are defined by the same data description entry. (See Overlapping Operands.)

Examples

In each of the following five examples of the INSPECT statement, COUNT-n is assumed to be zero immediately prior to execution of the statement. The results shown for each example, except the last, are the result of executing two successive INSPECT statements.

```
1:  INSPECT ITEM TALLYING
        COUNT-0 FOR ALL "AB" ALL "D"
        COUNT-1 FOR ALL "BC"
        COUNT-2 FOR LEADING "EF"
        COUNT-3 FOR LEADING "B"
        COUNT-4 FOR CHARACTERS

    INSPECT ITEM REPLACING
        ALL "AB" BY "XY" "D" BY "X"
        ALL "BC" BY "VW"
        LEADING "EF" BY "TU"
        LEADING "B" BY "S"
        FIRST "G" BY "R"
        FIRST "G" BY "P"
        CHARACTERS BY "Z"
```

Initial Value of ITEM	COUNT -0	COUNT -1	COUNT -2	COUNT -3	COUNT -4	Final Value of ITEM
EFABDBCGABCFGG	3	1	1	0	5	TUXYXVWRXYZZPZ
BABABC	2	0	0	1	1	SXYXYZ
BBBC	0	1	0	2	0	SSVW

```
2:  INSPECT ITEM TALLYING
        COUNT-0 FOR CHARACTERS
        COUNT-1 FOR ALL "A"

    INSPECT ITEM REPLACING
        CHARACTERS BY "Z"
        ALL "Z" BY "X"
```

Initial Value of ITEM	COUNT-0	COUNT-1	Final Value of ITEM
BBB	3	0	ZZZ
ABA	3	0	ZZZ

3:
```
INSPECT ITEM TALLYING
    COUNT-0 FOR ALL "AB" BEFORE "BC"
    COUNT-1 FOR LEADING "B" AFTER "D"
    COUNT-2 FOR CHARACTERS AFTER "A" BEFORE "C"

INSPECT ITEM REPLACING
    ALL "AB" BY "XY" BEFORE "BC"
    LEADING "B" BY "W" AFTER "D"
    FIRST "E" BY "V" AFTER "D"
    CHARACTERS BY "Z" AFTER "A" BEFORE "C"
```

Initial Value of ITEM	COUNT-0	COUNT-1	COUNT-2	Final Value of ITEM
BBEABDABABBCABEE	3	0	2	BBEXYZXYXYZCABVE
ADDDDC	0	0	4	AZZZZC
ADDDDA	0	0	5	AZZZZZ
CDDDDC	0	0	0	CDDDDC
BDBBBDB	0	3	0	BDWWWDB

4:
```
INSPECT ITEM TALLYING
    COUNT-0 FOR ALL "AB" AFTER "BA" BEFORE "BC"

INSPECT ITEM REPLACING
    ALL "AB" BY "XY" AFTER "BA" BEFORE "BC"
```

Initial Value of ITEM	COUNT-0	Final Value of ITEM
ABABABABC	1	ABABXYABC

5:
```
INSPECT ITEM CONVERTING
    "ABCD" TO "XYZX" AFTER QUOTE BEFORE "@"
```

The above INSPECT is equivalent to the following INSPECT:

```
INSPECT ITEM REPLACING
    ALL "A" BY "X" AFTER QUOTE BEFORE "@"
    ALL "B" BY "Y" AFTER QUOTE BEFORE "@"
    ALL "C" BY "Z" AFTER QUOTE BEFORE "@"
    ALL "D" BY "X" AFTER QUOTE BEFORE "@"
```

Initial Value of ITEM	Final Value of ITEM
AC"AEBDFBCD@AB"D	AC"XEYXFYZX@AB"D

Function

The LOCK statement places file record(s) in lock status thus restricting access by other system programs or processes.

General Format

Format 1 - Relative Organization

```
LOCK  file-name-1  AT KEY {identifier-1} ...
```

Format 2 - Indexed Organization

```
LOCK  file-name-1  [ ON PRIMARY                    ]
                   [ ON ALTERNATE data-name-1      ]

      AT KEY {identifier-1} ...
```

Syntax Rules

1. Data-name-1 may be qualified.

2. In format 1 identifier-1 must reference an unsigned integer data-item whose description does not contain the PICTURE symbol 'P'.

3. In format 2 identifier-1 must be the name of a data item with the equivalent data description to the data item specified as either a primary or an alternate key of file-name-1 depending on the ON clause implicitly or explicitly specified in the LOCK statement.

4. In format 2 data-name-1 must be the name of one of the data items specified as an alternate key of file-name-1.

General Rules

1. A LOCK statement may only be executed for a file in an open mode.

2. The LOCK statement restricts the access to indicated records of file-name-1 by other programs to input only. In fact, only the program executing the LOCK statement can perform output operations on it.

3. In format 2 when the ON clause is not specified, ON PRIMARY is assumed.

4. The AT KEY phrase defines the value of the key(s) of the record(s) to be locked. In format 1 the record is locked even if it does not exist in the file.

5. In format 2 if the ALTERNATE phrase is specified, the record indicated by the key value in the AT KEY phrase must exist. For a primary key value the record is locked even if it does not exist in the file.

6. In format 2 if data-name-1 was defined with DUPLICATES and records exist with that alternate key value, the first record written containing that value is locked.

7. A record is locked until it is unlocked by the program or process that locked it in one of the following ways:

   a) By execution of an UNLOCK statement specifying the same key value; or

   b) By execution of a REWRITE statement specifying the same key value that contains the NO LOCK phrase; or

   c) By execution of a CLOSE statement for the file; or

   d) By termination.

8. Any program or process that attempts to execute a LOCK statement for a record locked by another program or process receives an I-O error indication in the I-O status associated with file-name-1. (See I-O Status.)

Function

The MOVE statement transfers data, in accordance with the rules of editing, to one or more data areas.

General Format

Format 1:

```
MOVE { identifier-1 } TO {identifier-2} ...
     { literal-1    }
```

Format 2:

```
     . { CORRESPONDING }
MOVE   { CORR          }  identifier-1 TO identifier-2
```

Syntax Rules

1. Literal-1 or the data item referenced by identifier-1 represents the sending area. The data item referenced by identifier-2 represents the receiving area

2. CORR is an abbreviation for CORRESPONDING.

3. When the CORRESPONDING phrase is used, all identifiers must be group items.

4. An index data item must not appear as an operand of a MOVE statement.

General Rules

1. If the CORRESPONDING phrase is used, selected items within identifier-1 are moved to selected items within identifier-2, according to the rules specified in the appropriate paragraph. (See The CORRESPONDING Phrase.) The results are the same as if the user had referred to each pair of corresponding identifiers in separate MOVE statements.

2. Literal-1 or the content of the data item referenced by identifier-1 is moved to the data item referenced by each identifier-2 in the order in which it is specified. The rules referring to identifier-2 also apply to the other receiving areas. Any subscripting associated with identifier-2 is evaluated immediately before the data is moved to the respective data item.
   Any subscripting associated with identifier-1 is evaluated only once, immediately before data is moved to the first of the receiving operands.

The result of the statement

```
MOVE a (b) TO b, c (b)
```

is equivalent to:

```
MOVE a (b) TO temp
MOVE temp TO b
MOVE temp TO c (b)
```

where 'temp' is an intermediate result item provided by the compiler.

3. Any move in which the receiving operand is an elementary item and the sending operand is either a literal or an elementary item is an elementary move. Every elementary item belongs to one of the following categories: numeric, alphabetic, alphanumeric, numeric edited, alphanumeric edited (See The PICTURE Clause). Numeric literals belong to the category numeric and nonnumeric literals belong to the category alphanumeric. The figurative constant ZERO (ZEROS, ZEROES) belongs to the category numeric. The figurative constant SPACE (SPACES) belongs to the category alphabetic. All other figurative constants belong to the category alphanumeric.

   The following rules apply to an elementary move between these categories:

   a) The figurative constant SPACE, a numeric edited, alphanumeric edited, or alphabetic data item must not be moved to a numeric or numeric edited data item.

   b) A numeric literal, the figurative constant ZERO, a numeric data item or a numeric edited data item must not be moved to an alphabetic data item.

   c) A noninteger numeric literal or a noninteger numeric data item must not be moved to an alphanumeric or alphanumeric edited data item.

   d) All other elementary moves are legal and are performed according to the rules given in general rule 4.

4. Any necessary conversion of data from one form of internal representation to another takes place during legal elementary moves, along with any editing specified for the receiving data item:

   a) When an alphanumeric edited or alphanumeric item is a receiving item, alignment and any necessary space filling takes place as previously defined. (See Standard Alignment Rules.) If the sending item is described as being signed numeric, the operational sign is not moved; if the operational sign occupies a separate character position that character is not moved and the size of the sending item is considered to be one less than its actual size. If the usage of the sending operand is different from that of the receiving operand, conversion of the sending operand to the internal representation of the receiving operand takes place.

   b) When a numeric or numeric edited item is the receiving item, alignment by decimal point and any necessary zero-filling takes place as previously defined except where zeroes are replaced

because of editing requirements. (See Standard Alignment Rules.)

- When a signed numeric item is the receiving item, the sign of the sending item is placed in the receiving item. (See The SIGN Clause.) Conversion of the representation of the sign takes place as necessary. If the sending item is unsigned, a positive sign is generated for the receiving item.

- When an unsigned numeric item is the receiving item, the absolute value of the sending item is moved and no operational sign is generated for the receiving item.

- When the sending data item is described as alphanumeric, data is moved as if the sending item were described as an unsigned numeric integer.

c) When a receiving item is described as alphabetic, justification and any necessary space filling takes place as previously defined. (See Standard Alignment Rules.)

5. Any move that is not an elementary move is treated exactly as if it were an alphanumeric to alphanumeric elementary move, except that there is no conversion of data from one form of internal representation to another. In such a move, the receiving area will be filled without consideration for the individual elementary or group items contained within either the sending or receiving area, except as noted in the OCCURS clause. (See The OCCURS Clause.) If the sending operand is a signed integer literal, the form of the integer is display with a trailing zoned sign.

6. Summary of the legality of the various types of MOVE statements.

CATEGORY OF SENDING OPERAND	CATEGORY OF RECEIVING DATA ITEM		
	ALPHABETIC	ALPHANUMERIC EDITED ALPHANUMERIC	NUMERIC INTEGER NUMERIC NONINTEGER NUMERIC EDITED
ALPHABETIC	Yes/4c	Yes/4a	No/3a
ALPHANUMERIC	Yes/4c	Yes/4a	Yes/4b
ALPHANUMERIC EDITED	Yes/4c	Yes/4a	No/3a
NUMERIC INTEGER	No/3b	Yes/4a	Yes/4b
NUMERIC NONINTEGER	No/3b	No/3c	Yes/4b
NUMERIC EDITED	No/3b	Yes/4a	No/3a

The general rule reference indicates the rule that prohibits the move or describes the behavior of a legal move.

MULTIPLY

Function

The MULTIPLY statement causes numeric data items to be multiplied and sets the values of data items equal to the results.

General Format

Format 1:

```
MULTIPLY { identifier-1 } BY {identifier-2 [ROUNDED]} ...
         { literal-1    }

   [ON SIZE ERROR imperative-statement-1 [END-MULTIPLY]]
```

Format 2:

```
MULTIPLY { identifier-1 } BY { identifier-2 }
         { literal-1    }    { literal-2    }

    GIVING {identifier-3 [ROUNDED]} ...

    [ON SIZE ERROR imperative-statement-1 [END-MULTIPLY]]
```

Syntax Rules

1. Each identifier must refer to a numeric elementary item, except that in format 2 each identifier following the word GIVING must refer to either an elementary numeric item or an elementary numeric edited item.

2. Each literal must be a numeric literal.

3. The composite of operands, which is that hypothetical data item resulting from the superimposition of all receiving data items of a given statement aligned on their decimal points, must not contain more than eighteen (18) digits.

General Rules

1. When format 1 is used, literal-1 or the value of the data item referenced by identifier-1 is stored in a temporary data item. The value in this temporary data item is then multiplied by the value of the data item referenced by identifier-2. The value of the multiplier (the value of the data item referenced by identifier-2) is replaced

by this product; similarly, the temporary data item is multiplied by each successive occurrence of identifier-2 in the left-to-right order in which identifier-2 is specified.

2. When format 2 is used, literal-1 or the value of the data item referenced by identifier-1 is multiplied by literal-2 or the value of the data item referenced by identifier-2 and the result is stored in the data items referenced by each identifier-3.

3. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See Scope of Statements; The ROUNDED Phrase; The Arithmetic Statements; Overlapping Operands; The SIZE ERROR Phrase; and Multiple Results in Arithmetic Statements.)

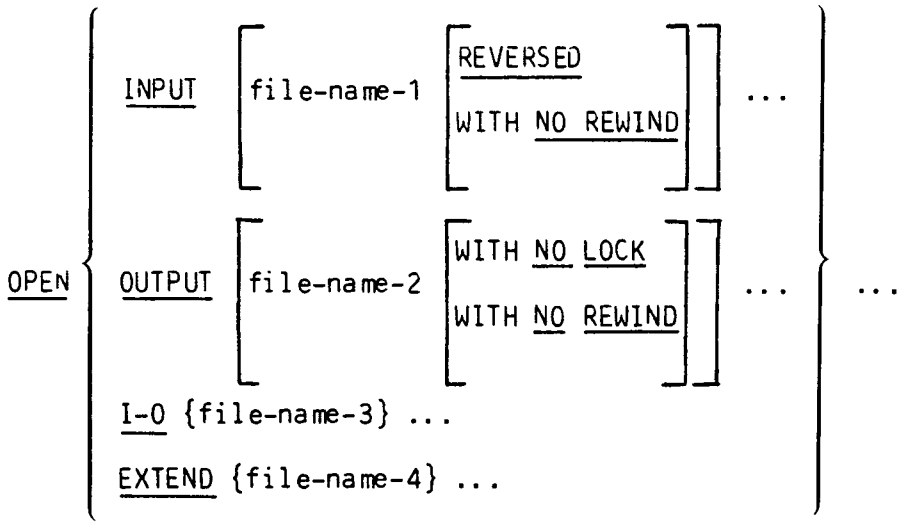
OPEN

Function

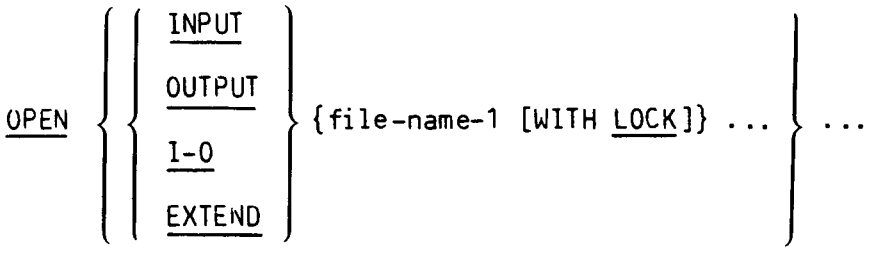
The OPEN statement initiates the processing of files.

General Format

Format 1 - Sequential Organization



Format 2 - Relative and Indexed Organization



Syntax Rules

1. The files referenced in the OPEN statement need not all have the same organization or access.

Format 1

2. The REVERSED or I-O phrase must not be specified for a TEXT-FILE.

3. The OUTPUT phrase is the only one allowed for a file assigned to SYSPRINT1, SYSPRINT2, SYSPRINT3, SYSPRINT4 or WS-PRINTER.

4. The NO LOCK phrase may only be specified for a TEXT-FILE or a file assigned to SYSPRINT1, SYSPRINT2, SYSPRINT3 or SYSPRINT4.

5. The EXTEND phrase may only be used for sequential files for which the LINAGE clause has not been specified.

Format 2

6. The EXTEND phrase may only be used for files in sequential access mode.

General Rules

1. The successful execution of an OPEN statement determines the availability of the file and results in the file being in an open mode. The successful execution of an OPEN statement associates the file with the file-name through the file control block.

   If the file is assigned to a device, the open is successful if the device is present and available. Otherwise a file is available if it is physically present and is recognized by the I-O control system. The table below shows the results of opening available and unavailable files.

Type of OPEN	File is Available	File is Unavailable
INPUT	Normal open	Open is unsuccessful
INPUT (optional file)	Normal open	Normal open; the first read causes the at end condition or invalid key cond.
I-O	Normal open	Open is unsuccessful
I-O (optional file)	Normal open	Open causes the file to be created
OUTPUT	Normal open then truncation at beginning	Open causes the file to be created
EXTEND	Normal open	Open is unsuccessful
EXTEND (optional file)	Normal open	Open causes the file to be created

Availability of a file

2. The successful execution of an OPEN statement makes the associated record area available to the program. If the file control block associated with file-name is an external file control block, there is only one record area associated with the file control block for the run unit.

3. An OPEN statement must be successfully executed prior to the execution of any of the permissible input-output statements. In the table below Permissible Statements, 'X' at an intersection indicates that the specified statement may be used with the open mode given at the top of the column. The '*' indicates not permitted for a TEXT-FILE.

File Organization	File Access Mode	Statement	Open Mode			
			Input	Output	I-O	Extend
Sequential	Sequential	READ	X		X	
		WRITE		X		X
		REWRITE			X*	
Relative Indexed	Sequential	READ	X		X	
		WRITE		X		
		REWRITE			X	
		START	X		X	
		DELETE			X	
	Random	READ	X		X	
		WRITE		X	X	
		REWRITE			X	
		DELETE			X	
		LOCK	X	X	X	
		UNLOCK	X	X	X	
	Dynamic	READ	X		X	
		WRITE		X	X	
		REWRITE			X	
		START	X		X	
		DELETE			X	
		LOCK	X	X	X	
		UNLOCK	X	X	X	

Permissible Statements

4. Execution of the OPEN statement does not obtain or release the first data record.

5. When a file is not in an open mode, no statement may be executed which references the file, either explicitly or implicitly, except for a MERGE or SORT statement with the USING or GIVING phrase, or an OPEN statement.

6. A file may be opened with the INPUT, OUTPUT, EXTEND, and I-O phrases in the same run unit. Following the initial execution of an OPEN statement for a file, each subsequent OPEN statement execution for that same file must be preceded by the execution of a CLOSE statement, without the LOCK phrase, for that file.

   The only exception is when there is an input-output statement for a file due to a problem on a remote machine (see I-O Status, value 9B). For the sequential file the execution of an OPEN statement is allowed without the previous successful execution of a CLOSE statement. For the relative and indexed files, if the file control entry for the file specifies ASSIGN TO data-name, the execution of an OPEN statement is allowed without the previous successful execution of a CLOSE statement.

7. If a file opened with the INPUT phrase is an optional file which is not present, the OPEN statement sets the file position indicator to indicate that an optional file is not present.

8. When files are opened with the INPUT or I-O phrase, the file position indicator is set to the first record currently existing within the file. For an indexed file the prime record key is established as the key of reference. If no records exist in the file, the file position indicator is set such that the next executed sequential READ statement for the file will result in an AT END condition.

9. When the EXTEND phrase is specified, the OPEN statement positions the file immediately after the last logical record for that file. Where the last logical record for a file is :

   a) In a sequential file, the last record written in the file.

   b) In a relative file, the record with the highest relative record number.

   c) In an indexed file, the record with the highest prime record key.

10. The OPEN statement with the I-O phrase must reference a mass storage file. The execution of the OPEN statement with the I-O phrase places the referenced file in the open mode for both input and output operations.

11. For an optional file that is unavailable, the successful execution of an OPEN statement with an EXTEND or I-O phrase creates the file. This creation takes place as if the following statements were executed in the order shown:

```
OPEN OUTPUT file-name.
CLOSE file-name.
```

These statements are followed by execution of the OPEN statement specified in the source program.

The successful execution of an OPEN statement with an OUTPUT phrase creates the file. After the successful creation of a file, that file contains no data records.

12. The execution of the OPEN statement causes the value of the I-O status associated with file-name to be updated. (See I-O Status.)

13. If more than one file-name is specified in an OPEN statement, the result of executing this OPEN statement is the same as if a separate OPEN statement had been written for each file-name in the same order as specified in the OPEN statement.

14. The maximum record size for a file is established at the time the file is created and must not subsequently be changed.

Format 1

15. If label records are specified for the file, the beginning labels are processed as follows:

    a) When the INPUT phrase is specified, the execution of the OPEN statement causes the labels to be checked in a manner to be defined.

    b) When the OUTPUT phrase is specified, the execution of the OPEN statement causes the labels to be written in a manner to be defined.

    The behavior of the OPEN statement when label records are specified but not present, or when label records are not specified but are present, is undefined.

16. If during the execution of an OPEN statement a file attribute condition occurs, the execution of the OPEN statement is unsuccessful. (See The File Attribute Conflict Condition.)

17. The REVERSED and NO REWIND phrases may only be used with sequential single reel/unit files.

18. The REVERSED and NO REWIND phrases will be ignored if it does not apply to the storage medium on which the file resides.

19. If the storage medium for the file permits rewinding, the following rules apply:

    a) When neither the REVERSED, the EXTEND nor the NO REWIND phrase is specified, execution of the OPEN statement causes the file to be positioned at its beginning.

    b) When the NO REWIND phrase is specified, execution of the OPEN statement does not cause the file to be repositioned; the file must be already positioned at its beginning prior to execution of the OPEN statement.

    c) When the REVERSED phrase is specified, the file is positioned at its end by execution of the OPEN statement.

20. When the REVERSED phrase is specified, the subsequent READ statements for the file make the data records of the file available in reversed order; that is starting with the last record.

21. When the EXTEND phrase is specified and the LABEL RECORDS clause indicates label records are present, the execution of the OPEN statement includes the following steps:

    a) The beginning file labels are processed only in the case of a single reel/unit file.

    b) The beginning reel/unit labels on the last existing reel/unit are processed as though the file was being opened with the INPUT phrase.

    c) The existing ending file labels are processed as though the file is being opened with the INPUT phrase. These labels are then deleted.

    d) Processing then proceeds as though the file had been opened with the OUTPUT phrase.

22. When the I-O phrase is specified and the LABEL RECORDS clause indicates label records are present, the execution of the OPEN statement includes the following steps:

    a) The labels are checked .

    b) The new labels are written .

23. Upon successful execution of the OPEN statement, the current volume pointer is set:

    a) To point to the first or only reel/unit for an available input or I-O file.

    b) To point to the reel/unit containing the last logical record for an extend file.

c) To point to the new reel/unit for an unavailable output, I-O, or extend file.

Format 2

24. If during the execution of an OPEN statement a file attribute conflict condition occurs, the execution of the OPEN statement is unsuccessful. (See File Attribute Conflict Condition.)

25. When the WITH LOCK phrase is specified for a file in an OPEN statement, successful execution of that OPEN statement restricts access to that file to the program executing that OPEN statement.

    A file is locked until it is unlocked by the program or process that locked it by execution of a CLOSE statement for the file; or by termination.

26. The execution of an OPEN statement for a file in LOCK status is unsuccessful and the I-O status associated with the file-name is updated appropriately.

Function

The PERFORM statement is used to transfer control explicitly to one or more procedures and to return control implicitly whenever execution of the specified procedure is complete. The PERFORM statement is also used to control execution of one or more imperative statements which are within the scope of that PERFORM statement.

General Format

Format 1

```
PERFORM [procedure-name-1 [{THROUGH} procedure-name-2]]
                           {THRU   }

    [imperative-statement-1 END-PERFORM]
```

Format 2

```
PERFORM [procedure-name-1 [{THROUGH} procedure-name-2]]
                           {THRU   }

    {identifier-1} TIMES [imperative-statement-1 END-PERFORM]
    {integer-1   }
```

Format 3

```
PERFORM [procedure-name-1 [{THROUGH} procedure-name-2]]
                           {THRU   }

    UNTIL condition-1 [imperative-statement-1 END-PERFORM]
```

Format 4

```
PERFORM [procedure-name-1 [{THROUGH} procedure-name-2]]
                           {THRU   }

    VARYING {identifier-2 } FROM {identifier-3 }
            {index-name-1 }      {index-name-2 }
                                 {literal-1    }

            BY {identifier-4} UNTIL condition-1
               {literal-2   }

    [AFTER {identifier-5 } FROM {identifier-6 }
           {index-name-3 }      {index-name-4 }
                                {literal-3    }

            BY {identifier-7} UNTIL condition-2] ...
               {literal-4   }

    [imperative-statement-1 END-PERFORM]
                            ----------
```

Syntax Rules

1. If procedure-name-1 is omitted, imperative-statement-1 and the END-PERFORM phrase must be specified; if procedure-name-1 is specified, imperative-statement-1 and the END-PERFORM phrase must not be specified.

2. In format 4, if procedure-name-1 is omitted, the AFTER phrase must not be specified.

3. Each identifier represents a numeric elementary item described in the Data Division. In format 2, identifier-1 must be described as a numeric integer.

4. Each literal represents a numeric literal.

5. The words THROUGH and THRU are equivalent.

6. If an index-name is specified in the VARYING or AFTER phrase, then:

   a) The identifier in the associated FROM and BY phrases must reference an integer data item.

b) The literal in the associated FROM phrase must be a positive integer.

c) The literal in the associated BY phrase must be a nonzero integer.

7. If an index-name is specified in the FROM phrase, then:

   a) The identifier in the associated VARYING or AFTER phrase must reference an integer data item.

   b) The identifier in the associated BY phrase must reference an integer data item.

   c) The literal in the associated BY phrase must be an integer.

8. Literal in the BY phrase must not be zero.

9. Condition-1, condition-2, ... , may be any conditional expression. (See Conditional Expressions.)

10. Where procedure-name-1 and procedure-name-2 are both specified and either is the name of a procedure in the declaratives portion of the Procedure Division, both must be procedure-names in the same declarative section.

11. The value of integer-1 in format 2 must not be greater than 32767.

General Rules

1. The data items referenced by identifier-4 and identifier-7 must not have a zero value.

2. If an index-name is specified in the VARYING or AFTER phrase, and an identifier is specified in the associated FROM phrase, the data item referenced by the identifier must have a positive value.

3. When procedure-name-1 is specified, the PERFORM statement is referred to as an out-of-line PERFORM statement; when procedure-name-1 is omitted, the PERFORM statement is referred to as an in-line PERFORM statement.

4. The statements contained within the range of procedure-name-1 (through procedure-name-2 if specified) for an out-of-line PERFORM statement or contained within the PERFORM statement itself for an in-line PERFORM statement are referred to as the specified set of statements.

5. The END-PERFORM phrase delimits the scope of the in-line PERFORM statement. (See Scope of Statements.)

6. An in-line PERFORM statement functions according to the following general rules for an otherwise identical out-of-line PERFORM statement, with the exception that the statements contained within the in-line PERFORM statement are executed in place of the statements contained within the range of procedure-name-1 (through procedure-

name-2 if specified). Unless specifically qualified by the word in-line or out-of-line, all the general rules which apply to the out-of-line PERFORM statement also apply to the in-line PERFORM statement.

7. When the PERFORM statement is executed, control is transferred to the first statement of the specified set of statements (except as indicated in general rules 10b, 10c, and 10d). This transfer of control occurs only once for each execution of a PERFORM statement. For those cases where a transfer of control to the specified set of statements does take place, an implicit transfer of control to the end of the PERFORM statement is established as follows:

   a) If procedure-name-1 is a paragraph-name and procedure-name-2 is not specified, the return is after the last statement of procedure-name-1.

   b) If procedure-name-1 is a section-name and procedure-name-2 is not specified, the return is after the last statement of the last paragraph in procedure-name-1.

   c) If procedure-name-2 is specified and it is a paragraph-name, the return is after the last statement of the paragraph.

   d) If procedure-name-2 is specified and it is a section-name, the return is after the last statement of the last paragraph in the section.

   e) If an in-line PERFORM statement is specified, an execution of the PERFORM statement is completed after the last statement contained within it has been executed.

8. There is no necessary relationship between procedure-name-1 and procedure- name-2 except that a consecutive sequence of operations is to be executed beginning at the procedure named procedure-name-1 and ending with the execution of the procedure named procedure-name-2. In particular, GO TO and PERFORM statements may occur between procedure-name-1 and the end of procedure-name-2. If there are two or more logical paths to the return point then procedure-name-2 may be the name of a paragraph consisting of the EXIT statement, to which all of these paths must lead.

9. If control passes to the specified set of statements by means other than a PERFORM statement, control will pass through the last statement of the set to the next executable statement as if no PERFORM statement referenced the set.

10. The PERFORM statements operate as follows:

   a) Format 1 is the basic PERFORM statement. The specified set of statements referenced by this type of PERFORM statement is executed once and then control passes to the end of the PERFORM statement.

   b) Format 2 is the PERFORM ... TIMES. The specified set of statements is performed the number of times specified by integer-1 or

by the initial value of the data item referenced by identifier-1 for that execution. If at the time of the execution of a PERFORM statement, the value of the data item referenced by identifier-1 is equal to zero or is negative, control passes to the end of the PERFORM statement, and, if the value is greater than 32767, the value will be truncated. Following the execution of the specified set of statements the specified number of times, control is transferred to the end of the PERFORM statement.

During execution of the PERFORM statement, reference to identifier-1 cannot alter the number of times the specified set of statements is to be executed from that which was indicated by the initial value of the data item referenced by identifier-1.

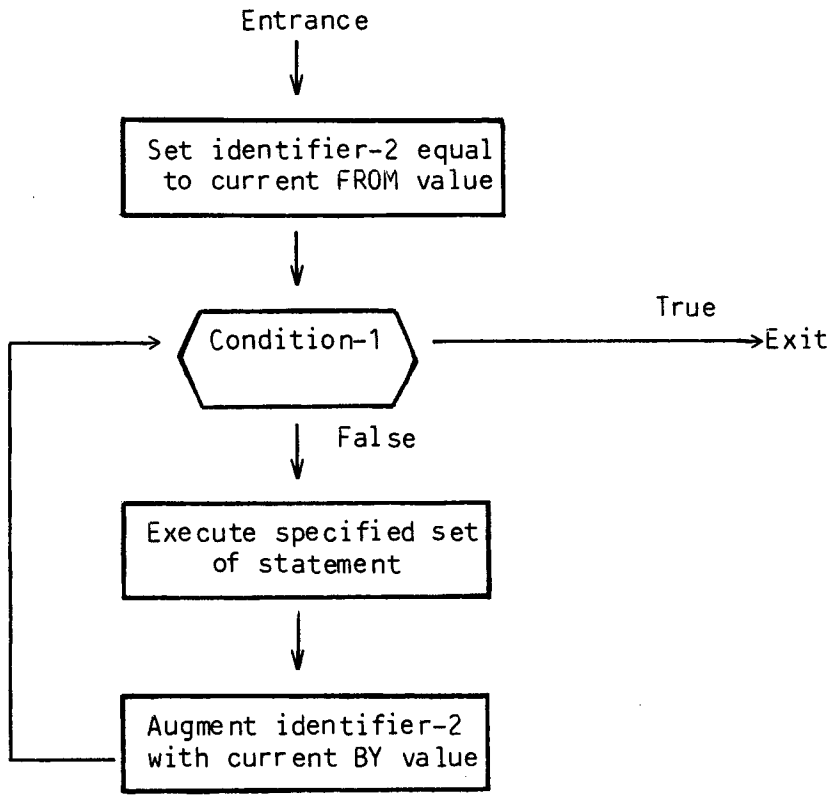
c) Format 3 is the PERFORM ... UNTIL. The specified set of statements is performed until the condition specified by the UNTIL phrase is true. When the condition is true, control is transferred to the end of the PERFORM statement. If the condition is true when the PERFORM statement is entered no transfer to procedure-name-1 takes place, and control is passed to the end of the PERFORM statement. Any subscripting associated with the operands specified in condition-1 is evaluated each time the condition is tested.

d) Format 4 is the PERFORM ... VARYING. This variation of the PERFORM statement is used to augment the values referenced by one or more identifiers or index-names in an orderly fashion during the execution of a PERFORM statement. In the following discussion, every reference to identifier as the object of the VARYING, AFTER, and FROM (current value) phrases also refers to index-names. If index-name-1 or index-name-3 is specified, the value of the associated index at the beginning of the PERFORM statement must be set to an occurrence number of an element in the table. If index-name-2 or index-name-4 is specified, the value of the data item referenced by identifier-2 or identifier-5 at the beginning of the PERFORM statement must be equal to an occurrence number of an element in a table associated with index-name-2 or index-name-4. Subsequent augmentation, as described below, of index-name-1 or index-name-3 must not result in the associated index being set to a value outside the range of the table associated with index-name-1 or index-name-3; except that, at the completion of the PERFORM statement, the index associated with index-name-1 may contain a value that is outside the range of the associated table by one increment or decrement value. If identifier-2 or identifier-5 is subscripted, the subscripts are evaluated each time the content of the data item referenced by the identifier is set or augmented. If identifier-3, identifier-4, identifier-6, or identifier-7 is subscripted, the subscripts are evaluated each time the content of the data item referenced by the identifier is used in a setting or augmenting operation. Any subscripting associated with the operands specified in condition-1 or condition-2 is evaluated each time the condition is tested.

Representations of the actions of several types of format 4 PERFORM statements are given below.

1. The VARYING option of a PERFORM statement having one condition.

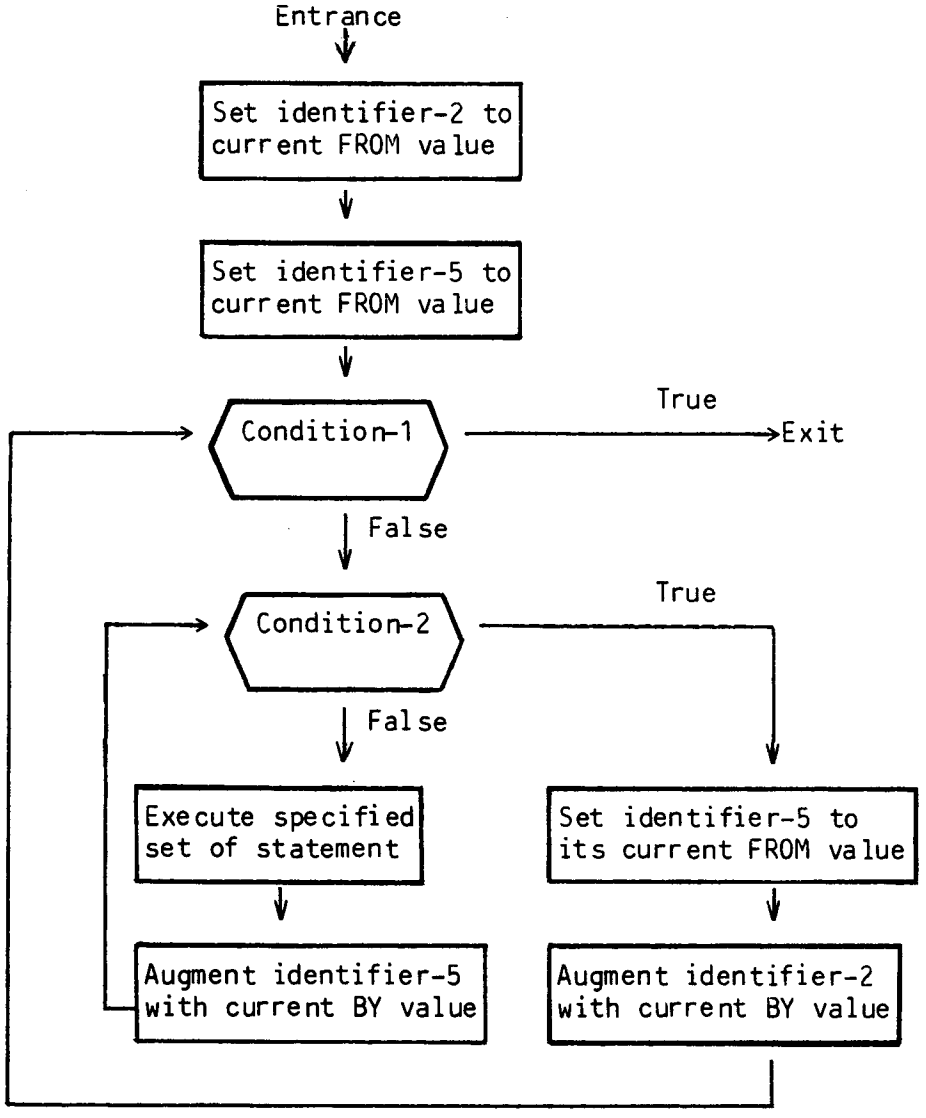
When the data item referenced by one identifier is varied, the content of the data item referenced by identifier-2 is set to literal-1 or the current value of the data item referenced by identifier-3 at the point of initial execution of the PERFORM statement; then, if the condition of the UNTIL phrase is false, the specified set of statements is executed once. The value of the data item referenced by identifier-2 is augmented by the specified increment or decrement value (literal-2 or the value of the data item referenced by identifier-4) and condition-1 is evaluated again. The cycle continues until this condition is true, at which point control is transferred to the end of the PERFORM statement. If condition-1 is true at the beginning of execution of the PERFORM statement, control is transferred to the end of the PERFORM statement.



2. The VARYING option of a PERFORM statement having two conditions.

When the data items referenced by two identifiers are varied, the content of the data item referenced by identifier-2 is set to literal-1 or the current value of the data item referenced by identifier-3 and then the content of the data item referenced by identifier-5 is set to literal-3 or the current value of the data item referenced by identifier-6. After the contents of the data items referenced by the identifiers have been set, condition-1 is

evaluated; if true, control is transferred to the end of the PERFORM statement; if false, condition-2 is evaluated. If condition-2 is false, the specified set of statements is executed once, then the content of the data item referenced by identifier-5 is augmented by literal-4 or the content of the data item referenced by identifier-7 and condition-2 is evaluated again. This cycle of evaluation and augmentation continues until this condition is true. When condition-2 is true, the content of the data item referenced by identifier-2 is augmented by literal-2 or the content of the data item referenced by identifier-4, the content of the data item referenced by identifier-5 is set to literal-3 or the current value of the data item referenced by identifier-6, and condition-1 is reevaluated. The PERFORM statement is completed if condition-1 is true; if not, the cycle continues until condition-1 is true.



At the termination of the PERFORM statement, the data item referenced by identifier-5 contains literal-3 or the current value of the data item referenced by identifier-6. The data item referenced by

identifier-2 contains a value that exceeds the last used setting by one increment or decrement value, unless condition-1 was true when the PERFORM statement was entered, in which case, the data item referenced by identifier-2 contains literal-1 or the current value of the data item referenced by identifier-3.

After the completion of a PERFORM statement, each data item varied by an AFTER phrase contains the current value of the data item referenced by the identifier in the associated FROM phrase. The data item referenced by identifier-2 has a value that exceeds its last used setting by one increment or decrement value, unless condition-1 is true when the PERFORM statement is entered, in which case the data item referenced by identifier-2 contains the current value of the data item referenced by identifier-3.

During the execution of the specified set of statements associated with the PERFORM statement, any change to the VARYING variable (the data item referenced by identifier-2 and index-name-1), the BY variable (the data item referenced by identifier-4), the AFTER variable (the data item referenced by identifier-5 and index-name-3), or the FROM variable (the data item referenced by identifier-3 and index-name-2) will be taken into consideration and will affect the operation of the PERFORM statement.

When the data items referenced by two identifiers are varied, the data item referenced by identifier-5 goes through a complete cycle (FROM, BY, UNTIL) each time the content of the data item referenced by identifier-2 is varied. When the contents of three or more data items referenced by identifiers are varied, the mechanism is the same as for two identifiers except that the data item being varied by each AFTER phrase goes through a complete cycle each time the data item being varied by the preceding AFTER phrase is augmented.

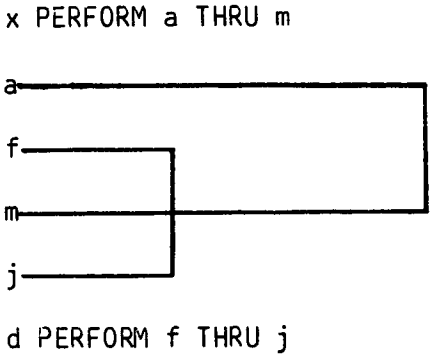
11. The range of a PERFORM statement consists logically of all those statements that are executed as a result of executing the PERFORM statement through execution of the implicit transfer of control to the end of the PERFORM statement. The range includes all statements that are executed as the result of a transfer of control by CALL, EXIT, GO TO, and PERFORM statements in the range of the PERFORM statement, as well as all statements in declarative procedures that are executed as a result of the execution of statements in the range of the PERFORM statement. The statements in the range of a PERFORM statement need not appear consecutively in the source program.

12. Statements executed as the result of a transfer of control caused by executing an EXIT PROGRAM statement within the range of a PERFORM statement are not considered to be part of that range.

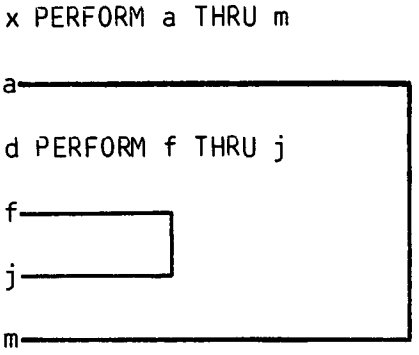
13. Procedure-name-1 and procedure-name-2 must not name sections or paragraphs in any other program in the run unit, irrespective of whether or not the other program contains or is contained within the program which includes the PERFORM statement. Statements in other programs in the run unit may only be obeyed as a result of executing a PERFORM statement, if the range of that PERFORM statement includes CALL and EXIT PROGRAM statements

14. If the range of a PERFORM statement includes another PERFORM statement, the sequence of procedures associated with the included PERFORM must itself either be totally included in, or totally excluded from, the logical sequence referred to by the first PERFORM. Thus, an active PERFORM statement, whose execution point begins within the range of another active PERFORM statement, must not allow control to pass to the exit of the other active PERFORM statement; furthermore, two or more such active PERFORM statements may not have a common exit. See the following illustrations for examples of legal PERFORM constructs:

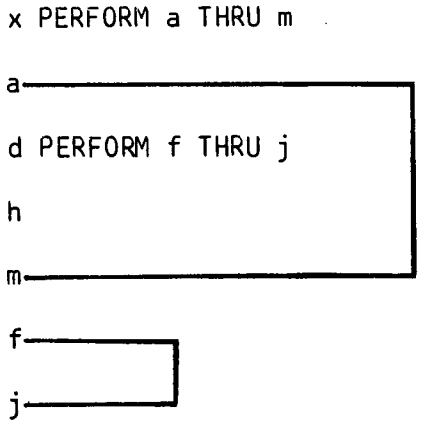
EXAMPLE 1



EXAMPLE 2



EXAMPLE 3



Example of the use of PERFORM statement

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. PERF
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. M-40.
       OBJECT-COMPUTER. M-40.

       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  WORK-AREA.
           02  COUNTER      PIC 99 VALUE ZEROS.
           02  TEXT-FIELD   PIC X(42).
           02  CONTROL      PIC 88 VALUE ZEROS.

       PROCEDURE DIVISION.
       INIZIO-SECTION.
       AA.
          MOVE "  exe perform until data-name = literal " TO TEXT-FIELD
          MOVE 20  TO  CONTROL.

          PERFORM UNTIL COUNTER = 20 OR COUNTER = CONTROL
                       ADD 1 TO COUNTER
                       DISPLAY WORK-AREA
          END-PERFORM.

          IF COUNTER = 20
                              DISPLAY "OK"
          ELSE
                              DISPLAY "ERROR".

          MOVE "  exe perform data-name times " to text-field.
          MOVE ZEROS TO COUNTER.
          MOVE 10  TO CONTROL.

          PERFORM CONTROL TIMES
                  ADD 1  TO COUNTER
                  DISPLAY WORK-AREA
          END-PERFORM.

          IF COUNTER = 10
                              DISPLAY "OK"
          ELSE
                              DISPLAY "ERROR".
          STOP RUN.
```

Function

For sequential access, the READ statement makes available the next (previous) logical record from a file. For random access, the READ statement makes available a specified record from a mass storage file.

General Format

Format 1 - Sequential Access Sequential Organization

```
READ file-name-1 [NEXT] RECORD [INTO identifier-1]

     [AT END imperative-statement-1 [END-READ]]
```

Format 2 - Sequential Access Relative and Indexed Organization

```
READ file-name-1 [ NEXT     ] RECORD [INTO identifier-1]
                 [ PREVIOUS ]

          [WITH LOCK]

     [AT END imperative-statement-1 [END-READ]]
```

Format 3 - Random Access Relative Organization

```
READ file-name-1 RECORD [INTO identifier-1] [WITH LOCK]

     [INVALID KEY imperative-statement-1 [END-READ]]
```

Format 4 - Random Access Indexed Organization

READ file-name-1 RECORD [INTO identifier-1] [WITH LOCK]

[KEY IS data-name-1]

[INVALID KEY imperative-statement-1 [END-READ]]

Syntax Rules

1. The storage area associated with identifier-1 and the record area associated with file-name-1 must not be the same storage area.

Format 1

2. The NEXT phrase is optional and has no effect on the execution of the READ statement.

3. The AT END phrase must be specified if no applicable USE AFTER STANDARD EXCEPTION procedure is specified for file-name-1.

Format 2

4. The PREVIOUS phrase must not be used for files in sequential access mode.

5. The NEXT phrase must be specified for files in dynamic access mode, when records are to be retrieved sequentially forward.

6. The PREVIOUS phrase must be specified for files in dynamic access mode when records are to be retrieved sequentially backward.

Formats 3 and 4

7. Format 3 or 4 is used for files in random access mode or for files in dynamic access mode when records are to be retrieved randomly.

8. The INVALID KEY phrase or the AT END phrase must be specified if no applicable USE AFTER STANDARD EXCEPTION procedure is specified for file-name-1.

Format 4

9. In the KEY clause for indexed files data-name-1 must be the name of a data item specified as a record key associated with file-name-1.

10. In the KEY clause for indexed files data-name-1 may be qualified.

General Rules

1. The file referenced by file-name-1 must be open in the input or I-O mode when the READ statement is executed. (See The OPEN Statement.)

2. The execution of the READ statement causes the value of the I-O status associated with file-name-1 to be updated. (See I-O Status.)

3. The setting of the file position indicator at the start of the execution of a Sequential READ statement is used in determining the record to be made available. The comparisons are made according to the collating sequence of the file for indexed files, or the relative record number for relative files, or the record number for a sequential file. For a Read Next Operation if the file position indicator,

   a) indicates that no valid record has been established, execution of the READ statement is unsuccessful.

   b) indicates that an optional input file is not present or that the at end condition already exists, execution proceeds as specified in general rule 9.

   c) indicates that no next logical record exists, the file position indicator is set to indicate that the at end condition already exists and execution proceeds as specified in general rule 9.

   d) If the file position indicator was established by a previous OPEN or START statement, the first existing record in the file whose key value is greater than or equal to the file position indicator is selected.

   e) was established by a previous READ statement, and the current key of reference does not allow duplicates, the first existing record in the file whose key value is greater than the file position indicator is selected.

   f) for an indexed file was established by a previous READ statement, and the current key of reference does allow duplicates, the first record in the file whose key value is either equal to the file position indicator and whose logical position within the set of duplicates is immediately after the the record that was made available by that previous READ statement, or whose key value is greater than the file position indicator, is selected.

   For a Read Previous Operation the above rules apply with the word 'previous' replacing 'next' and the word 'less' replacing 'greater'.

   If a record is found which satisfies the above rules, it is made available in the record area associated with file-name-1.

   If no record is found which satisfies the above rules, the file position indicator is set to indicate that no next (previous) logical record exists and execution proceeds as specified in general rule 9.

   If a record is made available, the file position indicator is set to

the value of the current key of reference or the relative record number of the record made available.

4. Regardless of the method used to overlap access time with processing time, the concept of the READ statement is unchanged in that a record is available to the object program prior to the execution of any statement following the READ statement.

5. When the logical records of a file are described with more than one record description, these records share the same record area in storage; this is equivalent to an implicit redefinition of the area. The contents of any data items which lie beyond the range of the current data record are undefined at the completion of the execution of the READ statement.

6. The result of the execution of a READ statement with the INTO phrase is equivalent to the application of the following rules in the order specified:

   a) The execution of the same READ statement without the INTO phrase.

   b) The current record is moved from the record area to the area specified by identifier-1 according to the rules for the MOVE statement without the CORRESPONDING phrase. The implied MOVE statement does not occur if the execution of the READ statement was unsuccessful. Any subscripting associated with identifier-1 is evaluated after the record has been read and immediately before it is moved to the data item. The record is available in both the record and the data item referenced by identifier-1.

7. If, at the time of the execution of a format 2 READ statement, the file position indicator indicates that an optional input file is not present, the invalid key condition exists and execution of the READ statement is unsuccessful. (See The Invalid Key Condition.)

8. For a sequential READ statement, if the file position indicator indicates that no next (previous) logical record exists, or that an optional file is not present, or that the at end condition already exists, the following occurs in the order specified:

   a) A value, derived from the setting of the file position indicator, is placed into the I-O status associated with file-name-1 to indicate the at end condition. (See I-O Status.)

   b) If the AT END phrase is specified in the statement causing the condition, control is transferred to imperative-statement-1 in the AT END phrase. Any USE AFTER STANDARD EXCEPTION procedure associated with file-name-1 is not executed.

   c) If the AT END phrase is not specified, a USE AFTER STANDARD EXCEPTION procedure must be associated with file-name-1, and that procedure is executed. Return from that procedure is to the next executable statement following the end of the READ statement.

When the at end condition occurs, execution of the input-output statement which caused the condition is unsuccessful.

9. Following the unsuccessful execution of a READ statement, the content of the associated record area is undefined, the key of reference is undefined for indexed files, and the file position indicator is set as follows:

   a) For a format 3 or 4 READ statement, the file position indicator is set to indicate that no valid next record has been established.

   b) For a sequential READ statement, the following rules apply:

      - If the file position indicator indicates that no next (previous) logical record exists, the file position indicator is unchanged.

      - If the file position indicator indicates that an optional input file is not present, the file position indicator is set to indicate that the at end condition already exists.

      - In all other cases, the file position indicator is set to indicate that no valid next (previous) record has been established.

10. The END-READ phrase delimits the scope of the READ statement. (See Scope of Statements.)

Format 1

11. If, during execution of a READ statement, the end of a reel/unit is recognized or a reel/unit contains no logical records, and the logical end of the file has not been reached, the following operations are executed:

    a) The standard ending reel/unit label procedure.

    b) A reel/unit swap. The current volume pointer is updated to point to the next reel/unit existing for the file.

    c) The standard beginning reel/unit label procedure.

12. When file-name-1 is defined as a TEXT-FILE and if the number of characters in the record that is read is greater than the maximum size specified by the record description entries for file-name-1, the record is truncated on the right to the maximum size. In this case, the READ statement is successful and an I-O status is set indicating a record length conflict has occurred. (See I-O Status.)

Format 2

13. For a file for which dynamic access mode is specified, a READ statement with the NEXT phrase specified causes the next logical record to be retrieved from that file.

14. For a file for which dynamic access mode is specified, a READ statement with the PREVIOUS phrase specified causes the previous logical record to be retrieved from that file.

15. For an indexed file being sequentially accessed, records having the same duplicate value in an alternate record key which is the key of reference are made available in the same order in which they are released by execution of WRITE statements, or by execution of REWRITE statements which create such duplicate values.

16. For a relative file, if the RELATIVE KEY phrase is specified for file-name-1, the execution of a READ statement moves the relative record number of the record made available to the relative key data item according to the rules for the MOVE statement. (See The MOVE Statement.)

Format 3

17. For a relative file, execution of a READ statement sets the file position indicator to the value contained in the data item referenced by the RELATIVE KEY clause for the file, and the record whose relative record number equals the file position indicator is make available in the record area associated with file-name-1. If the file does not contain such a record, the invalid key condition exists and execution of the READ statement is unsuccessful. (See I-O Status.)

Format 4

18. For an indexed file, if the KEY phrase is specified in a READ statement, data-name-1 is established as the key of reference for this retrieval. If the dynamic access mode is specified, this key of reference is also used for retrievals by any subsequent execution of format 2 READ statements for the file until a different key of reference is established for the file.

19. For an indexed file, if the KEY phrase is not specified in a READ statement, the prime record key is established as the key of reference for this statement. If the dynamic access mode is specified, this key of reference is also used for retrievals by any subsequent execution of format 2 READ statements for the file until a different key of reference is established for the file.

20. For an indexed file, execution of a READ statement sets the file position indicator to the value in the key of reference. This value is compared with the value contained in the corresponding data item of the stored records in the file, until the first record having an equal value is found. In the case of an alternate key with duplicate values, the first record found is the first record of a sequence of duplicates which was released to the MOS file system. The record so found is made available in the record area associated with file-name-1. If no record can be so identified, the invalid key condition exists and execution of the READ statement is unsuccessful. (See The Invalid Key Condition.)

Formats 3 and 4

21. The use of the WITH LOCK phrase locks the record in the sense that other system programs and processes can read that record but cannot write it.

A record is locked until it is unlocked by the program or process that locked it in one of the following ways:

a) By execution of an UNLOCK statement specifying the same key value; or

b) By execution of a REWRITE statement specifying the same key value that contains the NO LOCK phrase; or

c) By execution of a CLOSE statement for the file; or

d) By termination.

REWRITE

Function

The REWRITE statement logically replaces a record existing in a mass storage file.

General Format

Format 1 - Sequential Organization

REWRITE record-name-1 [FROM identifier-1]

Format 2 - Relative and Indexed Organization

REWRITE record-name-1 [FROM identifier-1] [WITH [NO] LOCK]

[INVALID KEY imperative-statement-1 [END-REWRITE]]

Syntax Rules

1. Record-name-1 and identifier-1 must not refer to the same storage area.

2. Record-name-1 is the name of a logical record in the File Section of the Data Division and may be qualified.

3. The file referenced by the file-name associated with record-name-1 may not be assigned to TEXT-FILE.

Format 2

4. The INVALID KEY phrase must be specified if an applicable USE AFTER STANDARD EXCEPTION procedure is not specified for the associated file-name.

General Rules

1. The file referenced by the file-name associated with record-name-1 must be a mass storage file and must be open in the I-O mode at the time of execution of this statement. (See The OPEN Statement.)

2. For files in the sequential access mode, the last input-output statement executed for the associated file prior to the execution of the REWRITE statement must have been a successfully executed READ statement. The MOS file system logically replaces the record that was accessed by the READ statement.

3. The logical record released by a successful execution of the REWRITE statement is no longer available in the record area unless the file-

name associated with record-name-1 is specified in a SAME RECORD AREA clause. In this case the logical record is available to the program as a record of other files whose file-names are specified in the same SAME RECORD AREA clause.

4. The result of the execution of a REWRITE statement with the FROM phrase is equivalent to the execution of the following statements in the order specified:

   a) The statement:

      MOVE identifier-1 TO record-name-1

   according to the rules specified for the MOVE statement.

   b) The same REWRITE statement without the FROM phrase.

5. After the execution of the REWRITE statement is complete, the information in the area referenced by identifier-1 is available, even though the information in the area referenced by record-name-1 is not available except as specified by the SAME RECORD AREA clause.

6. The file position indicator is not affected by the execution of a REWRITE statement.

7. The execution of the REWRITE statement causes the value of the I-O status of the file-name associated with record-name-1 to be updated. (See I-O Status.)

8. The execution of the REWRITE statement releases a logical record to the operating system.

Format 2

9. The END-REWRITE phrase delimits the scope of the REWRITE statement. (See Scope of Statements.)

10. The number of character positions in the record referenced by record-name-1 must be equal to the number of character positions in the record being replaced.

11. For an indexed file in the sequential access mode, the record to be replaced is specified by the value of the prime record key. When the REWRITE statement is executed the value of the prime record key of the record to be replaced must be equal to the value of the prime record key of the last record read from this file.

12. For a file in the random or dynamic access mode, the record to be replaced is specified by the prime record key or the relative key data item.

13. For an indexed file execution of the REWRITE statement for a record which has an alternate record key occurs as follows:

    a) When the value of a specific alternate record key is not changed, the order of retrieval when that key is the key of reference

remains unchanged.

b) When the value of a specific alternate record key is changed in a record, the subsequent order of retrieval of that record is changed when that specific alternate record key is the key of reference. When duplicate key values are permitted, the record is logically positioned last within the set of duplicate records containing the same alternate record key value as the alternate record key value that was placed in the record.

14. The invalid key condition exists under the following circumstances:

a) When a relative file does not contain the record specified by the relative key data item, or for indexed files

b) When the file is open in the sequential access mode, and the value of the prime record key of the record to be replaced is not equal to the value of the prime record key of the last record read from the file, or

c) When the file is open in the dynamic or random access mode, and the value of the prime record key of the record to be replaced is not equal to the value of the prime record key of any record record existing in the file, or

d) When the value of an alternate record key of the record to be replaced, for which duplicates are not allowed, equals the value of the corresponding data item of a record already existing in the file.

15. When the invalid key condition is recognized, the execution of the REWRITE statement is unsuccessful, the updating operation does not take place, the content of the record area is unaffected and the I-O status of the file-name associated with record-name-1 is set to a value indicating the cause of the condition. Execution of the program proceeds according to the rules for an invalid key condition. (See I-O Status, and The Invalid Key Condition.)

16. The use of the WITH NO LOCK phrase causes the current record to be made accessible to other system programs or processes.

17. An attempt to execute a REWRITE statement for a record placed in lock mode by another system program or process results in an I-O error indication in the I-O status of the file-name associated with record-name-1.

18. The use of the WITH LOCK phrase locks the record in the sense that other system programs and processes can read that record but cannot write it.

A record is locked until it is unlocked by the program or process that locked it in one of the following ways:

a) By execution of an UNLOCK statement specifying the same key value; or

b) By execution of a REWRITE statement specifying the same key value that contains the NO LOCK phrase; or

c) By execution of a CLOSE statement for the file; or

d) By termination.

SEARCH

Function

The SEARCH statement is used to search a table for a table element that satisfies the specified condition and to adjust the value of the associated index to indicate that table element.

General Format

Format 1

```
SEARCH identifier-1 [VARYING {identifier-2 }]
                             {index-name-1 }

    [AT END imperative-statement-1]

    { WHEN condition-1 { imperative-statement-2 } } ...
                       { NEXT SENTENCE          }

    [END-SEARCH]
```

Format 2

```
SEARCH ALL identifier-1  [AT END imperative-statement-1]

    WHEN { data-name-1      { IS EQUAL TO } { identifier-3                } }
         {                  {             } { literal-1                  } }
         {                  { IS =        } { arithmetic-expression-1    } }
         { condition-name-1                                                }

    [ AND { data-name-2      { IS EQUAL TO } { identifier-4                } } ... ]
          {                  {             } { literal-2                  } }
          {                  { IS =        } { arithmetic-expression-2    } }
          { condition-name-2                                                }
```

imperative-statement-2

NEXT SENTENCE

[END-SEARCH]

Syntax Rules

1. In both formats 1 and 2, identifier-1 must not be subscripted but its description must contain an OCCURS clause including an INDEXED BY phrase. The description of identifier-1 in format 2 must also contain the KEY IS phrase in its OCCURS clause.

2. Identifier-2 must reference a data item described as USAGE IS INDEX or as a numeric elementary data item without any positions to the right of the assumed decimal point.

3. .Identifier-2 may not be subscripted by the first (or only) index-name specified in the INDEXED BY phrase in the OCCURS clause associated with identifier-1.

4. In format 1, condition-1 may be any conditional expression. (See Conditional Expressions.)

5. In format 2, all referenced condition-names must be defined as having only a single value. The data-name associated with a condition-name must appear in the KEY IS phrase in the OCCURS clause referenced by identifier-1. Each data-name-1, data-name-2 may be qualified. Each data-name-1, data-name-2 must be subscripted by the first index-name associated with identifier-1 along with other indices or literals as required, and must be referenced in the KEY IS phrase in the OCCURS clause referenced by identifier-1. Identifier-3, identifier-4, or identifiers specified in arithmetic-expression-1, arithmetic-expression-2 must not be referenced in the KEY IS phrase in the OCCURS clause referenced by identifier-1 or be subscripted by the first index-name associated with identifier-1.

   In format 2, when a data-name in the KEY IS phrase in the OCCURS clause referenced by identifier-1 is referenced, or when a condition-name associated with a data-name in the KEY IS phrase in the OCCURS clause referenced by identifier-1 is referenced, all preceding data-names in the KEY IS phrase in the OCCURS clause referenced by identifier-1 or their associated condition names must also be referenced.

6. If the END-SEARCH phrase is specified, the NEXT SENTENCE phrase must not be specified.

General Rules

1. The scope of a SEARCH statement may be terminated by any of the following:

a) An END-SEARCH phrase at the same level of nesting.

b) A separator period.

c) An ELSE or END-IF phrase associated with a previous IF statement.

2. If format 1 of the SEARCH statement is used, a serial type of search operation takes place, starting with the current index setting.

   a) If, at the start of execution of the SEARCH statement, the index-name associated with identifier-1 contains a value that corresponds to an occurrence number that is greater than the highest permissible occurrence number for identifier-1, the search is terminated immediately. The number of occurrences of identifier-1, the last of which is the highest permissible, is discussed in the OCCURS clause. (See The OCCURS Clause.) Then, if the AT END phrase is specified, imperative-statement-1 is executed; if the AT END phrase is not specified, control passes to the end of the SEARCH statement.

   b) If, at the start of execution of the SEARCH statement, the index-name associated with identifier-1 contains a value that corresponds to an occurrence number that is not greater than the highest permissible occurrence number for identifier-1 (the number of occurrences of identifier-1, the last of which is the highest permissible, is discussed in the OCCURS clause), the SEARCH statement operates by evaluating the conditions in the order that they are written, making use of the index settings, wherever specified, to determine the occurrence of those items to be tested. If none of the conditions is satisfied, the index-name for identifier-1 is incremented to obtain reference to the next occurrence. The process is then repeated using the new index-name settings unless the new value of the index-name settings for identifier-1 corresponds to a table element outside the permissible range of occurrence values, in which case the search terminates as indicated in 2a above. If one of the conditions is satisfied upon its evaluation, the search terminates immediately, and control passes to the imperative statement associated with that condition, if present, or, if the NEXT SENTENCE phrase is associated with that condition, to the next executable sentence; the index-name remains set at the occurrence which caused the condition to be satisfied.

3. In a format 2 SEARCH statement, the results of the SEARCH ALL operation are predictable only when:

   a) The data in the table is ordered in the same manner as described in the KEY IS phrase of the OCCURS clause referenced by identifier-1, and

   b) The contents of the key(s) referenced in the WHEN phrase are sufficient to identify a unique table element.

4. If format 2 of the SEARCH statement is used, a nonserial type of search operation may take place; the initial setting of the index-name for identifier-1 is ignored and its setting is varied during the

search operation with the restriction that at no time is it set to a value that exceeds the value which corresponds to the last element of the table, or that is less than the value that corresponds to the first element of the table. The length of the table is discussed in the OCCURS clause. (See The OCCURS Clause.) If any of the conditions specified in the WHEN phrase cannot be satisfied for any setting of the index within the permitted range, control is passed to imperative-statement-1 of the AT END phrase, when specified, or to the next executable sentence when this phrase is not specified; in either case the final setting of the index is not predictable. If all the conditions can be satisfied, the index indicates an occurrence that allows the condition to be satisfied, and control passes to imperative-statement-2, if specified, or to the next executable sentence if the NEXT SENTENCE phrase is specified.

5. After execution of imperative-statement-1 or imperative-statement-2, that does not terminate with a GO TO statement, control passes to the end of the SEARCH statement.

6. In format 2, the index-name that is used for the search operation is the first (or only) index-name that appears in the INDEXED BY phrase in the OCCURS clause referenced by identifier-1. Any other index-names for identifier-1 remain unchanged.

7. In format 1, if the VARYING phrase is not used, the index-name that is used for the search operation is the first (or only) index-name that appears in the INDEXED BY phrase in the OCCURS clause referenced by identifier-1. Any other index-names for identifier-1 remain unchanged.

8. In format 1, if the VARYING index-name-1 phrase is specified, and if index-name-1 appears in the INDEXED BY phrase in the OCCURS clause referenced by identifier-1, that index-name is used for this search. If this is not the case, or if the VARYING identifier-2 phrase is specified, the first (or only) index-name given in the INDEXED BY phrase in the OCCURS clause referenced by identifier-1 is used for the search. In addition, the following operation will occur:

   a) If the VARYING index-name-1 phrase is used, and if index-name-1 appears in the INDEXED BY phrase in the OCCURS clause referenced by another table entry, the occurrence number represented by index-name-1 is incremented by the same amount as, and at the same time as, the occurrence number represented by the index-name associated with identifier-1 is incremented.

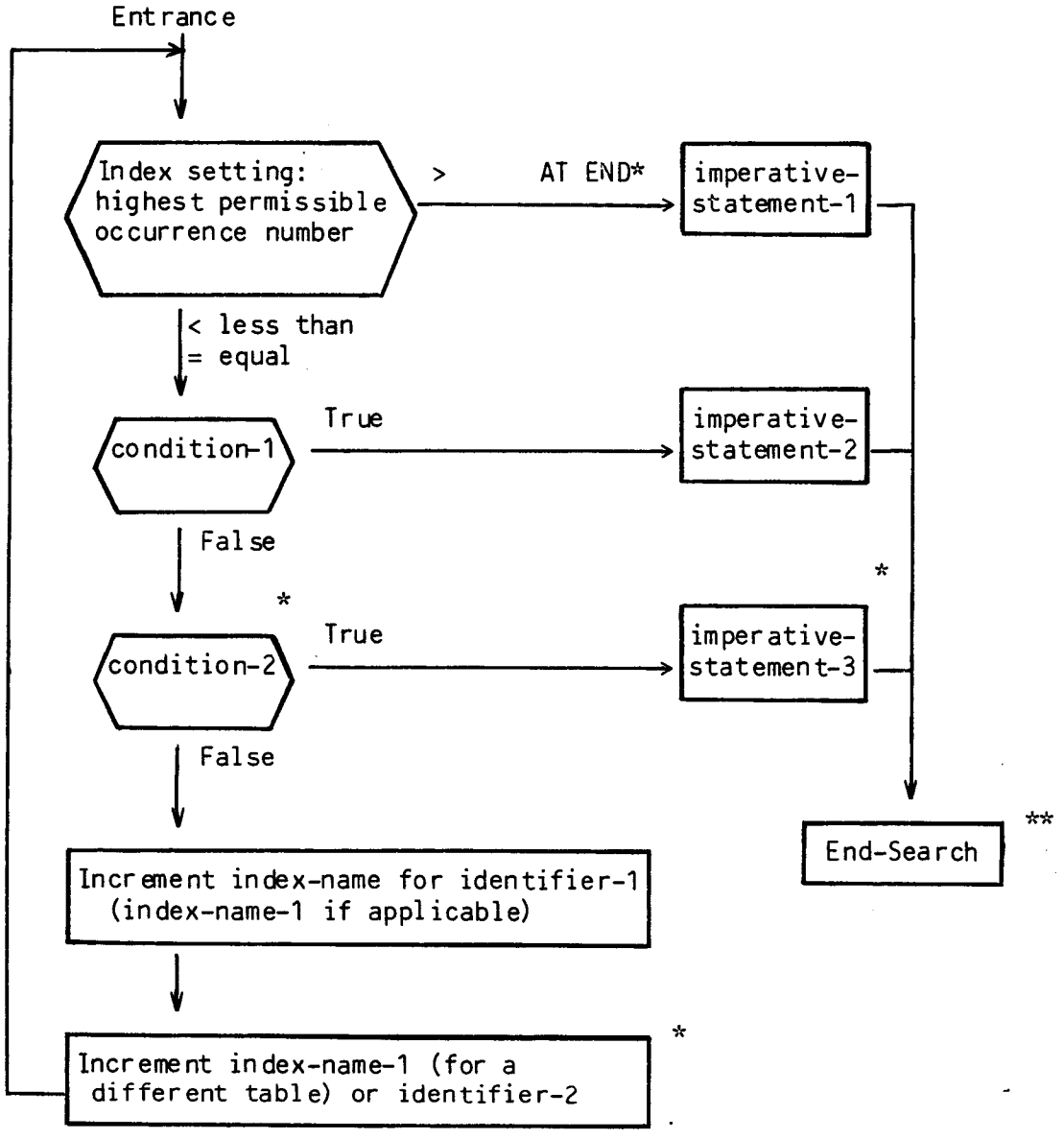
   b) If the VARYING identifier-2 phrase is specified, and identifier-2 is an index data item, then the data item referenced by identifier-2 is incremented by the same amount as, and at the same time as, the index associated with identifier-1 is incremented. If identifier-2 is not an index data item, the data item referenced by identifier-2 is incremented by the value one at the same time as the index referenced by the index-name associated with identifier-1 is incremented.

9. If identifier-1 references a data item subordinate to a data item that contains an OCCURS clause, an index-name must be associated with

each dimension of the table through the INDEXED BY phrase of the OCCURS clause. Only the setting of the index-name associated with identifier-1 (and the data item identifier-2 or index-name-1, if present) is modified by the execution of the SEARCH statement. To search a multi-dimensional table it is necessary to execute a SEARCH statement several times. Prior to each execution of a SEARCH statement, SET statements must be executed whenever index-names must be adjusted to appropriate settings.

10. The END-SEARCH phrase delimits the scope of the SEARCH statement. (See Scope of Statements.)

A representation of the action of a format 1 SEARCH statement containing two WHEN phrases is shown below.



* These operations are options included only when specified in the SEARCH statement.

** Each of these control transfers is to the end of the SEARCH statement unless the imperative-statement ends with a GO TO statement.

Function

1. The SET statement establishes reference points for table handling operations by setting indices associated with table elements.

2. The SET statement is also used to alter the status of external switches.

3. The SET statement is also used to alter the value of conditional variables.

General Format

Format 1:

```
SET { index-name-1  } ... TO { index-name-2  }
    { identifier-1  }        { identifier-2  }
                             { integer-1     }
```

Format 2:

```
SET {index-name-3} ... { UP BY   } { identifier-3 }
                       { DOWN BY } { integer-2    }
```

Format 3:

```
SET {mnemonic-name-1} ... TO { ON  }
                             { OFF }
```

Format 4:

```
SET {condition-name-1} ... TO TRUE
```

Syntax Rules

1. All references to index-name-1, identifier-1, and index-name-3 apply equally to all recursions thereof.

2. Identifier-1 and identifier-2 must each reference an index data item or an elementary item described as an integer.

3. Identifier-3 must reference an elementary numeric integer.

4. Integer-1 and integer-2 may be signed. Integer-1 must be positive.

5. Mnemonic-name-1 must be associated with an external switch, the status of which can be altered.

6. Condition-name-1 must be associated with a conditional variable.

General Rules

Formats 1 and 2

1. Index-names are associated with a given table by being specified in the INDEXED BY phrase of the OCCURS clause for that table.

2. If index-name-1 is specified, the value of the index after the execution of the SET statement must correspond to an occurrence number of an element in the table associated with index-name-1. The value of the index associated with an index-name after the execution of a PERFORM or SEARCH statement may be set to an occurrence number that is outside the range of its associated table.

   If index-name-2 is specified, the value of the index before the execution of the SET statement must correspond to an occurrence number of an element in the table associated with index-name-1.

   If index-name-3 is specified, the value of the index both before and after the execution of the SET statement must correspond to an occurrence number of an element in the table associated with index-name-3.

3. In format 1, the following action occurs:

   a) Index-name-1 is set to a value causing it to refer to the table element that corresponds in occurrence number to the table element referenced by index-name-2, identifier-2, or integer-1. If identifier-2 references an index data item, or if index-name-2 is related to the same table as index-name-1, no conversion takes place.

   b) If identifier-1 references an index data item, it may be set equal to either the content of index-name-2 or identifier-2 where identifier-2 also references an index data item; no conversion takes place in either case.

   c) If identifier-1 does not reference an index data item, it may be set only to an occurrence number that corresponds to the value of index-name-2. Neither identifier-2 nor integer-1 can be used in this case.

   d) The process is repeated for each recurrence of index-name-1, identifier-1, ..., if specified. Each time, the value of index-name-2 or the data item referenced by identifier-2 is used as it was at the beginning of the execution of the statement. Any subscripting associated with identifier-1, ... , is evaluated immediately before the value of the respective data item is changed.

Data in the following table represents the validity of various

operand combinations in format 1 of the SET statement. The general rule reference indicates the applicable general rule.

SENDING ITEM	RECEIVING ITEM		
	INTEGER DATA ITEM	INDEX	INDEX DATA ITEM
Integer literal	No/3c	Valid/3a	No/3b
Integer data item	No/3c	Valid/3a	No/3b
Index	Valid/3c	Valid/3a	Valid/3b*
Index data item	No/3c	Valid/3a*	Valid/3b*

Validity of Operand Combinations in Format 1 SET Statements.

* = No conversion takes place.

4. In Format 2, the content of index-name-3 are incremented (UP BY) or decremented (DOWN BY) by a value that corresponds to the number of occurrences represented by the value of integer-2 or the data item referenced by identifier-3; thereafter, the process is repeated for each recurrence of index-name-3. For each repetition the value of the data item referenced by identifier-3 is used as it was at the beginning of the execution of the statement.

Format 3

5. The status of each external switch associated with the specified mnemonic-name-1 is modified such that the truth value resultant from evaluation of a condition-name associated with that switch will reflect an on status if the ON phrase is specified, or an off status if the OFF phrase is specified.

Format 4

6. The literal in the VALUE clause associated with condition-name-1 is moved to the conditional variable as if a statement of the form "MOVE literal TO conditional-variable" were executed. If more than one literal is specified in the VALUE clause, the literal in the MOVE statement is the first literal in the VALUE clause.

7. If multiple condition-names are specified, the results are the same as if a separate SET statement had been written for each condition-name in the same order as specified in the SET statement.

Function

The START statement provides a basis for logical positioning within a file with relative or indexed organization, for subsequent sequential retrieval of records.

General Format

```
START file-name-1 [ KEY { IS EQUAL TO
                        { IS =
                        { IS GREATER THAN
                        { IS >
                        { IS NOT LESS THAN
                        { IS NOT <                } data-name-1 ]

[INVALID KEY imperative-statement-1 [END-START]]
```

Syntax Rules

1. File-name-1 must be the name of a file with sequential or dynamic access.

2. Data-name-1 may be qualified.

3. The INVALID KEY phrase must be specified if no applicable USE AFTER STANDARD EXCEPTION procedure is specified for file-name-1.

4. If the KEY phrase is specified, data-name-1 must reference either:

   a) For relative organization the data item specified in the RELATIVE KEY phrase in the ACCESS MODE clause of the associated file control entry, or for indexed organization

   b) A data item specified as a record key associated with file-name-1 (see The ALTERNATE RECORD KEY Clause, and The RECORD KEY Clause), or

   c) Any data item of category alphanumeric whose leftmost character position within a record of the file corresponds to the leftmost character position of a record key associated with file-name-1 whose length is not greater than the length of that record key.

General Rules

1. The file referenced by file-name-1 must be open in the input or I-O mode at the time that the START statement is executed. (See The OPEN Statement.)

2. If the KEY phrase is not specified, the relational operator 'IS EQUAL TO' is implied.

3. The execution of the START statement does not alter the contents of the record area.

4. The type of comparison specified by the relational operator in the KEY phrase occurs between a key associated with a record in the file referenced by file-name-1 and a data item as specified in general rule 10 for relative organization and general rules 11 and 12 for indexed organization.
   The comparison is made on the ascending key of reference according to the collating sequence of the file. If the operands are of unequal size, comparison proceeds as though the longer one was truncated on the right such that its length is equal to that of the shorter. All other nonnumeric comparison rules apply.

   a) The file position indicator is set to the value of the key of reference or relative record number in the first logical record currently existing in the file whose key or relative record number satisfies the comparison.

   b) If the comparison is not satisfied by any record in the file, the invalid key condition exists and the execution of the START statement is unsuccessful. (See The Invalid Key Condition.)

5. The execution of the START statement causes the value of the I-O status associated with file-name-1 to be updated. (See I-O Status.)

6. If, at the time of the execution of the START statement, the file position indicator indicates that an optional input file is not present, the invalid key condition exists and the execution of the START statement is unsuccessful. (See The Invalid Key Condition.)

7. Following the unsuccessful execution of a START statement, the file position indicator is set to indicate that no valid next record has been established. Also for indexed files, the key of reference is undefined.

8. The END-START phrase delimits the scope of the START statement. (See Scope of Statements.)

9. A key of reference is established as follows:

   a) If the KEY phrase is not specified, the prime record key specified for file-name-1 becomes the key of reference.

   b) If the KEY phrase is specified, and data-name-1 is specified as a record key for file-name-1, that record key becomes the key of reference.

   c) If the KEY phrase is specified, and data-name-1 is not specified as a record key for file-name-1, the record key whose leftmost character position corresponds to the leftmost character position of the data item specified by data-name-1, becomes the key of reference.

This key of reference is used to establish the ordering of records for the purpose of this START statement, see general rule 4; and, if the execution of the START statement is successful, the key of reference is also used for subsequent sequential READ statements. (See The READ Statement.)

10. The comparison described in general rule 4 uses the data item referenced by the RELATIVE KEY phrase of the ACCESS MODE clause associated with file-name-1.

11. If the KEY phrase is specified, the comparison described in general rule 4 uses the data item referenced by data-name-1.

12. If the KEY phrase is not specified, the comparison described in general rule 4 uses the data item referenced in the RECORD KEY clause associated with file-name-1.

STOP

Function

The STOP statement causes a permanent or temporary suspension of the execution of the object program.

General Format

```
STOP { RUN       }
     { literal-1 }
```

Syntax Rules

1. Literal-1 may not be a figurative constant that begins with the word ALL.

2. If a STOP RUN statement appears in a consecutive sequence of imperative statements within a sentence, it must appear as the last statement in that sequence.

3. If literal-1 is numeric, then it must be an unsigned integer.

General Rules

1. If the RUN phrase is used, execution of the run unit ceases and control is transferred to the operating system.

2. During the execution of a STOP RUN statement, an implicit CLOSE statement without any optional phrases is executed for each file that is in the open mode in the run unit. Any USE procedures associated with any of these files are not executed.

3. If STOP literal-1 is specified, literal-1 is communicated to the operator. Continuation of the object program begins with the execution of the next executable statement in sequence.

Function

The STRING statement provides juxtaposition of the partial or complete contents of one or more data items into a single data item.

General Format

```
STRING { { identifier-1 } ... DELIMITED BY { identifier-2 } } ...
       { { literal-1    }                  { literal-2    }
                                           { SIZE         }

   INTO identifier-3 [WITH POINTER identifier-4]

   [ON OVERFLOW imperative-statement [END-STRING]]
```

Syntax Rules

1. Each literal may be any figurative constant without the optional word ALL.

2. All literals must be described as nonnumeric literals, and all identifiers, except identifier-4, must be described implicitly or explicitly as usage is DISPLAY.

3. Identifier-3 must represent an elementary alphanumeric data item without editing symbols or the JUSTIFIED clause.

4. Identifier-4 must be described as an elementary numeric integer data item of sufficient size to contain a value equal to 1 plus the size of the area referenced by identifier-3. The symbol 'P' may not be used in the PICTURE character-string of identifier-4.

5. Where identifier-1 or identifier-2 is an elementary numeric data item, it must be described as an integer without the symbol 'P' in its PICTURE character-string.

General Rules

1. All references to identifier-1, literal-1, apply equally to all recursions thereof.

2. Identifier-1 or literal-1 represents the sending item. Identifier-3 represents the receiving item.

3. Literal-2 or the content of the data item referenced by identifier-2 indicate the character(s) delimiting the move. If the SIZE phrase is used, the content of the complete data item defined by identifier-1 or literal-1 is moved. When a figurative constant is used as the delimiter, it stands for a single character nonnumeric literal.

4. When a figurative constant is specified as literal-1 or literal-2, it refers to an implicit one character data item whose usage is DISPLAY.

5. When the STRING statement is executed, the transfer of data is governed by the following rules:

   a) Those characters from literal-1 or from the content of the data item referenced by identifier-1 are transferred to the data item referenced by identifier-3 in accordance with the rules for alphanumeric to alphanumeric moves, except that no space filling will be provided. (See The MOVE Statement.)

   b) If the DELIMITED phrase is specified without the SIZE phrase, the content of the data item referenced by identifier-1, or the value of literal-1, is transferred to the receiving data item in the sequence specified in the STRING statement beginning with the leftmost character and continuing from left to right until the end of the data item is reached or until the character(s) specified by literal-2, or by the content of the data item referenced by identifier-2, are encountered. The character(s) specified by literal-2 or by the data item referenced by identifier-2 are not transferred.

   c) If the DELIMITED phrase is specified with the SIZE phrase, the entire content of literal-1, or the content of the data item referenced by identifier-1, is transferred, in the sequence specified in the STRING statement, to the data item referenced by identifier-3 until all data has been transferred or the end of the data item referenced by identifier-3 has been reached.

   This behavior is repeated until all occurrences of literal-1 or data items referenced by identifier-1 have been processed.

6. If the POINTER phrase is specified, the data item referenced by identifier-4 must be set to an initial value greater than zero prior to the execution of the STRING statement.

7. If the POINTER phrase is not specified, the following general rules apply as if the user had specified identifier-4 referencing a data item with an initial value of 1.

8. When characters are transferred to the data item referenced by identifier-3, the moves behave as though the characters were moved one at a time from the source into the character positions of the data item referenced by identifier-3 designated by the value of the data item referenced by identifier-4 (provided the value of the data item referenced by identifier-4 does not exceed the length of the data item referenced by identifier-3), and then the data item referenced by identifier-4 was increased by one prior to the move of the next character or prior to the end of execution of the STRING statement. The value of the data item referenced by identifier-4 is changed during execution of the STRING statement only by the behavior specified above.

9. At the end of execution of the STRING statement, only the portion of the data item referenced by identifier-3 that was referenced during

the execution of the STRING statement is changed. All other portions of the data item referenced by identifier-3 will contain data that was present before this execution of the STRING statement.

10. Before each move of a character to the data item referenced by identifier-3, if the value associated with the data item referenced by identifier-4 is either less than one or exceeds the number of character positions in the data item referenced by identifier-3, no (further) data is transferred to the data item referenced by identifier-3, and control is transferred to the end of the STRING statement or, if the ON OVERFLOW phrase is specified, to imperative-statement-1. If control is transferred to imperative-statement-1, execution continues according to the rules for each statement specified in imperative-statement-1. If a procedure branching or conditional statement which causes explicit transfer of control is executed, control is transferred in accordance with the rules for that statement; otherwise, upon completion of the execution of imperative-statement-1, control is transferred to the end of the STRING statement.

11. The END-STRING phrase delimits the scope of the STRING statement. (See Scope of Statements.)

12. If identifier-1, or identifier-2, occupies the same storage area as identifier-3, or identifier-4, or if identifier-3 and identifier-4 occupy the same storage area, the result of the execution of this statement is undefined, even if they are defined by the same data description entry. (See Overlapping Operands.)

SUBTRACT

Function

The SUBTRACT statement is used to subtract one, or the sum of two or more, numeric data items from one or more items, and set the values of one or more items equal to the results.

General Format

Format 1:

```
SUBTRACT { identifier-1 } ... FROM {identifier-2 [ROUNDED] }...
         { literal-1    }

    [ON SIZE ERROR imperative-statement-1 [END-SUBTRACT]]
```

Format 2:

```
SUBTRACT { identifier-1 } ... FROM { identifier-2 }
         { literal-1    }         { literal-2    }

   GIVING {identifier-3 [ROUNDED]} ...

   [ON SIZE ERROR imperative-statement-1 [END-SUBTRACT]]
```

Format 3:

```
SUBTRACT { CORRESPONDING } identifier-1 FROM identifier-2 [ROUNDED]
         { CORR          }

    [ON SIZE ERROR imperative-statement-1 [END-SUBTRACT]]
```

Syntax Rules

1. Each identifier must refer to a numeric elementary item except that:

   a) In format 2, each identifier following the word GIVING must refer to either an elementary numeric item or an elementary numeric edited item.

b) In format 3, where each identifier must refer to a group item.

2. Each literal must be a numeric literal.

3. The composite of operands must not contain more than 18 digits. (See The Arithmetic Statements.)

   a) In format 1 the composite of operands is determined by using all of the operands in a given statement.

   b) In format 2 the composite of operands is determined by using all of the operands in a given statement excluding the data items that follow the word GIVING.

   c) In format 3 the composite of operands is determined separately for each pair of corresponding data items.

4. CORR is an a abbreviation for CORRESPONDING.

General Rules

1. When format 1 is used, the values of the operands preceding the word FROM are added together and the sum is stored in a temporary data item. The value in this temporary data item is subtracted from the value of the data item referenced by identifier-2, storing the result into the data item referenced by identifier-3, and repeating this process for each successive occurrence of identifier-2 in the left-to-right order in which identifier-2 is specified.

2. In format 2, all literals and the values of the data items referenced by the identifiers preceding the word FROM are added together, the sum is subtracted from literal-2 or the value of the data item referenced by identifier-2 and the result of the subtraction is stored as the new content of each data item referenced by identifier-3.

3. If format 3 is used, data items in identifier-1 are subtracted from and stored into corresponding data items in identifier-2.

4. The compiler insures enough places are carried so as not to lose significant digits during execution.

5. Additional ruling and explanations relative to this statement are given under the appropriate paragraphs. (See Scope of Statements; The ROUNDED Phrase; The Arithmetic Statements; The SIZE ERROR Phrase; The CORRESPONDING Phrase; and Multiple Results in Arithmetic Statements.)

UNLOCK

Function

The UNLOCK statement places file record(s) in unlock status, thus allowing access by other system programs or processes.

General Format

Format 1 - Relative and Indexed Organization

```
UNLOCK  file-name-1  { RECORD  }
                     { RECORDS }
```

Format 2 - Relative Organization

```
UNLOCK file-name-1  AT KEY {identifier-1} ...
```

Format 3 - Indexed Organization

```
UNLOCK  file-name-1  [ ON PRIMARY                    ]
                     [ ON ALTERNATE data-name-1      ]

      AT KEY {identifier-1} ...
```

Syntax Rules

1. Data-name-1 may be qualified.

2. In format 2, identifier-1 must reference an unsigned integer data-item whose description does not contain the PICTURE symbol 'P'.

3. In format 3, data-name-1 must be the name of one of the data items specified as an alternate key of file-name-1.

4. In format 3, identifier-1 must be the name of a data item with the equivalent data description to the data item specified as either primary or alternate key of file-name-1 depending on the ON clause implicitly or explicitly specified in the LOCK statement.

General Rules

1. When a format 1 UNLOCK is executed all records in the file are unlocked, use of format 2 or 3 unlocks records by specific key value.

2. An UNLOCK statement may only be executed for a file in an open mode.

3. When the ON clause is not specified, ON PRIMARY is assumed.

4. The AT KEY phrase defines the value of the key(s) of the record(s) to be unlocked. In format 2 the record is unlocked even if it does not exist in the file.

5. If the ALTERNATE phrase is specified in format 3, the record indicated by the key value in the AT KEY phrase must exist. For a primary key value the record is unlocked even if it does not exist in the file.

6. If data-name-1 was defined with DUPLICATES and records exist with that alternate key value, the first record written containing that value is unlocked.

7. An unlocked record can be accessed by any system program or process by use of an input-output statement unless the unlocked record is a member of a file that has been locked by the present program. That is, if the file is locked, unlocking a record does not make the record available for access by another program or process.

8. The execution of an UNLOCK statement for a record that is not in a locked status by the present program has no effect.

UNSTRING

Function

The UNSTRING statement causes contiguous data in a sending field to be separated and placed into multiple receiving fields.

General Format

```
UNSTRING identifier-1

    [ DELIMITED BY [ALL] { identifier-2 }
                         { literal-1    }

        [ OR [ALL] { identifier-3 } ]... ]
                   { literal-2    }

    INTO {identifier-4 [DELIMITER IN identifier-5]

       [COUNT IN identifier-6] } ...

    [WITH POINTER identifier-7] [TALLYING IN identifier-8]

    [ON OVERFLOW imperative-statement-1 [END-UNSTRING]]
```

Syntax Rules

1. Literal-1 and literal-2 must be nonnumeric literals and neither can be a figurative constant that begins with the word ALL.

2. Identifier-1, identifier-2, identifier-3, and identifier-5 must reference data items described, implicitly or explicitly, as category alphanumeric.

3. Identifier-4 may be described as either alphabetic, alphanumeric, or numeric (except that the symbol 'P' may not be used in the PICTURE character-string), and must be described, implicitly or explicitly, as USAGE IS DISPLAY.

4. Identifier-6 and identifier-8 must reference integer data items (except that the symbol 'P' may not be used in the PICTURE character-string).

5. Identifier-7 must be described as an elementary numeric integer data item of sufficient size to contain the value equal to 1 plus the size

of the data item referenced by identifier-1. The symbol 'P' may not be used in the PICTURE character-string of identifier-7.

6. The DELIMITER IN phrase and the COUNT IN phrase may be specified only if the DELIMITED BY phrase is specified.

General Rules

1. All references to identifier-2 and literal-1 apply equally to identifier-3 and literal-2, respectively, and all recursions thereof.

2. The data item referenced by identifier-1 represents the sending area.

3. The data item referenced by identifier-4 represents the data receiving area. The data item referenced by identifier-5 represents the receiving area for delimiters.

4. Literal-1 or the data item referenced by identifier-2 specifies a delimiter.

5. The data item referenced by identifier-6 represents the count of the number of characters within the data item referenced by identifier-1 isolated by the delimiters for the move to the data item referenced by identifier-4. This value does not include a count of the delimiter character(s).

6. The data item referenced by identifier-7 contains a value that indicates a relative character position within the area defined by identifier-1.

7. The data item referenced by identifier-8 is a counter which is incremented by 1 for each occurrence of the data item referenced by identifier-4 accessed during the UNSTRING operation.

8. When a figurative constant is used as the delimiter, it stands for a single character nonnumeric literal.

   When the ALL phrase is specified, one occurrence or two or more contiguous occurrences of literal-1 (figurative constant or not) or the content of the data item referenced by identifier-2 are treated as if it were only one occurrence, and one occurrence of literal-1 or the data item referenced by identifier-2 is moved to the receiving data item according to the rules in general rule 13d.

9. When any examination encounters two contiguous delimiters, the current receiving area is space filled if it is described as alphabetic or alphanumeric, or zero filled if it is described as numeric.

10. Literal-1 or the content of the data item referenced by identifier-2 can contain any character in the computer's character set.

11. Each literal-1 or the data item referenced by identifier-2 represents one delimiter. When a delimiter contains two or more characters, all of the characters must be present in contiguous positions of the sending item, and in the order given, to be recognized as a delimiter.

12. When two or more delimiters are specified in the DELIMITED BY phrase, an OR condition exists between them. Each delimiter is compared to the sending field. If a match occurs, the character(s) in the sending field is considered to be a single delimiter. No character(s) in the sending field can be considered a part of more than one delimiter.

    Each delimiter is applied to the sending field in the sequence specified in the UNSTRING statement.

13. When the UNSTRING statement is initiated, the current receiving area is the data item referenced by identifier-4. Data is transferred from the data item referenced by identifier-1 to the data item referenced by identifier-4 according to the following rules:

    a) If the POINTER phrase is specified, the string of characters referenced by identifier-1 is examined beginning with the relative character position indicated by the content of the data item referenced by identifier-7. If the POINTER phrase is not specified, the string of characters is examined beginning with the leftmost character position.

    b) If the DELIMITED BY phrase is specified, the examination proceeds left to right until either a delimiter specified by literal-1 or the value of the data item referenced by identifier-2 is encountered. (See general rule 11.) If the DELIMITED BY phrase is not specified, the number of characters examined is equal to the size of the current receiving area. However, if the sign of the receiving item is defined as occupying a separate character position, the number of characters examined is one less than the size of the current receiving area.

       If the end of the data item referenced by identifier-1 is encountered before the delimiting condition is met, the examination terminates with the last character examined.

    c) The characters thus examined (excluding the delimiting character(s), if any) are treated as an elementary alphanumeric data item, and are moved into the current receiving area according to the rules for the MOVE statement (See The MOVE Statement.)

    d) If the DELIMITER IN phrase is specified, the delimiting character(s) are treated as an elementary alphanumeric data item and are moved into the data item referenced by identifier-5 according to the rules for the MOVE statement. (See The MOVE Statement.) If the delimiting condition is the end of the data item referenced by identifier-1, then the data item referenced by identifier-5 is space-filled.

    e) If the COUNT IN phrase is specified, a value equal to the number of character thus examined (excluding the delimiter character(s), if any) is moved into the area referenced by identifier-6 according to the rules for an elementary move

    f) If the DELIMITED BY phrase is specified, the string of characters is further examined beginning with the first character to the

right of the delimiter. If the DELIMITED BY phrase is not specified, the string of characters is further examined beginning with the character to the right of the last character transferred.

g) After data is transferred to the data item referenced by identifier-4, the current receiving area is the data item referenced by the next recurrence of identifier-4. The behavior described in paragraphs 13b through 13f is repeated until either all the characters are exhausted in the data item referenced by identifier-1, or until there are no more receiving areas.

14. The initialization of the contents of the data items associated with the POINTER phrase or the TALLYING phrase is the responsibility of the user.

15. The content of the data item referenced by identifier-7 will be incremented by one for each character examined in the data item referenced by identifier-1. When the execution of an UNSTRING statement with a POINTER phrase is completed, the content of the data item referenced by identifier-7 will contain a value equal to the initial value plus the number of characters examined in the data item referenced by identifier-1.

16. When the execution of an UNSTRING statement with a TALLYING phrase is completed, the content of the data item referenced by identifier-8 contains a value equal to its value at the beginning of the execution of the statement plus a value equal to the number of identifier-4 receiving data items accessed during execution of the statement.

17. Either of the following situations causes an overflow condition:

    a) An UNSTRING is initiated, and the value in the data item referenced by identifier-7 is less than 1 or greater than the size of the data item referenced by identifier-1.

    b) If, during execution of an UNSTRING statement, all receiving areas have been acted upon, and the data item referenced by identifier-1 contains characters that have not been examined.

18. When an overflow condition exists, the UNSTRING operation is terminated and control is transferred to the end of the UNSTRING statement or, if the ON OVERFLOW phrase is specified, to imperative-statement-1. If control is transferred to imperative-statement-1, execution continues according to the rules for each statement specified in imperative-statement-1. If a procedure branching or conditional statement which causes explicit transfer of control is executed, control is transferred in accordance with the rules for that statement; otherwise, upon completion of the execution of imperative-statement-1, control is transferred to the end of the UNSTRING statement.

19. The evaluation of subscripting for the identifiers is as follows:

    a) Any subscripting associated with identifier-1, identifier-7, identifier-8 is evaluated only once, immediately before any data is transferred as the result of the execution of the UNSTRING

statement.

b) Any subscripting associated with identifier-2 or identifier-3 is evaluated only once, immediately before the examination of the sending field for its respective set of delimiters.

c) Any subscripting associated with identifier-4, identifier-5, identifier-6 is evaluated immediately before the transfer of data into the respective data item.

d) If identifier-6, identifier-7 or identifier-8 are used as subscripts within the UNSTRING statement, their evaluation occurs only once, at the beginning of the execution of the statement.

20. The END-UNSTRING phrase delimits the scope of the UNSTRING statement. (See Scope of Statements.)

21. If identifier-1, identifier-2, or identifier-3 occupies the same storage area as identifier-4, identifier-5, identifier-6, identifier-7, or identifier-8, or if identifier-4, identifier-5, or identifier-6, occupies the same storage area as identifier-7 or identifier-8, or if identifier-7 and identifier-8 occupy the same storage area, the result of the execution of this statement is undefined, even if they are defined by the same data description entry. (See Overlapping Operands.)

Function

The USE statement specifies procedures for input-output error handling that are in addition to the standard procedures provided by the input-output control system.

General Format

```
USE AFTER STANDARD { EXCEPTION } PROCEDURE ON { {file-name-1}... }
                   { ERROR     }              { INPUT          }
                                              { OUTPUT         }
                                              { I-O            }
                                              { EXTEND         }
```

Syntax Rules

1. A USE statement, when present, must immediately follow a section header in the declaratives portion of the Procedure Division and must appear in a sentence by itself. The remainder of the section must consist of zero, one, or more procedural paragraphs that define the procedures to be used.

2. The USE statement is never executed; it merely defines the conditions calling for the execution of the USE procedures.

3. Appearance of file-name-1 in a USE statement must not cause the simultaneous request for execution of more than one USE procedure.

4. The words ERROR and EXCEPTION are synonymous and may be used interchangeably.

5. The files referenced in the USE statement need not all have the same organization or access.

6. The INPUT, OUTPUT, I-O and EXTEND phrases may each be specified only once in the declarative portion of a given Procedure Division.

General Rules

1. Declarative procedures may be included in any COBOL source program irrespective of whether the program contains or is contained within another program. A declarative is invoked when any of the conditions described in the USE statement which prefaces the declarative occurs while the program is being executed.

2. Within a declarative, there must be no reference to any nondeclarative procedures.

3. Procedure-names associated with a USE statement may be referenced in a different declarative section or in a nondeclarative procedure only

with a PERFORM statement.

4. When file-name-1 is specified explicitly, no other USE statement for the same condition applies to file-name-1.

5. The procedures associated with a USE statement are executed as follows upon the unsuccessful execution of an input-output operation unless an AT END or INVALID KEY phrase takes precedence:

   a) If file-name-1 is specified, the associated procedure is executed when the condition described in the USE statement occurs.

   b) If INPUT is specified, the associated procedure is executed when the condition described in the USE statement occurs for any file open in the input mode or in the process of being opened in the input mode, except those files referenced by file-name-1 in another USE statement specifying the same condition.

   c) If OUTPUT is specified, the associated procedure is executed when the condition described in the USE statement occurs for any file open in the output mode or in the process of being opened in the output mode, except those files referenced by file-name-1 in another USE statement specifying the same condition.

   d) If I-O is specified, the associated procedure is executed when the condition described in the USE statement occurs for any file open in the I-O mode or in the process of being opened in the I-O mode, except those files referenced by file-name-1 in another USE statement specifying the same condition.

   e) If EXTEND is specified, the associated procedure is executed when the condition described in the USE statement occurs for any file open in the extend mode or in the process of being opened in the extend mode, except those files referenced by file-name-1 in another USE statement specifying the same condition.

6. The procedures designated by the USE statement are executed by the input-output system after completing the standard input-output error routine, or upon recognition of the invalid key or at end condition, when the INVALID KEY phrase or AT END phrase, respectively, has not been specified in the input-output statement.

7. After execution of the USE procedure, control is returned to the invoking routine.

8. Within a USE procedure, there must not be the execution of any statement that would cause the execution of a USE procedure that had previously been invoked and had not yet returned control to the invoking routine.

Function

The WRITE statement releases a logical record for an output or input-output file. It can also be used for vertical positioning of lines within a logical page.

General Format

Format 1 - Sequential Organization

```
WRITE record-name-1 [FROM identifier-1]

[ { BEFORE } ADVANCING { { identifier-2 } [ LINE ] }                    ]
[ { AFTER  }           { { integer-1    } [ LINES ] }                   ]
[                      { { mnemonic-name-1 }                           ]
[                      { { PAGE            }                           ]

[ AT { END-OF-PAGE } imperative-statement-1 [END-WRITE] ]
     { EOP         }
```

Format 2 - Relative and Indexed Organization

```
WRITE record-name-1 [FROM identifier-1] [WITH LOCK]

    [INVALID KEY imperative-statement-1 [END-WRITE]]
```

Syntax Rules

1. Record-name-1 and identifier-1 must not reference the same storage area.

2. Record-name-1 is the name of a logical record in the File Section of the Data Division and may be qualified.

Format 1

3. The ADVANCING mnemonic-name-1 phrase cannot be specified when writing a record to a file whose file description entry contains the LINAGE clause.

4. Identifier-2 must be the name of an integer data item.

5. Integer-1 may be positive or zero, but must not be negative.

6. When mnemonic-name-1 is specified, the name is associated with an implementor-name of the form 'LINE n'. Mnemonic-name-1 is defined in the SPECIAL-NAMES paragraph of the Environment Division.

7. The phrases ADVANCING PAGE and END-OF-PAGE must not both be specified in a single WRITE statement.

8. If the END-OF-PAGE phrase is specified, the LINAGE clause must be specified in the file description entry for the associated file.

9. The words END-OF-PAGE and EOP are equivalent.

10. If the BEFORE/AFTER clause is specified, the file referenced by the file-name associated with record-name-1 must be assigned to TEXT-FILE, SYSPRINT1, SYSPRINT2, SYSPRINT3, SYSPRINT4, or WS-PRINTER.

Format 2

11. The INVALID KEY phrase must be specified if an applicable USE AFTER STANDARD EXCEPTION procedure is not specified for the associated file-name.

General Rules

1. The file referenced by the file-name associated with record-name-1 must be open in the output or extend mode when the WRITE statement is executed.

2. The logical record released by the execution of the WRITE statement is no longer available in the record area unless the file-name associated with record-name-1 is specified in a SAME RECORD AREA clause. In this case, the logical record is available to the program as a record of other files whose file-names are specified in the same SAME RECORD AREA clause.

3. The result of the execution of a WRITE statement with the FROM phrase is equivalent to the execution of the following statements in the order specified:

   a) The statement:

      MOVE identifier-1 TO record-name-1

   according to the rules specified for the MOVE statement.

b) The same WRITE statement without the FROM phrase.

4. After the execution of the WRITE statement is complete, the information in the area referenced by identifier-1 is available, even though the information in the area referenced by record-name-1 is not available except as specified by the SAME RECORD AREA clause.

5. The file position indicator is not affected by the execution of a WRITE statement.

6. The execution of the WRITE statement causes the value of the I-O status of the file-name associated with record-name-1 to be updated.

7. The execution of the WRITE statement releases a logical record to the operating system.

8. The END-WRITE phrase delimits the scope of the WRITE statement. (See Scope of Statements.)

Format 1

9. The successor relationship of a sequential file is established by the order of execution of WRITE statements when the file is created. The relationship does not change except when records are added to the end of a file.

10. When a sequential file is open in the extend mode, the execution of the WRITE statement will add records to the end of the file as though the file were open in the output mode. If there are records in the file, the first record written after the execution of the OPEN statement with the EXTEND phrase is the successor of the last record in the file.

11. When an attempt is made to write beyond the externally defined boundaries of a sequential file, an exception condition exists and the contents of the record area are unaffected. The following actions take place:

    a) The value of the I-O status of the file-name associated with record-name-1 is set to a value indicating a boundary violation.

    b) If a USE AFTER STANDARD EXCEPTION declarative is explicitly or implicitly specified for the file-name associated with record-name-1, that declarative procedure will then be executed.

    c) If a USE AFTER STANDARD EXCEPTION declarative is not explicitly or implicitly specified for the file-name associated with record-name-1, the result is undefined.

12. The maximum record size for a file is established at the time the file is created and must not subsequently be changed.

13. The number of character positions on a mass storage device required to store a logical record in a file may or may not be equal to the number of character positions defined by the logical description of that record in the program.

14. Both the ADVANCING phrase and the END-OF-PAGE phrase allow control of the vertical positioning of each line on a representation of a printed page. If the ADVANCING phrase is not used, automatic advancing will be provided to act as if the user specified AFTER ADVANCING 1 LINE. If the ADVANCING phrase is used, advancing is provided as follows:

    a) If integer-1 or the value of the data item referenced by identifier-2 is positive, the representation of the printed page is advanced the number of lines equal to that value.

    b) If the value of the data item referenced by identifier-2 is negative, the results are undefined.

    c) If integer-1 or the value of the data item referenced by identifier-2 is zero, no repositioning of the representation of the printed page is performed.

    d) If mnemonic-name-1 is specified, the representation of the printed page is advanced to the line indicated by the associated implementor-name.

    e) If the BEFORE phrase is used, the line is presented before the representation of the printed page is advanced according to rules a, b, c, and d above.

    f) If the AFTER phrase is used, the line is presented after the representation of the printed page is advanced according to rules a, b, c, and d above.

    g) If PAGE is specified and the LINAGE clause is specified in the associated file description entry, the record is presented on the logical page before or after (depending on the phrase used) the device is repositioned to the next logical page. The repositioning is to the first line that can be written on the next logical page as specified in the LINAGE clause.

    h) If PAGE is specified and the LINAGE clause is not specified in the associated file description entry, the record is presented on the physical page before or after (depending on the phrase used) the device is repositioned to the next physical page. The repositioning to the next physical page is accomplished by a new page control character. If physical page has no meaning in conjunction with a specific device, advancing will be provided to act as if the user had specified BEFORE or AFTER (depending on the phrase used) ADVANCING 1 LINE.

15. If the logical end of the representation of the printed page is reached during the execution of a WRITE statement with the END-OF-PAGE phrase, imperative-statement-1 specified in the END-OF-PAGE phrase is executed. The logical end is specified in the LINAGE clause associated with record-name-1.

16. An end-of-page condition occurs when the execution of a given WRITE statement with the END-OF-PAGE phrase causes printing or spacing within the footing area of a page body. This occurs when the

execution of such a WRITE statement causes the LINAGE-COUNTER to equal or exceed the value specified by integer-2 or the data item referenced by data-name-2 of the LINAGE clause if specified. In this case, the WRITE statement is executed and then imperative-statement-1 in the END-OF-PAGE phrase is executed.

17. A page overflow condition occurs when the execution of a given WRITE statement (with or without an END-OF-PAGE phrase) cannot be fully accommodated within the current page body.

18. This occurs when a WRITE statement, if executed, would cause the LINAGE-COUNTER to exceed the value specified by integer-1 or the data item referenced by data-name-1 of the LINAGE clause. In this case, the record is presented on the logical page before or after (depending on the phrase used) the device is repositioned to the first line that can be written on the next logical page as specified in the LINAGE clause. Imperative-statement-1 in the END-OF-PAGE clause, if specified, is executed after the record is written and the device has been repositioned.

19. A page overflow occurs when the execution of a given WRITE statement would cause LINAGE-COUNTER to simultaneously exceed the value of both integer-2 or the data item referenced by data-name-2 of the LINAGE clause and integer-1 or the data item referenced by data-name-1 of the LINAGE clause.

Format 2

20. Execution of a WRITE statement causes the contents of the record area to be released. The MOS file system utilizes the contents of the record keys in such a way that subsequent access of the record may be made based upon any of those specified record keys.

21. For indexed files the value of the prime record key must be unique within the records in the file.

22. For indexed files the data item specified as the prime record key must be set by the program to the desired value prior to the execution of the WRITE statement.

23. If an indexed file is open in the sequential access mode, records must be released to the MOS file system in ascending order of prime record key values according to the collating sequence of the file. When the file is open in the extend mode, the first record released to the MOS file system must have a prime record key whose value is greater than the highest prime record key value existing in the file.

24. If an indexed file is open in the random or dynamic access mode, records may be released to the MOS file system in any program-specified order.

25. When the ALTERNATE RECORD KEY clause is specified in the file control entry or file description entry for an indexed file, the value of the alternate record key may be nonunique only if the DUPLICATES phrase is specified for that data item. In this case the MOS file system provides storage of records such that when records are accessed

sequentially, the order of retrieval of those records is the order in which they are released to the MOS file system.

26. When a relative file is opened in the output mode, records may be placed into the file by one of the following:

    a) If the access mode is sequential, the WRITE statement causes a record to be released to the MOS file system. The first record has a relative record number of one, and subsequent records released have relative record numbers of 2, 3, 4, ... If the RELATIVE KEY phrase is specified for the file-name associated with record-name-1, the relative record number of the record being released is moved into the relative key data item by the MOS file system during execution of the WRITE statement according to the rules for the MOVE statement. (See The MOVE Statement.)

    b) If the access mode is random or dynamic, prior to the execution of the WRITE statement the value of the relative key data item must be initialized by the program with the relative record number to be associated with the record in the record area. That record is then released to the MOS file system by execution of the WRITE statement.

27. When a relative file is open in the extend mode, records are inserted into the file. The first record released to the MOS file system has a relative record number one greater than the highest relative record number existing in the file. Subsequent records released to the MOS file system have consecutively higher relative record numbers. If the RELATIVE KEY phrase is specified for the file-name associated with record-name-1, the relative record number of the record being released is moved into the relative key data item by the MOS file system during execution of the WRITE statement according to the rules for the MOVE statement. (See The MOVE Statement.)

28. When a relative file is opened in the I-O mode and the access mode is random or dynamic, records are to be inserted in the associated file. The value of the relative key data item must be initialized by the program with the relative record number to be associated with the record in the record area. Execution of the WRITE statement then causes the content of the record area to be released to the MOS file system.

29. The invalid key condition exists under the following circumstances:

    a) When an indexed file is open in the sequential access mode, and the file also is open in the output or extend mode, and the value of the prime record key is not greater than the value of the prime record key of the previous record, or

    b) When an indexed file is open in the output or I-O mode, and the value of the prime record key equals the value of a prime record key of a record already existing in the file, or

    c) When an indexed file is open in the output, extend, or I-O mode, and the value of an alternate record key for which duplicates are not allowed equals the value of the corresponding data item of a

record already existing in the file, or

d) When an attempt is made to write beyond the externally-defined boundaries of the file, or

e) When the access mode is random or dynamic, and the relative key data item specifies a record which already exists in the file, or

f) When the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file.

30. When the invalid key condition is recognized, the execution of the WRITE statement is unsuccessful, the content of the record area is unaffected, and the I-O status of the file-name associated with record-name-1 is set to a value indicating the cause of the condition. Execution of the program proceeds according to the rules for an invalid key condition. (See I-O Status, and The Invalid Key Condition.)

31. The use of the WITH LOCK phrase locks the record in the sense that other system programs and processes can read that record but cannot write it.

    A record is locked until it is unlocked by the program or process that locked it in one of the following ways:

    a) By execution of an UNLOCK statement specifying the same key value; or

    b) By execution of a REWRITE statement specifying the same key value that contains the NO LOCK phrase; or

    c) By execution of a CLOSE statement for the file; or

    d) By termination.

32. An attempt to execute a WRITE statement for a record placed in lock mode by another system program or process results in an I-O error indication in the I-O status associated with file-name-1.

6. SCREEN HANDLING

Screen Handling provides an extension to the source language facilities for the formatting and the handling of data on the terminal.

SCREEN HANDLING IN THE DATA DIVISION

SCREEN SECTION

The Screen Section is located in the Data Division of a source program. The Screen Section contains the description of the data to be processed by the Screen Handling facility.

The general format of the Screen Section is shown below:

SCREEN SECTION.

[screen-record-description-entry] ...

SCREEN RECORD DESCRIPTION STRUCTURE

A screen record description consists of a set of screen description entries which describe the characteristics of all or a portion (a range of lines) of a screen. The screen description entry is very much like a data description entry in other sections of the Data Division. Each entry consists of a level-number followed by a series of independent clauses as required. A screen record description may have a hierarchical structure and therefore the clauses used within an entry may vary considerably, depending upon whether or not it is followed by subordinate entries. The structure of a screen record description and the elements allowed in a screen record description entry are explained in Concept of Levels; and in The Screen Description Entry.

THE SCREEN DESCRIPTION ENTRY

Function

The screen description entry describes a screen data element (literal screen element entry and data-item screen element entry) or its hierarchical structure (group entry). Screen data elements are elementary items. Each field appearing on a screen is described by a screen element. The screen element specifies the field's positioning on the screen, the source and/or destination of the data it is to contain or its literal value, and any screen properties associated with it (bell, blanking, blinking, etc.). An input field is defined by a data-item screen element entry (format 3) containing a TO phrase in its PICTURE clause. An output field is defined by a literal screen element entry (format 2) or a data-item screen element entry (format 3) containing a FROM phrase in its PICTURE clause. An input-output field is defined by a data-item screen element entry (format 3) containing a USING phrase or FROM and TO phrases in its PICTURE clause.

General Format

Format 1 (Group Entry)

```
level-number [ screen-name-1 ]
             [ FILLER        ]

    [BLANK SCREEN]

    [BELL]

    [BLINK]

    [BOX [integer-1]]

    [HIGHLIGHT]

    [REVERSE]

    [UNDERLINE]

    [LINE NUMBER IS [PLUS] integer-2]

    [AUTO]
```

[SECURE]

[REQUIRED]

[FULL]

[HEBREW].

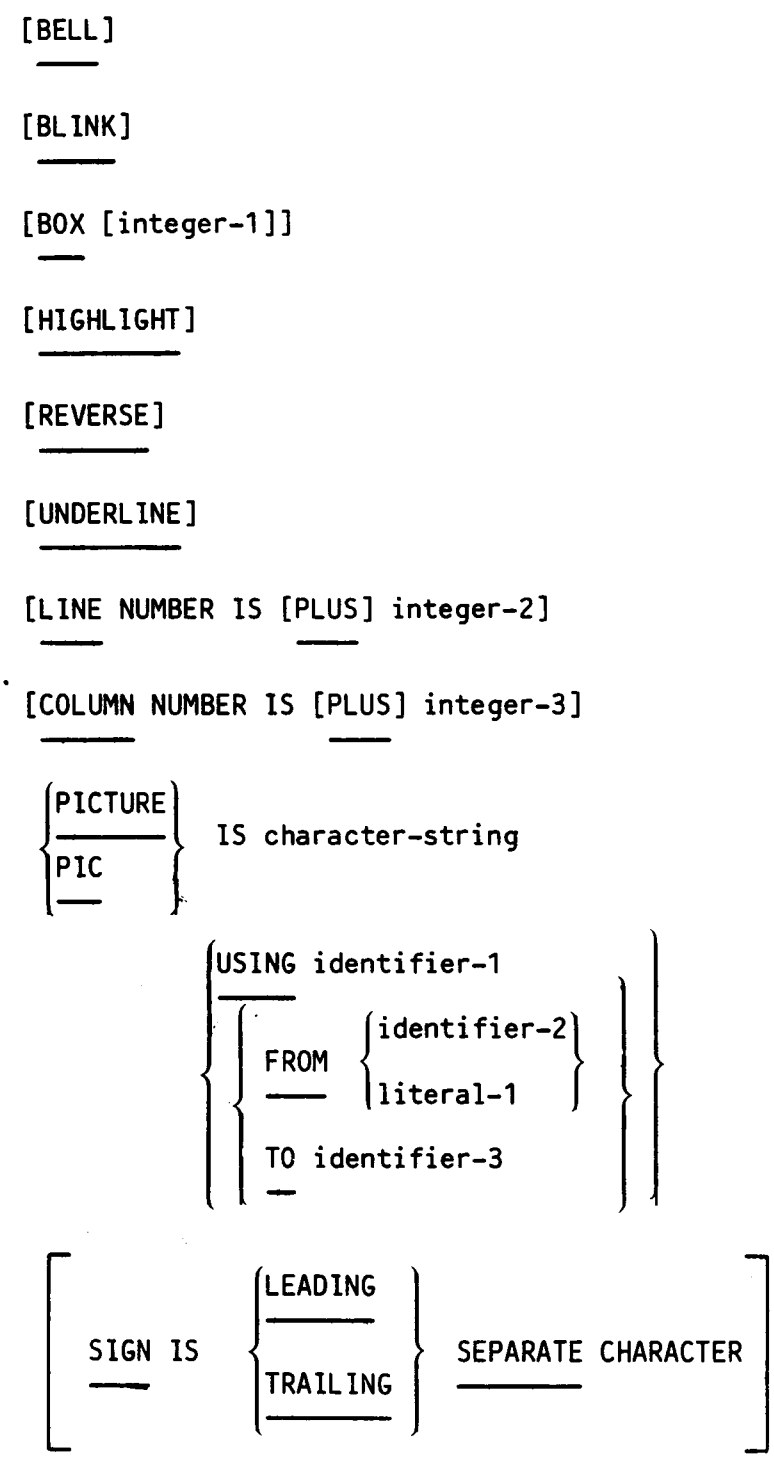
Format 2 (Literal Screen Element Entry)

```
level-number [ screen-name-1 ]
             [ FILLER      ]

[ BLANK { SCREEN } ]
        { LINE   }

[BLANK WHEN ZERO]

[BELL]

[BLINK]

[BOX [integer-1]]

[HIGHLIGHT]

[REVERSE]

[UNDERLINE]

[LINE NUMBER IS [PLUS] integer-2]

[COLUMN NUMBER IS [PLUS] integer-3]
```

```
{PICTURE}
{PIC    }  IS character-string

[ SIGN IS {LEADING }  SEPARATE CHARACTER ]
          {TRAILING}

VALUE IS literal-1.
```

Format 3 (Data-Item Screen Element Entry)

```
level-number [screen-name-1]
             [FILLER       ]

[BLANK {SCREEN}]
       {LINE  }
```


[BELL]
[BLINK]
[BOX [integer-1]]
[HIGHLIGHT]
[REVERSE]
[UNDERLINE]
[LINE NUMBER IS [PLUS] integer-2]
[COLUMN NUMBER IS [PLUS] integer-3]
PICTURE
PIC
IS character-string
USING identifier-1
FROM
identifier-2
literal-1
TO identifier-3
SIGN IS
LEADING
TRAILING
SEPARATE CHARACTER

```
[BLANK WHEN ZERO]
 ----- ----

[ {JUSTIFIED} RIGHT ]
 { ---------- }
 { JUST      }
 { ----      }

[AUTO]
 ----

[SECURE]
 ------

[REQUIRED]
 --------

[FULL]
 ----

[HEBREW].
 ------
```

Syntax Rules

1. Level-number may be any number from 01 through 49.

2. Screen-name-1 or FILLER clause, if specified, must immediately follow the level-number. The remaining clauses may be written in any order.

3. The PICTURE clause is required for each elementary item.

General Rules

1. If the SECURE clause is specified in format 1, it applies to each input and input-output field in the group.

2. If AUTO, REQUIRED, FULL or HEBREW clause is specified in format 1, it applies to each input and input-output field in the group.

3. If the BLINK clause is specified in format 1, it applies to each input-output and output item in the group.

4. If the BELL, HIGHLIGHT, UNDERLINE, or REVERSE clause is specified in format 1, it applies to each elementary item in the group.

5. An entry containing the LINE clause without the PLUS phrase specified must not be subordinate to an entry which contains a LINE clause without the PLUS phrase.

6. LINE and COLUMN clauses may not be specified within a screen record description entry in such a way that fields overlap on the screen.

Function

The AUTO clause causes the cursor to be automatically moved to the next field when, in ACCEPT context, the last character of the input or input-output field whose definition contains this clause is entered.

General Format

AUTO

Syntax Rules

1. The AUTO clause may not be specified in a literal screen element entry.

General Rules

1. If the AUTO clause is specified at group level, it applies to each input and input-output item in that group.

2. The AUTO clause is significant in ACCEPT context. This includes the execution of the ACCEPT statement; the ACCEPTing of an input field during the execution of the EDIT statement for a group; and in certain situations the EDITing of an input-output field. (See the EDIT statement).

3. The AUTO clause is ignored for an elementary output field.

4. If there is only one field to be input, or if the field is the last one in a group, the ACCEPT statement is complete.

BELL clause

Function

The BELL clause causes the work station audio tone to sound.

General Format

BELL

Syntax Rules

1. The BELL clause may be specified in any screen description entry.

General Rules

1. The use of this clause results in the audio tone sounding when the screen data element in which it is specified is processed during the execution of an ACCEPT, a DISPLAY or an EDIT statement. The audio tone sounds at the start of the display of the element.

2. If the BELL clause is specified at group level, it applies to each elementary item in that group.

Function

The BLANK clause clears all or a portion of a screen line before the accept or display of a screen data element or clears the whole screen before the display of a screen data element.

General Format

BLANK { SCREEN / LINE }

Syntax Rules

1. The BLANK SCREEN clause may be specified in any screen description entry.

2. The BLANK LINE clause may be specified only for an elementary screen item.

General Rules

1. When BLANK SCREEN is specified, blanking begins at the line and column coordinates specified for the screen data element in whose description it is included and continues through the end of the screen. After the screen is cleared, the cursor remains positioned at the coordinates at which the group or elementary item begins.

2. When BLANK LINE is specified, blanking begins at the line and column coordinates specified for the screen data element in whose description it is included and continues through the end of the line.

3. Blanking, whether of the screen or of a line or portion thereof, is always performed before the actual accept, display, or edit of the field takes place.

4. If the BLANK clause is not specified, only the particular character positions corresponding to the screen data element are modified when the element is displayed. The rest of the screen content remains the same.

BLANK WHEN ZERO clause

Function

The BLANK WHEN ZERO clause displays blanks (space characters) when the value of a data item is zero.

General Format

BLANK WHEN ZERO

Syntax Rules

1. The BLANK WHEN ZERO clause can be specified only for an elementary item whose PICTURE is specified as numeric or numeric edited (see The PICTURE Clause).

General Rules

1. The BLANK WHEN ZERO clause is ignored for an elementary input field.

2. During the execution of a DISPLAY statement for an element containing the BLANK WHEN ZERO clause, if the value of the element is zero, spaces are displayed on the screen.

3. During the execution of an EDIT statement for an element containing the BLANK WHEN ZERO clause, if the modified value of the element is zero, spaces are displayed on the screen.

4. When the BLANK WHEN ZERO clause is specified for an element whose PICTURE clause is numeric, the category of the element is considered to be numeric edited.

Function

The BLINK clause blinks an output item when it is displayed on the screen.

General Format

BLINK

Syntax Rules

1. The BLINK clause may be specified in any screen description entry.

General Rules

1. If the BLINK clause is specified at group level, it applies to each input-output and output item in that group.

2. The BLINK clause is ignored for an elementary input field.

BOX clause

Function

The BOX clause displays a box around the item or group which specifies the clause.

General Format

BOX [integer-1]

Syntax Rules

1. The BOX clause may be specified in any screen description entry.

2. Integer-1, if present, may be zero.

3. Integer-1, if present, cannot be greater than 79.

General Rules

1. If the BOX clause is specified at elementary level, a box is drawn around the field.

2. If the BOX clause is specified at group level, a box is drawn around the group. The coordinates are chosen using the lowest and highest line numbers and the lowest column number and the highest ending column number of all the fields in the group.

3. If integer-1 is not specified, zero is assumed.

4. Integer-1 or its default represents the size of the boundary surrounding the item or group. If it is zero, the box is drawn immediately. Otherwise, each unit represents a boundary of one column and one line. That is, integer-1 or its default is subtracted from the first column and line number of the box and added to the last column and line number to determine the actual coordinates used to draw the box.

5. If the coordinates thus calculated lie outside the screen boundary is used.

Function

The COLUMN clause specifies the horizontal screen coordinate for a screen data element.

General Format

COLUMN NUMBER IS [PLUS] integer-1

Syntax Rules

1. Integer-1 must be in the range 1 to 80 inclusive.

2. The COLUMN clause can be specified only for an elementary item.

General Rules

1. The COLUMN clause, in conjunction with the LINE clause, establishes the starting coordinates for a screen data element. The COLUMN clause specifies the horizontal coordinate.

2. The COLUMN clause without the PLUS phrase specifies the absolute screen column position of the leftmost character of the screen data element.

3. The COLUMN clause with the PLUS phrase specifies a screen column number relative to the rightmost column of the last defined screen data element.

4. If the COLUMN clause is not specified:

   a) If a LINE clause is specified, and no previous screen data element has been defined for that line, COLUMN 1 is assumed.

   b) If no LINE clause is specified, but a previous screen data element has been defined, COLUMN PLUS 1 is assumed.

   c) If no LINE clause is specified and no previous screen data element has been defined, COLUMN 1 of LINE 1 is assumed.

5. The implicit or explicit value of integer-1, whether as an absolute or relative column position, must be such that the rightmost character position of the element is not outside the number of columns available on the line on which that element appears.

FULL clause

Function

The FULL clause specifies that, in ACCEPT context, the entire field must be filled by characters by the user, or that no characters may be entered.

General Format

FULL

Syntax Rules

1. The FULL clause may not be specified in a literal screen element entry.

2. If the FULL clause is specified in a data-item screen element entry, the JUSTIFIED clause may not be specified.

General Rules

1. If the FULL clause is specified at group level, it applies to each input and input-output field in that group.

2. The FULL clause is significant in ACCEPT context. This includes the execution of the ACCEPT statement; the ACCEPTing of an input field during the execution of the EDIT statement for a group; and in certain situations the EDITing of an input-output field. (See the EDIT statement).

3. The FULL clause requires that the user must enter the entire field before processing continues. However, if only the ENTER key is typed, the field is bypassed.

4. The specification of the FULL and REQUIRED clauses together requires that the user must always entirely fill the field.

5. The FULL clause is ignored for an elementary output field.

Function

The COLUMN clause specifies the horizontal screen coordinate for a screen data element.

General Format

COLUMN NUMBER IS [PLUS] integer-1

Syntax Rules

1. Integer-1 must be in the range 1 to 80 inclusive.

2. The COLUMN clause can be specified only for an elementary item.

General Rules

1. The COLUMN clause, in conjunction with the LINE clause, establishes the starting coordinates for a screen data element. The COLUMN clause specifies the horizontal coordinate.

2. The COLUMN clause without the PLUS phrase specifies the absolute screen column position of the leftmost character of the screen data element.

3. The COLUMN clause with the PLUS phrase specifies a screen column number relative to the rightmost column of the last defined screen data element.

4. If the COLUMN clause is not specified:

   a) If a LINE clause is specified, and no previous screen data element has been defined for that line, COLUMN 1 is assumed.

   b) If no LINE clause is specified, but a previous screen data element has been defined, COLUMN PLUS 1 is assumed.

   c) If no LINE clause is specified and no previous screen data element has been defined, COLUMN 1 of LINE 1 is assumed.

5. The implicit or explicit value of integer-1, whether as an absolute or relative column position, must be such that the rightmost character position of the element is not outside the number of columns available on the line on which that element appears.

FULL clause

Function

The FULL clause specifies that, in ACCEPT context, the entire field must be filled by characters by the user, or that no characters may be entered.

General Format

FULL

Syntax Rules

1. The FULL clause may not be specified in a literal screen element entry.

General Rules

1. If the FULL clause is specified at group level, it applies to each input and input-output field in that group.

2. The FULL clause is significant in ACCEPT context. This includes the execution of the ACCEPT statement; the ACCEPTing of an input field during the execution of the EDIT statement for a group; and in certain situations the EDITing of an input-output field. (See the EDIT statement).

3. The FULL clause requires that the user must enter the entire field before processing continues. However, if only the ENTER key is typed, the field is bypassed.

4. The specification of the FULL and REQUIRED clauses together requires that the user must always entirely fill the input field.

5. The FULL clause is ignored for an elementary output field.

Function

The HEBREW clause specifies that the user must enter the field from right to left rather than from left to right.

General Format

HEBREW

Syntax Rules

1. The HEBREW clause may not be specified in a literal screen element entry.

General Rules

1. If the HEBREW clause is specified at group level, it applies to each input and input-output field in that group.

2. The HEBREW clause is ignored for an elementary output field.

HIGHLIGHT clause

Function

The HIGHLIGHT clause specifies that the input or output field is to appear on the screen with the highest intensity.

General Format

HIGHLIGHT

Syntax Rules

1. The HIGHLIGHT clause may be specified in any screen description entry.

General Rules

1. If the HIGHLIGHT clause is specified at group level, it applies to each elementary item in that group.

Function

The JUSTIFIED clause specifies non-standard positioning of data within a screen data element.

General Format

```
{ JUSTIFIED } RIGHT
{ JUST      }
```

Syntax Rules

1. The JUSTIFIED clause can be specified only for an elementary item.

2. The JUSTIFIED clause may not be specified for a screen data element described as numeric, numeric edited, or alphanumeric edited or for a screen data element described as FULL.

3. JUSTIFIED and JUST are equivalent and can be used interchangeably.

General Rules

1. When the JUSTIFIED clause is specified for a screen data element being accepted, data are aligned in the screen data element beginning with its rightmost character position and the leftmost characters not occupied by transmitted data are space filled. Thus data can be entered left-justified for ease of the user and will be right-justified in the screen data element before the move to the TO or USING field.

2. When the JUSTIFIED clause is specified for a screen data element being displayed:

   a) If the sending data item is larger than the screen data element, data are aligned beginning with the rightmost character position of the screen data element and the leftmost characters in excess of the capacity of the screen data element are truncated.

   b) If the screen data element is larger than the sending data item, the leftmost character positions of the screen data element not occupied by transmitted data are space-filled.

3. When this clause is not used data are justified to the left when transferred to the screen data element.

LEVEL NUMBER description

Function

The level-number indicates the position of a screen item within the hierarchical structure of a logical record.

General Format

level-number

Syntax Rules

1. A level-number in the range 01 through 49 is required as the first element in a screen description entry.

General Rules

1. The first entry in each screen record description must have a level number 01. Level numbers 02 through 49 are used in the entries that describe the elements subordinate to the 01 entry (if the 01 is a group item).

2. In effect, the 01 level item identifies the beginning of a 'complete' or a 'partial' screen. Thus, beginning with the 01 level and including any subordinate level numbers, the description must be such that the entire element can appear on the screen at one time (taking into account the positioning of each elementary item within the screen data element as defined by the LINE and COLUMN clauses).

Function

The LINE clause specifies the vertical screen coordinate for a screen data element.

General Format

LINE NUMBER IS [PLUS] integer-1

Syntax Rules

1. Integer-1 must be in the range 1 to 24 inclusive.

2. Within a given screen record description entry, an entry that contains a LINE clause without the PLUS phrase may not be subordinate to an entry that contains a LINE clause.

General Rules

1. The LINE clause, in conjunction with the COLUMN clause, establishes the starting coordinates for a screen data element. The LINE clause specifies the vertical coordinate.

2. The LINE clause without the PLUS phrase specifies the absolute screen line position of the screen data element.

3. The LINE clause with the PLUS phrase specifies a screen column number relative to the line containing the last defined screen data element. Integer-1 must be such that the line occupied by the element being described is not outside the available number of lines on the screen.

4. If the LINE clause is omitted:

   a) If no previous screen data element has been defined, LINE 1 of the screen is assumed.

   b) If a previous screen data element has been defined, the line of that previous element is assumed.

5. The implicit or explicit value of integer-1, whether as an absolute or relative line number, must be such that the line number of the element is not outside the number of lines available on the screen.

PICTURE clause

Function

The PICTURE clause describes the general characteristics and editing requirements of an elementary screen data element. It is also used to specify the data item whose contents are to be displayed when this element appears on the screen and the data item in which the contents of this element are to be stored when it is accepted or modified.

General Format

Format 1 (Literal Screen Element Entry)

```
{PICTURE}
{------ } IS character-string
{PIC    }
{---    }
```

Format 2 (Data-Item Screen Element Entry)

```
{PICTURE}                                  {USING identifier-1                                    }
{------ } IS character-string {              {-----                                               }
{PIC    }                    {              {     FROM {identifier-2}                             }
{---    }                    {              {     ---- {literal-1   }                             }
                             {              {                                                     }
                             {              {     TO identifier-3                                 }
                             {              {     --                                              }
```

Syntax Rules

1. In format 1 (and format 2 without the optional phrases), the PICTURE clause in Screen Handling is the same as the PICTURE clause in the DATA DIVISION. Thus the specifications for this portion of the clause (i.e., syntax and general rules for character-string) are located in the PICTURE Clause in the Data Division chapter.

   Note that a screen item described with a picture containing a 'V' or 'P's is accepted and displayed without scaling and without a decimal-point. There are no character positions allocated on the screen for these characters.

2. If the character 'S' appears in the character-string, the SIGN clause must be present.

3. If the USING or FROM phrase is specified, the SECURE clause may not be present.

4. If the USING/FROM/TO phrases are not present, the VALUE clause must be present.

5. If the USING/FROM/TO phrases are specified, the VALUE clause may not be present.

6. Identifier-1, identifier-2 and identifier-3 may not be defined in the Report or Screen Section.

General Rules

Format 2 only

1. The USING, FROM and TO phrases have meaning only when the screen element's name or a screen-name in its hierarchy is specified in an ACCEPT, a DISPLAY, or an EDIT statement.

2. When data is to be transferred to the screen from one data item, possibly edited, and stored in a different data item, both the FROM and the TO phrases must be used in the PICTURE clause of the screen data element.

3. When data is to be transferred to the screen, possibly modified, and stored in the same data item (as when reading, modifying and rewriting records of a file), the USING phrase must be used in the PICTURE clause of the screen data element.

4. Storage locations (identifier-1, identifier-2, identifier-3) need not be the same length as the screen data element. Transfers between storage locations and the screen data element are made in accordance with normal COBOL MOVE statement rules (see The MOVE Statement) as follows:

   a) Identifier-1, identifier-2 or literal-1 must be defined such that a MOVE statement from identifier-1, identifier-2 or literal-1 to an item specified with a PICTURE of character-string is legal. In addition, if identifier-1 or identifier-2 is defined as alphanumeric edited, the screen data element may not be modified by the execution of an EDIT statement, only accepted.

   b) Identifier-1 or identifier-3 must be defined such that a MOVE statement from an item specified with a PICTURE of character-string to identifier-1 or identifier-3 is legal with one exception: if the screen element is defined to be numeric edited, the rules used are those of a numeric integer/noninteger sending operand rather than numeric edited. Thus, if the screen data element is numeric edited, neither identifier-1 nor identifier-2 may be numeric edited. However, identifier-3 may be numeric edited.

5. When the FROM phrase is specified:

   a) On DISPLAY statement execution, data are transferred from identifier-2 or literal-1, after being edited in accordance with character-string, and displayed on the screen. The display begins at the screen position defined either implicitly or explicitly by the LINE and COLUMN clauses.

   b) The FROM phrase has no meaning in execution of an ACCEPT statement.

c) During EDIT statement execution for a group, an elementary screen item defined with only the FROM phrase is displayed as in the execution of a DISPLAY statement. Otherwise, literal-1 or data from identifier-2 are transferred after being edited in accordance, with character-string and the special modifications (see the EDIT statement), and displayed on the screen. The display begins at the screen position defined either implicitly or explicitly by the LINE and COLUMN clauses.

6. When the TO phrase is specified:

a) On ACCEPT statement execution, the data introduced for the screen data element are transferred to identifier-3 after being edited in accordance to the PICTURE character-string specified for identifier-3. If the screen data element is defined as numeric edited, the data introduced is converted to trailing separate sign format with the decimal point, if present, used for alignment removed before the transfer to identifier-3. If the screen data element is defined as alphanumeric edited, the data introduced is treated as alphanumeric without conversion before the transfer to identifier-3.

b) The TO phrase has no meaning in execution of a DISPLAY statement. If the FROM phrase is not specified for the same screen data element, spaces are displayed (see The DISPLAY Statement).

c) On EDIT statement execution, after the display according to the FROM phrase as described in rule 5a, the changes introduced for the screen data element are applied and the modified data are transferred to identifier-3 after being edited in accordance to the PICTURE character-string specified for identifier-3. If the screen data element is defined as numeric edited, the modified data is converted to trailing separate sign format with the decimal point, if present, used for alignment and any commas and other simple insertion characters removed before the transfer to identifier-3. If the screen data element is defined as alphanumeric edited, the modified data is converted to alphanumeric by removing any simple insertion characters before the transfer to identifier-3.

7. The USING clause is equivalent to a FROM and TO clause with a common identifier. When the USING phrase is specified:

a) On DISPLAY statement execution, data are transferred from identifier-1 as described in rule 5a.

b) On ACCEPT statement execution, the data introduced for the screen data element are transferred to identifier-1 as described in rule 6a.

c) On EDIT statement execution, data are first transferred from identifier-1, as described in 5c. Then the behavior for editing and transfer to identifier-1 is as described in rule 6c.

Function

The REQUIRED clause specifies that, in ACCEPT context, the user must enter at least one character in the input or input-output field.

General Format

REQUIRED

Syntax Rules

1. The REQUIRED clause may not be specified in a literal screen element entry.

General Rules

1. If the REQUIRED clause is specified at group level, it applies to each input and input-output field in that group.

2. The REQUIRED clause is significant in ACCEPT context. This includes the execution of the ACCEPT statement; the ACCEPTing of an input field during the execution of the EDIT statement for a group; and in certain situations the EDITing of an input-output field. (See the EDIT statement).

3. The specification of the FULL and REQUIRED clauses together requires that the input field must always be filled entirely by the user.

4. The REQUIRED clause is ignored for an elementary output field.

REVERSE clause

Function

The REVERSE clause specifies that the data is represented as black characters on a white background.

General Format

REVERSE

Syntax Rules

1. The REVERSE clause may be specified in any screen description entry.

General Rules

1. If the REVERSE clause is specified at group level, it applies to each elementary item in that group.

Function

A screen-name specifies the name of the screen item being described. The key word FILLER may be used to specify a screen item which is not referenced.

General Format

```
[screen-name-1]
[FILLER       ]
```

Syntax Rules

1. Screen-name-1 or the key word FILLER, if either is specified, must be the first word following the level-number in each screen description entry.

General Rules

1. If this clause is omitted, the screen item being described is treated as though FILLER had been specified.

2. The key word FILLER may be used to name a screen item. Under no circumstances can a FILLER item be referred to explicitly.

SECURE clause

Function

The SECURE clause causes the cursor not to be moved when the input field to which it applies is entered and no characters are displayed.

General Format

SECURE

Syntax Rules

1. The SECURE clause may not be specified in a literal screen element entry.

2. The SECURE clause may not be specified in a screen element entry containing a PICTURE clause with the USING or FROM phrase.

General Rules

1. If the SECURE clause is specified at group level, it applies to each input item in that group.

2. When the SECURE clause is used, characters introduced for the input field do not appear on the screen.

Function

The SIGN clause specifies the position and the mode of representation of the operational sign when it is necessary to describe these properties explicitly.

General Format

```
[SIGN IS]  { LEADING  }  SEPARATE CHARACTER
           { TRAILING }
```

Syntax Rules

1. The SIGN clause may be specified only for a numeric data description entry whose PICTURE contains the character 'S', the SIGNE clause must be specified.

2. When the SIGN clause is included in a screen description entry, the SEPARATE CHARACTER phrase must be specified.

3. If the PICTURE clause in the screen description entry contains the character 'S', the SIGN clause must be specifies.

General Rules

1. The optional SIGN clause, if present, specifies the position and the mode of representation of the operational sign for the numeric data description entry to which it applies. The SIGN clause applies only to numeric data description entries whose PICTURE contains the character 'S'; the 'S' indicates the presence of, but not necessarily, the position of the operational sign.

2. Since a SIGN clause in a screen description entry must specify the SEPARATE CHARACTER phrase, then:

   a) The operational sign will be presumed to be the leading (or trailing) character position of the elementary numeric data item; this character position is not a digit position.

   b) The letter 'S' in a PICTURE character-string is counted in determining the size of the item (in terms of standard data format characters).

   c) The operational signs for positive and negative are the standard data format characters '+' and '-', respectively.

3. Every numeric data description entry whose PICTURE contains the character 'S' is a signed numeric data description entry. If conversion is necessary for purposes of assignment, conversion takes place automatically.

Function

The UNDERLINE clause specifies that each character of the field is underlined before it is accepted and when it is displayed on the screen.

General Format

UNDERLINE

Syntax Rules

1. The UNDERLINE clause may be specified in any screen description entry.

General Rules

1. If the UNDERLINE clause is specified at group level, it applies to each elementary item in that group.

VALUE clause

Function

The VALUE clause defines the constant value of an output screen data element.

General Format

VALUE IS literal-1

Syntax Rules

1. The VALUE clause can be specified only at the elementary level.

2. The VALUE clause must not be used if the PICTURE clause for the element being described includes the USING, the FROM, or the TO phrase.

3. The VALUE clause in Screen Handling is the same as for the format 1 VALUE clause in the DATA DIVISION. Thus the specifications for the clause are located in the VALUE Clause in the Data Division chapter.

SCREEN HANDLING IN THE PROCEDURE DIVISION

INTRODUCTION

The Procedure Division statements that pertain to the screen handling facility are the ACCEPT and the DISPLAY statements. The same basic rules apply to these statements as described in the PROCEDURE DIVISION chapter; however, certain other actions and assumptions exist when a screen data element is referenced. These are explained in the following pages.

ACCEPT

Function

The ACCEPT statement causes data to be transferred from the screen to the specified data item.

General Format

Format 1

```
ACCEPT screen-name-1 [FROM {CONSOLE          }]
                           {mnemonic-name-1  }

       [ON ESCAPE imperative-statement-1 [END-ACCEPT]]
```

Format 2

```
ACCEPT identifier-1 FROM ESCAPE KEY
```

Format 3

```
ACCEPT screen-name-1 [FROM {CONSOLE          }]
                           {mnemonic-name-1  }

       TIME-OUT {integer-1   }
                {identifier-2}

       ON EXCEPTION imperative-statement-2 [END-ACCEPT]
```

Syntax Rules

1. Mnemonic-name-1 must be defined in the SPECIAL-NAMES paragraph of the Environment Division.

2. Screen-name-1 must be defined in a screen description entry in the SCREEN SECTION.

3. If screen-name-1 is an elementary item, the PICTURE clause specified in the associated screen description entry must contain either the USING or the TO option.

4. If screen-name-1 is a group item, at least one of the elementary items subordinate to it must be described according to rule 3 above.

5. Identifier-1 must be described as an integer whose picture description contains at least two digit positions and no P's.

6. Identifier-2 must reference an unsigned numeric integer data item.

7. Integer-1 or the value of the item referenced by identifier-2 may not be greater than 3200.

General Rules

1. If format 1 or format 3 of the ACCEPT statement is specified, the following actions take place:

   a) If screen-name-1 is a group and has the BELL clause specified, the work station audio tone is sounded (see The BELL Clause).

   b) If screen-name-1 is a group and has the BOX clause specified, a box is drawn around the group (see The BOX Clause).

   c) The cursor is positioned to the field that corresponds to elementary screen-name-1, or the first elementary data-item in screen-name-1 whose screen description entry contains either the USING or TO phrase in the PICTURE clause (see The PICTURE Clause).

   d) The work station audio tone is sounded, if the BELL clause is specified for the field (see The BELL Clause).

   e) A box is drawn around the field if the BOX clause is specified (see The BOX Clause).

   f) If the field is an input field (only TO phrase in the PICTURE clause), a string of underscore characters which has the same length as the field is displayed.

   g) Blanking of the remaining characters of the line takes place, if BLANK LINE clause is specified (see The BLANK Clause).

   h) The operator introduces the data. The input is validated on the basis of the PICTURE character-string after the field is terminated. The operator may use the left arrow key and right arrow key to move across and change characters which have been already entered. However, for a field defined AUTO, once the last character is entered, the field is terminated. Otherwise, the ENTER key terminates the field. If before the field is terminated the RESET key is struck, the entire field is cleared; a string of underscore characters which has the same length as the field is displayed; and the cursor is repositioned at the beginning of the field. Invalid data cause an error message to be displayed, and the cursor to be repositioned to the first column of the field. The data is still visible and the user is required to reenter data (i. e. the ENTER key may not be immediately pressed). If valid, the data are moved to the corresponding item defined in the USING or TO clause following the Standard Rules of the MOVE statement. The MOVE does not take place if the field is terminated before any characters are entered.

i) If screen-name-1 is a group, the cursor is positioned to the next input or input-output field, if any. Rules d through h are repeated until the last input or input-output field in the group has been accepted.

2. The rules for accepting numeric screen data are:

   a) A screen item described with a picture containing a 'V' or trailing 'P's is accepted without scaling and without a decimal-point.

   b) A screen item described with a picture containing 'S', '+', or '-' is accepted with a leading or trailing separate sign. If positive, no sign specification is necessary.

   c) A screen item described with a picture containing 'CR' or 'DB' is accepted as if the picture contained '+' or '-'; see rule b above.

   d) A screen item described with a picture containing a decimal point is accepted with a decimal point. The character is removed before the move takes place. The position of the decimal point is used for alignment.

   e) Insertion characters other than '.', '+', and '-' are not acceptable.

3. A screen item described as alphanumeric edited is accepted without consideration of insertion characters. Any insertion characters entered are treated as data.

4. If an escape character (EXIT character) is typed at any time, the ACCEPT is terminated, and the move of the current field is not executed. In addition, if the ON ESCAPE clause is specified, imperative-statement-1 is executed. The value 99 is stored for an eventual ACCEPT statement of format 2.

5. The use of format 3 places a time limit on each field accepted. Integer-1 or the value of the data item referenced by identifier-2 is the number of seconds the user has to enter the screen item starting from the moment when the cursor is positioned at the beginning of the field. If the indicated time has elapsed before the user has finished typing the field, the ACCEPT is terminated, the move of the current field is not executed, and control is transferred to imperative-statement-2 in the ON EXCEPTION clause. Execution continues according to the rules for each statement specified in imperative-statement-2.

6. The striking of a function key also terminates the ACCEPT statement. If the current field is defined without the FULL clause in its screen description entry and at least one character has been entered, the move is executed. The code of the function key typed is stored for an eventual ACCEPT statement of format 2. The various codes for the function keys of the standard and multifunctional keyboards are specified in the following table. Note that some of the codes for the standard keyboard have no equivalents on the multifunctional keyboard. Note also that EXIT is present in the table but the behavior

of this function key is different from the others since it aborts the current operation (see rule 4 above).

CODE	STANDARD FUNCTION KEY	MULTIFUNCTIONAL FUNCTION KEY	VALUE
1	F1	F1	ESC C
2	F2	F2	ESC B
3	F3	F3	ESC A
4	F4	F4	ESC @
5	F5	F5	ESC ?
6	F6	F6	ESC >
7	F7 (FUNCT)	F7	ESC =
8	F8	F8	ESC <
9	F9	F9	ESC ;
10	F10	F10	ESC :
11	F11	F11	ESC 9
12	F12	F12	ESC 8
13	F13	F13	ESC 7
14	F14	F14	ESC 6
15	F15 (SHIFT FUNCT)	F15	ESC 5
16	F16	F16	ESC 4
17	S5	F27	ESC D
18	S4	F25	ESC E
19	S3	F23	ESC F
20	S2	F21	ESC G
21	SEND	Not present	ESC M
22	CTRL S5	F28	ESC +
23	CTRL S4	F26	ESC ,
24	CTRL S3	F24	ESC -
25	CTRL S2	F22	ESC .
26	ERASE	Not present	ESC [
27	CL.ERR	Not present	ESC Q
28	CTRL EXIT	CTRL EXIT	ESC /
29	IC	IC	ESC P
30	DC	DC	ESC O
31	IL	IL	ESC J
32	DL	DL	ESC I
33	|←	|←	ESC W
34	→|	→|	ESC V
35	CTRL Ω	CTRL Ω	ESC 0
36	CTRL V	CTRL V	ESC 1
37	CTRL ←	CTRL ←	ESC 3
38	CTRL →	CTRL →	ESC 2
39	HOME	HOME	ESC Z
99	EXIT	EXIT	ESC H

In addition, the following codes are available for the standard keyboard when the scientific keyboard is present and for the multifunctional keyboard always.

CODE	STANDARD FUNCTION KEY	MULTIFUNCTIONAL FUNCTION KEY	VALUE
40	P1	F19	ESC *
41	P2	F20	ESC)
42	P3	CTRL |←	ESC (
43	P4	CTRL →|	ESC '
44	P5	CTRL HOME	ESC &
45	CTRL P1	Not present	ESC %
46	CTRL P2	Not present	ESC $
47	CTRL P3	Not present	ESC #
48	CTRL P4	Not present	ESC "
49	CTRL P5	Not present	ESC !

7. When accepting a nonnumeric screen data item defined with the JUSTIFIED clause, the data may be entered either left or right justified by the operator and is right-justified before being moved to the USING or TO identifier.

8. For more details about the specific rules related to the definition of each clause in the screen description entry, refer to its entry under SCREEN DESCRIPTION ENTRY.

9. If the format 2 of the ACCEPT statement is executed, the code of the last function key typed by the operator is moved into identifier-1.

10. If format 2 of the ACCEPT statement is executed before an ACCEPT statement of format 1 or if no function key has been typed by the operator, the value 00 is moved into identifier-1.

Function

The DISPLAY statement causes data to be transferred from the specified data items to the screen.

General Format

```
DISPLAY {screen-name-1} ... [ UPON { CONSOLE          } ]
                            [      { mnemonic-name-1 } ]
```

Syntax Rules

1. Mnemonic-name-1 must be defined in the SPECIAL-NAMES paragraph of the Environment Division.

2. Screen-name-1 must be defined in a screen description entry in the SCREEN SECTION.

3. If only one screen-name-1 is specified, it may be either a group item or an elementary item.

4. If more than one screen-name-1 is specified:

   a) Each must be defined in the same program with an 01 level number; or

   b) Each must be subordinate to the same level 01 entry, and no screen-name can be specified which is subordinate to any other specified screen-name.

5. The maximum number of screen items represented by screen-name-1 which may be present is 100.

General Rules

1. On execution of a DISPLAY statement, the following actions take place for each screen-name-1:

   a) If screen-name-1 is a group and has the BLANK SCREEN clause specified, blanking occurs (see The BLANK Clause).

   b) If screen-name-1 is a group and has a BELL clause specified, the work station audio tone is sounded (see The BELL Clause).

   c) If screen-name-1 is a group and has the BOX clause specified, a box is drawn around the group (see The BOX Clause).

   d) The cursor is positioned to the first elementary field.

   e) The work station audio tone is sounded, if the BELL clause is specified for the field (see The BELL Clause).

f) Blanking of all or a portion of the screen takes place, if BLANK SCREEN is specified for the field and the screen has not yet been blanked during the execution of the DISPLAY statement.

g) Blanking of the remaining characters of the line takes place, if the BLANK LINE clause is specified (see The BLANK Clause).

h) A box is drawn around the field if the BOX clause is specified (see The BOX Clause).

i) If the field is an input field, i.e. defined with a TO phrase and no FROM phrase in the PICTURE clause, a character string of as many spaces as the length specified for the item is displayed.

j) If the field is an output or input_output field, i.e. defined with a FROM or USING phrase in the PICTURE clause, the item is displayed according to the clauses present in its screen description entry.

k) If screen-name-1 is a group, the cursor is positioned to the next elementary field, if any. Rules e through k are repeated until the last elementary field in the group has been displayed.

2. For more details about the specific actions related to the definition of each clause, refer to its entry under SCREEN DESCRIPTION ENTRY.

Function

The EDIT statement for an input-output elementary screen data item is used to display, modify and then redisplay the associated data items specified in the FROM and TO phrases or the USING phrase. The EDIT statement for a group causes the group to be displayed and then allow each input field in the group (TO phrase only) to be accepted and each input-output field in the group to be edited and redisplayed.

General Format

Format 1

```
EDIT screen-name-1 [ FROM { CONSOLE          } ]
                          { mnemonic-name-1  }

     [ON ESCAPE imperative-statement-1 [END-EDIT]]
```

Format 2

```
EDIT screen-name-1 [ FROM { CONSOLE          } ]
                          { mnemonic-name-1  }

     TIME-OUT { integer-1    }
              { identifier-1 }

     ON EXCEPTION imperative-statement-2 [END-EDIT]
```

Syntax Rules

1. Mnemonic-name-1 must be defined in the SPECIAL-NAMES paragraph of the Environment Division.

2. Screen-name-1 must be defined in a screen description entry in the Screen Section.

3. If screen-name-1 is an elementary item, the PICTURE clause specified in the associated screen description entry must contain either the USING phrase or the FROM and TO phrases.

4. If screen-name-1 is a group item, at least one of the elementary items subordinate to it must be described according to rule 3.

5. Identifier-1 must reference an unsigned numeric integer data item.

6. Integer-1 or the value of the data item referenced by identifier-1 may not be greater than 3200.

General Rules

1. If screen-name-1 is an elementary item, the following actions take place:

   a) The cursor is positioned to the field that corresponds to the elementary data item for screen-name-1.

   b) The work station audio tone is sounded, if the BELL clause is specified for the field (see the BELL clause).

   c) The field is displayed in a special manner to make modification easier. See rule 3 for a description of the display.

   d) Characters introduced by the terminal operator are validated on the basis of the PICTURE character-string as they are entered. See rule 4 for a description of the rules for modifying data.

   e) If valid, any editing characters are stripped out of the modified data. The resulting data are moved to the appropriate item defined in the USING or TO clause following the Standard Rules of the MOVE statement. The MOVE does not take place if the field is terminated before any characters are entered.

   f) The new value of the field is presented reformatted. This includes: consideration of the presence of BLANK WHEN ZERO; elimination of the underscore characters considering the presence of floating characters and the fill characters 'Z' and '*' in the PICTURE clause; and formatting the value of an input-output field which has been cleared and input rather than modified. Note that this may not be the same as the value in the USING or TO field if, for example, truncation took place during the move.

2. If screen-name-1 is a group item, the following actions take place:

   a) The group is displayed exactly the same as in execution of the DISPLAY statement for the group (see the DISPLAY statement).

   b) The cursor is positioned to the field that corresponds to the first elementary data-item whose screen description entry contains either the USING or TO phrase in the PICTURE clause (see the PICTURE clause).

   c) If the field is described as input only (a PICTURE containing only a TO phrase) , the field is accepted normally as described in the ACCEPT statement.

   d) If the field is described as input-output, the field is again displayed, if necessary (see rule 3). After modification (see rule 4), the field is moved to the receiving field as described for elementary items in rule 1e. The new value of the field is

then redisplayed as described in rule 1f.

e) The cursor is positioned to the next input or input-output field.

f) Rules c through e are repeated until the last input or input-output field specified has been accepted or edited, respectively.

3. The rules for the display of an elementary screen item available for modification are:

a) If the screen item to be displayed has the value zero (numeric, numeric edited) or the value spaces (alphabetic, alphanumeric, alphanumeric edited) or if the field and its FROM or USING field are described as alphanumeric edited, the field is displayed as a string of underscore characters as in the execution of an ACCEPT statement.

b) Otherwise, the treatment depends on the data type as follows:

- Screen items defined as numeric are displayed normally, i.e. zero filled.

- Screen items defined as alphanumeric or alphabetic are displayed with appropriate justification. For a field defined as justified left, trailing blank are replaced by underscores. For a field defined as justified right or HEBREW, initial blanks are replaced by underscores.

- Screen items defined as alphanumeric edited are displayed in their edited form except that all trailing blanks not including simple insertion characters are displayed as underscores (see the PICTURE clause).

- Screen items defined as numeric edited are displayed in their edited form except that the floating insertion characters '+', '-', and currency sign are displayed only once in the first character position and any initial zero digit positions are displayed as underscores. In addition, if the field is defined with 'CR' or 'DB' and contains a positive value, the characters '++' are displayed. (See the PICTURE clause).

4. The rules for modifying an elementary screen item are as follows:

a) If the screen item to be modified has the value zero (numeric, numeric edited) or the value spaces (alphabetic, alphanumeric, alphanumeric edited) or if the field and its FROM or USING field are described as alphanumeric edited, the field is treated exactly as described for an ACCEPT statement (see the ACCEPT statement).

b) Initially, the cursor is positioned as follows:

- For numeric and numeric edited data not defined as FULL, at the first non insertion character position not containing zero.
- For numeric and numeric edited data defined as FULL, at the first non insertion character position.
- For alphanumeric and alphanumeric edited data defined as justified left, at the first non insertion character position.
- For alphanumeric and alphanumeric edited data defined as justified right, at the first non insertion character position not containing a space.
- For data defined as HEBREW, at the rightmost character position.

c) The operator uses the left arrow key and right arrow key to move a cross the field to the position where characters are to be changed.

d) The entering of an illegal character is indicated immediately.

e) The operator may change the floating or fixed insertion character '+' to '-' or '-' to '+'; the fixed insertion characters 'CR' or 'DB' to ' ' or '++'; or the characters '++' to 'CR' (if declared CR), 'DB' (if declared DB) or '--'. All other insertion characters (currency sign, 'B', '0', '/', '.', ',') may be not changed. The cursor automatically skips over these positions. (See the PICTURE clause).

f) The presence in the screen description entry of any of the clauses AUTO, REQUIRED and FULL is ignored.

g) The ENTER key terminates the modification of the field.

h) The SKIP key also terminates the modification of the field with the added function of clearing the field starting with the current position of the cursor. The significance of this depends upon the data type of the screen field: for numeric, the rest of the field is filled with zeros; for an item defined as HEBREW, the positions to the left of the cursor are filled with spaces; for other alphanumeric and alphabetic data, the positions to the right of the cursor are filled with blanks; for numeric edited, the rest of the noninsertion digit positions are filled with zeros; and for alphanumeric edited, the rest of the noninsertion character positions are filled with blanks.

i) If before the field is terminated the RESET key is struck, the entire field is replaced with underscore characters and its value becomes zero if numeric or spaces if alphanumeric. The field is then treated as an input field exactly as in the ACCEPT statement including the processing of AUTO, REQUIRED and FULL. (See the ACCEPT statement).

5. If an escape character (EXIT character) is typed at any time, the EDIT is terminated without modification of the current field. In addition, if the ON ESCAPE clause is specified, imperative-statement-1 is executed. The value 99 is stored for an eventual ACCEPT statement of format 2 (see the ACCEPT statement).

6. The use of format 2 places a time limit on each field modified. Integer-1 or the value of the data item referenced by identifier-1 is the number of seconds the user has to enter or edit the screen item starting from the moment when the cursor is positioned at the beginning of the field. If the indicated time has elapsed before the user has finished with the field, the EDIT is terminated; the move of the current field is not executed; and control is transferred to imperative-statement-2 in the ON EXCEPTION phrase. Execution continues according to the rules for each statement specified in imperative-statement-2.

7. If a function key is struck, the EDIT is terminated. If at least one character in the current field has been modified, the move is executed. The code of the function key typed is stored for an eventual ACCEPT statement of format 2. For the specification of the format rules and function key codes see the ACCEPT statement.

7. INTER-PROGRAM COMMUNICATION

Inter-Program Communication provides a facility by which a program can communicate with one or more programs. This communication is provided by: (a) the ability to transfer control from one program to another within a run unit and (b) the ability to pass parameters between programs to make certain data values available to a called program. Inter-Program Communication also permits communication between two programs by the sharing of data and the sharing of files.

LANGUAGE CONCEPTS

Complex data processing problems are frequently solved by developing a set of separately compilable but logically coordinated programs which at some time prior to execution may be compiled and assembled into a complete problem solution. The organization of COBOL programs and run units supports this approach of dividing large problem solutions into small, more manageable, portions which may be programmed and validated independently.

PROGRAM AND RUN UNIT ORGANIZATION

There are two levels of computer programs in a COBOL environment. These are the source level and the object level.

At the source level, the most inclusive unit of a computer program is a source program. A source program may contain other source programs. A source program is a syntactically correct set of COBOL statements as specified in this document and consists of an Identification Division followed optionally by an Environment Division and/or a Data Division and/or a Procedure Division. In general, a source program which is contained within another program cannot itself be converted by a compiler into an object program, since a contained source program is permitted to reference data in a containing source program.

At the object level the most inclusive unit of organization of computer programs is the run unit. A run unit is a complete problem solution consisting of an object program or of several inter-communicating object programs. A run unit is an independent entity that can be executed without communicating with, or being coordinated with, any other run unit except that it may process data files and messages or set and test switches that were written or will be read by other run units.

When a program is called, parameters upon which it is to operate may be passed to it by the program which calls it. As any separately compiled program may be the first program executed in a run unit, the first

executed program of a run unit may receive parameters.

A run unit may also contain object code and data storage areas derived from the compilation of programs written in languages other than COBOL: in this case the requirements for the relationship between the COBOL and the non-COBOL programs are defined in the COBOL Program Preparation and Execution manual or the equivalent manual for the other language.

GLOBAL NAMES AND LOCAL NAMES

A data-name names a data item. A file-name names a file control block. A screen-name names a screen item. A report-name names a report. All these names are classified as either global or local.

A global name may be used to refer to the object with which it is associated either within the program in which the global name is declared, or within any other program contained in the one program which declares the global name.

A local name, however, may only be used to refer to the object with which it is associated within the program in which the local name is declared.

A record-name is global if the GLOBAL clause is specified in the record description entry which declares the record-name or, for record description entries in the File Section, if the GLOBAL clause is specified in the file description entry for the file-name associated with the record description entry.

A data-name is global if the GLOBAL clause is specified in the data description entry which that data description entry is subordinate.

A condition-name declared in a data description entry is global if that entry is subordinate to another one in which the GLOBAL clause is specified.

A file-name is global if the GLOBAL clause is specified in the file description entry which declares the file-name.

A report-name is global if the GLOBAL clause is specified in the report description entry which declares the report-name.

A screen-name is global if the GLOBAL clause is specified in the screen data description entry which declares the screen-name, or in another entry to which that screen data description entry is subordinate.

If the GLOBAL clause is not specified in a description entry which contains data-name, file-name, condition-name, screen-name, or report-name, the name is local.

EXTERNAL OBJECTS AND INTERNAL OBJECTS

Accessible data items usually require that certain representations of data be stored. File control blocks usually require that certain information concerning files be stored. The storage associated with a data item or a file control block may be external or internal to the program in which the object is declared.

A data item or file control block is external if the storage associated with that object is associated with the run unit rather than with any particular program of run unit.

An external object may be referenced by any program in the run-unit which describes the object.

References to an external object from different programs using separate descriptions of the object always concern the same object.

An object is internal if the storage associated with that object is only associated with the program which describes the object.

External and internal objects may have global or local names.

A data record described in the Working-Storage Section is given the external attribute if the EXTERNAL clause is present in the corresponding description entry. Any data item subordinate to the description of an external record is also external.

If the EXTERNAL clause is not specified in a record description, a data description, or in a file control block, the object is internal only to the program describing the object.

Data records, subordinate data items, and various associated control information described in the Linkage, Report, and Screen Sections of a program are always considered to be internal to the program describing that data. Special considerations apply to data described in the Linkage Section whereby an association is made between the data records described and other data items accessible to other programs.

EXAMPLE

The following example shows the use of the GLOBAL and the EXTERNAL clauses.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EXTMAIN.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 PAR1      PIC 9(2) VALUE 1.
 01 PAR2      PIC 9(2).
 PROCEDURE DIVISION.
 INIT.
     CALL "PUT" USING PAR1.
     CALL "GET" USING PAR2.
     DISPLAY PAR2.
```

```
    EXIT PROGRAM.
    STOP RUN.
```

The EXTMAIN program contains the description of PAR1 and PAR2 data item, returned by the following sub-programs.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PUT.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SHARED  IS EXTERNAL IS GLOBAL  PIC 9(2).
 LINKAGE SECTION.
 01 PAR     PIC 9(2).
 PROCEDURE DIVISION USING PAR.
 INIT.
     CALL "SUBPUT" USING PAR.
     EXIT PROGRAM.


 IDENTIFICATION DIVISION.
 PROGRAM-ID. SUBPUT.
 DATA DIVISION.
 LINKAGE SECTION.
 01 SUBPAR  PIC 9(2).
 PROCEDURE DIVISION USING SUBPAR.
 INIT.
      MOVE SUBPAR TO SHARED.
      EXIT PROGRAM.
 END PROGRAM SUBPUT.

 END PROGRAM PUT.
```

The first sub-program, PUT, contains the description of the SHARED data, which is declared EXTERNAL so that the area associated to the data is shared by other sub-program; SHARED is also declared GLOBAL so that the SUBPUT program which is internal to PUT, can access it.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. GET.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 SHARED  IS EXTERNAL    PIC 9(2).
 LINKAGE SECTION.
 01 PAR        PIC 9(2).
 PROCEDURE DIVISION USING PAR.
 INIT.
     MOVE SHARED TO PAR.
     EXIT PROGRAM.
```

The second sub-program, GET, repeats the description of SHARED, which is the data shared with the proceeding program.

COMMON PROGRAMS AND INITIAL PROGRAMS

All programs which form part of a run unit may possess none, one, or more of the following attributes: common and initial.

A common program is one which, despite being directly contained within another program, may be called by any program directly or indirectly contained in that other program. The common attribute is attained by specifying the COMMON clause in a program's Identification Division. The COMMON clause facilitates the writing of subprograms which are to be used by all the programs contained within a program.

An initial program is one whose program state is initialized when the program is called. Thus, whenever an initial program is called, its program state is the same as when the program was first called in that run unit. During the process of initializing an initial program, that program's internal data is initialized; thus an item of the program's internal data whose description contains a VALUE clause is initialized to that defined value, but an item whose description does not contain a VALUE clause is initialized to an undefined value. The initial attribute is attained by specifying the INITIAL clause in the program's Identification Division.

SHARING DATA

Two programs in a run unit may reference common data in the following circumstances:

1. The data content of an external data record may be referenced from any program provided that program has described that data record.

2. If a program is contained within another program, both programs may refer to data possessing the global attribute either in the containing program or in any program which directly or indirectly contains the containing program.

3. The mechanism whereby a parameter value is passed from a calling program to a called program establishes a common data item; the called program, which may use a different identifier, may refer to a data item in the calling program.

SHARING FILES

Two programs in a run unit may reference common file control blocks in the following circumstances:

1. An external file control block may be referenced from any program which describes that file control block.

2. If a program is contained within another program, both programs may refer to a common file control block by referring to an associated global file-name either in the containing program or in any program which directly or indirectly contains the containing program.

SCOPE OF NAMES

When programs are directly or indirectly contained within other programs, each program may use identical user-defined words to name objects independent of the use of these user-defined words by other programs. (See User-Defined Words.) When identically named objects exist, a program's reference to such a name, even when it is a different type of user-defined word, is to the object which that program described rather than to the object, possessing the same name, described in another program.

The following types of user-defined words may be referenced only by statements and entries in that program in which the user-defined word is declared:

1. paragraph-name
2. section-name

The following types of user-defined words when they are declared in a Report Section may be referenced only by statements and entries in that program which contains that section:

1. condition-name
2. data-name
3. record-name

Since a Configuration Section cannot be specified for a program which is contained within another program, the following types of names may be referenced by statements and entries either in that program which contains a Configuration Section or in any program directly or indirectly contained within that program:

1. alphabet-name
2. condition-name
3. mnemonic-name

Specific conventions, for declarations and references, apply to the following types of user-defined words when the conditions listed above do not apply:

1. condition-name
2. data-name
3. file-name
4. index-name
5. program-name
6. record-name
7. report-name
8. screen-name

Conventions for Program-Names

The program-name of a program is declared in the PROGRAM-ID paragraph of the program's Identification Division. A program-name may be referenced only by the CALL and CANCEL statements and the end program header. The program-names allocated to programs constituting a run unit are not necessarily unique but, when two programs in a run unit are identically named, at least one of those two programs must be directly or indirectly contained within another separately compiled program which does not contain the other of those two programs.

The following rules regulate the scope of a program-name.

1. If the program-name is that of a program which does not possess the common attribute and which is directly contained within another program, that program-name may be referenced only by statements included in that containing program.

2. If the program-name is that of a program which does possess the common attribute and which is directly contained within another program, that program-name may be referenced only by statements included in that containing program and any programs, directly or indirectly contained within that containing program, except programs contained within the program possessing the common attribute.

3. If the program-name is that of a program which is separately compiled, that program-name may be referenced by statements included in any other program in the run unit, except itself and any programs it directly or indirectly contains.

Conventions for Condition-, Data-, File-, Record-, Report-, and Screen-names

When condition-names, data-names, file-names, record-names, report-names, and screen-names are declared in a source program, these names may be referenced only by that program except when one or more of the names is global and the program contains other programs.

The requirements governing the uniqueness of the names allocated by a single program to be condition-names, data-names, file-names, record-names, report-names, and screen-names are explained in chapter two. (See User-Defined Words.)

A program cannot reference any condition-name, data-name, file-name, record-name, report-name, or screen-name declared in any program it contains.

A global name may be referenced in the program in which it is declared or in any programs which are directly or indirectly contained within that program.

When a program, program B, is directly contained within another program, program A, both programs may define a condition-name, a data-name, a file-name, a record-name, a report-name, or a screen-name using the same user-defined word. When such a duplicated name is referenced in program

B, the following rules are used to determine the referenced object:

1. The set of names to be used for determination of a referenced object consists of all names which are defined in program B and all global names which are defined in program A and in any programs which directly or indirectly contain program A. Using this set of names, the normal rules for qualification and any other rules for uniqueness of reference are applied until one or more objects is identified.

2. If only one object is identified, it is the referenced object.

3. If more than one object is identified, no more than one of them can have a name local to program B. If zero or one of the objects has a name local to program B, the following rules apply:

   a) If the name is declared in program B, the objet in program B is the referenced object.

   b) Otherwise, if program A is contained within another program, the referenced object is:

      - The object in program A if the name is declared in program A.

      - The object in the containing program if the name is not declared in program A and is declared in the program containing program A. This rule is applied to further containing programs until a single valid name has been found.

Conventions for Index-Names

If a data item possessing the global attribute includes a table accessed with an index, that index also possesses the global attribute. Therefore, the scope of an index-name is identical to that of the data-name which names the table whose index is named by that index-name and the scope of name rules for data-names apply. If a data item possessing the external attribute includes a table accessed with an index, that index does not possess the external attribute. Index-names cannot be qualified.

LINK UNITS AND RUN UNITS

A link unit consists of one or more object programs linked together. Therefore, in the simplest case, a run unit consists of one link unit which, in turn, consists of a single object program. A more complicated run unit consists of more than one object program.

Using the linker, without changing the source code, these object programs can be linked into a run unit containing overlays. Object programs in the root portion, which is memory resident, or the object programs in the overlay, may be called using either version of the CALL statement (CALL identifier or CALL literal). These overlays are automatically unloaded when necessary.

A run unit may be composed of a root link unit and one or more dynamically loadable link units. The root link unit is always memory resident except for any overlays it may contain. The dynamically loadable link units may contain one or more object programs, but only one of these object programs may be defined as the entry-point of the link unit.

The dynamically loadable link unit may not themselves contain overlays. Execution begins with the root link unit and control is passed to the entry-point of a dynamically loadable link unit using the CALL data-name statement. From a dynamically loadable link unit:

- The object program in the root link unit may be called only with the CALL literal statement.
- Other object programs in the link unit may be called using either form of the CALL statement.
- Other dynamically loadable link units may be called using the CALL identifier statement, as long as there is no conflict in memory occupation between the link unit to be loaded and those already loaded.

Dynamically loadable link units may be unloaded using the CANCEL statement.

Data and files may be shared between link units using the EXTERNAL mechanism (see EXTERNAL clause). However, any data or file description entries which are declared with the EXTERNAL clause must be declared in a program contained in the root link unit, even if they are to be referenced exclusively in a dynamically loadable link unit.

SPECIAL REGISTER MESSAGE

The reserved word MESSAGE is a name for a register whose value is set to indicate the status of the execution of a CALL data-name statement. The implicit description is that of an array of two unsigned integer as follows:

```
01  FILLER
    02  MESSAGE PIC 99 COMP OCCURS 2 TIMES.
```

MESSAGE may be referenced only in Procedure Division statements; however, only the call/cancel control system may chance the value of MESSAGE.

In certain cases of abnormal termination of the CALL data-name statement, the ON OVERFLOW/EXCEPTION phrase, if present, is activated. The user may interrogate the value of MESSAGE to determine the cause of the termination. In other cases of abnormal termination, such as calling a program which has already been called and has not returned, mismatch in number of parameters, system error, or if the ON OVERFLOW/EXCEPTION phrase is not present, the application is always aborted.

The following is a list of the values placed in the MESSAGE register after the execution of a CALL identifier statement when control is returned to the calling program. Note that the value returned is given in the form x.y where 'x' is MESSAGE(1) and 'y' is MESSAGE(2).

1. MESSAGE = 0.0. The CALL identifier statement is successfully executed.

2. MESSAGE = 3.13. Name not found.

3. MESSAGE = 3.16. Memory conflict. Segment needed to load the requested program are not free.

There are some special additional values which are returned in the register MESSAGE when an attempt is made to call a transaction unit. For further information on these errors, see the MTS - MODULAR TRANSACTION SYSTEM - Programmer Guide

TRANSFER OF CONTROL

The CALL statement provides the means whereby control may be transferred from one program to another program within a run unit. A called program may itself contain CALL statements.

When control is transferred to a called program, execution proceeds from statement to statement beginning with the first nondeclarative statement. If control reaches a STOP RUN statement, this signals the logical end of the run unit. If control reaches an EXIT PROGRAM statement, this signals the logical end of the called program only, and control then reverts to the next executable statement following the CALL statement in the calling program. Thus the EXIT PROGRAM statement terminates only the execution of the program in which it occurs, while the STOP RUN statement terminates the execution of a run unit.

The CALL statement may be used to call a program which is not written in COBOL. A COBOL program may also be called from a program which is not written in COBOL. In both the above cases, only those parts of the parameter passing mechanism which apply to the COBOL programs are specified in this manual.

Scope of the CALL Statement

In the following, the calling program may or may not possess any of the program attributes, it may either be separately compiled or not, and it may either be contained within programs or contain other programs:

1. Any calling program may call any separately compiled program in the same link unit or in the root link unit, or any separately compiled program which is the entry-point of a dynamically loadable link unit.

2. A calling program may call any program which is directly contained within the calling program.

3. Any calling program may call any program possessing the common attribute which is directly contained within a program which itself directly or indirectly contains the calling program, unless the calling program is itself contained within the program possessing the common attribute.

4. A calling program may call a program which neither possesses the common attribute nor is separately compiled, if and only if, that program is directly contained within the calling program. If the calling program specifies the called program using an identification, the called program must be separately compiled.

PASSING PARAMETERS TO PROGRAMS

Data passed as a parameter by a program calling another program must be accessible to the calling program and the data receiving the data must be declared in the Data Division of the called program. The data values passed as parameters also may identify some data to be shared; hence the two methods are not mutually independent.

Identifying Parameters

In the called program the parameters required are identified by listing references to the names assigned, in that program's data description entries, to the parameter in that program's Procedure Division header. In the calling program the values of the parameters to be passed by the calling program are identified by listing references in the CALL statement used to call the called program. These lists establish, on a positional basis at object time, the correspondence between the values as they are known to each program; that is, the first parameter on one list corresponds to the first identifier on the other, the second to the second, etc. Thus a program, which may be called by another program, may include:

```
PROGRAM-ID.  EXAMPLE.

PROCEDURE DIVISION USING NUM, PCODE, COST.
```

and may be called by executing:

```
CALL EXAMPLE USING NBR, PTYPE, PRICE.
```

thereby establishing the following correspondence:

Called program (EXAMPLE)	Calling program
NUM	NBR
PCODE	PTYPE
COST	PRICE

Only the positions of the data-names are significant, not the names themselves.

Values of Parameters

The calling program controls the methods by which a called program evaluates the values of the parameters passed to it and by which the called program returns results as modified parameter values.

The individual parameters referenced in the CALL statement's USING phrase are passed by reference. A called program is allowed to access and modify the value of the data item referenced in the calling program's CALL statement as a parameter passed by reference.

The parameters referenced in a called program's Procedure Division header must be described in the Linkage Section of that program's Data Division.

NESTED SOURCE PROGRAMS

A COBOL source program is a syntactically correct set of COBOL statements. A COBOL source program may contain other COBOL source programs and these contained programs may reference some of the resources of the program within which they are contained.

When a program, program B, is contained in another program, program A, it may be directly contained or indirectly contained. Program B is directly contained in program A if there is no program contained in program A that also contains program B. Program B is indirectly contained in program A if there exists a program contained in program A that also contains program B.

ORGANIZATION

With the exception of COPY statements and the end program header, the statements, entries, paragraphs, and sections of a COBOL source program are grouped into four divisions which are sequenced in the following order:

The Identification Division
The Environment Division
The Data Division
The Procedure Division

The end of a COBOL source program is indicated by either the end program header, if specified, or by the absence of additional source program lines.

STRUCTURE

The following gives the general format and order of presentation of the entries and statements which constitute a COBOL source program. The generic terms identification-division, environment-division, data-division, procedure-division, source-program, and end-program-header represent a COBOL Identification Division, a COBOL Environment Division, a COBOL Data Division, a COBOL Procedure Division, a COBOL source program, and a COBOL end program header, respectively.

General Format

identification-division

[environment-division]

[data-division]

procedure-division

[source-program] ...

[end-program-header]

Syntax Rules

1. End-program-header must be present if:

   a) The COBOL source program contains one or more other COBOL source programs; or

   b) The COBOL source program is contained within another COBOL source program.

General Rules

1. The beginning of a division in a program is indicated by the appropriate division header. The end of a division is indicated by one of the following:

   a) The division header of a succeeding division in that program.

   b) An Identification Division header which indicates the start of another source program.

   c) The end program header.

   d) If the division is the Procedure Division, that physical position after which no more source program lines occur.

2. A COBOL source program which is directly or indirectly contained within another program is considered in these specifications as a separate program that may additionally reference certain resources defined in the containing program.

3. The object code, resulting from compiling a source program contained within another program, is considered in these specifications to be inseparable from the object code resulting from compiling the containing program.

INITIAL STATE OF A PROGRAM

The initial state of a program is the state of a program the first time it is called in a run unit.

Characteristics of a Program

1. The program's internal data contained in the Working-Storage Section are initialized. If a VALUE clause is used in the description of the data item, the data item is initialized to the defined value. If a VALUE clause is not associated with a data item, the initial value of the data item is undefined.

2. Files with internal file control blocks associated with the program are not in the open mode.

Programs in the Initial State

A program is in the initial state:

1. The first time the program is called in a run unit.

2. The first time the program is called after the execution of a CANCEL statement referencing the program or a CANCEL statement referencing a program that directly or indirectly contains the program.

3. Every time the program is called, if it possesses the initial attribute.

4. The first time the program is called after the execution of a CALL statement referencing a program that possesses the initial attribute, and that directly or indirectly contains the program.

END PROGRAM HEADER

Function

The end program header indicates the end of the named COBOL source program.

General Format

END PROGRAM program-name.

Syntax Rules

1. The program-name must conform to the rules for forming a user-defined word.

2. The program-name must be identical to a program-name declared in a preceding PROGRAM-ID paragraph. (See The PROGRAM-ID Paragraph.)

3. If a PROGRAM-ID paragraph declaring a specific program-name is stated between the PROGRAM-ID paragraph and the end program header declaring and referencing, respectively, another program-name, the end program header referencing the former program-name must precede that referencing the latter program-name.

General Rules

1. The end program header must be present in every program which either contains or is contained within another program.

2. The end program header indicates the end of the specified COBOL source program.

3. If the program terminated by the end program header is contained within another program, the next statement must either be an Identification Division header or another end program header which terminates the containing program.

4. If the program terminated by the end program header is not contained within another program and if the next source statement is a COBOL statement, it must be the Identification Division header of a program to be compiled separately from that program terminated by the end program header.

THE PROGRAM-ID PARAGRAPH AND NESTED SOURCE PROGRAMS

Function

The PROGRAM-ID paragraph specifies the name by which a program is identified and assigns selected program attributes to that program.

General Format

```
PROGRAM-ID.  program-name  [ IS { COMMON  } ]  PROGRAM  .
                               { INITIAL }
```

Syntax Rules

1. The program-name must conform to the rules for formation of a user-defined word.

2. A program contained within another program must not be assigned the same name as that of any other program contained within the separately compiled program which contains this program.

3. The optional COMMON clause may be used only if the program is contained within another program.

General Rules

1. The program-name identifies the source program, the object program, and all listings pertaining to a particular program.

2. The COMMON clause specifies that the program is common. A common program is contained within another program but may be called from programs other than that containing it. (See Scope of Names.)

3. The INITIAL clause specifies that the program is initial. When an initial program is called, it and any programs contained within it are placed in their initial state (see Initial State of a Program) and the internal data, excluding Linkage Section data, in each program is initialized.

LINKAGE SECTION

The Linkage Section is located in the Data Division of a source program. The Linkage Section appears in the called program and describes data items that are to be referred to by the calling program and the called program.

The Linkage Section in a program is meaningful if and only if the object program is to function under the control of a CALL statement, and the CALL statement in the calling program contains a USING phrase.

The Linkage Section is used for describing data that is available through the calling program but is to be referred to in both the calling and the called program. The mechanism by which a correspondence is established between the data items described in the Linkage Section of a called program and data items described in the calling program is described elsewhere in these specifications. (See The Procedure Division Header, and The CALL Statement.) In the case of index-names, no such correspondence is established and index-names in the called and calling programs always refer to separate indexes.

The structure of the Linkage Section is the same as that previously described for the Working-Storage Section, beginning with a section header, followed by noncontiguous data items and/or record description entries.

The general format of the Linkage Section is shown below.

```
LINKAGE SECTION.

[77-level-description-entry  ]
[record-description-entry    ] ...
```

If a data item in the Linkage Section is accessed in a program which is not a called program, the effect is undefined.

Noncontiguous Linkage Storage

Items in the Linkage Section that bear no hierarchical relationship to one another need not be grouped into records and are classified and defined as noncontiguous elementary items. Each of these items is defined in a separate data description entry which begins with the special level-number 77.

The following data clauses are required in each data description entry:

a. level-number 77
b. data-name

c. the PICTURE clause or the USAGE IS INDEX clause

Other data description clauses are optional and can be used to complete the description of the item if necessary.

Linkage Records

Data elements in the Linkage Section which bear a definite hierarchical relationship to one another must be grouped into records according to the rules for formation of record descriptions. Data elements in the Linkage Section which bear no hierarchical relationship to any other data item may be described as records which are single elementary items.

Initial Values

The VALUE clause must not be specified in the Linkage Section except in condition-name entries (level 88).

THE FILE DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

Function

Within Inter-Program Communication, the file description entry determines the internal or external attributes of a file control block, of the associated data records, and of the associated data items. The file description entry also determines whether a file-name is a local name or a global name.

General Format

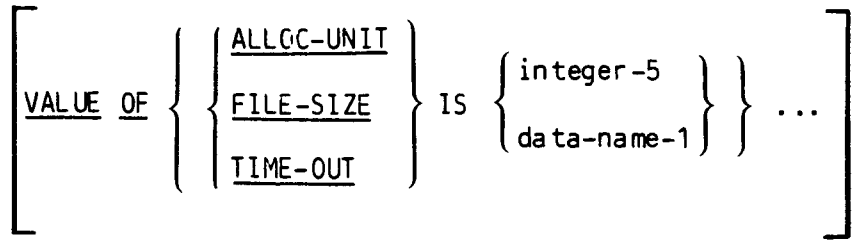
Format 1: - Sequential Organization

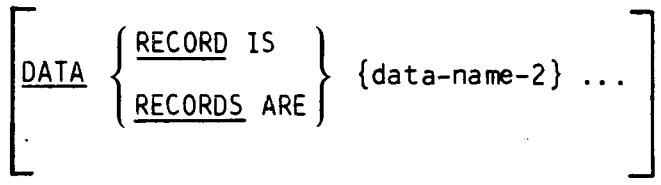
```
FD file-name-1

    [IS EXTERNAL]

    [IS GLOBAL]

    [BLOCK CONTAINS [integer-1 TO] integer-2 {RECORDS    }]
                                             {CHARACTERS }

    [RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

    [LABEL {RECORD IS  } {STANDARD}]
           {RECORDS ARE} {OMITTED }

    [VALUE OF {{ALLOC-UNIT} IS {integer-5 }} ...]
              {{FILE-SIZE }    {data-name-1}}

    [DATA {RECORD IS  } {data-name-2} ...]
          {RECORDS ARE}
```

```
[LINAGE IS {data-name-3 } LINES
           {integer-6   }

          [WITH FOOTING AT {data-name-4}]
                           {integer-7  }

     [LINES AT TOP  {data-name-5}]
                    {integer-8  }

               [LINES AT BOTTOM {data-name-6}]
                                {integer-9  }]]

[CODE-SET IS alphabet-name-1].
```

Format 2: - Relative Organization

```
FD file-name-1

    [IS EXTERNAL]

    [IS GLOBAL]

    [BLOCK CONTAINS [integer-1 TO] integer-2 {RECORDS   }]
                                             {CHARACTERS}

    [RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

    [LABEL {RECORD IS  } {STANDARD}]
           {RECORDS ARE} {OMITTED }]
```



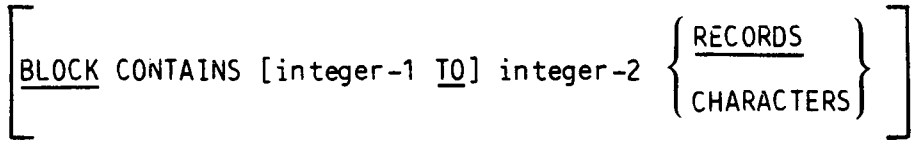


Format 3: - Indexed Organization

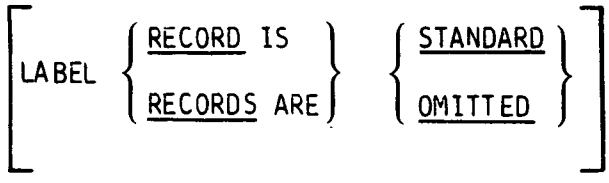
FD file-name-1

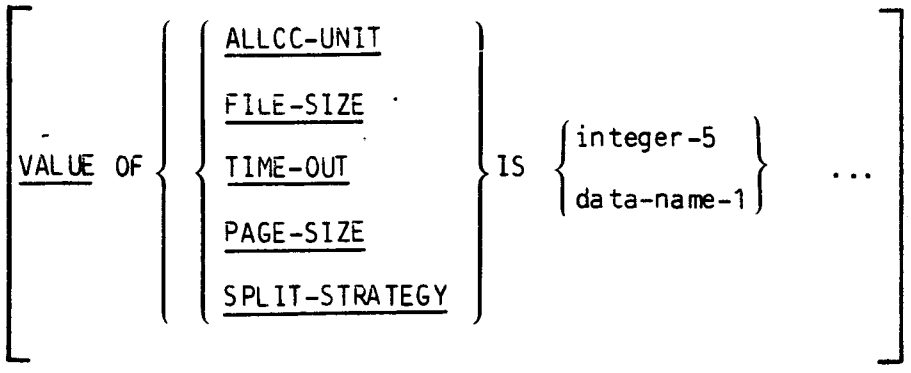
[IS EXTERNAL]

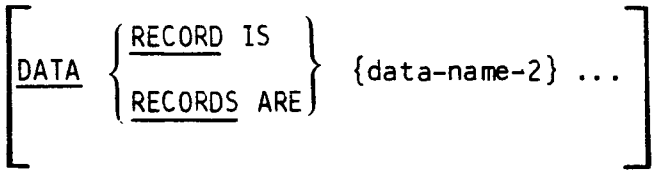
[IS GLOBAL]



[RECORD CONTAINS [integer-3] TO integer-4 CHARACTERS]





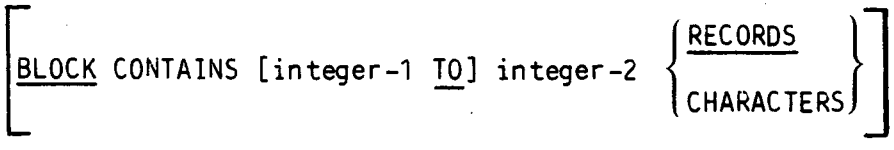


Format 4: - Sequential Organization Report File

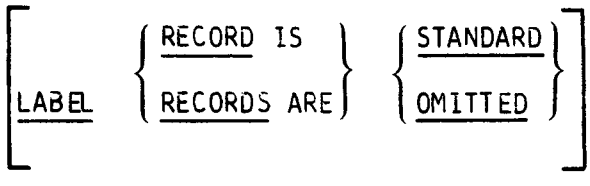
FD file-name-1

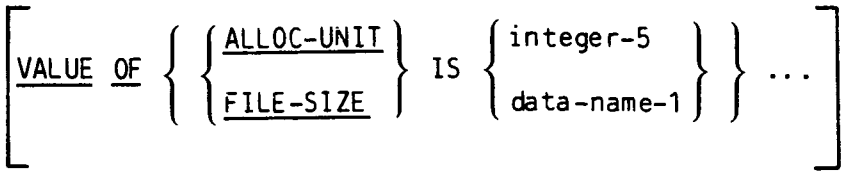
[IS EXTERNAL]

[IS GLOBAL]



[RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]





[CODE-SET IS alphabet-name-1]

Syntax Rules

1. Format 1 is the file description entry for a file with sequential organization.

2. Format 2 is the file description entry for a file with relative organization.

3. Format 3 is the file description entry for a file with indexed organization.

4. Format 4 is the file description entry for a report file.

General Rules

1. The presence of the EXTERNAL clause in a file description entry causes the file control block, the associated data records, and all subordinate data items to have the external attribute.

2. The absence of the EXTERNAL clause from a file description entry causes the file control block, the associated data records, and all subordinate data items to have the internal attribute.

3. The presence of the GLOBAL clause in a file description entry causes the file-name, the associated record-names, all subordinate data-names, all subordinate condition-names, and all subordinate index-names to be global names.

4. The absence of the GLOBAL clause from a file description entry causes the file-name to be a local name. The associated record-names, subordinate data-names, subordinate condition-names, and subordinate index-names are local names unless the GLOBAL clause is specified in the level 01 data description entry subordinate to the file description entry.

5. If the file description entry for a sequential file contains the LINAGE clause and the EXTERNAL clause, the LINAGE-COUNTER data item is an external data item. If the file description entry for a sequential file contains the LINAGE clause and the GLOBAL clause, the special register LINAGE-COUNTER is a global name.

6. The EXTERNAL clause and the GLOBAL clause are presented in this chapter. All other clauses in the file description entry are presented in the IDENTIFICATION AND ENVIRONMENT DIVISION chapter.

THE DATA DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

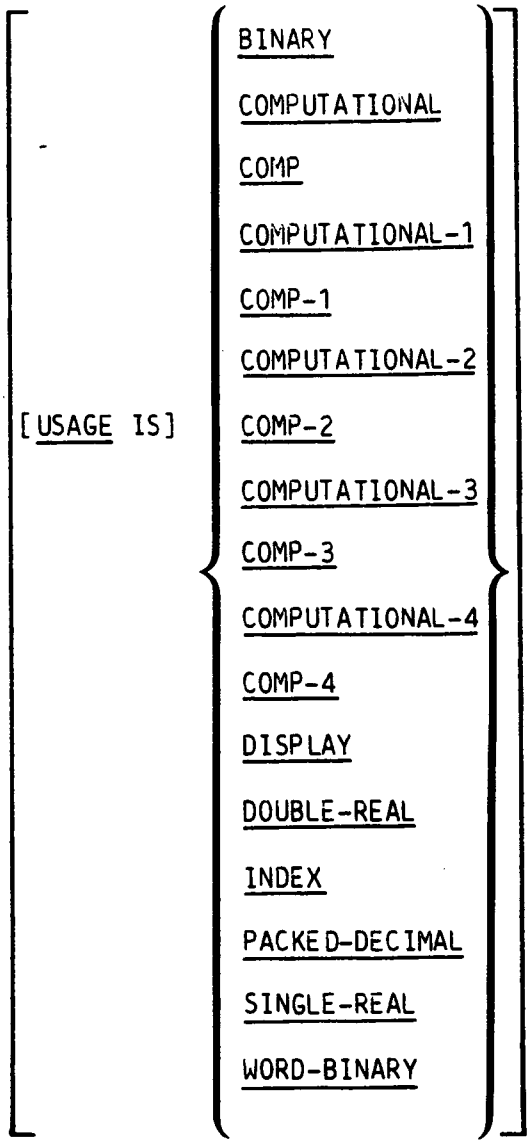
Function

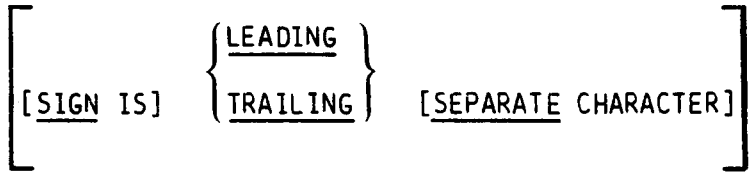
Within Inter-Program Communication, a level 01 data description entry within the Working-Storage or File Section determines whether the data record and its subordinate data items have local names or global names.

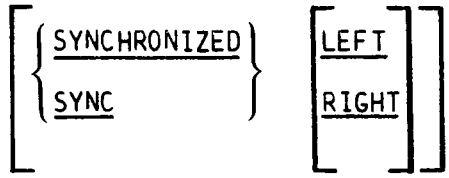
Within Inter-Program Communication, a level 01 data description entry within the Working-Storage Section determines the internal or external attribute of the data record and its subordinate data items.

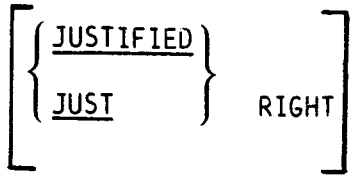
General Format

```
01  [ data-name-1 ]
    [ FILLER      ]

    [REDEFINES data-name-2]

    [IS EXTERNAL]

    [IS GLOBAL]

    [ { PICTURE } IS character-string ]
    [ { PIC     }                     ]
```


[USAGE IS]
BINARY
COMPUTATIONAL
COMP
COMPUTATIONAL-1
COMP-1
COMPUTATIONAL-2
COMP-2
COMPUTATIONAL-3
COMP-3
COMPUTATIONAL-4
COMP-4
DISPLAY
DOUBLE-REAL
INDEX
PACKED-DECIMAL
SINGLE-REAL
WORD-BINARY


[SIGN IS]
LEADING
TRAILING
[SEPARATE CHARACTER]


SYNCHRONIZED
SYNC
LEFT
RIGHT



[BLANK WHEN ZERO]

[VALUE IS literal-1].

Syntax Rules

1. The EXTERNAL clause may be specified only in data description entries in the Working-Storage Section whose level-number is 01.

2. The EXTERNAL clause and the REDEFINES clause must not be specified in the same data description entry.

3. The GLOBAL clause may be specified only in data description entries in the Working-Storage or File Section whose level-number is 01.

4. Data-name-1 must be specified for any entry containing the GLOBAL or EXTERNAL clause, or for record descriptions associated with a file description entry which contains the EXTERNAL or GLOBAL clause.

General Rules

1. The presence of the EXTERNAL clause in a data description entry causes the data record and all subordinate data items to have the external attribute.

2. The absence of the EXTERNAL clause from a data description entry causes the data record and all subordinate data items to have the internal attribute.

3. The presence of the GLOBAL clause in a data description entry causes the data-name, all subordinate data-names, all subordinate condition-names, and all subordinate index-names to be global names.

4. The absence of the GLOBAL clause from a data description entry causes the data-name, all subordinate data-names, all subordinate condition-names, and all subordinate index-names to be local names.

5. The EXTERNAL and the GLOBAL clause are described in this chapter. All other clauses in the data description entry are presented in the DATA DIVISION chapter.

THE SCREEN DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

Function

Within Inter-Program Communication, a level 01 screen description entry within the Screen Section determines whether the screen record and its subordinate screen items have local names or global names.

General Format

Format 1 (Group Entry)

```
01  [screen-name-1]
    [FILLER       ]

    [IS GLOBAL]

    [BLANK SCREEN]

    [BELL]

    [BLINK]

    [BOX [integer-1]]

    [HIGHLIGHT]

    [REVERSE]

    [UNDERLINE]

    [LINE NUMBER IS [PLUS] integer-2]

    [AUTO]

    [SECURE]
```

[REQUIRED]

[FULL]

[HEBREW].

Format 2 (Literal Screen Element Entry)

```
01  [ screen-name-1 ]
    [ FILLER        ]
```

[IS GLOBAL]

```
[ BLANK { SCREEN } ]
        { LINE   }
```

[BLANK WHEN ZERO]

[BELL]

[BLINK]

[BOX [integer-1]]

[HIGHLIGHT]

[REVERSE]

[UNDERLINE]

[LINE NUMBER IS [PLUS] integer-2]

[COLUMN NUMBER IS [PLUS] integer-3]

{PICTURE / PIC} IS character-string

VALUE IS literal-1.

Format 3 (Data-Item Screen Element Entry)

01 [screen-name-1 / FILLER]

[IS GLOBAL]

[BLANK {SCREEN / LINE}]

[BELL]

[BLINK]

[BOX [integer-1]]

[HIGHLIGHT]

[REVERSE]

[UNDERLINE]

[LINE NUMBER IS [PLUS] integer-2]

[COLUMN NUMBER IS [PLUS] integer-3]

```
{PICTURE}
{PIC    } IS character-string

                {USING identifier-1                                  }
                {     {     {FROM {identifier-2}     }     }       }
                {     {     {     {literal-1   }     }     }       }
                {     {     {TO identifier-3         }     }       }

[BLANK WHEN ZERO]

[{JUSTIFIED} RIGHT]
 {JUST     }

[AUTO]

[SECURE]

[REQUIRED]

[FULL]

[HEBREW].
```

Syntax Rules

1. The GLOBAL clause may be specified only in data description entries in the Screen Section whose level-number is 01.

2. Data-name-1 must be specified for any entry containing the GLOBAL clause.

General Rules

1. The presence of the GLOBAL clause in a screen data description entry causes the screen-name and all subordinate screen-names to be global names.

2. The absence of the GLOBAL clause from a data description entry causes the screen-name to be a local name. The subordinate screen-names are local names.

3. The GLOBAL clause is presented in this chapter. All other clauses in the screen data description entry are presented in the Screen Handling chapter.

THE REPORT DESCRIPTION ENTRY IN INTER-PROGRAM COMMUNICATION

Function

Within Inter-Program Communication, the report description entry determines whether a report-name is a local name or a global name.

General Format

```
RD report-name-1

  [IS GLOBAL]

  [CODE literal-1]

  [ { CONTROL IS  } { {data-name-1}          } ]
  [ { CONTROLS ARE} { FINAL [data-name-1]... } ]

  [ PAGE [ LIMIT IS  ] integer-1 [ LINE  ] [HEADING integer-2] ]
         [ LIMITS ARE]           [ LINES ]

       [FIRST DETAIL integer-3] [LAST DETAIL integer-4]

       [FOOTING integer-5] ] .
```

General Rules

1. The presence of the GLOBAL clause in a report description entry causes the report-name to be a global name.

2. The absence of the GLOBAL clause from a report description entry causes the report-name to be a local name.

3. If the report description entry contains the GLOBAL clause, the special registers LINE-COUNTER and PAGE-COUNTER are global names.

4. The GLOBAL clause is presented in this chapter. All other clauses in the report description entry are presented in the Report Writer chapter.

Function

The EXTERNAL clause specifies that a data item or a file control block is external. The constituent data items and group data items of an external data record are available to every program in the run unit which describes that record.

General Format

IS EXTERNAL

Syntax Rules

1. The EXTERNAL clause may be specified only in file description entries or in record description entries in the Working-Storage Section.

2. In the same program, the data-name specified as the subject of the entry whose level-number is 01 that includes the EXTERNAL clause must not be the same data-name specified for any other data description entry which includes the EXTERNAL clause.

3. The VALUE clause must not be used in any data description entry which includes, or is subordinate to, an entry which includes the EXTERNAL clause. The VALUE clause may be specified for condition-name entries associated with such data description entries.

General Rules

1. The data contained in the record named by the data-name clause is external and may be accessed and processed by any program in the run unit which describes and, optionally, redefines it subject to the following general rules.

2. Within a run unit, if two or more programs describe the same external data record, the associated data description entries, including all subordinate data-names and data items and their redefinitions, must be identical except that GLOBAL clauses need not occur identically in the description. However, a program which describes an external record may contain a data description entry including the REDEFINES clause which redefines the complete external record, and this complete redefinition need not occur identically in other programs in the run unit. (See The REDEFINES Clause.)

3. Use of the EXTERNAL clause does not imply that the associated file-name or data-name is a global name. (See The GLOBAL Clause.)

4. The file control block associated with this description entry is an external file control block.

GLOBAL clause

Function

The GLOBAL clause specifies that a data-name, a file-name, or a screen-name is a global name. A global name is available to every program contained within the program which declares it.

General Format

IS GLOBAL

Syntax Rules

1. The GLOBAL clause may be specified only in data description entries whose level-number is 01 in the File Section or the Working-Storage Section, file description entries, report description entries, or screen data description entries whose level number is 01.

2. In the same Data Division, the data description entries for any two data items for which the same data-name is specified must not include the GLOBAL clause.

3. If the SAME RECORD AREA clause is specified for several files, the record description entries or the file description entries for these files must not include the GLOBAL clause.

General Rules

1. A data-name, file-name, screen-name, or report-name described using a GLOBAL clause is a global name. All data-names subordinate to a global name are global names. All condition-names associated with a global name are global names.

2. A statement in a program contained directly or indirectly within a program which describes a global name may reference that name without describing it again. (See Scope of Names.)

3. If the GLOBAL clause is used in a data description entry which contains either the REDEFINES clause or the RENAMES clause, it is only the subject of that REDEFINES or RENAMES clause, as appropriate, which possesses the global attribute.

INTER-PROGRAM COMMUNICATION IN THE PROCEDURE DIVISION

THE PROCEDURE DIVISION HEADER

The Procedure Division is identified by, and must begin with, the following header:

PROCEDURE DIVISION [USING {data-name-1} ...].

The USING phrase is necessary only if the object program is to be invoked by a CALL statement and that statement includes a USING phrase.

The USING phrase of the Procedure Division header identifies the names used by the program for any parameters passed to it by a calling program. The parameters passed to a called program are identified in the USING phrase of the calling program's CALL statement. The correspondence between the two lists of names is established on a positional basis.

Syntax Rules

1. Data-name-1 must be defined as a level 01 entry or a level 77 entry in the Linkage Section.

2. A particular user-defined word may not appear more than once as data-name-1.

3. The record description entry for data-name-1 must not contain a REDEFINES clause. Data-name-1 may, however, be the object of a REDEFINES clause elsewhere in the Linkage Section.

The following additional rules apply:

1. The object program operates as if the data item in the called program occupies the same storage area as the corresponding item in the calling program. The description of the data item in the called program must describe the same number of character positions as described by the description of the corresponding data item in the calling program.

2. At all times in the called program, references to data-name-1 are resolved in accordance with the description of the item given in the Linkage Section of the called program.

3. Data items defined in the Linkage Section of the called program may be referenced within the Procedure Division of that program if, and only if, they satisfy one of the following conditions:

   a) They are operands of the USING phrase of the Procedure Division header.

b) They are subordinate to operands of the USING phrase of the Procedure Division header.

c) They are defined with a REDEFINES or RENAMES clause, the object of which satisfies the condition in point a above.

d) They are items subordinate to any item which satisfies the condition in point c above.

e) They are condition-names or index-names associated with data items that satisfy any of the above four conditions.

Function

The CALL statement causes control to be transferred from one object program to another, within the run unit.

General Format

```
CALL { identifier-1 } [USING {identifier-2}...]
     { literal-1    }

    [ON { OVERFLOW  } imperative-statement-1 [END-CALL]]
        { EXCEPTION }
```

Syntax Rules

1. Literal-1 must be a nonnumeric literal other than a figurative constant which contains the program-name. Therefore, the maximum length allowed is thirty characters.

2. Identifier-1 must be defined as an alphanumeric data item.

3. Each of the operands in the USING phrase must have been defined as a data item in the File Section, Working-Storage Section, or Linkage Section. A binary operand must be defined with the SYNCRONIZED (RIGHT) clause. Otherwise, identifier-2 must be word-aligned either naturally or by being defined with the SYNCHRONIZED (LEFT) clause. Note that if the operand is a group item, the SYNC clause may not be specified explicitly. Therefore, the record must be defined carefully in order for the group to be aligned naturally.

4. The ON OVERFLOW or ON EXCEPTION phrase is allowed only if identifier-1 is specified.

General Rules

1. Literal-1 or the content of the data item referenced by identifier-1 identifies the called program or link unit. The program in which the CALL statement appears is the calling program. If the program or link unit being called is written in COBOL, the rules for the specification of literal-1 or the content of the data-name referenced by identifier-1 are as described in rules 2 through 4. If the program being called is not a COBOL program, the rules for formation of the program-name are defined by the language the program is written in, but must obey the MOS file system file-name constraints.

2. If a program is being called which is not compiled separately:

   a) Literal-1 must be specified.

   b) Literal-1 may not contain lower-case letters.

   c) Literal-1 must contain the program-name contained in the PROGRAM-ID paragraph of the called program.

3. If a separately compiled program is being called:

   a) Literal-1 or the content of the data item referenced by identifier-1 must contain the program-name contained in the PROGRAM-ID paragraph of the called program.

   b) If literal-1 is specified, upper and lower-case letters are treated equivalently.

   c) If identifier-1 is specified, the content of the data item referenced by identifier-1 may not contain lower-case letters.

4. If a link unit is being called:

   a) Identifier-1 must be specified.

   b) The content of the data item referenced by identifier-1 specifies the path name of the link unit file on the physical device. If the content of the name contains a space character, only the characters up until the first blank are considered to be part of the name. If the physical link unit name starts with the character '/', then it is assumed to be a complete path name. Otherwise, the content of the data item referenced by identifier-1 is assumed to be a partial path name. In either case, the allowed maximum length until the first blank is fifty-nine characters.

   c) When calling a link unit which has already been called and has not yet been cancelled, the content of the data item referenced by identifier-1 must be identical to that of the first call. For example, the link unit may not be called using a complete path name and successively called using a relative path name unless a CANCEL statement is first executed.

5. If, when a CALL statement is executed for a program, the program specified by the CALL statement is made available for execution, control is transferred to the called program.

   If, when a CALL statement is executed for a link unit, the link unit specified by the CALL statement is made available for execution, the link unit is loaded, if necessary, and control is transferred to the program that is the entry-point of a link unit. Therefore, this entry-point program is the called program.

   After control is returned from the called program, the ON OVERFLOW/EXCEPTION phrase, if specified, is ignored and control is transferred to the end of the CALL statement.

6. If it is determined, when a CALL statement specifying identifier-1 is executed, that the program or link unit specified by the CALL statement is not present in the run unit or conflicts with other object programs or link units already loaded and the ON OVERFLOW/EXCEPTION phrase is specified, control is transferred to imperative-statement-1. The special register MESSAGE contains information concerning the type of exception that occurred. (See Special Register MESSAGE).

7. If it is determined, when a CALL statement specifying identifier-1 is executed, that the program or link unit specified by the CALL statement is not present in the run unit or that it conflicts with other object programs or link units already loaded and the ON OVERFLOW/EXCEPTION phrase is not specified, execution of the run unit is terminated.

8. Two or more programs in the run unit may have the same program-name, and the reference in a CALL statement with literal-1 to such a program-name is resolved by using the scope of names conventions for program-names. (See Conventions for Program-Names.)

   For example, when only two programs in the run unit have the same name as that specified in a CALL statement:

   a) One of those two programs must also be contained directly or indirectly either within the separately compiled program which includes that CALL statement or within the separately compiled program which itself directly or indirectly contains the program which includes that CALL statement, and

   b) The other of those two programs must be a different separately compiled program.

   The mechanism used in this example is as follows:

   a) If one of the two programs having the same name as that specified in the CALL statement with literal-1 is directly contained within the program which includes that CALL statement, that program is called.

   b) If one of the two programs having the same name as that specified in the CALL statement with literal-1 possesses the common attribute and is directly contained within another program which directly or indirectly contains the program which includes the CALL statement, that common program is called unless the calling program is contained within that common program.

   c) Otherwise, the separately compiled program is called.

9. If identifier-1 is specified in a CALL statement, reference is always to a separately compiled program or a separate link unit. If a separately compiled program within the link unit has a program-name which is the same as the file name of a link unit in the program directory of the root link unit, and the data item referenced by identifier-1 contains this name, the separately compiled program within the link unit is called.

10. If the called program does not possess the initial attribute, it, and each program directly or indirectly contained within it, is in its initial state the first time it is called within a run unit and the first time it is called after a CANCEL to the called program.

    On all other entries into the called program, the state of the program and each program directly or indirectly contained within it remains unchanged from its state when last exited.

11. If the called program possesses the initial attribute, it and each program directly or indirectly contained within it, is placed into its initial state every time the called program is called within a run unit.

12. Files associated with a called program's internal file control blocks are not in the open mode when the program is in an initial state. (See Initial State of a Program.)

    On all other entries into the called program, the status and positioning of all such files is the same as when the called program was last exited.

13. The process of calling a program or exiting from a called program does not alter the status or positioning of a file associated with any external file control block.

14. If the program being called is a COBOL program, the USING phrase is included in the CALL statement only if there is a USING phrase in the Procedure Division header of the called program, in which case the number of operands in each USING phrase must be identical. If the program being called is other than a COBOL program, the use of the USING phrase and the mapping of parameters is the responsibility of the user.

15. The sequence of appearance of data-names in the USING phrase of the CALL statement and in the corresponding USING phrase in the called program's Procedure Division header determines the correspondence between the data-names used by the calling and called programs. This correspondence is positional and may not be name equivalence; the first data-name in one USING phrase corresponds to the first data-name in the other, the second to the second, etc.

16. The values of the parameters referenced in the USING phrase of the CALL statement are made available to the called program at the time the CALL statement is executed.

17. The object program operates as if the corresponding data item in the called program occupies the same storage area as the data item in the calling program. The description of the data item in the called program must describe the same number of character positions as described by the description of the corresponding data item in the calling program. In addition, if the data item in the called program is numeric, its data description must be the same as the data description of the corresponding data item in the calling program.

18. Called programs may contain CALL statements. However, a called program must not execute a CALL statement that directly or indirectly calls the calling program. If a CALL statement is executed within the range of a declarative, that CALL statement cannot directly or indirectly reference any called program to which control has been transferred and which has not completed execution.

19. The END-CALL phrase delimits the scope of the CALL statement. (See Scope of Statements.)

CANCEL

Function

The CANCEL statement ensures that the next time the referenced program is called it will be in its initial state.

General Format

```
CANCEL { identifier-1 } ...
       { literal-1    }
```

Syntax Rules

1. Literal-1 must be a nonnumeric literal other than a figurative constant which contains the program-name or the path name of the link unit to be cancelled.

2. Identifier-1 must reference an alphanumeric data item such that its contents can be as described above.

3. Literal-1 or the content of identifier-1 may not contain lower case letters.

General Rules

1. Literal-1 or the content of the data item referenced by identifier-1 identifies the program or link unit to be cancelled. The rules for the specification of literal-1 or the content of the data-name referenced by identifier-1 are as described in rules 2 through 3.

2. If a separately compiled program is being cancelled:

   a) Literal-1 or the content of the data item referenced by identifier-1 must contain the program-name contained in the PROGRAM-ID paragraph of the program to be cancelled.

   b) Literal-1 or the content of the data item referenced by identifier-1 may not contain lower-case letters.

3. If a link unit is being cancelled:

   a) Literal-1 or the content of the data item referenced by identifier-1 specifies the path name of the link unit file on the physical device. If the content of the name contains a space character, only the characters up until the first blank are considered to be part of the name. If the physical link unit name starts with the character '/', then it is assumed to be a complete path name. Otherwise, the content of the data item referenced by identifier-1 is assumed to be a partial path name. In either case, the allowed maximum length until the first blank is fifty-nine characters.

b) Literal-1 or the content of the data item referenced by identifier-1 must be identical to that of the call which caused the dynamic load of link unit. For example, the link unit may not be called using complete a path name and successively cancelled using a relative path name.

4. If a separately compiled program within the link unit has a program-name which is the same as the file name of a link unit in the program directory of the root link unit, and the data item referenced by identifier-1 contains this name, the separately compiled program within the link unit is cancelled.

5. Subsequent to the execution of an explicit or implicit CANCEL statement, the program or link unit referred to therein ceases to have any logical relationship to the run unit in which the CANCEL statement appears. In addition, in an explicit CANCEL statement referencing a link unit, that link unit ceases to have any physical relationship to the run unit, i.e. it is unloaded. If the program or link unit referenced by a successfully executed explicit or implicit CANCEL statement in a run unit is subsequently called in that run unit, that program is in its initial state. (See Initial State of a program).

6. A program named in a CANCEL statement must be a COBOL program compiled separately within the run unit.

7. When an explicit or implicit CANCEL statement is executed, all programs contained within the program or link unit referenced by the CANCEL statement are also cancelled.

8. A program or link unit named in the CANCEL statement must not refer directly or indirectly to any program that has been called and has not yet executed an EXIT PROGRAM statement.

9. A logical relationship to a cancelled program is established only by execution of a subsequent CALL statement naming that program or link unit containing that program.

10. A called program is cancelled either by being referred to as the operand of a CANCEL statement, by the termination of the run unit of which the program is a member, or by execution of an EXIT PROGRAM statement in a called program that possesses the initial attribute.

11. No action is taken when an explicit or implicit CANCEL statement is executed naming a program that has not been called in this run unit or has been called and is at present cancelled. Control is transferred to the next executable statement following the explicit CANCEL statement.

12. The contents of data items in external data records described by a program are not changed when that program is cancelled.

13. During execution of an explicit or implicit CANCEL statement, an implicit CLOSE statement without any optional phrases is executed for each file in the open mode that is associated with an internal file connector in the program or link unit named in the CANCEL statement. Any USE procedures associated with any of these files are not executed.

Function

The EXIT PROGRAM statement marks the logical end of a called program.

General Format

EXIT PROGRAM

Syntax Rules

1. If an EXIT PROGRAM statement appears in a consecutive sequence of imperative statements within a sentence, it must appear as the last statement in that sequence.

2. The EXIT PROGRAM statement is not permitted in the DECLARATIVE section of a program.

General Rules

1. If the EXIT PROGRAM statement is executed in a program which is not under the control of a calling program, the EXIT PROGRAM statement causes execution of the program to continue with the next executable statement.

2. The execution of an EXIT PROGRAM statement in a called program which does not possess the initial attribute causes execution to continue with the next executable statement following the CALL statement in the calling program. The program state of the calling program is not altered and is identical to that which existed at the time it executed the CALL statement except that the contents of data items and the contents of data files shared between the calling and called program may have been changed. The program state of the called program is not altered except that the ends of the ranges of all PERFORM statements executed by that called program are considered to have been reached.

3. Besides the actions specified in general rule 2, the execution of an EXIT PROGRAM statement in a called program which possesses the initial attribute is equivalent also to executing a CANCEL statement referencing that program. (See The CANCEL Statement.)

USE

Function

Within Inter-Program Communication, the USE statement determines whether the associated declarative procedures are invoked during the execution of any program contained within the program which includes the USE statement.

General Format

```
USE [GLOBAL] AFTER STANDARD { EXCEPTION }
                            { ERROR     }

                                  { {file-name-1}... }
                                  { INPUT            }
PROCEDURE ON                      { OUTPUT           }
                                  { I-O              }
                                  { EXTEND           }
```

Syntax Rules

1. If file-name-1 is specified, the GLOBAL phrase may not be specified.

2. File-name-1 must be defined in the program containing the USE statement.

General Rules

1. Special precedence rules are followed when programs are contained within other programs. In applying these rules, only the first qualifying declarative will be selected for execution. The declarative which is selected for execution must satisfy the rules for execution of that declarative. The selection depends upon where the file is declared and where the USE statement is present.

   a) The declarative within the program that contains the definition of the file.

   b) The declarative in which the GLOBAL phrase is specified and which is within the program directly containing the program which was last examined for a qualifying declarative.

   c) Any declarative selected by applying rule 1b to each more inclusive containing program until rule 1b is applied to the outermost program. If no qualifying declarative is found, none is executed.

8. SEGMENTATION

Segmentation provides a means by which the user may communicate with the compiler to specify object program overlay requirements.

Segmentation is treated as commentary information.

SCOPE

COBOL segmentation deals only with segmentation of procedures. As such, only the Procedure Division and the Environment Division are considered in determining segmentation requirements for an object program.

ORGANIZATION

Although it is not mandatory, the Procedure Division for a source program is usually written as a consecutive group of sections, each of which is composed of a series of closely related operations that are designed to collectively perform a particular function. However, when segmentation is used, the entire Procedure Division must be in sections. Each section must then be specified as belonging to either the nonoverlayable or overlayable portion of the program.

The use of segmentation only affects the physical management of the object program during execution. Segmentation neither imposes any syntactic restrictions nor implies any semantic differences over the same program written without segmentation.

Nonoverlayable Portion

The nonoverlayable portion is defined as that portion of the program which must not be overlaid once it has been referenced explicitly or implicitly.

Overlayable Portion

The overlayable portion is defined as that portion of the program which can overlay any other overlayable segment or be overlaid by any nonoverlayable or overlayable segment.

SEGMENT CLASSIFICATION

Sections which are to be segmented are classified, using a system of segment-numbers and the following criteria:

- Frequency of Use - Generally, the more frequently a section is referred to, the lower its segment-number; the less frequently it is referred to, the higher its segment-number.

- Relationship to Other Sections - Sections which frequently communicate with one another should be given the same segment-numbers.

SEGMENTATION CONTROL

The logical sequence of the program is the same as the sequence in which statements are written in the source program, except for specific transfers of control. (See Explicit and Implicit Transfers of Control.) If any reordering of the object program is required to handle the flow from segment to segment, according the the rules in this section, control transfers are provided to maintain the logical flow specified in the source program.

CONFIGURATION SECTION

Information concerning the Configuration Section is located in the IDENTIFICATION AND ENVIRONMENT DIVISIONS chapter.

THE OBJECT-COMPUTER PARAGRAPH

Function

The OBJECT-COMPUTER paragraph provides a means of describing the computer on which the program is to be executed.

General Format

```
OBJECT-COMPUTER.  [computer-name

    [PROGRAM COLLATING SEQUENCE IS alphabet-name-1]

    [SEGMENT-LIMIT IS segment-number] . ]
```

Syntax Rules

1. Computer-name is a system-name.

General Rules

1. All clauses of the OBJECT-COMPUTER paragraph apply to the program in which they are explicitly or implicitly specified and to any program contained within that program.

2. General rules concerning computer-name and the PROGRAM COLLATING SEQUENCE clause are presented in the IDENTIFICATION AND ENVIRONMENT DIVISION chapter.

SEGMENT-LIMIT clause

Function

The SEGMENT-LIMIT clause provides the user with a means of specifying how each segment-number will be interpreted as belonging to either the collection of nonoverlayable or overlayable segments. The logical properties of these segments remain unchanged.

General Format

SEGMENT-LIMIT IS segment-number

Syntax Rules

1. Segment-number must be an integer ranging in value from 00 through 99.

General Rules

1. When the SEGMENT-LIMIT clause is specified, only those segments having segment-numbers less than the segment limit are the nonoverlayable portion of the program in which the clause is specified and the nonoverlayable portions of any programs contained within that program.

2. Segments which have segment-numbers equal to or greater than the segment limit are the overlayable segments.

3. When the SEGMENT-LIMIT clause is omitted, all segments are nonoverlayable.

SEGMENTATION IN THE PROCEDURE DIVISION

GENERAL DESCRIPTION

The Procedure Division contains sections with segment-numbers when Segmentation is used in a COBOL source program. Shown below is the general format of the Procedure Division when sections and segment-numbers are present.

```
PROCEDURE DIVISION.

[DECLARATIVES.

{section-name SECTION  [segment-number] .

    USE statement.

[paragraph-name.

    [sentence] ... ] ... } ...

END DECLARATIVES.]

{section-name SECTION  [segment-number] .

[paragraph-name.

    [sentence] ... ] ... } ...
```

SEGMENT-NUMBERS

Function

Section classification is accomplished by means of a system of segment-numbers. The segment-number is included in the section header within the Procedure Division.

General Format

section-name SECTION [segment-number].

Syntax Rules

1. The segment-number must be an integer ranging in value from 00 through 99.

2. If the segment-number is omitted from the section header, the segment-number is assumed to be 00.

General Rules

1. All sections within a separately compiled program which have the same segment-number constitute a program segment.

2. Segments with segment-numbers less than the number specified in the SEGMENT-LIMIT clause belong to the nonoverlayable portion of the program.

3. Segments with segment-numbers equal to or greater than the number specified in the SEGMENT-LIMIT clause belong to the overlayable portion of the program.

9. SORT-MERGE

Sort-Merge provides the capability to order one or more files of records, or to combine two or more identically ordered files of records, according to a set of user-specified keys contained within each record. Optionally, each record may be processed by input or output procedures. This processing may be applied before and/or after the records are ordered by the SORT or after the records have been combined by the MERGE.

LANGUAGE CONCEPTS

SORT FILE

A sort file is a collection of records to be sorted by a SORT statement. The sort file has no label procedures which the programmer can control and the rules for blocking and for allocation of internal storage are peculiar to the SORT statement. The RELEASE and RETURN statements imply nothing with respect to buffer areas, blocks, or reels. A sort file, then, may be considered as an internal file which is created (RELEASE statement) from the input file, processed (SORT statement), and then made available (RETURN statement) to the output file.

A sort file is named by a file control entry and is described by a sort-merge file description entry. A sort file is referred to by the RELEASE, RETURN, and SORT statements.

MERGE FILE

A merge file is a collection of records to be merged by a MERGE statement. The merge file has no label procedures which the programmer can control and the rules for blocking and for allocation of internal storage are peculiar to the MERGE statement. The RETURN statement implies nothing with respect to buffer areas, blocks, or reels. A merge file, then, may be considered as an internal file which is created from input files by combining them (MERGE statement) as the file is made available (RETURN statement) to the output file.

A merge file is named by a file control entry and is described by a sort-merge file description entry. A merge file is referred to by the RETURN and MERGE statements.

RELATIONSHIP WITH SEQUENTIAL I-O MODULE

The files specified in the USING and GIVING phrases of the SORT and MERGE statements must be described implicitly or explicitly in the FILE-CONTROL paragraph as having sequential organization. No input-output statement may be executed for the file named in the sort-merge file description.

SORT-MERGE IN THE ENVIRONMENT DIVISION

INPUT-OUTPUT SECTION

Information concerning the Input-Output Section is located in the IDENTIFICATION AND ENVIRONMENT DIVISION chapter.

THE FILE-CONTROL PARAGRAPH

Information concerning the FILE-CONTROL paragraph is located in the IDENTIFICATION AND ENVIRONMENT DIVISION chapter.

FILE-CONTROL ENTRY

Function

The file control entry declares the relevant physical attributes of a sort or merge file.

General Format

```
SELECT [OPTIONAL] file-name-1

ASSIGN TO TEXT-FILE { literal-1   } [ SYNC         ]
                    { data-name-1 } [ SYNCHRONIZED ]
```

Syntax Rules

1. Each sort or merge file described in the Data Division must be specified only once in the FILE-CONTROL paragraph. Each sort or merge file specified in the SELECT clause must have a sort-merge file description entry in the Data Division of the same program.

2. Since file-name-1 represents a sort or merge file, only the ASSIGN clause is permitted to follow file-name-1 in the FILE-CONTROL paragraph.

3. Literal-1 must be a non-numeric literal other than a figurative constant.

4. Data-name-1 may be qualified.

5. Data-name-1 must reference an alphanumeric data item.

6. Literal-1 or the content of the data item referenced by data-name-1 at the time of execution of an OPEN statement or MERGE statement referencing file-name-1 specifies the name of the file on the physical device. If the name contains a space character, only the characters up until the first blank are considered to be part of the name.

If the physical file name starts with the character '/', then it is assumed to be a complete path-name with an allowed maximum length of fiftynine characters. Otherwise, literal-1 or the content of the data item referenced by data-name-1 is assumed to be a local name or the name of a file in the working directory, and may not contain the character '/'. In this case, the maximum length allowed is twelve characters.

7. When data-name-1 is specified, the above length restriction applies to the number of characters before the first blank character, if any, or the length of the data item referenced by data-name-1.

General Rules

1. The ASSIGN clause specifies the association of the file referenced by file-name-1 to a storage medium referenced by literal-1 or the content of the data item referenced by data-name-1.

2. The presence of the SYNC or SYNCHRONIZED phrase in the ASSIGN clause indicates that the output to the indicated file is synchronized, that is, the user program receives control when the output operation has terminated. The default is asynchronous output which means that the output operation is requested immediately, but control is returned to the COBOL program before the operation is completed.

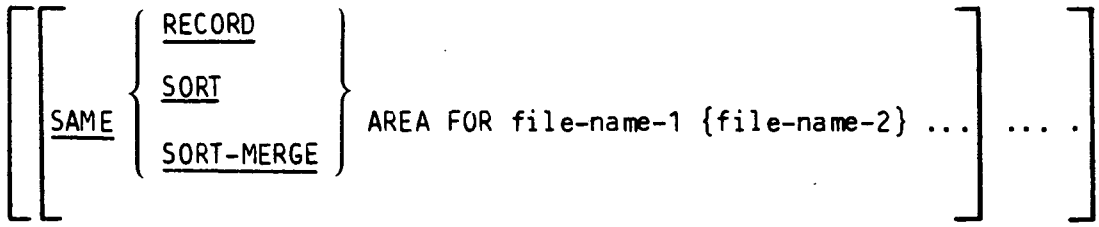
THE I-O-CONTROL PARAGRAPH

Function

The I-O-CONTROL paragraph specifies the memory area which is to be shared by different files.

General Format

I-O-CONTROL.



Syntax Rules

1. The I-O-CONTROL paragraph is optional.

General Rules

1. The SAME AREA clause for the Sort-Merge is presented on the next page.

SAME AREA clause

Function

The SAME AREA clause specifies the memory area which is to be shared by different files at least one of which is a sort or merge file.

General Format

```
SAME { RECORD    } AREA FOR file-name-1 {file-name-2} ...
     { SORT      }
     { SORT-MERGE }
```

Syntax Rules

1. Each file-name specified in the SAME AREA clause must be specified in the FILE-CONTROL paragraph of the same program.

2. File-name-1 and file-name-2 may not reference an external file control block.

3. SORT and SORT-MERGE are equivalent.

4. The SAME SORT AREA and SAME SORT-MERGE AREA clauses are treated as comments. There is, thus, only one form of the SAME clause with semantic meaning: SAME RECORD AREA.

5. More than one SAME AREA clause may be included in a program; however file-name-1 must not appear in more than one SAME RECORD AREA clause.

6. The files referenced in the SAME AREA clause need not all have the same organization or access.

General Rules

1. The SAME RECORD AREA clause specifies that two or more files referenced by file-name-1, file-name-2 are to use the same memory area for processing of the current logical record. All of these files may be in the open mode at the same time. A logical record in the SAME RECORD AREA is considered as a logical record of each file open in the output mode whose file-name appears in this SAME RECORD AREA clause and of the most recently read file open in the input mode whose file-name appears in this SAME RECORD AREA clause. This is equivalent to an implicit redefinition of the area, i.e., records are aligned on the leftmost character position.

SORT-MERGE IN THE DATA DIVISION

FILE SECTION

The File Section is located in the Data Division of a source program. The File Section defines the structure of sort files and merge files. Each sort file or merge file is defined by a sort-merge file description entry and one or more record description entries. Record description entries are written immediately following the sort-merge file description entry.

The general format of the File Section in the Sort-Merge is shown below.

FILE SECTION.

[sort-merge-file-description-entry {record-description-entry} ...] ...

SORT-MERGE FILE DESCRIPTION ENTRY

In a COBOL program, the sort-merge file description entry (SD) represents the highest level of organization in the File Section. The File Section header is followed by a sort-merge file description entry consisting of a level indicator (SD), a file-name, and a series of independent clauses. The clauses of a sort-merge file description entry (SD entry) specify the size and the names of the data records associated with a sort file or a merge file. There are no label procedures which the user can control, and the rules for blocking and internal storage are peculiar to the SORT and MERGE statements. The sort-merge file description entry is terminated by a period.

RECORD DESCRIPTION STRUCTURE

A record description consists of a set of data description entries which describe the characteristics of a particular record. Each data description entry consists of a level-number followed by the data-name or FILLER clause, if specified, followed by a series of independent clauses as required. A record description may have a hierarchical structure and therefore the clauses used with an entry may vary considerably, depending upon whether or not it is followed by subordinate entries. The structure of a record description and the elements allowed in a record description entry are explained in paragraphs Concept of Levels, and The Data Description Entry.

INITIAL VALUES

The initial value of data items in the File Section is undefined.

THE SORT-MERGE FILE DESCRIPTION ENTRY

Function

The sort-merge file description furnishes information concerning the physical structure and record names pertaining to a sort or merge file.

General Format

```
SD  file-name-1

    [RECORD CONTAINS  [integer-1 TO]  integer-2 CHARACTERS]

[ DATA { RECORD IS  } {data-name-1} ... ] .
       { RECORDS ARE}
```

Syntax Rules

1. The level indicator SD identifies the beginning of the sort-merge file description and must precede file-name-1.

2. The clauses which follow file-name-1 are optional, and their order of appearance is immaterial.

3. One or more record description entries must follow the sort-merge file description entry; however, no input-output statements may be executed for this sort or merge file.

General Rules

1. The DATA RECORDS clause for the Sort-Merge is the same as the DATA RECORDS clause for Sequential I-O , See the DATA RECORDS clause.

2. The RECORD CONTAINS clause for the Sort-Merge is the same as the RECORD CONTAINS clause for Sequential I-O . See the RECORD CONTAINS clause.

SORT-MERGE IN THE PROCEDURE DIVISION

The Procedure Division statements for the Sort-Merge are presented on the following pages in alphabetical order.

MERGE

Function

The MERGE statement combines two or more identically sequenced files on a set of specified keys, and during the process makes records available, in merged order, to an output procedure or to an output file.

General Format

```
MERGE file-name-1 { ON { ASCENDING  } KEY {data-name-1}... } ...
                   {     { DESCENDING }                    }

    [COLLATING SEQUENCE IS alphabet-name-1]

    USING file-name-2 {file-name-3} ...

{ OUTPUT PROCEDURE IS                                          }
{                     procedure-name-1 [{ THROUGH } procedure-name-2] }
{                                      { THRU    }                  }
{ GIVING {file-name-4} ...                                     }
```

Syntax Rules

1. MERGE statements may appear anywhere except in the declaratives portion of the Procedure Division or in an input or output procedure associated with a SORT or MERGE statement.

2. File-name-1 must be described in a sort-merge file description entry in the Data Division.

3. Data-name-1 is a key data-name. Key data-names are subject to the following rules:

   a) The data items identified by key data-names must be described in records associated with file-name-1.

   b) Key data-names may be qualified.

   c) The data items identified by key data-names must not name group items which contain variable occurrence data items.

   d) If file-name-1 has more than one record description, the data items identified by key data-names need be described in only one of the record descriptions. The same character positions referenced by a key data-name in one record description entry are taken as the key in all records of the file.

e) None of the data items identified by key data-names can be described by an entry which either contains an OCCURS clause or is subordinate to an entry which contains an OCCURS clause.

4. A maximum number of sixty key data-names may be specified.

5. File-name-2, file-name-3, and file-name-4 must be described in a file description entry, not in a sort-merge file description entry, in the Data Division. The actual size of the logical record(s) described for file-name-2, file-name-3, and file-name-4, must be equal to the actual size of the logical record(s) described for file-name-1. If the data descriptions of the elementary items that make up these records are not identical, it is the programmer's responsibility to describe the corresponding records in such a manner so as to cause an equal number of character positions to be allocated for the corresponding records.

6. File-names must not be repeated within the MERGE statement.

7. The only file-names in a MERGE statement that can be specified in the same SAME RECORD AREA clause are those associated with the GIVING phrase.

8. The words THRU and THROUGH are equivalent.

General Rules

1. The MERGE statement merges all records contained on the files referenced by file-name-2 and file-name-3.

2. The data-names following the word KEY are listed from left to right in the MERGE statement in order of decreasing significance without regard to how they are divided into KEY phrases. The leftmost data-name is the major key, the next data-name is the next more significant key, etc.

   a) When the ASCENDING phrase is specified, the merged sequence will be from the lowest value of the contents of the data items identified by the key data-names to the highest value, according to the rules for comparison of operands in a relation condition.

   b) When the DESCENDING phrase is specified, the merged sequence will be from the highest value of the contents of the data items identified by the key data-names to the lowest value, according to the rules for comparison of operands in a relation condition.

3. When, according to the rules for the comparison of operands in a relation condition, the contents of all the key data items of one data record are equal to the contents of the corresponding key data items of one or more other data records, the order of return of these records:

   a) follows the order of the associated input files as specified in the MERGE statement.

b) is such that all records associated with one input file are returned prior to the return of records from another input file.

4. The collating sequence that applies to the comparison of the non-numeric key data items specified is determined in the following order of precedence:

   a) First, the collating sequence established by the COLLATING SEQUENCE phrase, if specified, in that MERGE statement.

   b) Second, the collating sequence established as the program collating sequence.

5. The results of the merge operations are undefined unless the records in the files referenced by file-name-2 and file-name-3 are ordered as described in the ASCENDING or DESCENDING KEY phrases associated with the MERGE statement.

6. All the records in the files referenced by file-name-2 and file-name-3 are transferred to the file referenced by file-name-1. At the start of execution of the MERGE statement, the files referenced by file-name-2 and file-name-3 must not be in the open mode. For each of the files referenced by file-name-2 and file-name-3 the execution of the MERGE statement causes the following actions to be taken:

   a) The processing of the file is initiated. The initiation is performed as if an OPEN statement with the INPUT phrase had been executed. If an output procedure is specified, this initiation is performed before control passes to the output procedure.

   b) The logical records are obtained and released to the merge operation. Each record is obtained as if a READ statement with the NEXT and the AT END phrases had been executed.

   c) The processing of the file is terminated. The termination is performed as if a CLOSE statement without optional phrases had been executed. If an output procedure is specified, this termination is not performed until after control passes the last statement in the output procedure.

   These implicit functions are performed such that any associated USE AFTER EXCEPTION/ERROR procedures are executed.

7. The output procedure may consist of any procedure needed to select, modify, or copy the records that are made available one at a time by the RETURN statement in merged order from the file referenced by file-name-1. The range includes all statements that are executed as the result of a transfer of control by CALL, EXIT, GO TO, and PERFORM statements in the range of the output procedure, as well as all statements in declarative procedures that are executed as a result of the execution of statements in the range of the output procedure. The range of the output procedure must not cause the execution of any MERGE, RELEASE, or SORT statement. (See Explicit and Implicit Specifications.)

8. If an output procedure is specified, control passes to it during execution of the MERGE statement. The compiler inserts a return mechanism at the end of the last section in the output procedure. When control passes the last statement in the output procedure, the return mechanism provides for termination of the merge, and then passes control to the next executable statement after the MERGE statement. Before entering the output procedure, the merge procedure reaches a point at which it can select the next record in merged order when requested. The RETURN statements in the output procedure are the requests for the next record.

9. During the execution of the output procedure, no statement may be executed manipulating the file referenced by, or accessing the record area associated with, file-name-2 or file-name-3. During the execution of any USE AFTER STANDARD EXCEPTION procedure implicitly invoked while executing the MERGE statement, no statement may be executed manipulating the file referenced by, or accessing the record area associated with, file-name-2, file-name-3, or file-name-4.

10. If the GIVING phrase is specified, all the merged records are written on the file referenced by file-name-4 as the implied output procedure for the MERGE statement. At the start of execution of the MERGE statement, the file referenced by file-name-4 must not be in the open mode. For each of the files referenced by file-name-4, the execution of the MERGE statement causes the following actions to be taken:

    a) The processing of the file is initiated. The initiation is performed as if an OPEN statement with the OUTPUT phrase had been executed.

    b) The merged logical records are returned and written onto the file. Each record is written as if a WRITE statement without any optional phrases had been executed.

    c) The processing of the file is terminated. The termination is performed as if a CLOSE statement without optional phrases had been executed.

    These implicit functions are performed such that any associated USE AFTER EXCEPTION/ERROR procedures are executed; however the execution of such a USE procedure must not cause the execution of any statement manipulating the file referenced by, or accessing the record area associated with, file-name-4. On the first attempt to write beyond the externally defined boundaries of the file, any USE AFTER STANDARD EXCEPTION/ERROR procedure specified for the file is executed; if control is returned from that USE procedure or if no such USE procedure is specified, the processing of the file is terminated as in paragraph 10c above.

RELEASE

Function

The RELEASE statement transfers records to the initial phase of a sort operation.

General Format

RELEASE record-name-1 [FROM identifier-1]

Syntax Rules

1. Record-name-1 must be the name of a logical record in a sort-merge file description entry and it may be qualified.

2. A RELEASE statement may be used only within the range of an input procedure associated with a SORT statement for the file-name whose sort-merge file description entry contains record-name-1.

3. Record-name-1 and identifier-1 must not refer to the same storage area.

General Rules

1. The execution of a RELEASE statement causes the record named by record-name-1 to be released to the initial phase of a sort operation.

2. The logical record released by the execution of the RELEASE statement is no longer available in the record area unless the sort-merge file-name associated with record-name-1 is specified in a SAME RECORD AREA clause. In this case, the logical record is available to the program as a record of other files whose file-names are specified in the same SAME RECORD AREA clause.

3. The result of the execution of a RELEASE statement with the FROM phrase is equivalent to the execution of the following statements in the order specified:

   a) The statement:

      MOVE identifier-1 TO record-name-1

      according to the rules specified for the MOVE statement.

   b) The same RELEASE statement without the FROM phrase.

4. After the execution of the RELEASE statement is complete, the information in the area referenced by identifier-1 is available, even though the information in the area referenced by record-name-1 is not available except as specified by the SAME RECORD AREA clause.

Function

The RETURN statement obtains either sorted records from the final phase of a SORT operation or merged records during a MERGE operation.

General Format

RETURN file-name-1 RECORD [INTO identifier-1]

AT END imperative-statement-1 [END-RETURN]

Syntax Rules

1. The storage area associated with identifier-1 and the record area associated with file-name-1 must not be the same storage area.

2. File-name-1 must be described by a sort-merge file description entry in the Data Division.

3. A RETURN statement may only be used within the range of an output procedure associated with a SORT or MERGE statement for file-name-1.

General Rules

1. When the logical records of a file are described with more than one record description, these records automatically share the same storage area; this is equivalent to an implicit redefinition of the area. The contents of any data items which lie beyond the range of the current data record are undefined at the completion of the execution of the RETURN statement.

2. The execution of the RETURN statement causes the next existing record in the file referenced by file-name-1, as determined by the keys listed in the SORT or MERGE statement, to be made available in the record area associated with file-name-1. If no next logical record exists in the file referenced by file-name-1, the at end condition exists and control is transferred to imperative-statement-1 of the AT END phrase. When the at end condition occurs, execution of the RETURN statement is unsuccessful and the contents of the record area associated with file-name-1 are undefined.

3. The END-RETURN phrase delimits the scope of the RETURN statement. (See Scope of Statements.)

4. The result of the execution of a RETURN statement with the INTO phrase is equivalent to the application of the following rules in the order specified:

   a) The execution of the same RETURN statement without the INTO phrase.

b) The current record is moved from the record area to the area specified by identifier-1 according to the rules for the MOVE statement without the CORRESPONDING phrase. The implied MOVE statement does not occur if the execution of the RETURN statement was unsuccessful. Any subscripting associated with identifier-1 is evaluated after the record has been read and immediately before it is moved to the data item. The record is available in both the record area and the data item associated with identifier-1.

Function

The SORT statement creates a sort file by executing input procedures or by transferring records from another file, sorts the records in the sort file on a set of specified keys, and, in the final phase of the sort operation, makes available each record from the sort file, in sorted order, to some output procedures or to an output file.

General Format

```
SORT file-name-1  { ON { ASCENDING  } KEY {data-name-1}... } ...
                       { DESCENDING }

    [WITH DUPLICATES IN ORDER]

    [COLLATING SEQUENCE IS alphabet-name-1]

{ INPUT PROCEDURE IS                                          }
{                    procedure-name-1 [ { THROUGH } procedure-name-2 ] }
{                                       { THRU    }                   }
{ USING {file-name-2} ...                                     }

{ OUTPUT PROCEDURE IS                                         }
{                     procedure-name-3 [ { THROUGH } procedure-name-4 ] }
{                                        { THRU    }                   }
{ GIVING {file-name-3} ...                                    }
```

Syntax Rules

1. A SORT statement may appear anywhere in the Procedure Division except in the declaratives portion or in an input procedure or output procedure associated with a SORT or MERGE statement.

2. File-name-1 must be described in a sort-merge file description entry in the Data Division.

3. Data-name-1 is a key data-name. Key data-names are subject to the following rules:

a) The data items identified by key data-names must be described in records associated with file-name-1.

b) Key data-names may be qualified.

c) The data items identified by key data-names must not be group items which contain variable occurrence data items.

d) If file-name-1 has more than one record description, then the data items identified by key data-names need be described in only one of the record descriptions. The same character positions which are referenced by a key data-name in one record description entry are taken as the key in all records of the file.

e) None of the data items identified by key data-names can be described by an entry which either contains an OCCURS clause or is subordinate to an entry which contains an OCCURS clause.

4. A maximum number of sixty key data-names may be specified.

5. The words THRU and THROUGH are equivalent.

6. File-name-2 and file-name-3 must be described in a file description entry, not in a sort-merge file description entry, in the Data Division. The actual size of the logical record(s) described for file-name-2 and file-name-3 must be equal to the actual size of the logical record(s) described for file-name-1. If the data descriptions of the elementary items that make up these records are not identical, it is the programmer's responsibility to describe the corresponding records in such a manner so as to cause equal amounts of character positions to be allocated for the corresponding records.

General Rules

1. The data-names following the word KEY are listed from left to right in the SORT statement in order of decreasing significance without regard to how they are divided into KEY phrases. The leftmost data-name is the major key, the next data-name is the next most significant key, etc.

   a) When the ASCENDING phrase is specified, the sorted sequence will be from the lowest value of the contents of the data items identified by the key data-names to the highest value, according to the rules for comparison of operands in a relation condition.

   b) When the DESCENDING phrase is specified, the sorted sequence will be from the highest value of the contents of the data items identified by the key data-names to the lowest value, according to the rules for comparison of operands in a relation condition.

2. If the DUPLICATES phrase is specified and the contents of all the key data items associated with one data record are equal to the contents of the corresponding key data items associated with one or more other data records, then the order of return of these records is:

a) the order of the associated input files as specified in the SORT statement. Within a given input file the order is that in which the records are accessed from that file.

b) the order in which these records are released by an input procedure, when an input procedure is specified.

3. If the DUPLICATES phrase is not specified and the contents of all the key data items associated with one data record are equal to the contents of the corresponding key data items associated with one or more other data records, then the order of return of these records is undefined.

4. The collating sequence that applies to the comparison of the nonnumeric key data items specified is determined in the following order of precedence:

   a) First, the collating sequence established by the COLLATING SEQUENCE phrase, if specified, in the SORT statement.

   b) Second, the collating sequence established as the program collating sequence.

5. The execution of the SORT statement consists of three distinct phases as follows:

   a) Records are made available to the file referenced by file-name-1. This is achieved either by the execution of RELEASE statements in the input procedure or by the implicit execution of READ statements for file-name-2. When this phase commences, the file referenced by file-name-2 must not be in the open mode. When this phase terminates, the file referenced by file-name-2 is not in the open mode.

   b) The file referenced by file-name-1 is sequenced. No processing of the files referenced by file-name-2 and file-name-3 takes place during this phase.

   c) The records of the file referenced by file-name-1 are made available in sorted order. The sorted records are either written to the file referenced by file-name-3 or, by the execution of a RETURN statement, are made available for processing by the output procedure. When this phase commences, the file referenced by file-name-3 must not be in the open mode. When this phase terminates, the file referenced by file-name-3 is not in the open mode.

6. The input procedure may consist of any procedure needed to select, modify, or copy the records that are made available one at a time by the RELEASE statement to the file referenced by file-name-1. The range includes all statements that are executed as the result of a transfer of control by CALL, EXIT, GO TO, and PERFORM statements in the range of the input procedure, as well as all statements in declarative procedures that are executed as a result of the execution of statements in the range of the input procedure. The range of the input procedure must not cause the execution of any MERGE, RETURN, or

SORT statement. (See Explicit and Implicit Specifications.)

7. If an input procedure is specified, control is passed to the input procedure before the file referenced by file-name-1 is sequenced by the SORT statement. The compiler inserts a return mechanism at the end of the last statement in the input procedure and when control passes the last statement in the input procedure, the records that have been released to file-name-1 are sorted.

8. If the USING phrase is specified, all the records in the file(s) referenced by file-name-2 are transferred to the file referenced by file-name-1. For each of the files referenced by file-name-2 the execution of the SORT statement causes the following actions to be taken:

   a) The processing of the file is initiated. The initiation is performed as if an OPEN statement with the INPUT phrase had been executed.

   b) The logical records are obtained and released to the sort operation. Each record is obtained as if a READ statement with the NEXT and the AT END phrases had been executed.

   c) The processing of the file is terminated. The termination is performed as if a CLOSE statement without optional phrases had been executed. This termination is performed before commencing the execution of any output procedure.

   These implicit functions are performed such that any associated USE AFTER STANDARD EXCEPTION procedures are executed; however, the execution of such a USE procedure must not cause the execution of any statement manipulating the file referenced by, or accessing the record area associated with, file-name-2.

9. The output procedure may consist of any procedure needed to select, modify, or copy the records that are made available one at a time by the RETURN statement in sorted order from the file referenced by file-name-1. The range includes all statements that are executed as the result of a transfer of control by CALL, EXIT, GO TO, and PERFORM statements in the range of the output procedure, as well as all statements in declarative procedures that are executed as a result of the execution of statements in the range of the output procedure. The range of the output procedure must not cause the execution of any MERGE, RELEASE, or SORT statement. (See Explicit and Implicit Specifications.)

10. If an output procedure is specified, control passes to it after the file referenced by file-name-1 has been sequenced by the SORT statement. The compiler inserts a return mechanism at the end of the last statement in the output procedure and when control passes the last statement in the output procedure, the return mechanism provides for termination of the sort and then passes control to the next executable statement after the SORT statement. Before entering the output procedure, the sort procedure reaches a point at which it can select the next record in sorted order when requested. The RETURN statements in the output procedure are the requests for the next record.

11. If the GIVING phrase is specified, all the sorted records are written on the file referenced by file-name-3 as the implied output procedure for the SORT statement. For each of the files referenced by file-name-3, the execution of the SORT statement causes the following actions to be taken:

    a) The processing of the file is initiated. The initiation is performed as if an OPEN statement with the OUTPUT phrase had been executed. This initiation is performed after the execution of any input procedure.

    b) The sorted logical records are returned and written onto the file. The records are written as if a WRITE statement without any optional phrases had been executed.

    c) The processing of the file is terminated. The termination is performed as if a CLOSE statement without optional phrases had been executed.

    These implicit functions are performed such that any associated USE AFTER EXCEPTION/ERROR procedures are executed; however, the execution of such a USE procedure must not cause the execution of any statement manipulating the file referenced by, or accessing the record area associated with, file-name-3. On the first attempt to write beyond the externally defined boundaries of the file, any USE AFTER STANDARD EXCEPTION/ERROR procedure specified for the file is executed; if control is returned from that USE procedure or if no such USE procedure is specified, the processing of the file is terminated as in paragraph 11c above.

10. DEBUG

Debug provides a means by which the user can describe his debugging algorithm including the conditions under which data items or procedures are to be monitored during the execution of the object program.

The decisions of what to monitor and what information to display on the output device are explicitly in the domain of the user. The COBOL debug facility simply provides a convenient access to pertinent information.

Most of the features of the Debug module are not supported by NSPL COBOL.

LANGUAGE CONCEPTS

DEBUG FEATURES

The features of the COBOL language that support Debug are:

- A compile time switch -- WITH DEBUGGING MODE clause.
- An object time switch.
- A USE FOR DEBUGGING statement.
- A special register -- DEBUG-ITEM.
- Debugging lines.

The compile time switch and debugging lines are the only debugging features supported.

SPECIAL REGISTER DEBUG-ITEM

The reserved word DEBUG-ITEM is the name for a special register generated automatically by the code that supports the debugging facility. Only one DEBUG-ITEM is allocated per program. The names of the subordinate data items in DEBUG-ITEM are also reserved words. However, in NSPL COBOL, these words are treated as unimplemented reserved words.

COMPILE TIME SWITCH

The WITH DEBUGGING MODE clause is written as part of the SOURCE-COMPUTER paragraph. It serves as a compile time switch over the debugging lines written in the separately compiled program.

When the WITH DEBUGGING MODE clause is specified in a separately compiled program, all debugging lines are compiled as specified in this section of the document. When the WITH DEBUGGING MODE clause is not specified, all debugging lines are compiled as if they were comment lines. All debugging sections are always compiled as if they were comment lines.

The presence or absence of the WITH DEBUGGING MODE clause is logically determined after all COPY statements are processed.

AN OBJECT TIME SWITCH

An object time switch dynamically activates the debugging code inserted by the compiler. This switch cannot be addressed in the program: it is controlled outside the COBOL environment. If the switch is on, all the effects of the debugging language written in the source program are permitted. If the switch is off, only the effects described in paragraph, USE FOR DEBUGGING Statement, are inhibited. Recompilation of the source program is not required to provide or take away this facility.

The object time switch has no effect on the execution of the object program if the WITH DEBUGGING MODE clause was not specified in the source program at compile time.

In the NSPL COBOL environment, no such object time switch is defined.

DEBUG IN THE ENVIRONMENT DIVISION

Debug only effects the SOURCE-COMPUTER paragraph in the Environment Division, in which the WITH DEBUGGING MODE clause may be specified.

Function

The WITH DEBUGGING MODE clause in the SOURCE-COMPUTER paragraph indicates that all debugging lines are to be compiled. If this clause is not specified, all debugging lines are compiled as if they were comment lines. In NSPL COBOL, this clause has no effect on debugging sections.

General Format

SOURCE-COMPUTER. [computer-name [WITH DEBUGGING MODE] .]

General Rules

1. If the WITH DEBUGGING MODE clause is specified in the program, all debugging lines are compiled.

2. If the WITH DEBUGGING MODE clause is not specified in a program and the program is not contained within a program including a WITH DEBUGGING MODE clause, any debugging lines are compiled as if they were comment lines.

3. Irregardless of the presence or absence of the WITH DEBUGGING MODE clause, all USE FOR DEBUGGING statements are analyzed syntactically and treated as a comment semantically. The associated debugging sections are treated as a comment.

DEBUG IN THE PROCEDURE DIVISION

GENERAL DESCRIPTION

The Procedure Division contains declarative procedures when the USE FOR DEBUGGING statement from Debug is present in a COBOL source program. Shown below is the general format of the Procedure Division when the USE FOR DEBUGGING statement is present.

```
PROCEDURE DIVISION.

DECLARATIVES.

{section-name SECTION.

    USE FOR DEBUGGING statement.

[paragraph-name.

    [sentence] ... ] ... } ...

[section-name SECTION.

    USE statement.

[paragraph-name.

    [sentence] ... ] ... ] ...

END DECLARATIVES.

{section-name SECTION.

[paragraph-name.

    [sentence] ... ] ... } ...
```

Function

The USE FOR DEBUGGING statement identifies the user items that are to be monitored by the associated debugging section.

General Format

```
                                  { [ALL REFERENCES OF] identifier-1 }
                                  {                                 }
USE FOR DEBUGGING ON              { file-name-1                     }  ... .
                                  { procedure-name-1                }
                                  { ALL PROCEDURES                  }
```

Syntax Rules

1. Debugging section(s), if specified, must appear together immediately after the DECLARATIVES header.

2. Except in the USE FOR DEBUGGING statement itself, there must be no reference to any non-declarative procedure within the debugging section.

3. Statements appearing outside of the set of debugging sections must not reference procedure-names defined within the set of debugging sections.

4. Except for the USE FOR DEBUGGING statement itself, statements appearing within a given debugging section may reference procedure-names defined within a different USE procedure only with a PERFORM statement.

5. Procedure-names defined within debugging sections must not appear within USE FOR DEBUGGING statements.

6. Any given identifier, file-name, or procedure-name may appear in only one USE FOR DEBUGGING statement and may appear only once in that statement.

7. The ALL PROCEDURES phrase can appear only once in a program.

8. When the ALL PROCEDURES phrase is specified, procedure-name-1 must not be specified in any USE FOR DEBUGGING statement.

9. Identifier-1 must not reference any data item defined in the Report Section except sum counters.

10. Identifier-1 must not reference any data item defined in the Screen Section.

11. If the data description entry of the data item referenced by identifier-1 contains an OCCURS clause or is subordinate to a data description entry that contains an OCCURS clause, identifier-1 must be specified without the subscripting normally required.

12. References to the special register DEBUG-ITEM are restricted to references from within a debugging section.

13. Identifier-1 or file-name-1 must be defined in the program containing the USE statement.

General Rules

1. The USE FOR DEBUGGING statement is treated as a comment. The debugging sections are discarded.

2. A reference to identifier-1, file-name-1, or procedure-name-1 as a qualifier does not constitute a reference to that item for the debugging described in the general rules below. For purposes of debugging, a reference to identifier-1 with or without its associated subscripting constitutes an explicit reference to identifier-1 and causes the associated debugging section to be executed in accordance with general rules 6 and 7. If identifier-1 refers to an external data item, a reference to identifier-1 from another program which also describes the external data item is not considered to constitute a reference to identifier-1 in the program in which the debugging section is specified.

3. When more than one sequence of qualifiers may be used to reference a given data item or procedure-name, any form of explicit reference to such data item or procedure-name specified in the USE FOR DEBUGGING statement will cause the execution of the debugging section.

4. Automatic execution of a debugging section is not caused by a statement appearing in a debugging section.

5. The implicit CLOSE, OPEN, READ, and WRITE statements which are executed as part of a sort or merge operation are considered to be explicit references for debugging purposes.

6. The ALL PROCEDURES phrase causes the effects described in general rule 9 to occur for every procedure-name in the program, except those appearing within a debugging section.

7. When the ALL REFERENCES OF identifier-1 phrase is specified, that debugging section is executed for every statement that explicitly references identifier-1 at each of the following times:

   a) In the case of a RELEASE, REWRITE or WRITE statement, immediately before the execution of that RELEASE, REWRITE or WRITE statement, and after the execution of any implicit move resulting from the presence of the FROM phrase.

   b) In the case of a GO TO statement with a DEPENDING ON phrase, immediately before control is transferred and prior to the execution of any debugging section associated with the procedure-name to which control is to be transferred.

   c) In the case of a PERFORM statement in which a VARYING, AFTER, or UNTIL phrase references identifier-1, immediately after each initialization, modification or evaluation of the content of the

data item referenced by identifier-1.

d) In the case of a conditional statement other than the SEARCH statement, immediately after the determination of the truth value(s) and before the execution either of any associated imperative statement or of any other statement.

e) In the case of a SEARCH statement:

- In which a WHEN phrase references identifier-1, immediately after the determination of the truth value of that WHEN phrase.

- In which a VARYING phrase references identifier-1, immediately after modification of the contents of the data item referenced by identifier-1.

The debugging sections are executed one for each evaluation and/or modification of identifier-1 for every iteration of the search loop.

f) In the case of any other COBOL statement, immediately after execution of that statement.

If identifier-1 is specified in a phrase or a part of a statement that is not executed or evaluated, or if identifier-1 is specified only in the FROM or BY phrase of a PERFORM statement, or if identifier-1 is specified only as a table being searched in a SEARCH statement, the associated debugging section is not executed.

8. When identifier-1 is specified without the ALL REFERENCES OF phrase, that debugging section is executed at each of the following times:

a) In the case of a RELEASE, REWRITE or WRITE statement that explicitly references identifier-1, immediately before the execution of that RELEASE, REWRITE or WRITE statement and after the execution of any implicit move resulting from the presence of the FROM phrase.

b) In the case of a PERFORM statement in which a VARYING or AFTER phrase references identifier-1, immediately after each initialization, modification or evaluation of the content of the data item referenced by identifier-1.

c) In the case of a SEARCH statement in which a VARYING phrase references identifier-1, immediately after modification of the content of the data item referenced by identifier-1.

The debugging sections are executed once for each modification of identifier-1 for every iteration of the search loop.

d) Immediately after the execution of any other COBOL statement that explicitly references the data item referenced by identifier-1 as a receiving operand.

If identifier-1 is specified in a phrase or a part of a statement

that is not executed or evaluated, or if identifier-1 is specified only in the FROM or BY phrase of a PERFORM statement, or if identifier-1 is specified only as a table being searched in a SEARCH statement, the associated debugging section is not executed.

9. When file-name-1 is specified in a USE FOR DEBUGGING statement, that debugging section is executed at each of the following times:

   a) Immediately after the execution of any CLOSE or OPEN statement that references file-name-1.

   b) Immediately after the execution of any READ statement (after any other specified USE procedure) not resulting in the execution of an associated AT END or INVALID KEY imperative statement.

   c) Immediately after the execution of any DELETE or START statement that references file-name-1.

   d) Immediately after the execution of any RETURN statement not resulting in the execution of an associated AT END imperative-statement.

10. When procedure-name-1 is specified in a USE FOR DEBUGGING statement, that debugging section is executed immediately before each execution of the named procedure.

11. The associated debugging section is not executed for a specific operand more than once as a result of the execution of a single statement, regardless of the number of times that operand is explicitly specified. In the case of the identifier, the associated debugging section is executed once for each unique identifier. However, each subscripted operand is treated, for the purposes of debugging, as if it were different from every other operand in the statement. In the case of a PERFORM statement which causes iterative execution of a referenced procedure, the associated debugging section is executed once for each iteration.

12. Associated with each execution of a debugging section is the special register DEBUG-ITEM, which provides information about the conditions that caused the execution of a debugging section. DEBUG-ITEM has the following implicit description:

```
01  DEBUG-ITEM.
    02 DEBUG-LINE       PICTURE IS X(6).
    02 FILLER           PICTURE IS X VALUE SPACE.
    02 DEBUG-NAME       PICTURE IS X(30).
    02 FILLER           PICTURE IS X VALUE SPACE.
    02 DEBUG-SUB-1      PICTURE IS S9999 SIGN IS
                                   LEADING SEPARATE CHARACTER.
    02 FILLER           PICTURE IS X VALUE SPACE.
    02 DEBUG-SUB-2      PICTURE IS S9999 SIGN IS
                                   LEADING SEPARATE CHARACTER.
    02 FILLER           PICTURE IS X VALUE SPACE.
    02 DEBUG-SUB-3      PICTURE IS S9999 SIGN IS
                                   LEADING SEPARATE CHARACTER.
    02 FILLER           PICTURE IS X VALUE SPACE.
    02 DEBUG-CONTENTS   PICTURE IS X(n).
```

13. Prior to each execution of a debugging section, the content of the data item referenced by DEBUG-ITEM are space-filled. The contents of data items subordinate to DEBUG-ITEM are then updated, according to the following general rules, immediately before control is passed to that debugging section. The content of any data item not specified in the following general rules remain spaces.

    Updating is accomplished in accordance with the rules for the MOVE statement; the sole exception being the move to DEBUG-CONTENTS, when the move is treated exactly as if it was an alphanumeric to alphanumeric elementary move with no conversion of data from one form of internal representation to another.

14. The content of DEBUG-LINE is the source line number used in the source listing to identify a particular source statement.

15. DEBUG-NAME contains the first 30 characters of the name, as specified in the USE FOR DEBUGGING statement, that caused the debugging section to be executed. Each word of the name is separated by a single space.

    All qualifiers of the name are separated in DEBUG-NAME by the word 'IN' or 'OF'. Subscripts, if any, are not entered into DEBUG-NAME.

16. If the reference to a data item that causes the debugging section to be executed is subscripted, the occurrence number of each level in entered in DEBUG-SUB-1, DEBUG-SUB-2, DEBUG-SUB-3 respectively as necessary.

17. DEBUG-CONTENTS is a data item that is large enough to contain the data required by the following general rules.

18. If the first execution of the first nondeclarative procedure in the program causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the first statement of that procedure.

    b) DEBUG-NAME contains the name of that procedure.

c) DEBUG-CONTENTS contains 'START PROGRAM'.

19. If the transfer of control associated with the execution of a GO TO statement causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the GO TO statement whose execution transfers control to procedure-name-1.

    b) DEBUG-NAME contains procedure-name-1.

20. If a reference to file-name-1 causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the source statement that references file-name-1.

    b) DEBUG-NAME contains the name of file-name-1.

    c) For READ or RETURN, DEBUG-CONTENTS contains the entire record read.

    d) For all other references to file-name-1, DEBUG-CONTENTS contains spaces.

21. If a reference to identifier-1 causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the source statement that references identifier-1.

    b) DEBUG-NAME contains the name of identifier-1.

    c) If the data description entry of the data item referenced by identifier-1 contains an OCCURS clause or is subordinate to a data description entry that contains an OCCURS clause, DEBUG-CONTENTS contains only the content of the explicit table element which caused the USE FOR DEBUGGING statement to be executed.

    d) DEBUG-CONTENTS contains the content of the data item referenced by identifier-1 at the time that control passes to the debugging section. (See general rules 6 and 7.)

22. If reference to procedure-name-1 in the INPUT or OUTPUT phrase of a SORT or MERGE statement causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the SORT or MERGE statement that references procedure-name-1.

    b) DEBUG-NAME contains procedure-name-1.

    c) DEBUG-CONTENTS contains:

        - If the reference to procedure-name-1 is in the INPUT phrase of a SORT statement, 'SORT INPUT'.

- If the reference to procedure-name-1 is in the OUTPUT phrase of a SORT statement, 'SORT OUTPUT'.

- If the reference to procedure-name-1 is in the OUTPUT phrase of a MERGE statement, 'MERGE OUTPUT'.

23. If the transfer of control from the control mechanism associated with a PERFORM statement caused the debugging section associated with procedure-name-1 to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the PERFORM statement that references procedure-name-1.

    b) DEBUG-NAME contains procedure-name-1.

    c) DEBUG-CONTENTS contains 'PERFORM LOOP'.

24. If procedure-name-1 is a USE procedure that is to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the statement that causes execution of the USE procedure.

    b) DEBUG-NAME contains procedure-name-1.

    c) DEBUG-CONTENTS contains 'USE PROCEDURE'.

25. If an implicit transfer of control from the previous sequential paragraph to procedure-name-1 causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the previous statement.

    b) DEBUG-NAME contains procedure-name-1.

    c) DEBUG-CONTENTS contains 'FALL THROUGH'.

26. If the flow of control passes sequentially through a paragraph-name or section-name to procedure-name-1 without encountering any intervening executable statements and causes the debugging section to be executed, the following conditions exist:

    a) DEBUG-LINE identifies the previous paragraph-name or section-name.

    b) DEBUG-NAME contains procedure-name-1.

    c) DEBUG-CONTENTS contains 'FALL THROUGH'.

DEBUGGING LINES

A debugging line is any line with a 'D' in the indicator area of the line. Any debugging line that consists solely of spaces from margin A to margin R is considered the same as a blank line.

The content of a debugging line must be such that a syntactically correct program is formed with or without the debugging lines being considered as comment lines.

After all COPY statements have been processed, a debugging line will be considered to have all the characteristics of a comment line, if the WITH DEBUGGING MODE clause is not specified in the SOURCE-COMPUTER paragraph.

Successive debugging lines are allowed.

A debugging line is only permitted in the separately compiled program after the OBJECT-COMPUTER paragraph.

11. THE REPORT WRITER

The Report Writer provides the facility for producing reports by specifying the physical appearance of a report rather than requiring specification of the detailed procedures necessary to produce that report.

A hierarchy of levels is used in defining the logical organization of a report. Each report is divided into report groups, which in turn are divided into sequences of items. Such a hierarchical structure permits explicit reference to a report group with implicit reference to other levels in the hierarchy. A report group contains one or more items to be presented on one or more lines.

LANGUAGE CONCEPTS

The report named in the report description entry is not assigned directly to an output file. Instead, it is associated with a file-name in the File Section and that file-name is associated with a file when an OPEN statement specifying the file-name is executed. More than one report may be associated with the same file-name and the CODE clause is used to differentiate among the reports. For an external file connector referenced by a file-name, separately compiled programs may specify different reports for the same file-name. The file description entry of a file-name to which a report is assigned may not contain record description entries which describe data records. This file description entry must specify the name of a report description entry for each report associated with that file-name in this program.

The report description entry contains a set of clauses that names the report and supplies specific information about the format of the printed page and the organization of the subdivisions of the report. An identification code may be given in the report description entry so that each report may be identified separately in an intermediate output file.

Following each report description entry are one or more 01 level-number entries, each followed by a hierarchical structure similar to COBOL record descriptions. Each 01 level-number entry and its subordinate entries describes a report group. Each report group consists of one or more print lines that are regarded as a unit. A report group that is to be printed is printed entirely on one logical page; it is never split across pages.

REPORT STRUCTURE

When structuring a report, consideration must be given to vertical and horizontal spacing requirements, manipulation of data, and the physical and logical subdivisions of a report.

Vertical Spacing

The report writer feature allows the user to describe report groups containing multiple lines. The vertical positioning of the lines on a page is specified by the LINE NUMBER clause that is associated with each line. The NEXT GROUP clause indicates how many lines to space after presenting the last line of the group. The first LINE NUMBER clause of the next group indicates additional spacing information to be used in positioning of that group.

Horizontal Spacing

The report writer allows the user to position the fields of data on a report line by means of the COLUMN NUMBER clause. The report writer control system supplied space fill between all defined fields.

Data Manipulation

When the report writer feature is used, data movement to a report group is directed by Report Section clauses rather than Procedure Division statements. A data item in a report group is presented only if it has a COLUMN NUMBER clause specifying where it is to be presented. The value that is placed in a printable item is determined by the SOURCE, SUM, or VALUE clause stated in the report group description. Under no circumstances may a report group printable item receive a value directly via a Procedure Division statement.

Report Subdivisions

The physical and logical organization of a report interact to determine what is presented on a page.

Physical Subdivision of a Report

The PAGE clause specifies the length of the page, the size of the heading and footing areas, and the size of the area in which the detail lines will appear. The report writer control system uses the LINE NUMBER and NEXT GROUP clauses to position these report groups, and when necessary, to advance to a new page with automatic production of PAGE HEADING and PAGE FOOTING report groups.

Logical Subdivision of a Report

DETAIL report groups may be structured into a nested set of control groups. Each control group may begin with a CONTROL HEADING report group and end with a CONTROL FOOTING report group.

When nested control groups are defined, the recognition of a change in value of a control data item in a control hierarchy is called a control break and the heading and footing lines associated with the control data-name are called CONTROL HEADING and CONTROL FOOTING report groups.

During the execution of a GENERATE statement, the report writer control system uses the control hierarchy to check for control breaks. If a control break has occurred, all controls that are minor to it are considered to have changed, even though they may not in fact have changed. The occurrence of a control break causes the following sequence of events to take place:

1. All CONTROL FOOTING report groups are presented up to and including the one at the level at which the control break occurred.

2. All CONTROL HEADING report groups are presented from the control break level down to the most minor control.

3. The DETAIL report group named in the GENERATE statement is presented.

REPORT FILE

A report file is an text-file or printer output file having sequential organization. A report file has a file description entry containing a REPORT clause. The content of a report file consists of records that are written under control of the report writer control system (RWCS).

A report file is named by a file control entry and is described by a file description entry containing a REPORT clause. A report file is referred to and accessed by the OPEN, GENERATE, INITIATE, SUPPRESS, TERMINATE, USE BEFORE REPORTING, and CLOSE statements.

SPECIAL REGISTER PAGE-COUNTER

The reserved word PAGE-COUNTER is a name for a special register that is generated for each report description entry in the Report Section of the Data Division. The implicit description is that of an unsigned integer that must be capable of representing a range of values from 1 through 999999. The usage is WORD-BINARY. The value in PAGE-COUNTER is maintained by the report writer control system (RWCS) and is used by the program to number the page of a report. PAGE-COUNTER may be referenced only in the SOURCE clause of the Report Section and in Procedure Division statements. (See PAGE-COUNTER Rules.)

SPECIAL REGISTER LINE-COUNTER

The reserved word LINE-COUNTER is a name for a special register that is generated for each report description entry in the Report Section of the Data Division. The implicit description is that of an unsigned integer that must be capable of representing a range of values from 0 through 999999. The usage is WORD-BINARY. The value in LINE-COUNTER is maintained by the report writer control system (RWCS), and is used to determine the vertical positioning of a report. LINE-COUNTER may be referenced only in the SOURCE clause of the Report Section and in Procedure Division statements; however, only the RWCS may change the value of LINE-COUNTER. (See LINE-COUNTER Rules.)

SUBSCRIPTING

In the Report Section, neither a sum counter nor the special registers LINE-COUNTER and PAGE-COUNTER can be used as a subscript.

INPUT-OUTPUT SECTION

Information concerning the Input-Output Section is located in the IDENTIFICATION AND ENVIRONMENT DIVISIONS chapter.

THE FILE-CONTROL PARAGRAPH

Information concerning the FILE-CONTROL paragraph is located in the IDENTIFICATION AND ENVIRONMENT DIVISIONS chapter.

THE FILE CONTROL ENTRY

Function

The file control entry declares the relevant physical attributes of a report file.

General Format

```
SELECT [OPTIONAL] file-name-1

                                  {literal-1  }   [SYNC        ]
                  {TEXT-FILE      {           }   [            ]
                  {               {data-name-1}   [SYNCHRONIZED]
                  {SYSPRINT1                      [            ]
                  {
                  {SYSPRINT2
       ASSIGN TO  {
                  {SYSPRINT3
                  {
                  {SYSPRINT4
                  {
                  {WS-PRINTER

     [RESERVE integer-1 [AREA ]]
                        [AREAS]

     [[ORGANIZATION IS] SEQUENTIAL]

     [ACCESS MODE IS SEQUENTIAL]

     [FILE STATUS IS data-name-2].
```

Syntax Rules

1. The SELECT clause must be specified first in the file control entry. The clauses which follow the SELECT clause may appear in any order.

2. Each report file described in the Data Division must be specified only once in the FILE-CONTROL paragraph. Each report file specified in the SELECT clause must have a file description entry containing a REPORT clause in the Data Division of the same program.

3. Literal-1 must be a non-numeric literal other than a figurative constant.

4. Data-name-1 may be qualified.

5. Data-name-1 must reference an alphanumeric data item.

6. Literal-1 or the content of the data item referenced by data-name-1 at the time of execution of an OPEN statement referencing file-name-1 specifies the name of the file on the physical device. If the name contains a space character only the characters up until the first blank are considered to be part of the name. If the physical file

name starts with the character '/', then it is assumed to be a complete path-name with an allowed maximum length of 59 characters. Otherwise, literal-1 or the content of the data item referenced by data-name-1 is assumed to be a local name or the name of a file in the working directory, and may not contain the character '/'. In this case, the maximum length allowed is twelve characters.

7. When data-name-1 is specified, the above length restriction applies to the number of characters before the first blank character, if any, or the length of the data item referenced by data-name-1.

General Rules

1. If the file control block referenced by file-name-1 is an external file control block, all file control entries in the run unit which reference this file control block must have:

   a) The same specification for the OPTIONAL and SYNCHRONIZED phrases.

   b) A consistent specification for literal-1, that is a non-numeric literal containing the external file-name.

   c) The same external data item for data-name-1 in the ASSIGN clause.

   d) The same value for integer-1 in the RESERVE clause.

   e) The same organization.

   f) The same access mode.

2. The OPTIONAL phrase applies only to a report file opened in the extend mode. Its specification is required for a report file that is not necessarily present each time the object program is executed.

3. The ASSIGN clause specifies the association of the report file referenced by file-name-1 to a storage medium referenced by literal-1 or the content of the data item referenced by data-name-1, SYSPRINT1, SYSPRINT2, SYSPRINT3, SYSPRINT4, or WS-PRINTER.

4. A report file has sequential organization. Thus all clauses within the file control entry for a report file shown in the general format above are present within the Sequential I-O.

5. If literal-1 or data-name-1 is specified in the ASSIGN clause, the presence of the SYNC or SYNCHRONIZED phrase indicates that the output to the indicates file is synchronized, that is, the user program receives control when the output operation has terminated. The default is asynchronous output which means that the output operation is requested immediately, but control is returned to the COBOL program before operation is completed.

THE I-O-CONTROL PARAGRAPH

Function

The I-O-CONTROL paragraph specifies the memory area which is to be shared by different files and the location of files on a multiple file reel.

General Format

```
I-O-CONTROL.

    [[SAME AREA FOR file-name-1 {file-name-2} ...] ...

     [MULTIPLE FILE TAPE CONTAINS {file-name-3

           [POSITION IS integer-1]} ...] ... .]
```

Syntax Rules

1. The I-O-CONTROL paragraph is optional.
2. The MULTIPLE FILE clause is treated as a comment.
3. The SAME clause without the RECORD phrase is treated as a comment.

FILE SECTION

The File Section is located in the Data Division of a source program. The File Section defines the structure of report files. Each report file is defined by a file description entry having a REPORT clause. A file description entry for a report file is not followed by record description entries.

The general format of the File Section in the Report Writer is shown below.

FILE SECTION.

[file-description-entry] ...

In a COBOL program, the file description entry (FD entry) represents the highest level of organization in the File Section. The File Section header is followed by a file description entry consisting of a level indicator (FD), a file-name, and a series of independent clauses. For a report file, the file description entry must contain the REPORT clause specifying the names of the reports to be written onto the report file. No record description entries may follow the file description entry for a report file.

THE FILE DESCRIPTION ENTRY

Function

The file description entry furnishes information concerning the physical structure, identification, and report-names pertaining to a report file.

General Format

```
FD  file-name-1

    [ BLOCK CONTAINS [integer-1 TO] integer-2 { RECORDS    } ]
                                              { CHARACTERS }

    [RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

    [ LABEL { RECORD IS  } { STANDARD } ]
            { RECORDS ARE} { OMITTED  }

    [ VALUE OF { { ALLOC-UNIT } IS { integer-5  } } ... ]
               { { FILE-SIZE  }   { data-name-1 } }

    [ DATA { RECORD IS  } {data-name-2} ... ]
           { RECORDS ARE}

    [CODE-SET IS alphabet-name-1]

    { REPORT IS  }
    { REPORTS ARE}  {report-name-1} ... .
```

Syntax Rules

1. The level indicator FD identifies the beginning of a file description entry for a report file and must precede the file-name of the report file.

2. The clauses which follow file-name-1 may appear in any order.

3. File-name-1 may only reference a sequential file.

4. No record description entries may follow the file description entry.

5. The subject of a file description entry that specifies a REPORT clause may be referenced in the Procedure Division only by the USE statement, the CLOSE statement, or the OPEN statement with the OUTPUT or EXTEND phrase.

General Rules

1. A file description entry associates file-name-1 with a file connector.

2. The report writer logical record structure of the file associated with file-name-1 is defined by the implementor.

3. With the exception of the REPORT clause, all clauses within the file description entry for a report file shown above are presented within the Sequential I-O.

REPORT clause

Function

The REPORT clause specifies the names of reports that comprise a report file.

General Format

```
REPORT IS

REPORTS ARE    {report-name-1} ...
```

Syntax Rules

1. Report-name-1 must be the subject of a report description entry in the Report Section. The order of appearance of the report-names is not significant.

2. Report-name-1 must appear in only one REPORT clause.

3. The subject of a file description entry that specifies a REPORT clause may be referenced in the Procedure Division only by the USE statement, the CLOSE statement, or the OPEN statement with the OUTPUT or EXTEND phrase.

General Rules

1. The presence of more than one report-name in a REPORT clause indicates that the file contains more than one report.

2. After execution of an INITIATE statement and before the execution of a TERMINATE statement for the same report file, the report file is under the control of the report writer control system (RWCS). While a report file is under the control of the RWCS, no input-output statement may be executed which references that report file.

3. If the associated file connector is an external file connector, every file description entry in the run unit which is associated with that file connector must describe it as a report file.

REPORT SECTION

The Report Section is located in the Data Division of a source program. The Report Section describes the reports to be written onto report files. The description of each report must begin with a report description entry (RD entry) and be followed by one or more report group description entries.

The general format of the Report Section is shown below.

REPORT SECTION.

[report-description-entry {report-group-description-entry} ...] ...

REPORT Description Entry

In addition to naming the report, the report description entry (RD entry) defines the format of each page of the report by specifying the vertical boundaries of the region within which each type of report group may be printed. The report description entry also specifies the control data items. When the report is produced, changes in the values of the control data items cause the detail information of the report to be processed in groups called control groups.

Each report named in the REPORTS clause of an file description entry in the File Section must be the subject of an report description entry in the Report Section. Furthermore each report in the Report Section must be named in one and only one file description entry.

REPORT GROUP Description Entry

The report groups that will comprise the report are described following the report description entry. The description of each report group begins with a report group description entry; that is an entry that has a 01 level-number and a TYPE clause. Subordinate to the report group description entry, there may appear group and elementary entries that further describe the characteristics of the report group.

THE REPORT DESCRIPTION ENTRY

Function

The report description entry names a report, specifies any identifying characters to be appended to each print line, and describes the physical structure and organization of that report.

General Format

```
RD  report-name-1

    [CODE literal-1]

[ {CONTROL IS  }   {{data-name-1} ...          } ]
[ {CONTROLS ARE}   {FINAL [data-name-1] ...    } ]

[ PAGE [LIMIT IS  ] integer-1 [LINE  ] ]
       [LIMITS ARE]           [LINES ]

    [HEADING integer-2]  [FIRST DETAIL integer-3]

    [LAST DETAIL integer-4] [FOOTING integer-5] ]
```

Syntax Rules

1. Report-name-1 must appear in one and only one REPORT clause.

2. The order of appearance of the clauses following report-name-1 is immaterial.

3. Report-name-1 is the highest permissible qualifier that may be specified for LINE-COUNTER, PAGE-COUNTER and all data-names defined within the Report Section.

General Rules

1. The CODE clause, the CONTROL clause, and the PAGE clause are presented in alphabetical order.

PAGE-COUNTER Rules

1. PAGE-COUNTER is the reserved word to reference a special register that is created for each report specified in the Report Section. (See Special Register PAGE-COUNTER).

2. In the Report Section, a reference to PAGE-COUNTER can only appear in a SOURCE clause. In the Procedure Division, PAGE-COUNTER may be used in any context in which a data-name with an integer value can appear.

3. If more than one PAGE-COUNTER exists in a program, PAGE-COUNTER must be qualified by a report-name whenever it is referenced in the Procedure Division.

   In the Report Section an unqualified reference to PAGE-COUNTER is qualified implicitly by the name of the report in whose report description entry the reference is made. Whenever the PAGE-COUNTER of a different report is referenced, PAGE-COUNTER must be explicitly qualified by the report-name associated with the different report.

4. Execution of the INITIATE statement causes the report writer control system to set the PAGE-COUNTER of the referenced report to one.

5. PAGE-COUNTER is incremented by one each time the report writer control system executes a page advance.

6. PAGE-COUNTER may be altered by Procedure Division statements.

LINE-COUNTER Rules

1. LINE-COUNTER is the reserved word used to reference a special register that is created for each report specified in the Report Section. (See Special Register LINE-COUNTER).

2. In the Report Section a reference to LINE-COUNTER can only appear in a SOURCE clause. In the Procedure Division, LINE-COUNTER may be used in any context in which a data-name with an integer value may appear. However, only the report writer control system can change the content of LINE-COUNTER.

3. If more than one LINE-COUNTER exists in a program, LINE-COUNTER must be qualified by a report-name whenever it is referenced in the Procedure Division.

   In the Report Section an unqualified reference to LINE-COUNTER is implicitly qualified by the name of the report in which the reference is made. Whenever the LINE-COUNTER of a different report is referenced, LINE-COUNTER must be explicitly qualified by that report-name.

4. Execution of an INITIATE statement causes the report writer control system to set the LINE-COUNTER of the referenced report to zero. The report writer control system also resets LINE-COUNTER to zero each time it executes a page advance.

5. The value of LINE-COUNTER is not affected by the processing of nonprintable report groups nor by the processing of a printable report group whose printing is suppressed by means of the SUPPRESS statement.

6. At the time each print line is presented, the value of LINE-COUNTER represents the line number on which the print line is presented. The value of LINE-COUNTER after the presentation of a report group is governed by the presentation rules for the report group. (See Presentation Rules Tables.)

Function

The CODE clause specifies a two-character literal that identifies each print line as belonging to a specific report.

General Format

CODE literal-1

Syntax Rules

1. Literal-1 must be a two-character nonnumeric literal.

2. If the CODE clause is specified for any report in a file, it must be specified for all reports in that file.

General Rules

1. When the CODE clause is specified, literal-1 is placed in the first two character positions of each report writer logical record.

2. The positions occupied by literal-1 are not included in the description of the print line, but are included in the logical record size.

CONTROL clause

Function

The CONTROL clause establishes the levels of the control hierarchy for the report.

General Format

```
{ CONTROL IS  }  { {data-name-1} ...          }
{ CONTROLS ARE}  { FINAL [data-name-1] ...    }
```

Syntax Rules

1. Data-name-1 must not be defined in the Report Section. Data-name-1 may be qualified.

2. Each recurrence of data-name-1 must identify a different data item.

3. Data-name-1 must not have subordinate to it a variable occurrence data item.

General Rules

1. Data-name-1 and the word FINAL specify the levels of the control hierarchy. FINAL, if specified, is the highest control, data-name-1 is the major control, the next recurrence of data-name-1 is an intermediate control, etc. The last recurrence of data-name-1 is the minor control.

2. The execution of the chronologically first GENERATE statement for a given report causes the report writer control system (RWCS) to save the values of all control data items associated with that report. On subsequent executions of all GENERATE statements for that report, control data items are tested by the RWCS for a change of value. A change of value in any control data item causes a control break to occur. This control break is associated with the highest level for which a change of value is noted. (See The GENERATE Statement.)

3. The report writer control system (RWCS) tests for a control break by comparing the content of each control data item with the prior content of each control data item that was saved when the previous GENERATE statement for the same report was executed. The RWCS applies the inequality relation test as follows:

   a) If the control data item is a numeric data item, the relation test is for the comparison of two numeric operands.

   b) If the control data item is an index data item, the relation test is for the comparison of two index data items.

c) If the control data item is a data item other than as described in paragraph 3a and 3b, the relation test is for the comparison of two nonnumeric operands.

The inequality relation test is further explained in the appropriate paragraph, see Relation Condition.

4. FINAL is used when the most inclusive control group in the report is not associated with a control data-name.

PAGE clause

Function

The PAGE clause defines the length of a page and the vertical subdivisions within which report groups are presented.

General Format

```
PAGE [LIMIT IS  ] integer-1 [LINE ]
     [LIMITS ARE]           [LINES]

    [HEADING integer-2]  [FIRST DETAIL integer-3]

    [LAST DETAIL integer-4] [FOOTING integer-5]
```

Syntax Rules

1. The HEADING, FIRST DETAIL, LAST DETAIL and FOOTING phrases may be written in any order.

2. Integer-1 must not exceed three significant digits in length and must be greater than or equal to integer-5.

3. Integer-2 must be greater than or equal to one.

4. Integer-3 must be greater than or equal to integer-2.

5. Integer-4 must be greater than or equal to integer-3.

6. Integer-5 must be greater than or equal to integer-4.

7. The following rules indicate the vertical subdivision of the page in which each type of report group may appear when the PAGE clause is specified. (See Page Regions).

   a) A report heading report group that is to be presented on a page by itself, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-2 to the line number specified by integer-1, inclusive.

      A report heading report group that is not to be presented on a page by itself, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-2 to the line number specified by integer-3 minus 1, inclusive.

   b) A page heading report group, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-2 to the

line number specified by integer-3 minus 1, inclusive.

c) A control heading or detail report group, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-3 to the line number specified by integer-4, inclusive.

d) A control footing report group, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-3 to the line number specified by integer-5, inclusive.

e) A page footing report group, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-5 plus 1 to the line number specified by integer-1, inclusive.

f) A report footing report group that is to be presented on a page by itself, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-2 to the line number specified by integer-1, inclusive.

A report footing report group that is not to be presented on a page by itself, if defined, must be defined such that it can be presented in the vertical subdivision of the page that extends from the line number specified by integer-5 plus 1 to the line number specified by integer-1, inclusive.

8. All report groups must be described such that they can be presented on one page. The report writer control system (RWCS) never splits a multi-line report group across page boundaries.

General Rules

1. The vertical format of a report page is established using the integer values specified in the PAGE clause.

a) Integer-1 defines the size of a report page by specifying the number of lines available on each page.

b) HEADING integer-2 defines the first line number on which a report heading or page heading report group may be presented.

c) FIRST DETAIL integer-3 defines the first line number on which a body group may be presented. Report heading and page heading report groups may not be presented on or beyond the line number specified by integer-3.

d) LAST DETAIL integer-4 defines the last line number on which a control heading or detail report group may be presented.

e) FOOTING integer-5 defines the last line number on which a control footing report group may be presented. Page footing and report footing report groups must follow the line number specified by integer-5.

2. If the PAGE clause is specified the following implicit values are assumed for any omitted phrases:

   a) If the HEADING phrase is omitted, a value of one is assumed for integer-2.

   b) If the FIRST DETAIL phrase is omitted, a value equal to integer-2 is given to integer-3.

   c) If the LAST DETAIL and the FOOTING phrases are both omitted, the value of integer-1 is given to both integer-4 and integer-5.

   d) If the FOOTING phrase is specified and the LAST DETAIL phrase is omitted, the value of integer-5 is given to integer-4.

   e) If the LAST DETAIL phrase is specified and the FOOTING phrase is omitted, the value of integer-4 is given to integer-5.

3. If the PAGE clause is omitted, the report consists of a single page of indefinite length.

4. The presentation rules for each type of report group are specified in the appropriate paragraph, see Presentation Rules Tables.

PAGE REGIONS

Page regions that are established by the PAGE clause are described below.

Report Groups That May Be Presented in the Region	First Line Num. of the Region	Last Line Num. of the Region
Report heading described with NEXT GROUP NEXT PAGE Report footing described with LINE integer-1 NEXT PAGE	integer-2	integer-1
Report heading not described with NEXT GROUP NEXT PAGE Page heading	integer-2	integer-3 minus 1
Control heading Detail	integer-3	integer-4
Control footing	integer-3	integer-5
Page Footing Report footing not described with LINE integer-1 NEXT PAGE	integer-5 plus 1	integer-1

THE REPORT GROUP DESCRIPTION ENTRY

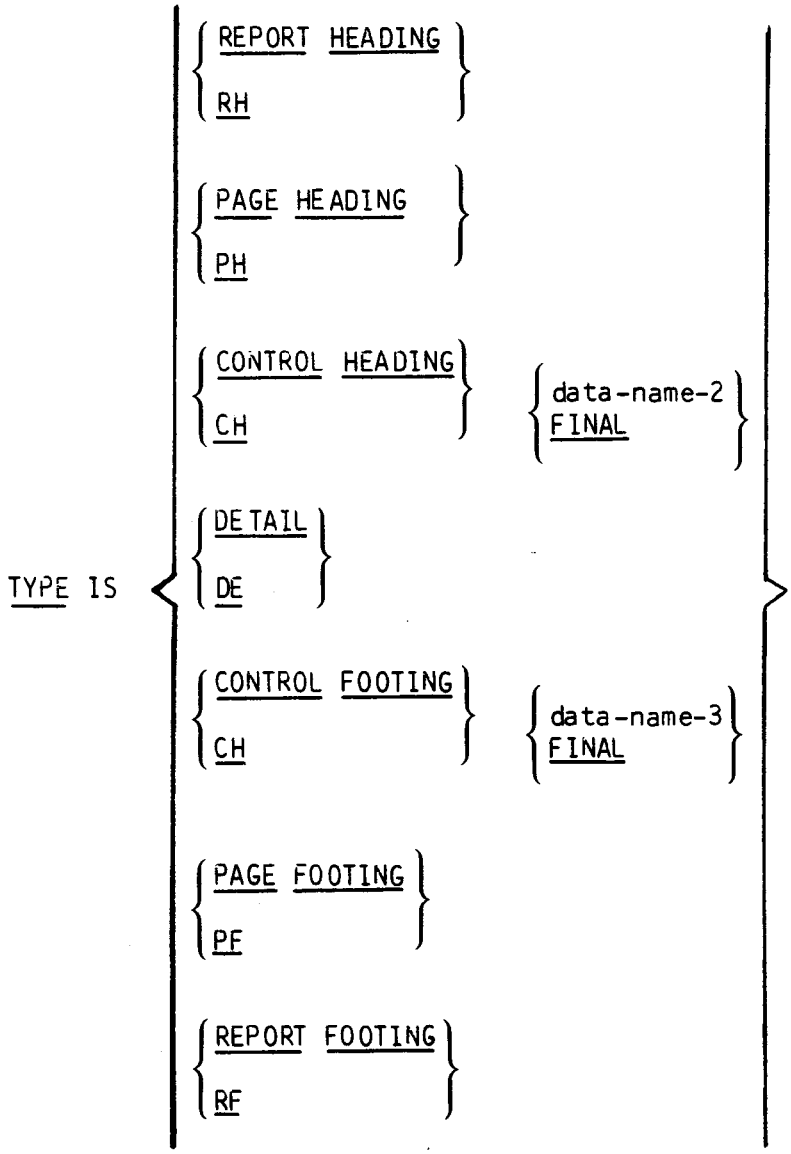
Function

The report group description entry specifies the characteristics of a report group and of the individual items within a report group.

General Format

Format 1:

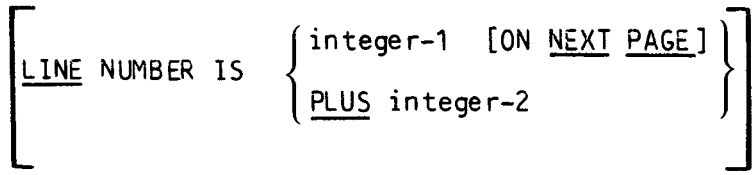
```
01 [data-name-1]

    [ LINE NUMBER IS { integer-1  [ON NEXT PAGE] } ]
                     { PLUS integer-2             }

    [ NEXT GROUP IS { integer-3          } ]
                    { PLUS integer-4     }
                    { NEXT PAGE          }
```



[[USAGE IS] DISPLAY]

Format 2:

level-number [data-name-1]



[[USAGE IS] DISPLAY]

Format 3:

```
level-number  [data-name-1] .

    { PICTURE }
    { PIC     } IS character-string

    [[USAGE IS]  DISPLAY]

    [                  { LEADING  }                          ]
    [SIGN IS]          { TRAILING }  SEPARATE CHARACTER       ]

    [ { JUSTIFIED } RIGHT ]
    [ { JUST      }       ]

    [BLANK WHEN ZERO]

    [ LINE NUMBER IS { integer-1  [ON NEXT PAGE] } ]
                     { PLUS integer-2             }

    [COLUMN NUMBER IS integer-3]

    { SOURCE IS identifier-1                                          }
    { VALUE IS literal-1                                              }
    { {SUM  {identifier-2} ... [UPON {data-name-2} ... ]} ...         }
    {     [ RESET ON { data-name-3 } ]                                }
    {                { FINAL       }                                  }

    [GROUP INDICATE] .
```

Syntax Rules

1. The report group description entry can appear only in the Report Section.

2. Except for the data-name clause, which when present must immediately follow the level-number, the clauses may be written in any sequence.

3. In format 2 the level-number may be any integer from 02 to 48 inclusive. In format 3 the level-number may be any integer from 02 to 49 inclusive.

4. The description of a report group may consist of one, two, or three hierarchic levels:

   a) The first entry that describes a report group must be a format 1 entry.

   b) Both format 2 and format 3 entries may be immediately subordinate to a format 1 entry.

   c) At least one format 3 entry must be immediately subordinate to a format 2 entry.

   d) Format 3 entries must define elementary items.

5. In a format 1 entry, data-name-1 is required only when:

   a) A detail report group is referenced by a GENERATE statement.

   b) A detail report group is referenced by the UPON phrase of a SUM clause.

   c) A report group is referenced in a USE BEFORE REPORTING sentence.

   d) The name of a control footing report group is used to qualify a reference to a sum counter.

   If specified, data-name-1 may be referenced only by a GENERATE statement, the UPON phrase of a SUM clause, a USE BEFORE REPORTING sentence, or as a sum counter qualifier.

6. A format 2 entry must contain at least one optional clause.

7. In a format 2 entry, data-name-1 is optional. If present it may be used only to qualify a sum counter reference.

8. In the Report Section, the USAGE clause is used only to declare the usage of printable items.

   a) If the USAGE clause appears in a format 3 entry, that entry must define a printable item.

   b) If the USAGE clause appears in a format 1 or format 2 entry, at least one subordinate entry must define a printable item.

9. An entry that contains a LINE NUMBER clause must not have a subordinate entry that also contains a LINE NUMBER clause.

10. In format 3:

    a) A GROUP INDICATE clause may appear only in a type detail report group.

    b) A SUM clause may appear only in a type control footing report group.

    c) An entry that contains a COLUMN NUMBER clause but no LINE NUMBER clause must be subordinate to an entry that contains a LINE NUMBER clause.

    d) Data-name-1 is optional but may be specified in any entry. It may be referenced only if the entry defines a sum counter.

    e) A LINE NUMBER clause must not be the only clause specified.

    f) An entry that contains a VALUE clause must also have a COLUMN NUMBER clause.

11. The following table shows all permissible clause combinations for a format 3 entry. The table is read from left to right along the chosen row. An 'M' indicates that the presence of the clause is mandatory. A 'P' indicates that the presence of the clause is permitted, but not required. A blank indicates that the clause is not permitted.

PIC	COLUMN	SOURCE	SUM	VALUE	JUST	BLANK WHEN ZERO	GROUP INDICATE	USAGE	LINE
M			M						P
M	M		M			P		P	P
M	P	M			P		P	P	P
M	P	M				P	P	P	P
M	M			M	P		P	P	P

Permissible Clause Combinations in Format 3 Entries

General Rules

1. Format 1 is the report group entry. The report group is defined by the contents of this entry and all of its subordinate entries.

2. The rules for the BLANK WHEN ZERO clause, the JUSTIFIED clause, and the PICTURE clause for the Report Writer are the same as those in the DATA DIVISION chapter. The other clauses of the report group description entry are presented in alphabetical order.

PRESENTATION RULES TABLES

Description

The tables and rules on the following pages specify:

1. The permissible combinations of LINE NUMBER and NEXT GROUP clauses for each type of report group,

2. The requirements that are placed on the use of these clauses, and

3. The interpretation that the report writer control system (RWCS) gives to these clauses.

Organization

There is an individual presentation rules table for each of the following types of report groups: report heading, page heading, page footing, report footing. In addition, detail report groups, control heading report groups, and control footing report groups are treated jointly in the body group presentation rules table. (See The Body Group Presentation Rules.)

Columns 1 and 2 of a presentation rules table list all of the permissible combinations of LINE NUMBER and NEXT GROUP clauses for the designated report group type. Consequently, for the purpose of identifying the set of presentation rules that apply to a particular combination of LINE NUMBER and NEXT GROUP clauses, a presentation rules table is read from left to right, along the selected row.

The applicable rules columns of a presentation rules table are partitioned into two parts. The first part specifies the rules that apply if the report description contains a PAGE clause, and the second part specifies the rules that apply if the PAGE clause is omitted. The purpose of the rules named in the applicable rules columns is discussed below:

1. Upper limit rules and lower limit rules:

   These rules specify the vertical subdivisions of the page within which the specified report group may be presented.

   In the absence of a PAGE clause the printed report is not considered to be partitioned into vertical subdivisions. Consequently, within the tables no upper limit rule and lower limit rule is specified for a report description in which the PAGE clause is omitted.

2. Fit test rules:

   The fit test rules are applicable only to body groups, and hence fit test rules are specified only within the body group presentation rules table. At object time the report writer control system (RWCS) applies the fit test rules to determine whether the designated body group can be presented on the page to which the report is currently positioned.

However, even for body groups there are no fit test rules when the PAGE clause is omitted from the report description entry.

3. First print line position rules:

   The first print line position rules specify where on the report medium the report writer control system (RWCS) shall present the first print line of the given report group.

   The presentation rules tables do not specify where on the report medium the report writer control system (RWCS) shall present the second and subsequent print lines (if any) of a report group. Certain general rules determine where the second and subsequent print lines of a report group shall be presented. Refer to the LINE NUMBER clause general rules for this information. (See The LINE NUMBER Clause.)

4. Next group rules:

   The next group rules relate to the proper use of the NEXT GROUP clause.

5. Final LINE-COUNTER setting rules:

   The terminal values that the report writer control system (RWCS) places in LINE-COUNTER after presenting report groups are specified by the final LINE-COUNTER setting rules.

LINE NUMBER Clause Notation

Column 1 of the presentation rules table uses a shorthand notation to describe the sequence of LINE NUMBER clauses that may appear in the description of a report group. The meaning of the abbreviations used in column 1 is as follows:

1. The letter 'A' represents one or more absolute LINE NUMBER clauses, none of which have the NEXT PAGE phrase, that appear in consecutive order within the sequence of LINE NUMBER clauses in the report group description entry.

2. The letter 'R' represents one or more relative LINE NUMBER clauses that appear in consecutive order within the sequence of LINE NUMBER clauses in the report group description entry.

3. The letters 'NP' represent one or more absolute LINE NUMBER clauses that appear in consecutive order within the sequence of LINE NUMBER clauses in the report group description entry, with the phrase NEXT PAGE appearing in the first, and only in the first, LINE NUMBER clause.

When two abbreviations appear together, they refer to a sequence of LINE NUMBER clauses that consists of the two specified consecutive sequences. For example 'A R' refers to a report group description entry within which the 'A' sequence (defined in rule 1 above) is immediately followed by the 'R' sequence (defined in rule 2 above).

LINE NUMBER Clause Sequence Substitutions

Where 'A R' is shown to be a permissible sequence in the presentation rules table, 'A' is also permissible and the same presentation rules are applicable.

When 'NP R' is shown to be a permissible sequence in the presentation rules table, 'NP' is also permissible and the same presentation rules are applicable.

SAVED NEXT GROUP INTEGER Description

Saved next group integer is a data item that is addressable only by the report writer control system (RWCS). When an absolute NEXT GROUP clause specifies a vertical positioning value which cannot be accommodated on the current page, the RWCS stores that value in saved next group integer. After page advance processing, the RWCS positions the next body group using the value stored in saved next group integer.

REPORT HEADING GROUP PRESENTATION RULES

The table below points to the appropriate presentation rules for all permissible combinations of LINE NUMBER and NEXT GROUP clauses in a report heading report group.

**		Applicable Rules ***						
		If the PAGE clause is specified.					If PAGE is omitted.	
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	Upper Limit	Lower Limit	First Print Line Pos.	Next Group	Final LINE-COUNTER Setting	First Print Line Pos.	Final LINE-COUNTER Setting
A R	Abs	1	2a	3a	4a	5a	Illegal+	
A R	Rel	1	2a	3a	4b	5b	Illegal+	
A R	NEXT PAGE	1	2b	3a	4c	5c	Illegal+	
A R		1	2a	3a		5d	Illegal+	
R	Abs	1	2a	3b	4a	5a	Illegal++	
R	Rel	1	2a	3b	4b	5b	3d	5b
R	NEXT PAGE	1	2b	3b	4c	5c	Illegal++	
R		1	2a	3b		5d	3d	5d
				3c		5e	3c	5e

* See LINE NUMBER Clause Notation, for a description of the abbreviations used in column 1.

** A blank entry in column 1 or column 2 indicates that the named clause is totally absent from the report group description entry.

*** A blank entry in an applicable rules column indicates the absence of the named rule for the given combination of LINE NUMBER and NEXT GROUP clauses.

\+ See the LINE NUMBER clause.

++ See the NEXT GROUP clause.

The report heading group presentation rules are as follows:

1. Upper limit rule:

   The first line number on which the report heading report group can be presented is the line number specified by the HEADING phrase of the PAGE clause.

2. Lower limit rules:

   a) The last line number on which the report heading report group can be presented is the line number that is obtained by subtracting 1 from the value of integer-3 of the FIRST DETAIL phrase of the PAGE clause.

   b) The last line number on which the report heading report group can be presented is the line number specified by integer-1 of the PAGE clause.

3. First print line position rules:

   a) The first print line of the report heading report group is presented on the line number specified by the integer of its LINE NUMBER clause.

   b) The first print line of the report heading report group is presented on the line number obtained by adding the integer of the first LINE NUMBER clause and the value obtained by subtracting 1 from the value of integer-2 of the HEADING phrase of the PAGE clause.

   c) The report heading report group is not presented.

   d) The first print line of the report heading report group is presented on the line number obtained by adding the content of its LINE-COUNTER (in this case, zero) to the integer of the first LINE NUMBER clause.

4. Next group rules:

   a) The NEXT GROUP integer must be greater than the line number on which the final print line of the report heading report group is presented. In addition, the NEXT GROUP integer must be less than the line number specified by the value of integer-3 of the FIRST DETAIL phrase of the PAGE clause.

   b) The sum of the NEXT GROUP integer and the line number on which the final print line of the report heading report group is presented must be less than the value of integer-3 of the FIRST DETAIL phrase of the PAGE clause.

   c) NEXT GROUP NEXT PAGE signifies that the report heading report group is to be presented entirely by itself on the first page of the report. The report writer control system (RWCS) processes no other report group while positioned to the first page of the report.

5. Final LINE-COUNTER setting rules:

   a) After the report heading report group is presented, the RWCS places the NEXT GROUP integer into LINE-COUNTER as the final LINE-COUNTER setting.

   b) After the report heading report group is presented, the RWCS places the sum of the NEXT GROUP integer and the line number on which the final print line of the report heading report group was presented into LINE-COUNTER as the final LINE-COUNTER setting.

   c) After the report heading report group is presented, the RWCS places zero into LINE-COUNTER as the final LINE-COUNTER setting.

   d) After the report heading report group is presented, the final LINE-COUNTER setting is the line number on which the final print line of the report heading report group was presented.

   e) LINE-COUNTER is unaffected by the processing of a nonprintable report group.

PAGE HEADING GROUP PRESENTATION RULES

THE table below points to the appropriate presentation rules for all permissible combinations of LINE NUMBER and NEXT GROUP clauses in a page heading report group.

**		Applicable Rules ***				
		If the PAGE clause is specified ****				
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	Upper Limit	Lower Limit	First Print Line Position	Next Group	Final LINE-COUNTER Setting
A R		1	2	3a		4a
R		1	2	3b		4a
				3c		4b

* See LINE NUMBER Clause Notation, for a description of the abbreviations used in column 1.

** A blank entry in column 1 or column 2 indicates that the named clause is totally absent from the report group description entry.

*** A blank entry in an applicable rules column indicates the absence of the named rule for the given combination of LINE NUMBER and NEXT GROUP clauses.

**** If the PAGE clause is omitted from the report description entry then a page heading report group may not be defined. (See the TYPE clause.)

The page heading group presentation rules are as follows:

1. Upper limit rule:

   If a report heading report group has been presented on the page on which the page heading report group is to be presented, then the first line number on which the page heading report group can be presented is one greater than the final LINE-COUNTER setting established by the REPORT HEADING.

   Otherwise the first line number on which the page heading report group can be presented is the line number specified by the HEADING phrase of the PAGE clause.

2. Lower limit rule:

   The last line number on which the page heading report group can be presented is the line number that is obtained by subtracting one from the value of integer-3 of the FIRST DETAIL phrase of the PAGE clause.

3. First print line position rules:

   a) The first print line of the page heading report group is presented on the line number specified by the integer of its LINE NUMBER clause.

   b) If a report heading report group has been presented on the page on which the page heading report group is to be presented, then the sum of the final LINE-COUNTER setting established by the report heading report group and the integer of the first LINE NUMBER clause of the page heading report group defines the line number on which the first print line of the page heading report group is presented.

      Otherwise the sum of the integer of the first LINE NUMBER clause of the page heading report group and the value obtained by subtracting one from the value of integer-2 of the HEADING phrase of the PAGE clause defines the line number on which the first print line of the page heading report group is presented.

   c) The page heading report group is not presented.

4. Final LINE-COUNTER setting rules:

   a) The final LINE-COUNTER setting is the line number on which the final print line of the page heading report group was presented.

   b) LINE-COUNTER is unaffected by the processing of a nonprintable report group.

BODY GROUP PRESENTATION RULES

The table below points to the appropriate presentation rules for all permissible combinations of LINE NUMBER and NEXT GROUP clauses in control heading, detail and control footing report groups.

**		Applicable Rules ***							
		If the PAGE clause is specified.						If PAGE is omitted.	
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	UPPER LIMIT	LOWER LIMIT	FIRST TEST	First Print Line Pos.	NEXT GROUP	Final LINE-COUNTER Setting	First Print Line Pos.	Final LINE-COUNTER Setting
A R	Abs	1	2	3a	4a	5	6a	Illegal+	
A R	Rel	1	2	3a	4a		6b	Illegal+	
A R	NEXT PAGE	1	2	3a	4a		6c	Illegal+	
A R		1	2	3a	4a		6d	Illegal+	
R	Abs	1	2	3b	4b	5	6a	Illegal++	
R	Rel	1	2	3b	4b		6b	4d	6f
R	NEXT PAGE	1	2	3b	4b		6c	Illegal++	
R		1	2	3b	4b		6d	4d	6d
NP R	Abs	1	2	3c	4a	5	6a	Illegal+	
NP R	Rel.	1	2	3c	4a		6b	Illegal+	
NP R	NEXT PAGE	1	2	3c	4a		6c	Illegal+	
NP R		1	2	3c	4a		6d	Illegal+	
					4c		6e	4c	6e

* See LINE NUMBER Clause Notation, for a description of the abbreviations used in column 1.

** A blank entry in column 1 or column 2 indicates that the named clause is totally absent from the report group description entry.

*** A blank entry in an applicable rules column indicates the absence of the named rule for the given combination of LINE NUMBER and NEXT GROUP clauses.

+ See the LINE NUMBER clause.

++ See the NEXT GROUP clause.

The body group presentation rules are as follows:

1. Upper limit rule:

   The first line number on which a body group can be presented is the line number specified by the FIRST DETAIL phrase of the PAGE clause.

2. Lower limit rules:

   The last line number on which a control heading report group or detail report group can be presented is the line number specified by the LAST DETAIL phrase of the PAGE clause.

   The last line number on which a control footing report group can be presented is the line number specified by the FOOTING phrase of the PAGE clause.

3. Fit test rules:

   a) If the value in LINE-COUNTER is less than the integer of the first absolute LINE NUMBER clause, then the body group shall be presented on the page to which the report is currently positioned.

      Otherwise the report writer control system (RWCS) executes page advance processing. After the page heading report group (if defined) has been processed, the RWCS determines whether the saved next group integer location was set when the final body group was presented on the preceding page. (See final LINE-COUNTER setting rule 6a.) If the saved next group integer was not so set, the body group shall be presented on the page to which the report is currently positioned. If the saved next group integer was so set, the RWCS moves the saved next group integer into LINE-COUNTER, resets the saved next group integer to zero, and reapplies fit test rule 3a.

   b) If a body group has been presented on the page to which the report is currently positioned, the RWCS computes a trial sum in a work location. The trial sum is computed by adding the content of LINE-COUNTER to the integers of all LINE NUMBER clauses of the report group. If the trial sum is not greater than the body group's lower limit integer, then the report group is presented on the current page. If the trial sum exceeds the body group's lower limit integer, then the RWCS executes page advance processing. After the page heading report group (if defined) has been processed, the RWCS reapplies fit test rule 3b.

If no body group has yet been presented on the page to which the report is currently positioned, the RWCS determines whether the saved next group integer location was set when the final body group was presented on the preceding page. (See final LINE-COUNTER setting rule 6a.)

If the saved next group integer was not so set, the body group shall be presented on the page to which the report is currently positioned.

If the saved next group integer was so set, the RWCS moves the saved next group integer into LINE-COUNTER, resets the saved next group integer to zero, and computes a trial sum in a work location.

The trial sum is computed by adding the content of LINE-COUNTER to the integer one and the integers of all but the first LINE NUMBER clause of the body group. If the trial sum is not greater than the body group's lower limit integer, then the body group is presented on the current page. If the trial sum exceeds the body group's lower limit integer, then the RWCS executes page advance processing. After the page heading report group (if defined) has been processed, the RWCS presents the body group on that page.

c) If a body group has been presented on the page to which the report is currently positioned, the RWCS executes page advance processing. After the page heading report group (if defined) has been processed, the RWCS reapplies fit test rule 3c.

If no body group has yet been presented on the page to which the report is currently positioned, the RWCS determines whether the saved next group integer location was set when the final body group was presented on the preceding page. (See final LINE-COUNTER setting rule 6a.) If the saved next group integer was not so set, the body group shall be presented on the page to which the report is currently positioned. If the saved next group integer was so set, the RWCS moves the saved next group integer into LINE-COUNTER and resets the saved next group integer to zero. If then the value in LINE-COUNTER is less than the integer of the first absolute LINE NUMBER clause, the body group shall be presented on the page to which the report is currently positioned. Otherwise the RWCS executes page advance processing. After the page heading report group (if defined) has been processed, the RWCS presents the body group on that page.

4. First print line position rules:

a) The first print line of the body group is presented on the line number specified by the integer of its LINE NUMBER clause.

b) If the value in LINE-COUNTER is equal to or greater than the line number specified by the FIRST DETAIL phrase of the PAGE clause, and if no body group has previously been presented on the page to which the report is currently positioned, then the first print line of the current body group is presented on the line immediately following the line indicated by the value contained in

LINE-COUNTER.

If the value in LINE-COUNTER is equal to or greater than the line number specified by the FIRST DETAIL phrase of the PAGE clause, and if a body group has previously been presented on the page to which the report is currently positioned, then the first print line of the current body group is presented on the line that is obtained by adding the content of LINE-COUNTER and the integer of the first LINE NUMBER clause of the current body group.

If the value in LINE-COUNTER is less than the line number specified by the FIRST DETAIL phrase of the PAGE clause, then the first print line of the body group is presented on the line specified by the FIRST DETAIL phrase.

c) The body group is not presented.

d) The sum of the content of LINE-COUNTER and the integer of the first LINE NUMBER clause defines the line number on which the first print line is presented.

5. Next group rule:

The integer of the absolute NEXT GROUP clause must specify a line number that is not less than that specified in the FIRST DETAIL phrase of the PAGE clause, and that is not greater than that specified in the FOOTING phrase of the PAGE clause.

6. Final LINE-COUNTER setting rules:

a) If the body group that has just been presented is a control footing report group and if the control footing report group is not associated with the highest level at which the report writer control system (RWCS) detected a control break, then the final LINE-COUNTER setting is the line number on which the final print line of the control footing report group was presented.

For all other cases the RWCS makes a comparison of the line number on which the final print line of the body group was presented and the integer of the NEXT GROUP clause. If the former is less than the latter, then the RWCS places the NEXT GROUP integer into LINE-COUNTER as the final LINE-COUNTER setting. If the former is equal to or greater than the latter, then the RWCS places the line number specified by the FOOTING phrase of the PAGE clause into LINE-COUNTER as the final LINE-COUNTER setting; in addition the RWCS places the NEXT GROUP integer into the saved next group integer location.

b) If the body group that has just been presented is a control footing report group, and if the control footing report group is not associated with the highest level at which the RWCS detected a control break, then the final LINE-COUNTER setting is the line number on which the final print line of the control footing report group was presented.

For all other cases the RWCS computes a trial sum in a work

location. The trial sum is computed by adding the integer of the NEXT GROUP clause to the line number on which the final print line of the body group was presented. If the sum is less than the line number specified by the FOOTING phrase of the PAGE clause, then the RWCS places that sum into LINE-COUNTER as the final LINE-COUNTER setting. If the sum is equal to or greater than the line number specified by the FOOTING phrase of the PAGE clause, then the RWCS places the line number specified by the FOOTING phrase of the PAGE clause into LINE-COUNTER as the final LINE-COUNTER setting.

c) If the body group that has just been presented is a control footing report group, and if the control footing report group is not associated with the highest level at which the RWCS detected a control break, then the final LINE-COUNTER setting is the line number on which the final print line of the control footing report group was presented.

   For all other cases the RWCS places the line number specified by the FOOTING phrase of the PAGE clause into LINE-COUNTER as the final LINE-COUNTER setting.

d) The final LINE-COUNTER setting is the line number on which the final print line of the body group was presented.

e) LINE-COUNTER is unaffected by the processing of a nonprintable body group.

f) If the body group that has just been presented is a control footing report group, and if the control footing report group is not associated with the highest level at which the RWCS detected a control break, then the final LINE-COUNTER setting is the line number on which the final print line of the control footing report group was presented.

   For all other cases the RWCS places the sum of the line number on which the final print line was presented and the NEXT GROUP integer into LINE-COUNTER as the final LINE-COUNTER setting.

PAGE FOOTING PRESENTATION RULES

The table below points to the appropriate presentation rules for all permissible combinations of LINE NUMBER and NEXT GROUP clauses in a page footing report group.

**		Applicable Rules ***				
		If the PAGE clause is specified ****				
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	Upper Limit	Lower Limit	First Print Line Pos.	Next Group	Final LINE-COUNTER Setting
A R	Absolute	1	2	3a	4a	5a
A R	Relative	1	2	3a	4b	5b
A R		1	2	3a		5c
				3b		5d

* See LINE NUMBER Clause Notation, for a description of the abbreviations used in column 1.

** A blank entry in column 1 or column 2 indicates that the named clause is totally absent from the report group description entry.

*** A blank entry in an applicable rules column indicates the absence of the named rule for the given combination of LINE NUMBER and NEXT GROUP clauses.

**** If the PAGE clause is omitted from the report description entry, then page footing report group may not be defined. (See The TYPE Clause.)

The page footing presentation rules are as follows:

1. Upper limit rule:

   The first line number on which the page footing report group can be presented, is the line number obtained by adding one to the value of integer-5 of the FOOTING phrase of the PAGE clause.

2. Lower limit rule:

   The last line number on which the page footing report group can be presented is the line number specified by integer-1 of the PAGE clause.

3. First print line position rules:

   a) The first print line of the page footing report group is presented on the line specified by the integer of its LINE NUMBER clause.

   b) The page footing report group is not presented.

4. Next group rules:

   a) The NEXT GROUP integer must be greater than the line number on which the final print line of the page footing report group is presented. In addition, the NEXT GROUP integer must not be greater than the line number specified by integer-1 of the PAGE clause.

   b) The sum of the NEXT GROUP integer and the line number on which the final print line of the page footing report group is presented must not be greater than the line number specified by integer-1 of the PAGE clause.

5. Final LINE-COUNTER setting rules:

   a) After the page footing report group is presented, the RWCS places the NEXT GROUP integer into LINE-COUNTER as the final LINE-COUNTER setting.

   b) After the page footing report group is presented, the RWCS places the sum of the NEXT GROUP integer and the line number on which the final print line of the page footing report group was presented into LINE-COUNTER as the final LINE-COUNTER setting.

   c) After the page footing report group is presented the final LINE-COUNTER setting is the line number on which the final print line of the page footing report group was presented.

   d) LINE-COUNTER is unaffected by the processing of a nonprintable report group.

REPORT FOOTING PRESENTATION RULES

The table below points to the appropriate presentation rules for all permissible combinations of LINE NUMBER and NEXT GROUP clauses in a report footing report group.

**		Applicable Rules ***						
		If the PAGE clause is specified.					If the PAGE clause is omitted.	
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	Upper Limit	Lower Limit	First Print Line Pos.	Next Group	Final LINE-COUNTER Setting	First Print Line Pos.	Final LINE-COUNTER Setting
A R		1a	2	3a		4a	Illegal+	
R		1a	2	3b		4a	3d	4a
NP R		1b	2	3c		4a	Illegal+	
				3e		4b	3e	4b

* See LINE NUMBER Clause Notation, for a description of the abbreviations used in column 1.

** A blank entry in column 1 or column 2 indicates that the named clause is totally absent from the report group description entry.

*** A blank entry in an applicable rules column indicates the absence of the named rule for the given combination of LINE NUMBER and NEXT GROUP clauses.

\+ See the LINE NUMBER clause.

The report footing presentation rules are as follows:

1. Upper limit rules:

   a) If a page footing report group has been presented on the page to which the report is currently positioned, then the first line number on which the report footing report group can be presented is one greater than the final LINE-COUNTER setting established by the page footing report group.

      Otherwise the first line number on which the report footing report group can be presented is the line number obtained by adding one and the value of integer-5 of the PAGE clause.

b) The first line number on which the report footing report group can be presented, is the line number specified by the HEADING phrase of the PAGE clause.

2. Lower limit rule:

The last line number on which the report footing report group can be presented is the line number specified by integer-1 of the PAGE clause.

3. First print line position rules:

a) The first print line of the report footing report group is presented on the line specified by the integer of its LINE NUMBER clause.

b) If a page footing report group has been presented on the page to which the report is currently positioned, then the sum of the final LINE-COUNTER setting established by the page footing report group and the integer of the first LINE NUMBER clause of the report footing report group defines the line number on which the first print line of the report footing report group is presented. Otherwise the sum of the integer of the first LINE NUMBER clause of the report footing report group, and the line number specified by the value of integer-5 of the FOOTING phrase of the PAGE clause defines the line number on which the first print line of the report footing report group is presented.

c) The NEXT PAGE phrase in the first absolute LINE NUMBER clause directs that the report footing report group is presented on a page on which no other report group has been presented. The first print line of the report footing report group is presented on the line number specified by the integer of its LINE NUMBER clause.

d) The sum of the content of LINE-COUNTER and the integer of the first LINE NUMBER clause defines the line number on which the first print line is presented.

e) The report footing report group is not presented.

4. Final LINE-COUNTER setting rules:

a) The final LINE-COUNTER setting is the line number on which the final print line of the report footing report group is presented.

b) LINE-COUNTER is unaffected by the processing of a nonprintable report group.

Function

The COLUMN NUMBER clause identifies a printable item and specifies the position of the item on a print line.

General Format

COLUMN NUMBER IS integer-1

Syntax Rules

1. The COLUMN NUMBER clause can only be specified at the elementary level within a report group. The COLUMN NUMBER clause, if present, must appear in or be subordinate to an entry that contains a LINE NUMBER clause.

2. Within a given print line, the printable items must be defined in ascending column number order such that each character defined occupies a unique position.

General Rules

1. The COLUMN NUMBER clause indicates that the object of a SOURCE clause or the object of a VALUE clause or the sum counter defined by a SUM clause is to be presented on the print line. The absence of a COLUMN NUMBER clause indicates that the entry is not to be presented on a print line.

2. Integer-1 specifies the column number of the leftmost character position of the printable item.

3. The report writer control system (RWCS) supplies space characters for all positions of a print line that are not occupied by printable items.

4. The first position of the print line is considered to be column number 1.

DATA-NAME clause

Function

A data-name specifies the name of the data being described.

General Format

data-name-1

Syntax Rules

1. In the Report Section data-name-1 need not appear in a data description entry.

General Rules

1. In the Report Section, data-name-1 must be given in the following cases:

   a) When data-name-1 represents a report group to be referred to by a GENERATE or a USE statement in the Procedure Division.

   b) When reference is to be made to the sum counter in the Procedure Division or Report Section.

   c) When a detail report group is referenced in the UPON phrase of the SUM clause.

   d) When data-name-1 is required to provide sum counter qualification.

Function

The GROUP INDICATE clause specifies that the associated printable item is presented only on the first occurrence of its report group after a control break or page advance.

General Format

GROUP INDICATE

Syntax Rules

1. The GROUP INDICATE clause must only be specified in a detail report group entry that defines a printable item.

General Rules

1.. If a GROUP INDICATE clause is specified, it causes the SOURCE or VALUE clause to be ignored and spaces supplied, except:

   a) On the first presentation of the detail report group in the report, or

   b) On the first presentation of the detail report group after every page advance, or

   c) On the first presentation of the detail report group after every control break.

2. If the report description entry specifies neither a PAGE clause nor a CONTROL clause, then a GROUP INDICATE printable item is presented the first time its DETAIL is presented after the INITIATE statement is executed. Thereafter spaces are supplied for indicated items with SOURCE or VALUE clauses.

LEVEL-NUMBER description

Function

The level-number indicates the position of a data item within the hierarchical structure of a report group.

General Format

level-number

Syntax Rules

1. A level-number is required as the first element in each data description entry.

2. Data description entries subordinate to an RD entry must have level-numbers 01 through 49 only.

General Rules

1. The level-number 01 identifies the first entry in a report group.

2. Multiple level 01 entries subordinate to a report description entry having the level indicator RD do not represent implicit redefinitions of the same area.

Function

The LINE NUMBER clause specifies vertical positioning information for its report group.

General Format

```
LINE NUMBER IS { integer-1  [ON NEXT PAGE] }
               { PLUS integer-2            }
```

Syntax Rules

1. Integer-1 and integer-2 must not exceed three significant digits in length.

   Neither integer-1 nor integer-2 may be specified in such a way as to cause any line of a report group to be presented outside of the vertical subdivision of the page designated for the report group type, as defined by the PAGE clause. (See The PAGE Clause.)

   Integer-2 may be zero.

2. Within a given report group description entry, an entry that contains a LINE NUMBER clause must not contain a subordinate entry that also contains a LINE NUMBER clause.

3. Within a given report group description entry, all absolute LINE NUMBER clauses must precede all relative LINE NUMBER clauses.

4. Within a given report group description entry, successive absolute LINE NUMBER clauses must specify integers that are in ascending order.

5. If the PAGE clause is omitted from a given report description entry, only relative LINE NUMBER clauses can be specified in any report group description entry within that report.

6. Within a given report group description entry a NEXT PAGE phrase can appear only once and, if present, must be in the first LINE NUMBER clause in that report group description entry.

7. A LINE NUMBER clause with the NEXT PAGE phrase can appear only in the description of body groups and in a report footing report group.

8. Every entry that defines a printable item (see The COLUMN NUMBER Clause) must either contain a LINE NUMBER clause, or be subordinate to an entry that contains a LINE NUMBER clause.

9. The first LINE NUMBER clause specified within a page footing report group must be an absolute LINE NUMBER clause.

General Rules

1. A LINE NUMBER clause must be specified to establish each print line of a report group.

2. The report writer control system (RWCS) effects the vertical positioning specified by a LINE NUMBER clause, before presenting the print line established by that LINE NUMBER clause.

3. Integer-1 specifies an absolute line number. An absolute line number specifies the line number on which the print line is presented.

4. Integer-2 specifies a relative line number. If a relative LINE NUMBER clause is not the first LINE NUMBER clause in the report group description entry, then the line number on which its print line is presented is determined by calculating the sum of the line number on which the previous print line of the report group was presented and integer-2 of the relative LINE NUMBER clause.

   If a relative LINE NUMBER clause is the first LINE NUMBER clause in the report group description entry, then the line number on which its print line is presented is determined by specified rules. (See Presentation Rules Tables.)

5. The NEXT PAGE phrase specifies that the report group is to be presented beginning on the indicated line number on a new page. (See Presentation Rules Tables.)

Function

The NEXT GROUP clause specifies information for vertical positioning of a page following the presentation of the last line of a report group.

General Format

```
NEXT GROUP IS { integer-1
              { PLUS integer-2
              {
              { NEXT PAGE
```

Syntax Rules

1. A report group entry must not contain a NEXT GROUP clause unless the description of that report group contains at least one LINE NUMBER clause.

2. Integer-1 and integer-2 must not exceed three significant digits in length.

3. If the PAGE clause is omitted from the report description entry only a relative NEXT GROUP clause may be specified in any report group description entry within that report.

4. The NEXT PAGE phrase of the NEXT GROUP clause must not be specified in a page footing report group.

5. The NEXT GROUP clause must not be specified in a report footing report group or in a page heading report group.

General Rules

1. Any positioning of the page specified by the NEXT GROUP clause takes place after the presentation of the report group in which the clause appears. (See Presentation Rules Tables.)

2. The vertical positioning information supplied by the NEXT GROUP clause is interpreted by the RWCS along with information from the TYPE and PAGE clauses, and the value in LINE-COUNTER, to determine a new value for LINE-COUNTER. (See Presentation Rules Tables.)

3. The NEXT GROUP clause is ignored by the RWCS when it is specified on a control footing report group that is at a level other than the highest level at which a control break is detected.

4. The NEXT GROUP clause of a body group refers to the next body group to be presented, and therefore can affect the location at which the next body group is presented. (See Presentation Rules Tables.)

5. The NEXT GROUP clause of a report footing report group can affect the location at which the page heading report group is presented. (See Presentation Rules Tables.)

6. The NEXT GROUP clause of a page footing report group can affect the location at which the report footing report group is presented. (See Presentation Rules Tables.)

Function

The SIGN clause specifies the position and the mode of representation of the operational sign when it is necessary to describe these properties explicitly.

General Format

```
[SIGN IS] { LEADING  } SEPARATE CHARACTER
          { TRAILING }
```

The following rules are in addition to or a limitation to the rules stated in the PROCEDURE DIVISION chapter.

Syntax Rules

1. When the SIGN clause is included in a report group description entry, the SEPARATE CHARACTER phrase must be specified.

General Rules

1. Since a SIGN clause in a report group description entry must specify the SEPARATE CHARACTER phrase, then:

   a) The operational sign will be presumed to be the leading (or respectively, trailing) character position of the elementary numeric data item; this character position is not a digit position.

   b) The letter 'S' in a PICTURE character-string is counted in determining the size of the item (in terms of standard data format characters).

   c) The operational signs for positive and negative are the standard data format characters '+' and '-', respectively.

SOURCE clause

Function

The SOURCE clause identifies the sending data item that is moved to an associated printable item defined within a report group description entry.

General Format

SOURCE IS identifier-1

Syntax Rules

1. Identifier-1 may be defined in any section of the Data Division. If identifier-1 is a Report Section item it can only be:

   a) PAGE-COUNTER, or

   b) LINE-COUNTER, or

   c) Sum counter that is part of the report within which the SOURCE clause appears.

2. Identifier-1 specifies the sending data item of the implicit MOVE statement that the report writer control system (RWCS) executes to move the content of the data item referenced by identifier-1 to the printable item. Identifier-1 must be defined such that it conforms to the rules for sending items in the MOVE statement. (See The MOVE Statement.)

General Rules

1. The RWCS formats the print lines of a report group just prior to presenting the report group. (See The TYPE Clause.) It is at this time that the implicit MOVE statements specified by SOURCE clauses are executed by the RWCS.

Function

The SUM clause establishes a sum counter and names the data items to be summed.

General Format

```
{SUM {identifier-1} ... [UPON {data-name-1} ... ] } ...

[RESET ON { data-name-2 }]
          { FINAL       }
```

Syntax Rules

1.. Identifier-1 must reference a numeric data item. If identifier-1 is defined in the Report Section, identifier-1 must reference a sum counter.

   If the UPON phrase is omitted, any identifiers in the associated SUM clause which are themselves sum counters must be defined either in the same report group that contains this SUM clause or in a report group which is at a lower level in the control hierarchy of this report.

   If the UPON phrase is specified, any identifiers in the associated SUM clause must not be sum counters.

2. Data-name-1 must be the name of a detail report group described in the same report as the control footing report group in which the SUM clause appears. Data-name-1 may be qualified by a report-name.

3. A SUM clause can appear only in the description of a control footing report group.

4. Data-name-2 must be one of the data-names specified in the CONTROL clause for this report. Data-name-2 must not be a lower level control than the associated control for the report group in which the RESET phrase appears.

   FINAL, if specified in the RESET phrase, must also appear in the CONTROL clause for this report.

5. The highest permissible qualifier of a sum counter is the report-name.

General Rules

1. The SUM clause establishes a sum counter. The sum counter is a numeric data item with an optional sign. At object time, the report writer control system adds into the sum counter the value in each

data item referenced by identifier-1. This addition is consistent with the rules for arithmetic statements. (See The Arithmetic Statements; and Overlapping Operands.)

2. The size of the sum counter is equal to the number of receiving character positions specified by the PICTURE clause that accompanies the SUM clause in the description of the elementary item.

3. Only one sum counter exists for an elementary report entry regardless of the number of SUM clauses specified in the elementary report entry.

4. If the elementary report entry for a printable item contains a SUM clause, the sum counter serves as a source data item. The RWCS moves the data contained in the sum counter, according to the rules of the MOVE statement, to the printable item for presentation.

5. If a data-name appears as the subject of an elementary report entry that contains a SUM clause, the data-name is the name of the sum counter; the data-name is not the name of the printable item that the entry may also define.

6. It is permissible for Procedure Division statements to alter the contents of sum counters.

7. Addition of the values of the data items referenced by identifiers into sum counters is performed by the report writer control system during the execution of GENERATE and TERMINATE statements. There are three categories of sum counter incrementing called subtotalling, crossfooting, and rolling forward. Subtotalling is accomplished only during execution of GENERATE statements and after any control break processing but before processing of the detail report group. (See The GENERATE Statement.) Crossfooting and rolling forward are accomplished during the processing of control footing report groups. (See The TYPE Clause.)

8. The UPON phrase provides the capability to accomplish selective subtotalling for the detail report groups named in the phrase.

9. The report writer control system adds each individual addend into the sum counter at a time that depends upon the characteristics of the addend.

   a) When the addend is a sum counter defined in the same control footing report group, then the accumulation of that addend into the sum counter is termed crossfooting.

      Crossfooting occurs when a control break takes place and at the time the control footing report group is processed.

      Crossfooting is performed according to the sequence in which sum counters are defined within the control footing report group. That is, all crossfooting into the first sum counter defined in the control footing report group is completed, and then all crossfooting into the second sum counter defined in the control footing report group is completed. This procedure is repeated

until all crossfooting operations are completed.

When one of the addends is the sum counter defined by the data description entry in which that SUM clause appears, the initial value of that sum counter at the time of summation is used in the summing operation.

b) When the addend is a sum counter defined in a lower level control footing report group, then the accumulation of that addend into the sum counter is termed rolling forward. A sum counter in a lower level control footing report group is rolled forward when a control break occurs and at the time that the lower level control footing report group is processed.

c) When the addend is not a sum counter the accumulation into a sum counter of such an addend is called subtotalling. If the SUM clause contains the UPON phrase, the addends are subtotalled when a GENERATE statement for the designated detail report group is executed. If the SUM clause does not contain the UPON phrase, the addends which are not sum counters are sub-totalled when any GENERATE data-name statement is executed for the report in which the SUM clause appears.

10. If two or more of the identifiers specify the same addend, then the addend is added into the sum counter as many times as the addend is referenced in the SUM clause. It is permissible for two or more of the data-names to specify the same detail report group. When a GENERATE data-name statement for such a detail report group is given, the incrementing occurs repeatedly, as many times as data-name appears in the UPON phrase.

11. The subtotalling that occurs when a GENERATE report-name statement is executed is discussed in the appropriate paragraph. (See The GENERATE Statement.)

12. In the absence of an explicit RESET phrase, the report writer control system (RWCS) will set a sum counter to zero at the time that the RWCS is processing the control footing report group within which the sum counter is defined. If an explicit RESET phrase is specified, then the RWCS will set the sum counter to zero at the time that the RWCS is processing the designated level of the control hierarchy. (See The TYPE Clause.)

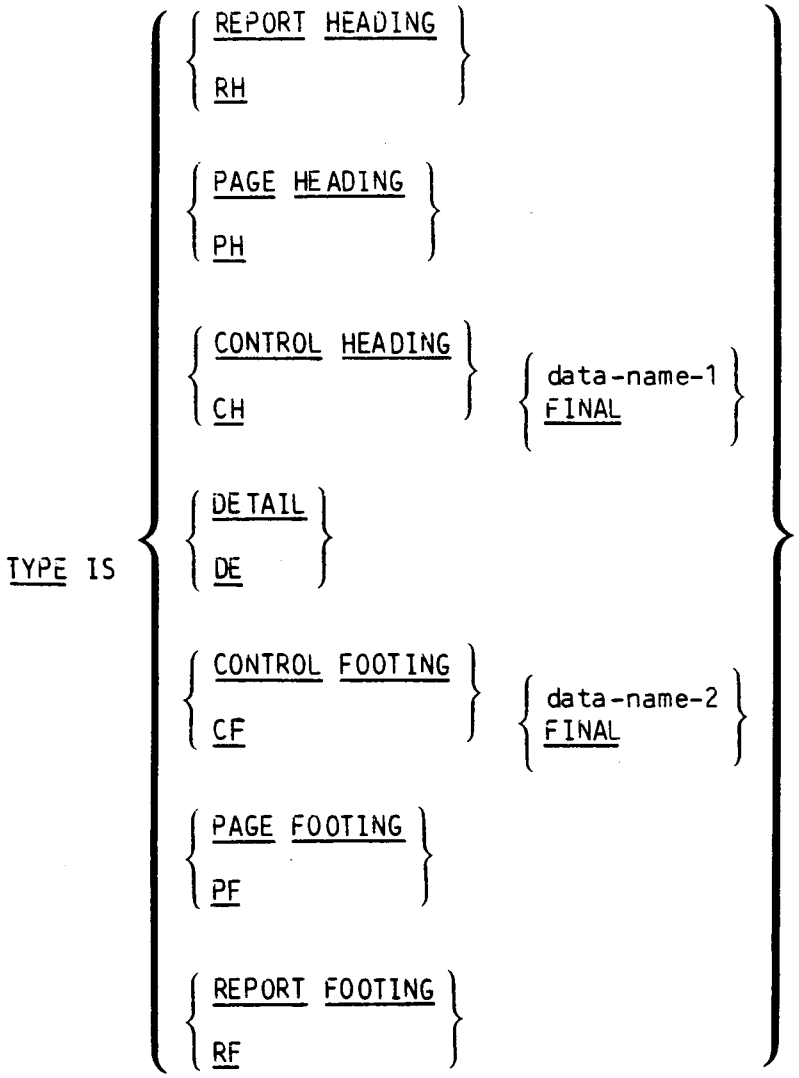
    Sum counters are initially set to zero by the report writer control system during the execution of the INITIATE statement for the report containing the sum counter.

TYPE clause

Function

The TYPE clause specifies the particular type of report group that is described by this entry and indicates the time at which the report group is to be processed by the report writer control system (RWCS).

General Format



Syntax Rules

1. RH, PH, CH, DE, CF, PF, RF are abbreviations for REPORT HEADING, PAGE HEADING, CONTROL HEADING, DETAIL, CONTROL FOOTING, PAGE FOOTING, and REPORT FOOTING respectively.

2. REPORT HEADING, PAGE HEADING, CONTROL HEADING FINAL, CONTROL FOOTING FINAL, PAGE FOOTING, and REPORT FOOTING report groups may each appear no more than once in the description of a report.

3. PAGE HEADING and PAGE FOOTING report groups may be specified only if a PAGE clause is specified in the corresponding report description entry.

4. Data-name-1, data-name-2 and FINAL, if present, must be specified in the CONTROL clause of the corresponding report description entry. At most, one CONTROL HEADING report group and one CONTROL FOOTING report group can be specified for each data-name or FINAL in the CONTROL clause of the report description entry. However, neither a CONTROL HEADING report group nor a CONTROL FOOTING report group is required for a data-name or FINAL specified in the CONTROL clause of the report description entry.

5. In CONTROL FOOTING, PAGE HEADING, PAGE FOOTING, and REPORT FOOTING report groups, SOURCE clauses and USE statements must not reference any of the following:

   a) Group data items containing a control data item.

   b) Data items subordinate to a control data item.

   c) A redefinition or renaming of any part of a control data item.

   In PAGE HEADING and PAGE FOOTING report groups, SOURCE clauses and USE statements must not reference control data-names.

6. When a GENERATE report-name statement is specified in the Procedure Division, the corresponding report description entry must include no more than one detail report group. If no GENERATE data-name statements are specified for such a report, a detail report group is not required.

7. The description of a report must include at least one body group.

General Rules

1. Detail report groups are processed by the report writer control system (RWCS) as a direct result of GENERATE statements. If a report group is other than TYPE DETAIL, its processing is an automatic RWCS function.

2. The REPORT HEADING phrase specifies a report group that is processed by the RWCS only once, per report, as the first report group of that report. The REPORT HEADING report group is processed during the execution of the chronologically first GENERATE statement for that report.

3. The PAGE HEADING phrase specifies a report group that is processed by the RWCS as the first report group on each page of that report except under the following conditions:

   a) A PAGE HEADING report group is not processed on a page that is to contain only a REPORT HEADING report group or only a REPORT FOOTING report group.

b) A PAGE HEADING report group is processed as the second report group on a page when it is preceded by a REPORT HEADING report group that is not to be presented on a page by itself.

(See Presentation Rules Tables.)

4. The CONTROL HEADING phrase specifies a report group that is processed by the RWCS at the beginning of a control group for a designated control data-name or, in the case of FINAL, is processed during the execution of the chronologically first GENERATE statement for that report. During the execution of any GENERATE statement at which the RWCS detects a control break, any CONTROL HEADING report groups associated with the highest control level of the break and lower levels are processed.

5. The DETAIL phrase specifies a report group that is processed by the RWCS when a corresponding GENERATE statement is executed.

6. The CONTROL FOOTING phrase specifies a report group that is processed by the RWCS at the end of a control group for a designated control data-name.

   In the case of FINAL, the CONTROL FOOTING report group is processed only once per report as the last body group of that report. During the execution of any GENERATE statement in which the RWCS detects a control break, any CONTROL FOOTING report group associated with the highest level of the control break or more minor levels is presented. All CONTROL FOOTING report groups are presented during the execution of the TERMINATE statement if there has been at least one GENERATE statement executed for the report. (See The TERMINATE Statement.)

7. The PAGE FOOTING phrase specifies a report group that is processed by the RWCS as the last report group on each page except under the following conditions:

   a) A PAGE FOOTING report group is not processed on a page that is to contain only a REPORT HEADING report group or only a REPORT FOOTING report group.

   b) A PAGE FOOTING report group is processed as the second to last report group on a page when it is followed by a REPORT FOOTING report group that is not to be processed on a page by itself.

   (See Presentation Rules Tables.)

8. The REPORT FOOTING phrase specifies a report group that is processed by the RWCS only once per report and as the last report group of that report. The REPORT FOOTING report group is processed during the execution of a corresponding TERMINATE statement, if there has been at least one GENERATE statement executed for the report. (See The TERMINATE Statement.)

9. The sequence of steps that the RWCS executes when it processes a REPORT HEADING, PAGE HEADING, CONTROL HEADING, PAGE FOOTING, or REPORT FOOTING report group is described below.

a) If there is a USE BEFORE REPORTING procedure that references the data-name of the report group, the USE procedure is executed.

b) If a SUPPRESS statement has been executed or if the report group is not printable, there is no further processing to be done for the report group.

c) If a SUPPRESS statement has not been executed and the report group is printable, the RWCS formats the print lines and presents the report group according to the presentation rules for that type of report group. (See Presentation Rules Tables.)

10. The sequence of steps that the RWCS executes when it processes a CONTROL FOOTING report group is described below.

The GENERATE rules specify that when a control break occurs, the RWCS produces the CONTROL FOOTING report groups beginning at the minor level, and proceeding upwards, through the level at which the highest control break was sensed. In this regard, it should be noted that even though no CONTROL FOOTING report group has been defined for a given control data-name, the RWCS will still have to execute the step described in paragraph 10f below if a RESET phrase within the report description specifies that control data-name.

a) Sum counters are crossfooted, i.e., all sum counters defined in this report group that are operands of SUM clauses in the same report group are added to their sum counters. (See the SUM clause.)

b) Sum counters are rolled forward, i.e., all sum counters defined in the report group that are operands of SUM clauses in higher level CONTROL FOOTING report groups are added to the higher level sum counters. (See the SUM clause.)

c) If there is a USE BEFORE REPORTING procedure that references the data-name of the report group the USE procedure is executed.

d) If a SUPPRESS statement has been executed or if the report group is not printable, the RWCS next executes the step described in paragraph 10f below.

e) If a SUPPRESS statement has not been executed and the report group is printable, the RWCS formats the print lines and presents the report group according to the presentation rules for CONTROL FOOTING report groups.

f) Then the RWCS resets those sum counters that are to be reset when the RWCS processes this level in the control hierarchy. (See the SUM clause.)

11. The detail report group processing that the RWCS executes in response to a GENERATE data-name statement is described in paragraphs 11a through 11e below.

When the description of a report includes exactly one detail report group, the detail-related processing that the RWCS executes in

response to a GENERATE report-name statement is described in paragraphs 11a through paragraph 11d below. These steps are performed as though a GENERATE data-name statement were being executed.

When the description of a report includes no detail report groups, the detail-related processing that the RWCS executes in response to a GENERATE report-name statement is described in paragraph 11a below. This step is performed as though the description of the report included exactly one detail report group, and a GENERATE data-name statement were being executed.

a) The RWCS performs any subtotalling that has been designated for the detail report group. (See the SUM clause.)

b) If there is a USE BEFORE REPORTING procedure that refers to the data-name of the report group, the USE procedure is executed.

c) If a SUPPRESS statement has been executed or if the report group is not printable there is no further processing done for the report group.

d) If the DETAIL report group is being processed as a consequence of a GENERATE report-name statement, there is no further processing done for the report group.

e) IF neither 11c nor 11d above applies, the RWCS formats the print lines and presents the report group according to the presentation rules for DETAIL report groups. (See Presentation Rules Tables.)

12. When the RWCS is processing a CONTROL HEADING, CONTROL FOOTING, or DETAIL report group, as described in general rules 9, 10, and 11, the RWCS may have to interrupt the processing of that body group after determining that the body group is to be presented, and execute a page advance (and process PAGE FOOTING and PAGE HEADING report groups) before actually presenting the body group.

13. During control break processing, the values of control data items that the RWCS used to detect a given control break are referred to as prior values.

a) During control break processing of a CONTROL FOOTING report group, any references to control data items in a USE procedure or SOURCE clause associated with that CONTROL FOOTING report group are supplied with prior values.

b) When a TERMINATE statement is executed, the RWCS makes the prior control data item values available to SOURCE clause or USE procedure references in CONTROL FOOTING and REPORT FOOTING report groups as though a control break had been detected in the highest control data-name.

c) All other data item references within report groups and their USE procedures access the current values that are contained within the data items at the time the report group is processed.

Function

The USAGE clause specifies the format of a data item in the computer storage.

General Format

[USAGE IS] DISPLAY

The following rules are in addition to or a limitation to the rules stated in the PROCEDURE DIVISION chapter.

Syntax Rules

1. The USAGE clause for a report group item can specify only USAGE IS DISPLAY.

VALUE clause

Function

The VALUE clause defines the value of Report Section printable items.

General Format

VALUE IS literal-1

The following rules are in addition to or a limitation to the rules stated in the PROCEDURE DIVISION chapter.

General Rules

1. In the Report Section, if the elementary report entry containing the VALUE clause does not contain a GROUP INDICATE clause, then the printable item will assume the specified value each time its report group is printed. However, when the GROUP INDICATE clause is also present, the specified value will be presented only when certain object time conditions exist (see The GROUP INDICATE Clause).

REPORT WRITER IN THE PROCEDURE DIVISION

GENERAL DESCRIPTION

The Procedure Division contains declarative procedures when the USE BEFORE REPORTING statement from the Report Writer is present in a COBOL source program. Shown below is the general format of the Procedure Division when the USE BEFORE REPORTING statement is present.

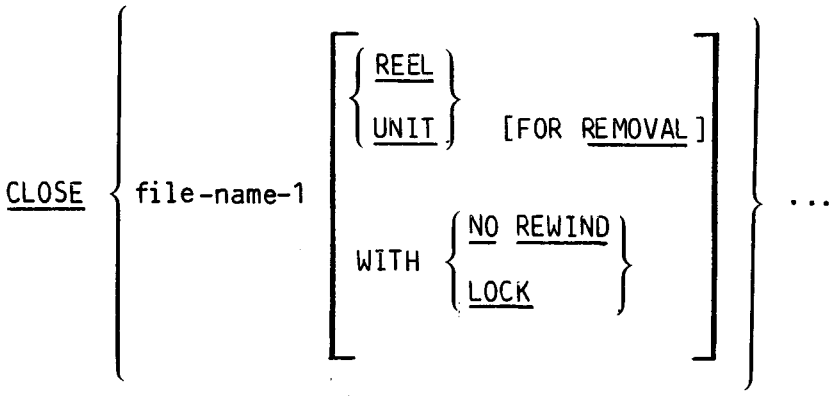
```
PROCEDURE DIVISION.

DECLARATIVES.

{section-name SECTION.

    USE BEFORE REPORTING statement.

[paragraph-name.

    [sentence] ... ] ... } ...

END DECLARATIVES.

{section-name SECTION.

[paragraph-name.

    [sentence] ... ] ... } ...
```

CLOSE

Function

The CLOSE statement terminates the processing of reels/units and files with optional rewind and/or lock or removal where applicable.

General Format



The following rules are in addition to or a limitation to the rules stated in the PROCEDURE DIVISION chapter.

General Rules

1. All reports associated with a report file that have been initiated must be ended with the execution of a TERMINATE statement before a CLOSE statement is executed for that report file.

Function

The GENERATE statement directs the report writer control system (RWCS) to produce a report in accordance with the report description that was specified in the Report Section of the Data Division.

General Format

GENERATE { data-name-1 / report-name-1 }

Syntax Rules

1. Data-name-1 must name a type detail report group and may be qualified by a report-name.

2. Report-name-1 may be used only if the referenced report description contains:

   a) A CONTROL clause, and

   b) Not more than one detail report group, and

   c) At least one body group.

General Rules

1. In response to a GENERATE report-name-1 statement, the RWCS performs summary processing. If all of the GENERATE statements that are executed for a report are of the form GENERATE report-name-1, then the report that is produced is called a summary report. A summary report is one in which no detail report group is presented.

2. In response to a GENERATE data-name-1 statement, the RWCS performs detail processing that includes certain processing that is specific for the detail report group designated by the GENERATE statement. Normally, the execution of a GENERATE data-name-1 statement causes the RWCS to present the designated detail report group.

3. During the execution of the chronologically first GENERATE statement for a given report, the RWCS saves the values within the control data items. During the execution of the second and subsequent GENERATE statements for the same report, and until a control break is detected, the RWCS utilizes this set of control values to determine whether a control break has occurred. When a control break occurs, the RWCS saves the new set of control values, which it thereafter uses to sense for a control break until another control break occurs.

4. During report presentation, an automatic function of the RWCS is to process page heading and page footing report groups, if defined, when the RWCS must advance the report to a new page for the purpose of presenting a body group. (See Presentation Rules Tables.)

5. When the chronologically first GENERATE statement for a given report is executed, the RWCS processes, in order, the report groups that are named below, provided that such report groups are defined within the report description. The RWCS also processes page heading and page footing report groups as described in general rule 4. The actions taken by the RWCS when it processes each type of report group are explained under the appropriate paragraph. (See the TYPE clause.)

   a) The report heading report group is processed.

   b) The page heading report group is processed.

   c) All control heading report groups are processed from major to minor.

   d) If a GENERATE data-name statement is being executed, the processing for the designated detail report group is performed. If a GENERATE report-name statement is being executed, certain of the steps that are involved in the processing of a detail report group are performed. (See the TYPE clause.)

6. When a GENERATE statement other than the chronologically first is executed for a given report, the RWCS performs the steps enumerated below, as applicable. The RWCS also processes page heading and page footing report groups as described in general rule 4. The actions taken by the RWCS when it processes each type of report group are explained under the appropriate paragraph. (See the TYPE clause.)

   a) Sense for control break. The rules for determining the equality of control data items are the same as those specified for relation conditions. If a control break has occurred then:

      - Enable the control footing USE procedures and control footing SOURCE clauses to access the control data item values that the report writer control system used to detect a given control break. (See the TYPE clause.)

      - Process the control footing report groups in the order minor to major. Only control footing report groups that are not more major than the highest level at which a control break occurred are processed.

      - Process the control heading report groups in the order major to minor. Only the control heading report groups that are not more major than the highest level at which a control break occurred are processed.

   b) If a GENERATE data-name statement is being executed, the processing for the designated detail report group is performed. If a GENERATE report-name statement is being executed, certain of the steps that are involved in the processing of a detail report group are performed. (See the TYPE clause.)

7. GENERATE statements for a report can be executed only after an INITIATE statement for the report has been executed and before a TERMINATE statement for the report has been executed.

Function

The INITIATE statement causes the report writer control system to begin the processing of a report.

General Format

INITIATE {report-name-1} ...

Syntax Rules

1. Report-name-1 must be defined by a report description entry in the Report Section of the Data Division.

General Rules

1. The INITIATE statement performs the following initialization functions for each named report:

   a) All sum counters are set to zero.

   b) LINE-COUNTER is set to zero.

   c) PAGE-COUNTER is set to one.

2. The INITIATE statement does not place the file associated with the report in the open mode; therefore, an OPEN statement with either the OUTPUT phrase or the EXTEND phrase for the file must be executed prior to the execution of the INITIATE statement.

3. A subsequent INITIATE statement for report-name-1 must not be executed unless an intervening TERMINATE statement has been executed for report-name-1.

4. If more than one report-name is specified in an INITIATE statement, the result of executing this INITIATE statement is the same as if a separate INITIATE statement had been written for each report-name in the same order as specified in the INITIATE statement.

OPEN

Function

The OPEN statement initiates the processing of report files.

General Format

```
OPEN { OUTPUT {file-name-2 [WITH NO REWIND]} ... } ...
     { EXTEND {file-name-4} ...                   }
```

The following rules are in addition to or a limitation to the rules stated in the PROCEDURE DIVISION chapter.

Syntax Rules

1. The OPEN statement for a report file must contain only the OUTPUT phrase or the EXTEND phrase.

General Rules

1. The OPEN statement for a report file must be executed prior to the execution of an INITIATE statement for any reports contained in the file.

2. A report file may be opened with the OUTPUT and EXTEND phrases in the same run unit. Following the initial execution of an OPEN statement for a file, each subsequent OPEN statement execution for that same report file must be preceded by the execution of a CLOSE statement, without the LOCK phrase, for that file.

Function

The SUPPRESS statement causes the report writer control system to inhibit the presentation of a report group.

General Format

SUPPRESS PRINTING

Syntax Rules

1. The SUPPRESS statement may only appear in a USE BEFORE REPORTING procedure.

General Rules

1. The SUPPRESS statement inhibits presentation only for the report group named in the USE procedure within which the SUPPRESS statement appears.

2. The SUPPRESS statement must be executed each time the presentation of the report group is to be inhibited.

3. When the SUPPRESS statement is executed, the RWCS is instructed to inhibit the processing of the following report group functions:

   a) The presentation of the print lines of the report group.

   b) The processing of all LINE clauses in the report group.

   c) The processing of the NEXT GROUP clause in the report group.

   d) The adjustment of LINE-COUNTER.

TERMINATE

Function

The TERMINATE statement causes the report writer control system (RWCS) to complete the processing of the specified reports.

General Format

TERMINATE {report-name-1} ...

Syntax Rules

1. Report-name-1 must be defined by a report description entry in the Report Section of the Data Division.

General Rules

1. The TERMINATE statement causes the RWCS to produce all the control footing report groups beginning with the minor control footing report group. Then the report footing report group is produced. The RWCS makes the prior set of control data item values available to the control footing and report footing SOURCE clauses and USE procedures, as though a control break has been sensed in the most major control data-name.

2. If no GENERATE statements have been executed for a report during the interval between the execution of an INITIATE statement and a TERMINATE statement, for that report, the TERMINATE statement does not cause the RWCS to produce any report groups or perform any of the related processing.

3. During report presentation, an automatic function of the RWCS is to process page heading and page footing report groups, if defined, when the RWCS must advance the report to a new page for the purpose of presenting a body group. (See Presentation Rules Table.)

4. The TERMINATE statement cannot be executed for a report unless the TERMINATE statement was chronologically preceded by an INITIATE statement for that report and for which no TERMINATE statement has yet been executed.

5. If more than one report-name is specified in a TERMINATE statement, the result of executing this TERMINATE statement is the same as if a separate TERMINATE statement had been written for each report-name in the same order as specified in the TERMINATE statement.

6. The TERMINATE statement does not close the file with which the report is associated; a CLOSE statement for the file must be executed. Every report in a file that is in an initiated condition must be terminated before a CLOSE statement is executed for that file.

Function

The USE BEFORE REPORTING statement specifies Procedure Division statements that are executed just before a report group named in the Report Section of the Data Division is produced.

General Format

USE BEFORE REPORTING identifier-1

Syntax Rules

1. A USE BEFORE REPORTING statement, when present, must immediately follow a section header in the declaratives portion of the Procedure Division and must appear in a sentence by itself. The remainder of the section must consist of zero, one, or more procedural paragraphs that define the procedures to be used.

2. Identifier-1 must reference a report group. Identifier-1 must not appear in more than one USE BEFORE REPORTING statement.

3. The GENERATE, INITIATE or TERMINATE statements must not appear in a paragraph within a USE BEFORE REPORTING procedure. A PERFORM statement in a USE BEFORE REPORTING procedure must not have GENERATE, INITIATE, or TERMINATE statements in its range.

4. A USE BEFORE REPORTING procedure must not alter the value of any control data item.

5. The USE BEFORE REPORTING statement itself is never executed; it merely defines the conditions calling for the execution of the USE procedures.

General Rules

1. Declarative procedures may be included in any COBOL source program irrespective of whether the program contains or is contained within another program. A declarative is invoked just before the named report group is produced during the execution of the program. The report group is named by identifier-1 in the USE BEFORE REPORTING statement which prefaces the declarative.

2. Within a declarative procedure, there must no references to any nondeclarative procedures.

3. Procedure-names associated with a USE BEFORE REPORTING statement may be referenced in a different declarative section or in a nondeclarative procedure only with a PERFORM statement.

4. In the USE BEFORE REPORTING statement, the designated procedures are executed by the report writer control system (RWCS) just before the named report group is produced. (See The TYPE Clause.)

5. Within a USE procedure, there must not be the execution of any statement that would cause the execution of a USE procedure that had previously been invoked and had not yet returned control to the invoking routine.

A. COBOL RESERVED WORDS

The following is the list of COBOL reserved words. A standard reserved word which is recognized by but not supported in the COBOL compiler is indicated by an asterisk (*). A reserved word which is an extension to the American National Standard list is indicated by two asterisks (**). A reserved word which has been added to the American National Standard (X2.23-74) is indicated by a plus (+).

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALL
ALPHABET+
ALPHABETIC
ALPHABETIC-LOWER+*
ALPHABETIC-UPPER+*
ALPHANUMERIC+*
ALPHANUMERIC-EDITED+*
ALSO
ALTERNATE
AND
ANY+
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
AUTO**

BEFORE
BELL**
BINARY
BLANK
BLINK**
BLOCK
BOTTOM
BOX**
BY

CALL
CANCEL
CD*
CF
CH
CHARACTER
CHARACTERS
CLOCK UNITS*
CLOSE
CLASS+
CODE
CODE-SET
COLUATING
COLUMN
COMMA
COMMON+
COMMUNICATION*
COMP
COMP-1**
COMP-2**
COMP-3**
COMP-4**
COMPUTIONAL
COMPUTIONAL-1**
COMPUTIONAL-2**
COMPUTIONAL-3**
COMPUTIONAL-4**
COMPUTE
CONFIGURATION
CONSOLE**
CONTAINS
CONTENT+*
CONTINUE+*
CONTROL
CONTROLS
COVERTING
COPY
CORR
CORRESPONDING
COUNT
CURRENCY

DATA
DATE
DATE-COMPILED
DATA_WRITTEN
DATE-WRITTEN
DAY
DAY-OF-WEEK
DE
DEBUG-CONTENTS*
DEBUG-ITEM*
DEBUG-LINE*
DEBUG-NAME*
DEBUG-SUB-1*
DEBUG-SUB-2*
DEBUG-SUB-3*
DEBUGGING
DECIMAL-POINT
DECLARATIVE
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION*
DETAIL
DISABLE*
DISPLAY
DIVIDE
DIVISION
DOUBLE-REAL**
DOWN
DUPLICATES
DYNAMIC

EDIT+
EGI*
ELSE
EMI*
ENABLE*
END
END-ACCEPT**
END-ADD+
END-CALL+
END-COMPUTE+
END-DELETE+
END-DIVIDE+
END-EDIT+
END-EVALUATE+
END-IF+
END-MULTIPLY+
END-OF-PAGE
END-PERFORM+
END-READ+
END-RECEIVE*+
END-RETURN+
END-REWRITE+
END-SEARCH+
END-START+
END-STRING+
END-SUBTRACT+
END-UNSTRING+
END-WRITE+
ENVIRONMENT
EOP
EQUAL
ERROR
ESCAPE**
ESI*
EVALUATE+
EXCEPTION
EXIT
EXTEND
EXTERNAL+

FALSE+
FD
FILE
FILE-CONTROL
FILLER
FINAL
FIRST
FOOTING
FOR
FROM
FULL**

GENERATE
GIVING
GLOBAL+
GO
GREATER
GROUP

HEADING
HEBREW**
HIGH-VALUE
HIGH-VALUES
HIGHLIGHT**

I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIALIZE+*
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS

JUST
JUSTIFIED

KEY

LABEL
LAST
LEADING
LEFT
LENGTH*
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINE-PARAM**
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES

MERGE
MESSAGE
MODE
MOVE
MULTIPLE
MULTIPLY

NATIVE
NEGATIVE
NEXT
NO
NOT
NUMBER
NUMERIC
NUMERIC-EDITED**

OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORDER+
OUTPUT
OVERFLOW

PACKED-DECIMAL
PADDING**
PAGE
PAGE-COUNTER
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
POSITIVE
PREVIOUS**
PRIMARY**
PRINTING
PROCEDURE
PROCEDURES
PROGRAM
PROGRAM-ID
PURGE**

QUEUE*
QUOTE
QUOTES

RANDOM
RD
READ
RECEIVE*
RECORD
RECORDS
REDEFINES
REEL
REFERENCE+*
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACE*
REPLACING
REPORT
REPORTING
REPORTS
REQUIRED**
RERUN*
REVERSE
RESET
RETURN
REVERSE**
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

SAME
SCREEN**
SD
SEARCH
SECTION
SECURE**
SECURITY
SEGMENT*
SEGMENT-LIMIT
SELECT
SEND*
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SINGLE-REAL**
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-1
STANDARD-2
START
STATUS
STOP
STRING
SUB-QUEUE-1*
SUB-QUEUE-2*
SUB-QUEUE-3*
SUBTRACT
SUM
SUPPRESS
SWTCH
SYMBOLIC*
SYNC
SYNCRONIZED

TABLE*
TALLYNG
TAPE
TERMINAL*
TERMINATE
TEST+*
TEXT*
THAN
THEN+
THROUGH
THRU
TIME
TIME-OUT**
TIMES
TO
TOP
TRILING
TRUE+
TYPE
UNDERLINE**
UNIT
UNLOCK**
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING

WHEN
WITH
WORD-BINARY**
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

+
-
*
/
**
>
<
=

SYSTEM NAMES

The system-names supported by the COBOL compiler are given in the following lists. The appearance of one of these names in the appropriate division of a program causes it to be treated as a reserved word. Some implementor-names (SWITCH and CONSOLE) have been added to the list of reserved words, and as such, appear in the reserved word list.

COMPUTER-NAMES

M-30 M-40 M-60

ENVIRONMENT DIVISION IMPLEMENTOR-NAMES

SYSPRINT1 SYSPRINT3 TEXT-FILE
SYSPRINT2 SYSPRINT4 WS-PRINTER

DATA DIVISION IMPLEMENTOR-NAMES

ALLOC-UNIT FILE-SIZE PAGE-SIZE
SPLIT-STRATEGY

PROCEDURE DIVISION IMPLEMENTOR-NAMES

LINE-PARAM

B. GLOSSARY

The terms are defined in accordance with their meaning as used in this document describing COBOL and may not have the same meaning for other languages.

These definitions are intended as either reference or introductory material to be read with a prior to reading the detailed language specifications. For this reason, these definitions are, in most instances, brief and do not include detailed syntactical rules.

DEFINITIONS

Abbreviated Combined Relation Condition. The combined condition that results from the explicit omission of a common subject or a common subject and common relational operator in a consecutive sequence of relation conditions.

Access Mode. The manner in which records are to be operated upon within a file.

Actual Decimal Point. The physical representation, using either of the decimal point characters period (.) or comma (,), of the decimal point position in a data item.

Alphabet-Name. A user-defined word, in the SPECIAL-NAMES paragraph of the Environment Division, that assigns a name to a specific character set and/or collating sequence. (See also SPECIAL-NAMES.)

Alphabetic Character. A letter or space character.

Alphanumeric Character. Any character in the computer's character set.

Alternate Record Key. A key, other than the prime record key, whose contents identify a record within an indexed file.

Arithmetic Expression. An identifier of a numeric elementary item, a numeric literal, such identifiers and literals separated by arithmetic operators, two arithmetic expressions separated by an arithmetic operator, or an arithmetic expression enclosed in parentheses.

Arithmetic Operation. The process caused by the execution of an arithmetic statement, or the evaluation of an arithmetic expression, that results in a mathematically correct solution to the arguments presented.

Arithmetic Operator. A single character, or a fixed two-character combination, that belongs to the following set:

Character	Meaning
+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Ascending Key. A key upon the values of which data is ordered starting with the lowest value of key up to the highest value of key in accordance with the rules for comparing data items.

Assumed Decimal Point. A decimal point position which does not involve the existence of an actual character in a data item. The assumed decimal point has logical meaning but no physical representation.

At End Condition. A condition caused:

1. During the execution of a READ statement for a sequentially accessed file, when no next logical record exists in the file, or when the number of significant digits in the relative record number is larger than the size of the relative key data item, or when an optional input file is not present, or when the at end condition already exists.

2. During the execution of a RETURN statement, when no next logical record exists for the associated sort or merge file.

3. During the execution of a SEARCH statement, when the search operation terminates without satisfying the condition specified in any of the associated WHEN phrases.

Block. A physical unit of data that is normally composed of one or more logical records. For mass storage files, a block may contain a portion of a logical record. The size of a block has no direct relationship to the size of the file within which the block is contained or to the size of the logical record(s) that are either contained within the block or that overlap the block. The term is synonymous with physical record.

Body Group. Generic name for a report group of TYPE DETAIL, CONTROL HEADING or CONTROL FOOTING.

Bottom Margin. An empty area which follows the page body.

Called Program. A program which is the object of a CALL statement combined at object time with the calling program to produce a run unit.

Calling Program. A program which executes a CALL to another program.

Character. The basic indivisible unit of the language.

Character Position. A character position is the amount of physical storage required to store a single standard data format character whose usage is DISPLAY (one byte).

Character-String. A sequence of contiguous characters which form a COBOL word, a literal, a PICTURE character-string, or a comment-entry. (See Character-String.)

Class Condition. The proposition, for which a truth value can be determined, that the content of an item is wholly alphabetic or is wholly numeric or consists exclusively of those characters listed in the definition of the class-name.

Class-Name. A user-defined word defined in the SPECIAL-NAMES paragraph of the Environment Division that assigns a name to the proposition for which a truth value can be defined, that the content of a data item consist exclusively of those characters listed in the definition of the class-name.

Clause. A clause is an ordered set of consecutive COBOL character-strings whose purpose is to specify an attribute of an entry.

COBOL Character Set. The complete COBOL character set consists of the characters listed below:

Character	Meaning	Character	Meaning
0,1,...,9	digit	$	currency sign
A,B,...,Z	uppercase letter	,	comma (decimal point)
a,b,...,z	lowercase letter	;	semicolon
	space	.	period (decimal point, full stop)
+	plus sign	"	quotation mark
-	minus sign (hyphen)	(	left parenthesis
*	asterisk	)	right parenthesis
/	slant (virgule, slash)	>	greater than symbol
=	equal sign	<	less than symbol

NOTE: Lowercase letters may be used in character-strings. Except when used in nonnumeric literals, each is equivalent to the corresponding uppercase letter.

COBOL Word. A character-string of not more than 30 characters which forms a user-defined word, a system-name, or a reserved word.

Collating Sequence. The sequence in which the characters that are acceptable in a computer are ordered for purposes of sorting, merging, comparing, and for processing indexed files sequentially.

Column. A character position within a print line. The columns are numbered from 1, by 1, starting at the leftmost character position of the print line and extending to the rightmost position of the print line.

Combined Condition. A condition that is the result of connecting two or more conditions with the 'AND' or the 'OR' logical operator.

Comment-Entry. An entry in the Identification Division that may be any combination of characters from the computer's character set.

Comment Line. A source program line represented by an asterisk (*) in the indicator area of the line and any characters from the computer's character set in area A and area B of that line. The comment line serves only for documentation in a program. A special form of comment line represented by a slant (/) in the indicator area of the line and any characters from the computer's character set in area A and area B of that line causes page ejection prior to printing the comment.

Common Program. A program which, despite being directly contained within another program, may be called from any program directly or indirectly contained in that other program.

Compile Time. The time at which a COBOL source program is translated, by a COBOL compiler, to a COBOL object program.

Compiler Directing Statement. A statement, beginning with a compiler directing verb, that causes the compiler to take a specific action during compilation.

Complex Condition. A condition in which one or more logical operators act upon one or more conditions. (See Negated Simple Condition, Combined Condition, and Negated Combined Condition.)

Computer-Name. A system-name that identifies the computer upon which the program is to be compiled or run.

Condition. A status of a program at execution time for which a truth value can be determined. Where the term 'condition' (condition-1, condition-2, ...) appears in these language specifications in or in reference to 'condition' (condition-1, condition-2, ...) of a general format, it is a conditional expression consisting of either a simple condition optionally parenthesized, or a combined condition consisting of the syntactically correct combinating of simple conditions, logical operators, and parentheses, for which a truth value can be determined.

Condition-Name. A user-defined word that assigns a name to a subset of values that a conditional variable may assume; or a user-defined word assigned to a status of a switch or device. When 'condition-name' is used in the general formats, it represents a unique data item reference consisting of a syntactically correct combination of a condition-name, together with qualifiers and subscripts, as required for uniqueness of reference.

Condition-Name Condition. The proposition, for which a truth value can be determined, that the value of a conditional variable is a member of the set of values attributed to a condition-name associated with the conditional variable.

Conditional Expression. A simple condition or a complex condition specified in an EVALUATE, IF, PERFORM, or SEARCH statement. (See Simple

Condition and Complex Condition.)

Conditional Statement. A conditional statement specifies that the truth value of a condition is to be determined and that the subsequent action of the object program is dependent on this truth value.

Conditional Variable. A data item one or more values of which has a condition-name assigned to it.

Configuration Section. A section of the Environment Division that describes overall specifications of source and object programs.

Connective. A reserved word that is used to:

1. Associate a data-name, paragraph-name, or condition-name with its qualifier.

2. Link two or more operands written in a series.

3. Form conditions (logical connectives). (See Logical Operator.)

Contiguous Items. Items that are described by consecutive entries in the Data Division, and that bear a definite hierarchical relationship to each other.

Control Break. A change in the value of a data item that is referenced in the CONTROL clause. More generally, a change in the value of a data item that is used to control the hierarchical structure of a report.

Control Break Level. The relative position within a control hierarchy at which the most major control break occurred.

Control Data Item. A data item, a change in whose content may produce a control break.

Control Data-Name. A data-name that appears in a CONTROL clause and refers to a control data item.

Control Footing. A report group that is presented at the end of the control group of which it is a member.

Control Group. A set of body groups that is presented for a given value of a control data item or of FINAL. Each control group may begin with a CONTROL HEADING, end with a CONTROL FOOTING, and contain DETAIL report groups.

Control Heading. A report group that is presented at the beginning of the control group of which it is a member.

Control Hierarchy. A designated sequence of report subdivisions defined by the positional order of FINAL and the data-names within a CONTROL clause.

Counter. A data item used for storing numbers or number representations in a manner that permits these numbers to be increased or decreased by the value of another number, or to be changed or reset to zero or to an

arbitrary positive or negative value.

Currency Sign. The character '$' of the COBOL character set.

Currency Symbol. The character defined by the CURRENCY SIGN clause in the SPECIAL-NAMES paragraph. If no CURRENCY SIGN clause is present in a COBOL source program, the currency symbol is identical to the currency sign.

Current Record. In file processing, the record which is available in the record area associated with the file.

Current Record Pointer. A conceptual entity that is used in the selection of the next record.

Current Volume Pointer. A conceptual entity that points to the current volume of a sequential file.

Data Clause. A clause appearing in a data description entry in the Data Division of a COBOL program, that provides information describing a particular attribute of a data item.

Data Description Entry. An entry in the Data Division of a COBOL program, that is composed of a level-number followed by a data-name, if required, and then followed by a set of data clauses, as required.

Data Item. A unit of data (excluding literals) defined by the COBOL program.

Data-Name. A user-defined word that names a data item described in a data description entry in the Data Division. When used in the general formats, 'data-name' represents a word which can neither be subscripted, indexed nor qualified unless specifically permitted by the rules for that format.

Debugging Line. A debugging line is any line with a 'D' in the indicator area of the line.

Debugging Section. A debugging section is a section that contains a USE FOR DEBUGGING statement.

Declarative Sentence. A compiler directing sentence consisting of a single USE statement terminated by the separator period.

Declaratives. A set of one or more special purpose sections, written at the beginning of the Procedure Division, the first of which is preceded by the key word DECLARATIVES and the last of which is followed by the key words END DECLARATIVES. A declarative is composed of a section header, followed by a USE compiler directing sentence, followed by a set of zero, one or more associated paragraphs.

Delimited Scope Statement. Any statement which includes its explicit scope terminator. (See Explicit and Implicit Scope Terminators.)

Delimiter. A character or a sequence of contiguous characters that identify the end of a string of characters and separates that string of

characters from the following string of characters. A delimiter is not part of the string of characters that it delimits.

Descending Key. A key upon the values of which data is ordered starting with the highest value of key down to the lowest value of key, in accordance with the rules for comparing data items.

Digit Position. A digit position is the amount of physical storage required to store a single digit. This amount may vary depending on the usage specified in the data description entry that defines the data item. If the data description entry specifies that usage is DISPLAY, then a digit position is synonymous with a character position.

Division. A set of zero, one, or more sections or paragraphs, called the division body, that are formed and combined in accordance with a specific set of rules. Each division consists of the division header and the related division body. There are four divisions in a COBOL program: Identification, Environment, Data, and Procedure.

Division Header. A combination of words, followed by a separator period, that indicates that beginning of a division. The division headers in a COBOL program are:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION [USING {data-name-1} ...].
```

Dynamic Access. An access mode in which specific logical records can be obtained from or placed into a mass storage file in a nonsequential manner and obtained from a file in a sequential manner during the scope of the same OPEN statement.

Dynamically Loadable Link Unit. A module, created by the linker, consisting of one or more object programs which is present in memory after the execution of CALL identifier statement until the execution of a CANCEL statement naming that link unit. Only one of the object programs can be specified as an entry-point of the link unit.

Editing Character. A single character or a fixed two-character combination belonging to the following set:

Character	Meaning	Character	Meaning
B	space	Z	zero suppress
0	zero	*	check protect
+	plus	$	currency sign
-	minus	,	comma (decimal point)
CR	credit	.	period (decimal point)
DB	debit	/	slant (virgule, slash)

Elementary Item. A data item that is described as not being further logically subdivided.

End of Procedure Division. The physical position in a COBOL source program after which no further procedures appear.

End Program Header. A combination of words, followed by a separator period, that indicates the end of a COBOL source program. The end program header is:

```
END PROGRAM program-name.
```

Entry. Any descriptive set of consecutive clauses terminated by a period and written in the Identification Division, Environment Division, or Data Division of a COBOL source program.

Environment Clause. A clause that appears as part of an Environment Division entry.

Execution Time. The time at which an object program is executed. The term is synonymous with object time.

Explicit Scope Terminator. A reserved word which terminates the scope of a particular Procedure Division statement.

Expression. An arithmetic or conditional expression.

Extend Mode. The state of a file after execution of an OPEN statement, with the EXTEND phrase specified, for that file and before the execution of a CLOSE statement for that file.

External Data. The data described in a program as external data items or external file control blocks.

External Data Item. A data item which is described as part of an external record in one or more programs of a run unit and which itself may be referenced from any program in which it is described.

External Data Record. A logical record which is described in one or more programs of a run unit and whose constituent data items may be referenced from any program in which they are described.

External File Control Block. A file control block which is accessible to one or more object programs in the run unit.

External Switch. A hardware or software device, defined and named by the implementor, which is used to indicate that one of two alternate states exists.

Figurative Constant. A compiler generated value referenced through the use of certain reserved words.

File. A collection of logical records.

File Attribute Conflict Condition. An unsuccessful attempt has been made to execute an input-output operation on a file and the file attributes, as specified for that file in the program, do not match the fixed attributes for that file.

File Clause. A clause that appears as part of any of the following Data Division entries: file description entry (FD entry) and sort-merge file description entry (SD entry.)

FILE-CONTROL. The name of an Environment Division paragraph in which the data files for a given source program are declared.

File Control Block. A storage area which contains information about a file and is used as the linkage between a file-name and a physical file and between a file-name and its associated record area.

File Control Entry. A SELECT clause and all its subordinate clauses which declare the relevant physical attributes of a file.

File Description Entry. An entry in the File Section of the Data Division that is composed of the level indicator FD, followed by a file-name, and then followed by a set of file clauses as required.

File-Name. A user-defined word that names a file described in a file description entry or a sort-merge file description entry within the File Section of the Data Division.

File Organization. The permanent logical file structure established at the time that a file is created.

File Position Indicator. A conceptual entity that contains the value of the current key within the key of reference for an indexed file, or the record number of the current record for a sequential file, or the relative record number of the current record for a relative file, or indicates that no next logical record exists, or that the number of significant digits in the relative record number is larger than the size of the relative key data item, or that an optional input file is not present, or that the at end condition already exists, or that no valid next record has been established.

File Section. The section of the Data Division that contains file description entries and sort-merge file description entries together with their associated record descriptions.

Fixed File Attributes. Information about a file which is established when a file is created and cannot subsequently be changed during the existence of the file. These attributes include the organization of the file (sequential, relative, or indexed), the prime record key, the alternate record keys, the code set, the record size, the collating sequence of the keys for indexed files, and the blocking factor.

Footing Area. The position of the page body adjacent to the bottom margin.

Format. A specific arrangement of a set of data.

Global Name. A name which is declared in only one program but which may be referenced from that program and from any program contained within that program. Condition-names, data-names, file-names, record-names, report-names, screen-names, and some special registers may be global names. (See Conventions of Condition-, Data-, File-, Record-, Report-,

and Screen-Names; general rule 5 concerning LINAGE-COUNTER; general rule 4 concerning LINE-COUNTER and PAGE-COUNTER.)

Group Item. A data item that is composed of subordinate data items.

High Order End. The leftmost character of a string of characters.

I-O-CONTROL. The name of an Environment Division paragraph in which object program requirements for sharing of same areas by several data files are specified.

I-O-CONTROL Entry. An entry in the I-O-CONTROL paragraph of the Environment Division which contains clauses which provide information required for the transmission and handling of data on named files during the execution of a program.

I-O Mode. The state of a file after execution of an OPEN statement, with the I-O phrase specified, for that file and before the execution of a CLOSE statement for that file.

I-O Status. A conceptual entity which contains the two-character value indicating the resulting status of an input-output operation. This value is made available to the program through the use of the FILE STATUS clause in the file control entry for the file.

Identifier. A syntactically correct combination of a data-name, with its qualifiers and subscripts as required for uniqueness of reference, that names a data item. The rules for 'identifier' associated with the general formats may, however, specifically prohibit qualification or subscripting.

Imperative Statement. A statement that either begins with an imperative verb and specifies an unconditional action to be taken or is a conditional statement that is delimited by its explicit scope terminator (delimited scope statement). An imperative statement may consist of a sequence of imperative statements.

Implementor-Name. A system-name that refers to a particular feature available on the L1 MOS systems.

Implicit Scope Terminator. A separator period which terminates the scope of any preceding unterminated statement, or a phrase of a statement which by its occurrence indicates the end of the scope of any statement contained within the preceding phrase.

Index. A computer storage area or register, the content of which represent the identification of a particular element in a table.

Index Data Item. A data item in which the value associated with an index-name can be stored.

Index-Name. A user-defined word that names an index associated with a specific table.

Indexed File. A file with indexed organization.

Indexed Organization. The permanent logical file structure in which each record is identified by the value of one or more keys within that record.

Initial Program. A program that is placed into an initial state every time the program is called in a run unit.

Initial State. The state of a program when it is first called in a run unit.

Input File. A file that is opened in the input mode.

Input Mode. The state of a file after execution of an OPEN statement, with the INPUT phrase specified, for that file and before the execution of a CLOSE statement for that file.

Input-Output File. A file that is opened in the I-O mode.

Input-Output Section. The section of the Environment Division that names the files and the external media required by an object program and which provides information required for transmission and handling of data during execution of the object program.

Input Procedure. A set of statements, to which control is given during the execution of a SORT statement, for the purpose of controlling the release of specified records to be sorted.

Integer. A numeric literal or a numeric data item that does not include any character positions to the right of the assumed decimal point. Where the term 'integer' appears in general formats, integer must not be a numeric data item, and must not be signed, nor zero unless explicitly allowed by the rules of that format.

Internal Data. The data described in a program excluding all external data items and external file control blocks. Items described in the Linkage Section of a program are treated as internal data.

Internal Data Item. A data item which is described in one program in a run unit. An internal data item may have a global name.

Internal File Control Block. A file control block which is accessible to only one object program in the run unit.

Intra-Record Data Structure. The entire collection of groups and elementary data items from a logical record which is defined by a contiguous subset of the data description entries which describe that record. These data description entries include all entries whose level-number is greater than the level-number of the first data description entry describing the intra-record data structure.

Invalid Key Condition. A condition, at object time, caused when a specific value of the key associated with an indexed or relative file is determined to be invalid.

Key. A data item which identifies the location of a record, or a set of data items which serves to identify the ordering of data.

Key of Reference. The key, either prime or alternate, currently being used to access records within an indexed file.

Key Word. A reserved word whose presence is required when the format in which the word appears is used in a source program.

Letter. A character that belongs to one of the following two sets:

1. uppercase letters: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;

2. lowercase letters: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

Level Indicator. Two alphabetic characters that identify a specific type of file or a position in hierarchy. The level indicators in the Data Division are FD, RD, and SD.

Level-Number. A user-defined word, expressed as a one or two digit number, which indicates the hierarchical position of a data item or which indicates special properties of a data description entry. Level-numbers in the range 1 through 49 indicate the position of a data item in the hierarchical structure of a logical record. Level-numbers in the range 1 through 9 may be written either as a single digit or as a zero followed by a significant digit. Level-numbers 66, 77, and 88 identify special properties of a data description entry.

Library Text. A sequence of text words, comment lines, the separator space, or the separator pseudo-text delimiter in a COBOL library.

LINAGE-COUNTER. A special register whose value points to the current position within the page body.

Line. A division of a page representing one row of horizontal character positions. Each character position of a report or screen line is aligned vertically beneath the corresponding character position of the report or screen line above it. Report lines are numbered from 1, by 1, starting at the top of the page. Screen lines are numbered from 1, by 1, starting at the top of the screen.

Line Number. An integer that denotes the vertical position of a report line on a page or a screen line on a screen.

Link Unit One or more object program linked together which with one another and perhaps other link units form a run unit.

Linkage Section. The section in the Data Division of the called program that describes data items available from the calling program. These data items may be referred to by both the calling and called program.

Literal. A character-string whose value is implied by the ordered-set of characters comprising the string.

Logical Operator. One of the reserved words AND, OR, or NOT. In the formation of a condition, either AND, or OR, or both, can be used as logical connectives. NOT can be used for logical negation.

Logical Page. A conceptual entity consisting of the top margin, the page body, and the bottom margin.

Logical Record. The most inclusive data item. The level-number for a record is 01. A record may be either an elementary item or a group item. The term is synonymous with record.

Low Order End. The rightmost character of a string of characters.

Mass Storage. A storage medium in which data may be organized and maintained in both a sequential and nonsequential manner.

Mass Storage File. A collection of records that is assigned to a mass storage medium.

Merge File. A collection of records to be merged by a MERGE statement. The merge file is created and can be used only by the merge function.

Mnemonic-Name. A user-defined word that is associated in the Environment Division with a specified implementor-name.

MOS File System. The input-output control system that directs, or controls, the processing of mass storage files.

Native Character Set. The character set associated with the computer specified in the OBJECT-COMPUTER paragraph.

Native Collating Sequence. The collating sequence associated with the computer specified in the OBJECT-COMPUTER paragraph.

Negated Combined Conditions. The 'NOT' logical operator immediately followed by a parenthesized combined condition.

Negated Simple Condition. The 'NOT' logical operator immediately followed by a simple condition.

Next Executable Sentence. The next sentence to which control will be transferred after execution of the current statement is complete.

Next Executable Statement. The next statement to which control will be transferred after execution of the current statement is complete.

Next Record. The record which logically follows the current record of a file.

Noncontiguous Items. Elementary data items, in the Working-Storage and Linkage Sections, which bear no hierarchic relationship to other data items.

Nonnumeric Item. A data item whose description permits its content to be composed of any combination of characters taken from the computer's character set. Certain categories of nonnumeric items may be formed from more restricted character sets.

Nonnumeric Literal. A literal bounded by quotation marks. The string of characters may include any character in the computer's character set. To

represent a single quotation mark character within a nonnumeric literal, two contiguous quotation marks must be used. If the literal is bounded by double quotation marks it is a hexadecimal literal and can contain only the characters '0' through '9' and 'A' through 'F'.

Numeric Character. A character that belongs to the following set of digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Numeric Item. A data item whose description restricts its content to a value represented by characters chosen from the digits '0' through '9'; if signed, the item may also contain a '+', '-', or other representation of an operational sign. (See The SIGN Clause.)

Numeric Literal. A literal composed of one or more numeric characters that also may contain either a decimal point, or an algebraic sign, or both. The decimal point must not be the rightmost character. The algebraic sign, if present, must be the leftmost character. A floating point literal is formed by two numeric literals separated by the character 'E'. Both literals may be signed. The first, the mantissa, must contain a decimal point. The second, the exponent, must not contain a decimal point.

OBJECT-COMPUTER. The name of an Environment Division paragraph in which the computer environment, within which the object program is executed, is described.

Object Computer Entry. An entry in the OBJECT-COMPUTER paragraph of the Environment Division which contains clauses which describe the computer environment in which the object program is to be executed.

Object of Entry. A set of operands and reserved words, within a Data Division entry of a COBOL program, that immediately follows the subject of the entry.

Object Program. A set or group of executable machine language instructions and other material designed to interact with data to provide problem solutions. In this context, an object program is generally the machine language result of the operation of a COBOL compiler on a source program. Where there is no danger of ambiguity, the word 'program' alone may be used in place of the phrase 'object program'.

Object Time. The time at which an object program is executed. The term is synonymous with execution time.

Obsolete Element. A COBOL language element that is to be deleted from the next revision of American National Standard COBOL.

Open Mode. The state of a file after execution of an OPEN statement for that file and before the execution of a CLOSE statement for that file. The particular open mode is specified in the OPEN statement as either INPUT, OUTPUT, I-O or EXTEND.

Operand. Whereas the general definition of operand is 'that component which is operated upon', for the purposes of this document, any lowercase word (or words) that appears in a statement or entry format may be considered to be an operand and, as such, is an implied reference to the

Report Description Entry. An entry in the Report Section of the Data Division that is composed of the level indicator RD, followed by a report name, followed by a set of report clauses as required.

Report File. An output file whose file description entry contains a REPORT clause. The contents of a report file consist of records that are written under control of the Report Writer Control System.

Report Footing. A report group that is presented only at the end of a report.

Report Group. In the Report Section of the Data Division, an 01 level-number entry and its subordinate entries.

Report Group Description Entry. An entry in the Report Section of the Data Division that is composed of the level-number 01, an optional data-name, a TYPE clause, and an optional set of report clauses.

Report Heading. A report group that is presented only at the beginning of a report.

Report Line. A division of a page representing one row of horizontal character positions. Each character position of a report line is aligned vertically beneath the corresponding character position of the report line above it. Report lines are numbered from 1, by 1, starting at the top of the page.

Report-Name. A user-defined word that names a report described in a report description entry within the Report Section of the Data Division.

Report Section. The section of the Data Division that contains one or more report description entries and their associated report group description entries.

Report Writer Control System (RWCS). An object time control system that accomplishes the construction of reports.

Report Writer Logical Record. A record that consists of the report writer print line and associated control information necessary for its selection and vertical positioning.

Reserved Word. A COBOL word specified in the list of words which may be used in COBOL source programs, but which must not appear in the programs as user-defined words or system-names.

Resource. A facility or service, controlled by the operating system, that can be used by an executing program.

Resultant Identifier. A user-defined data item that is to contain the result of an arithmetic operation.

Root Link Unit A module, created by the linker, consisting of one or more object programs which is always memory resident. In a run unit containing a single link unit, this link unit is the root link unit.

Routine-Name. A user-defined word that identifies a procedure written in a language other than COBOL.

Run Unit. One or more link units which interact with one another and which function, at object time, as an entity to provide problem solutions.

RWCS. Report Writer Control System; an object time control system that accomplishes the construction of reports.

Screen Clause. A clause, in the Screen Section of the Data Division, that appears in a screen data description entry.

Screen Description Entry. An entry in the Screen Section of the Data Division that is composed of a level-number followed by a data-name, if required, and then followed by a set of data clauses, as required.

Screen Data Item. A unit of data (excluding literals) defined in the Screen Section of the Data Division of the COBOL program.

Screen Line. A division of a screen representing one row of horizontal character positions. Each character position of a screen line is aligned vertically beneath the corresponding character position of the screen line above it. Screen lines are numbered from 1, by 1, starting at the top of the screen.

Screen-Name. A user-defined word that names a data item described in a screen data description entry in the Data Division. When used in the general formats, 'screen-name' represents a word which can not be qualified unless specifically permitted by the rules for that format.

Screen Section. The section of the Data Division that contains one or more screen description entries.

Section. A set of zero, one, or more paragraphs or entries, called a section body, the first of which is preceded by a section header. Each section consists of the section header and the related section body.

Section Header. A combination of words followed by a separator period that indicates the beginning of a section in the Environment, Data and Procedure Division.

In the Environment and Data Divisions, a section header is composed of reserved words followed by a separator period.

The permissible section headers in the Environment Division are:

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

The permissible section headers in the Data Division are:

```
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
SCREEN SECTION.
REPORT SECTION.
```

In the Procedure Division, a section header is composed of a section-name, followed by the reserved word SECTION, followed by a segment-number (optional), followed by a separator period.

Section-Name. A user-defined word which names a section in the Procedure Division.

Segment-Number. A user-defined word which classifies sections in the Procedure Division for purposes of segmentation. Segment-numbers may contain only the characters '0', '1', ..., '9'. A segment-number may be expressed either as a one or two digit number.

Sentence. A sequence of one or more statements, the last of which is terminated by a separator period.

Separately Compiled Program. A program which, together with its contained programs, is compiled separately from all other programs.

Separator. A character or two contiguous characters used to delimit character-strings.

Sequential Access. An access mode in which logical records are obtained from or placed into a file in a consecutive predecessor-to-successor logical record sequence determined by the order of records in the file.

Sequential File. A file with sequential organization.

Sequential Organization. The permanent logical file structure in which a record is identified by a predecessor-successor relationship established when the record is placed into the file.

Sign Condition. The proposition, for which a truth value can be determined, that the algebraic value of a data item or an arithmetic expression is either less than, greater than, or equal to zero.

Simple Condition. Any single condition chosen from the set:

```
relation condition
class condition
condition-name condition
switch-status condition
sign condition
(simple-condition)
```

Sort File. A collection of records to be sorted by a SORT statement. The sort file is created and can be used by the sort function only.

Sort-Merge File Description Entry. An entry in the File Section of the Data Division that is composed of the level indicator SD, followed by a

file-name, and then followed by a set of file clauses as required.

SOURCE-COMPUTER. The name of an Environment Division paragraph in which the computer environment, within which the source program is compiled, is described.

Source Computer Entry. An entry in the SOURCE-COMPUTER paragraph of the Environment Division which contains clauses which describe the computer environment in which the source program is to be compiled.

Source Item. An identifier designated by a SOURCE clause that provides the value of a printable item.

Source Program. Although it is recognized that a source program may be represented by other forms and symbols, in this document it always refers to a syntactically correct set of COBOL statements. A COBOL source program commences with the Identification Division or a COPY statement. A COBOL source program is terminated by the end program header, if specified, or by the absence of additional source program lines.

Special Character. A character that belongs to the following set:

Character.	Meaning
+	plus sign
-	minus sign
*	asterisk
/	slant (virgule, slash)
=	equal sign
$	currency sign
,	comma (decimal point, full stop)
;	semicolon
.	period (decimal point)
"	quotation mark
(	left parenthesis
)	right parenthesis
>	greater than symbol
<	less than symbol

Special-Character Word. A reserved word which is an arithmetic operator or a relation character.

SPECIAL-NAMES. The name of an Environment Division paragraph in which implementor-names are related to user specified mnemonic-names.

Special Names Entry. An entry in the SPECIAL-NAMES paragraph of the Environment Division which contains clauses which provides means for specifying the currency sign; choosing the decimal point; relating implementor-names to user-specified mnemonic-names; and relating alphabet-names to character sets or collating sequences.

Special Registers. Compiler generated storage areas whose primary use is to store information produced in conjunction with the user of specific COBOL features.

Standard Data Format. The concept used in describing data in a COBOL Data Division under which the characteristics or properties of the data are expressed in a form oriented to the appearance of the data on a printed page of infinite length and breadth, rather than a form oriented to the manner in which the data is stored internally in the computer or on a particular external medium.

Statement. A syntactically valid combination of words and symbols written in the Procedure Division beginning with a verb.

Subject of Entry. An operand or reserved word that appears immediately following the level indicator or the level-number in a Data Division entry.

Subprogram. A program which is the object of a CALL statement combined at object time with the calling program to produce a run unit. The term is synonymous with called program.

Subscript. An occurrence number represented by either an integer, a data-name optionally followed by an integer with the operator + or -, or an index-name optionally followed by an integer with the operator + or -, which identifies a particular element in a table.

Subscripted Data-Name. An identifier that is composed of a data-name followed by one or more subscripts enclosed in parentheses.

Sum Counter. A signed numeric data item established by a SUM clause in the Report Section of the Data Division. The sum counter is used by the Report Writer Control System to contain the result of designated summing operations that take place during production of a report.

Switch-Status Condition. The proposition, for which a truth value can be determined, that a switch, capable of being set to an 'on' or 'off' status, has been set to a specific status.

System-Name. A COBOL word which is used to communicate with the operating environment.

Table. A set of logically consecutive items of data that are defined in the Data Division by means of the OCCURS clause.

Table Element. A data item that belongs to the set of repeated items comprising a table.

Text Word. A character or sequence of contiguous characters between margin A and margin R in a COBOL library, source program, or in pseudo-text which is:

1. A separator, except for: space; a pseudo-text delimiter; and the opening and closing delimiters for nonnumeric literals. The right parenthesis and left parenthesis characters, regardless of context within the library, source program, or pseudo-text, are always considered text words.

2. A literal including, in the case of nonnumeric literals, the opening quotation mark and the closing quotation mark which bound the

literal.

3. 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.

Top Margin. An empty area which precedes the page body.

Truth Value. The representation of the result of the evaluation of a condition in terms of one of two values: true, false.

Unary Operator. A plus (+) or a minus (-) sign, which precedes a variable or a left parenthesis in an arithmetic expression and which has the effect of multiplying the expression of +1 or -1 respectively.

Unit. A discrete portion of a storage medium that contains part of a file, all of a file, or any number of files. The term is synonymous with reel and volume.

Unsuccessful Execution. The attempted execution of a statement that does not result in the execution of all the operations specified by that statement. The unsuccessful execution of a statement does not affect any data referenced by that statement, but may affect status indicators.

User-Defined Word. A COBOL word that must be supplied by the user to satisfy the format of a clause or statement.

Variable. A data item whose value may be changed by execution of the object program. A variable used in an arithmetic expression must be a numeric elementary item.

Variable Occurrence Data Item. A variable occurrence data item is a table element which is repeated a variable number of times. Such an item must contain an OCCURS DEPENDING ON clause in its data description entry, or be subordinate to such an item.

Verb. A word that expresses an action to be taken by a COBOL compiler or object program.

Volume. A discrete portion of a storage medium that contains part of a file, all of a file, or any number of files. The term is synonymous with reel and unit.

Word. A character-string of not more than 30 characters which forms a user-defined word, a system-name, or a reserved word. (See COBOL Words.)

Working-Storage Section. The section of the Data Division that describes working storage data items, composed either of noncontiguous items or of working storage records or of both.

77-Level-Description-Entry. A data description entry that describes a noncontiguous data item with the level-number 77.

C. I-O STATUS VALUES

THE I-O STATUS CONDITIONS

Upon completion of the input-output operation the value in the I-O status expresses the state of the outcome of the operation. The I-O status is a two character field that defines the condition and the cause of the error. The conditions are differentiated by the first character.

0x Successful Completion condition. The input-output statement was successfully executed.

1x At End condition. A sequential READ statement was unsuccessfully executed as a result an at end condition.

2x Invalid Key condition. The input-output statement was unsuccessfully executed as a result of an invalid key condition.

3x Permanent Error condition. The input-output statement was unsuccessfully executed as the result of an error that precluded further processing of the file.

4x Logical Error condition. The input-output statement was unsuccessfully executed as a result of an improper sequence of input-output operations that were performed on the file.

9x File system error condition, nonpermanent. The input-output statement was unsuccessfully executed as a result of a condition specifically related to the MOS file system.

Ax File system error condition, permanent. The input-output statement was unsuccessfully executed as a result of a condition specifically related to the MOS file system.

I-O STATUS VALUES

The following is a list of values that may be placed in the I-O status item following the execution of an input-output operation on a file. The file organization for which the value is appropriate is indicated in parentheses at the end of each item.

Successful Completion

00 The input-output statement was successfully executed and no further information is available concerning the input-output operation. (All)

02 The input-output statement was successfully executed but a duplicate key was detected. (Indexed)

- For a READ statement the key value for the current key of reference is equal to the value of the same key in the next record within the current key of reference. This occurs on the execution of a READ for the second through n duplicate keys read, but not the first.

- For a REWRITE or WRITE statement, the record just written created a duplicate key value for at least one alternate record key for which duplicates are allowed.

04 A READ statement was successfully executed but the length of the record being processed does not conform to the fixed file attributes for that file, i.e. too large. (Sequential)

05 An OPEN statement was successfully executed but the referenced optional file was not present at the time the OPEN was executed. If the open mode was I-O or EXTEND, the file has been created. (All)

07 The input-output statement was successfully executed. However, for a CLOSE or an OPEN with the NO REWIND phrase, the referenced file is on a non-reel/unit medium. (Sequential)

0A An OPEN statement for a text-file is successfully executed but the requested initial file allocation indicated in the VALUE OF FILE-SIZE clause is not available. The maximum amount available is reserved for the file. (Sequential)

0B An OPEN statement is successfully executed but the dynamic specification of a VALUE OF parameter (FILE -SIZE, ALLOC-UNIT, TIME-OUT, PAGE-SIZE, or SPLIT- STRATEGY) is illegal. The default value for the parameter is used. (All)

At End Condition With Unsuccessful Completion

10 A sequential READ was attempted and no next (previous) logical record exists in the file because the end (beginning) of the file has been reached. (All)

14 A sequential READ was attempted for a relative file and the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file. (Relative)

15 A sequential READ was attempted for the first time on an optional input file that is not present. (All)

16 A sequential READ was attempted and the at end condition already exists. (All)

Invalid Key Condition With Unsuccessful Completion

21 A sequence error exists for a sequentially accessed indexed file. The prime record key value has been changed by the program between the successful execution of a READ and the execution of the next REWRITE for the file, or the ascending sequence requirements for successive record key values are violated. (See The WRITE statement.) (Indexed)

22 An attempt was made to write a record that would create a duplicate key (relative), or duplicate prime record key, or to write or rewrite a record that would create a duplicate alternate record key without the DUPLICATES phrase specified. (Indexed)

23 An attempt was made to randomly access a record that does not exist in the file. (Relative, Indexed)

24 An attempt was made to write beyond the externally defined boundaries of an indexed or relative file (out of space on the medium). Or, a sequential WRITE was attempted for a relative file and the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file. In the first case, the user may release space by executing DELETE FILE statements. Space must be available before any WRITE for the file can be successfully executed. (Relative, Indexed)

25 A START or random READ statement is attempted on an optional input file that is not present. (Relative, Indexed)

Permanent Error Condition With Unsuccessful Completion

30 A permanent error exists and no further information is available concerning the input-output operation. If the file control entry for the file specifies ASSIGN TO data-name, stays in effect until the file is successfully opened. Otherwise, if it specifies ASSIGN TO literal, stay in effect for the rest of the run unit. (All)

34 Attempt to write beyond the externally defined boundaries of a sequential file (out of space for a magnetic tape file). Stays in effect until the file is closed. (Sequential)

35 An OPEN with the INPUT, I-O or EXTEND phrase was attempted on a non-optional file that is not present. Stays in effect until the file is successfully opened. (All)

37 An OPEN statement was attempted on a file that is required to be a mass storage file but is not a mass storage file. Stays in effect until the file is successfully opened. (All)

38 An OPEN was attempted on a file previously closed with lock. Stays in effect for the rest of the execution of this run-unit. (All)

39 The OPEN statement is unsuccessful because a conflict has been detected between the fixed file attributes and the attributes specified for that file in the program. If the file control entry for specifies ASSIGN TO data-name, stays in effect until the file is successfully opened. Otherwise, if it specifies ASSIGN TO literal, stays in effect for the rest of the run unit. (All)

Logic Error Condition With Unsuccessful Completion

41 An OPEN was attempted for a file in the open mode. (All)

42 A CLOSE was attempted for a file not in the open mode. (All)

43 In sequential access mode, the last input-output statement executed for the associated file prior to the attempted DELETE or REWRITE was not a successfully executed READ. (All)

46 A sequential READ was attempted on a file open in the INPUT or I-O mode and no valid next record has been established because:

- The preceding START was unsuccessful (Relative, Indexed), or
- The preceding READ was unsuccessful, but did not cause an at end condition. (All)

47 A READ or START was attempted on a file not open in the INPUT or I-O mode. (All)

48 A WRITE was attempted on a file not open in the I-O, OUTPUT or EXTEND mode. (All)

49 A DELETE or REWRITE was attempted on a file not open in the I-O mode. (Relative, Indexed)

MOS File System Nonpermanent Condition With Unsuccessful Completion

90 A DELETE FILE was attempted on a file which is not closed, logic error. (All)

91 A WRITE was attempted for a file open in the I-O mode with sequential access. (Relative, Indexed)

92 A DELETE, REWRITE or WRITE was attempted on a record which is locked. (Relative, Indexed)

93 A LOCK or UNLOCK was attempted on a file which is not open, logic error. (Relative,Indexed)

94 Attempt to LOCK a record which is already locked. (Relative, Indexed)

95 A DELETE FILE was attempted on a file which is not removable, that is a printer. (Sequential)

96 A WRITE was attempted for a device not ready (system and work station printers only). Note that before this condition occurs, a message is displayed followed by a wait of two minutes during which the device can be made available. (Sequential)

97 An attempt was made to exceed the maximum number of records LOCKed at one time. (Relative, Indexed)

98 A DELETE FILE was attempted on a file which does not exist. (All)

99 Attempt to LOCK or UNLOCK a nonexistent record using an alternate key. (Relative, Indexed)

9A A WRITE statement was attempted for a complete file and there is no more disk space available. The user may release space by executing DELETE FILE statements. Space must be available before any WRITE for the file can be successfully executed. (Sequential)

9B The execution of an input-output statement for a file resident on another machine is unsuccessful due to the line or machine being down. (All)

9C The execution of a LOCK or UNLOCK statement is attempted on an optional input file that is not present. (Relative, Indexed)

MOS File System Permanent Condition With Unsuccessful Completion

A0 Invalid path name (OPEN, DELETE FILE statements). If the file control entry for the file specifies ASSIGN TO data-name, stays in effect until the file successfully opened. Otherwise, if it specifies ASSIGN TO literal stays in effect for the rest of the run unit. (All)

A1 An OPEN was attempted on a file which is locked or an OPEN with LOCK was attempted on a file which is already opened by another process. Stays in effect until the file is successfully opened. (Relative, Indexed)

A2 An OPEN was attempted for a file on a device which is not ready. Stays in effect until the file is successfully opened. Note that before this condition occurs, a message is displayed followed by a wait of two minutes during which the device can be made available. (All)

A3 An OPEN with the INPUT phrase was attempted for a device which may not be open for input. Stays in effect until the file is successfully opened. (Sequential)

A4 An OPEN with the I-O phrase was attempted for a text-file. Stays in effect until the file is successfully opened. (Sequential)

A5 An OPEN was attempted which would result in too many files open at one time. Stays in effect until the file is successfully opened.(All)

A6 Illegal path-name detected during the execution of an OPEN or DELETE FILE. The dynamic specification of the path-name from the ASSIGN clause does not obey the syntax rules of a legal path-name. Stays in effect until the file is successfully opened. (All)

A7 The execution of an OPEN statement causes the creation of a file when there is not enough space available on the medium (disk). The user must change to a different medium; release space by executing DELETE FILE statements for unnecessary files; or if the file description entry for the file specifies the data-name form of the VALUE OF FILE-SIZE phrase, by reducing the value in the indentifier referenced by data-name . Stays in effect until the file is successfully opened. (All)

A8 Security violation. Stays in effect until the file is successfully opened. (All)

A9 The execution of an OPEN statement for a file resident on another machine is unsuccessful due to the line or machine being down.Stays in effect until the file is successfully opened. (All)

AA The execution of an input-output statement for a file resident on another machine is unsuccessful due to an invalid file connector. This occurs because the line or remote machine which was down as become available. Stays in effect until the file is successfully opened. (All)

D. ANALYTICAL INDEX

Since all statement and clause pages are individually numbered using the names of the statements/clauses, items referenced on those pages have been given page numbers in the following format in the index:

"chapter number"-"name"

although the chapter number does not appear on the pages in the body of the text. For example, 5-ACCEPT refers to the ACCEPT statement in chapter 5. Where the statement/clause description runs to several pages, individual page numbers within the description have been included. For example, 4-PICTURE-3 refers to the third page of the PICTURE clause in chapter 4.

"A" PICTURE symbol, 4-PICTURE-3
Abbreviated combined relation conditions, 5-11
ACCEPT statement, 5-ACCEPT
    Imperative statement, 2-47
    Mnemonic-name, 2-4
    Screen Handling, 6-ACCEPT
    SPECIAL-NAMES paragraph, 3-10
ACCESS MODE clause, 3-ACCESS MODE
Access modes, 2-26
ADD statement, 5-ADD
    Composite of operands, 5-16
    COMPUTE statement, 5-COMPUTE
    Conditional Statement, 2-46
    CORRESPONDING (CORR), 5-ADD
    Data conversion, 5-16
    Decimal alignment, 5-16
    Imperative statement, 2-47
    Maximum operand size, 5-16
    Multiple results, 5-17
ADVANCING phrase, 5-WRITE
AFTER phrase
    INSPECT statement, 5-INSPECT
    PERFORM statement, 5-PERFORM
    WRITE statement, 5-WRITE
Algebraic sign, 4-5
Alignment of data, 4-5
    ACCEPT statement, 5-ACCEPT
    MOVE statement, 5-MOVE
ALL
    INSPECT statement, 5-INSPECT
    SEARCH statement, 5-SEARCH
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
ALL literal, 2-9
    DISPLAY statement, 5-DISPLAY
    INSPECT statement, 5-INSPECT
    STOP statement, 5-STOP
    STRING statement, 5-STRING
    UNSTRING statement, 5-UNSTRING
ALL PROCEDURES phrase, 10-USE FOR DEBUGGING
ALL REFERENCES OF phrase, 10-USE FOR DEBUGGING
Alphabet-name, 2-3
    CODE-SET clause, 4-CODE-SET
    MERGE statement, 9-MERGE
    SORT statement, 9-SORT
Alphabet-name clause, 3-10
Alphabetic category, 4-4, 4-PICTURE-1, 5-MOVE
Alphabetic class, 4-4
Alphanumeric character, 4-4
Alphanumeric edited category, 4-PICTURE-2, 5-MOVE
ALSO phrase, 3-10
ALTERNATE RECORD KEY clause 3-ALTERNATE RECORD KEY
AND
    Abbreviated combined relation condition, 5-11
    Combined condition, 5-10
    Precedence, 5-12
    Negated combined condition, 5-10
Arithmetic expression, 5-3
    COMPUTE statement, 5-COMPUTE
    Relation condition, 5-5
    Sign condition, 5-9
Arithmetic operator, 5-3, B-2
Arithmetic statements, 5-3
ASCENDING KEY phrase
    MERGE statement, 9-MERGE
    OCCURS clause, 4-OCCURS
    SORT statement, 9-SORT
ASSIGN clause, 3-ASSIGN
    Sort-Merge, 9-3
Asterisk (*) comment line, 2-54
Asterisk (*) PICTURE symbol, 4-PICTURE-4
AT END condition, 2-31
    READ statement, 5-READ
    RETURN statement, 9-RETURN
    I-O status, 2-32
    USE statement, 5-USE
AT END phrase
    READ statement, 5-READ
    RETURN statement, 9-RETURN
    SEARCH statement, 5-SEARCH
    USE statement, 5-USE
AUTHOR paragraph, 3-2
AUTO clause, 6-AUTO

"B" PICTURE symbol, 4-PICTURE-3
BEFORE phrase
    INSPECT statement, 5-INSPECT
    WRITE statement, 5-WRITE
BELL clause, 6-BELL
Binary arithmetic operators, 5-3
BLANK clause, 6-BLANK
BLANK WHEN ZERO clause, 4-BLANK WHEN ZERO
    PICTURE clause, 4-PICTURE
    VALUE clause, 4-VALUE
BLINK clause, 6-BLINK

BLOCK CONTAINS clause, 4-BLOCK CONTAINS
    Report Writer, 11-10
Body group presentation rules, 11-36
BOX clause, 6-BOX
Braces, 1-6
Brackets, 1-6
BY
    COPY statement, 2-COPY
    DIVIDE statement, 5-DIVIDE
    INSPECT statement, 5-INSPECT
    MULTIPLY statement, 5-MULTIPLY
    PERFORM statement, 5-PERFORM

CALL statement, 7-CALL
    CANCEL statement, 7-CANCEL
    Imperative statement, 2-47
    Linkage Section, 4-27, 7-15
    Procedure Division header, 2-44
Called program, 7-1
Calling program, 7-8
CANCEL statement, 7-CANCEL
    CALL statement, 7-CALL
    CLOSE statement, 5-CLOSE
    EXIT PROGRAM statement, 7-EXIT PROGRAM
    Imperative statement, 2-47
Category of data, 4-PICTURE-1
    Editing, 4-PICTURE-4
    MOVE statement, 5-MOVE
    Nonnumeric literal, 2-7
    Numeric literal, 2-8
    PICTURE clause, 4-PICTURE
    VALUE clause, 4-VALUE
Category of statements, 2-48
CF (See CONTROL FOOTING)
CH (See CONTROL HEADING)
Character, 2-1
    Alphabetic, B-1
    Alphanumeric, B-1
    Editing, B-7
    Numeric, B-14
    Punctuation, B-17
    Relation, B-18
    Special, B-22
Character representation, 4-4
Character set, B-3
Character-string, 2-2
CHARACTERS
    OBJECT-COMPUTER paragraph, 3-8
    BLOCK CONTAINS clause, 4-BLOCK CONTAINS
    RECORD CONTAINS clause, 4-RECORD CONTAINS
Class condition, 5-7
Class of data, 4-4
    Incompatible, 5-17
Clause, 2-1, 2-56
CLOSE statement, 5-CLOSE
    AT END condition, 2-31
    Imperative statement, 2-47
    I-O status, 2-31
    OPEN statement, 5-OPEN
    READ statement, 5-READ
    Report Writer, 11-CLOSE
    TERMINATE statement, 11-TERMINATE
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
COBOL character set, 2-1, B-3
COBOL word, 2-3
CODE clause, 11-CODE
CODE-SET clause, 4-CODE-SET
    Report Writer, 11-10
COLLATING SEQUENCE clause, 3-8
COLLATING SEQUENCE phrase
    MERGE statement, 9-MERGE
    SORT statement, 9-SORT
COLUMN clause, 6-COLUMN
COLUMN NUMBER clause, 11-COLUMN NUMBER
Combined condition, 5-10
Comma
    DECIMAL-POINT IS COMMA clause, 3-10
    Identifier, 2-18
    Indices, 2-22
    Interchangeable with semicolon, 1-7
    Library text-word, 2-COPY
    PICTURE symbol, 4-PICTURE-3
    Separator, 2-1
    Subscripts, 2-18
Comment-entry, 2-11
    DATE-COMPILED paragraph, 3-4
Comment line, 2-54
    Debugging line, 10-12
    Library text, 2-COPY
    WITH DEBUGGING MODE clause, 10-WITH DEBUGGING MODE
COMP, 4-USAGE
COMP1 to 3, 4-USAGE
Compiler directing sentence, 2-47
Compiler directing statement, 2-46
Complex condition, 5-9
COMPUTATIONAL (COMP), 4-USAGE
COMPUTATIONAL1 to 3, 4-USAGE
COMPUTE statement, 5-COMPUTE
    Composite of operands, 5-16
    Conditional statement, 2-46
    Data conversion, 5-16
    Decimal alignment, 5-16
    Imperative statement, 2-47
    Maximum operand size, 5-16
    Multiple results, 5-17

Computer name, 3-7, A-4
Condition
    Abbreviated combined relation condition, 5-11
    Class condition, 5-7
    Combined condition, 5-10
    Complex condition, 5-9
    Condition-name condition, 5-8
    Evaluation rules, 5-12
    IF statement, 5-IF
    Negated combined condition, 5-10
    Negated simple condition, 5-10
    PERFORM UNTIL statement, 5-PERFORM
    Relation condition, 5-5
    Sign condition, 5-9
    Simple condition, 5-5
    SIZE ERROR condition, 5-14
    Switch-status condition, 5-9
Condition-name, 2-19
    Indexed, 2-22
    Level-number, 4-3
    Qualified, 2-14
    REDEFINES clause, 4-REDEFINES
    SPECIAL-NAMES paragraph, 3-10
    Subscripted, 2-17
    VALUE clause, 4-VALUE, 6-VALUE
Condition-name condition, 5-8
Condition-name data description entry, 4-30, 4-VALUE
Conditional expression, 5-5
Conditional sentence, 2-46
Conditional statement, 2-46
Conditional variable, 5-8
Configuration Section, 3-6
Continuation line, 2-54
Continuation of lines, 2-54
    Comment-entries, 2-11
    Comment lines, 2-52
    Debugging lines, 10-12
    Library pseudo-text, 2-55
Continued line, 2-54
CONTROL clause, 11-CONTROL
Control break
    CONTROL clause, 11-CONTROL
    GENERATE statement, 11-GENERATE
    GROUP INDICATE clause, 11-GROUP INDICATE
    TYPE clause, 11-TYPE
CONTROL FOOTING
    Body group presentation rules, 11-36
    Presentation rules table, 11-28
CONTROL HEADING
    Body group presentation rules, 11-31
    Presentation rules table, 11-28
COPY statement, 2-COPY
    Compiler directing statement, 2-46
CORR
CORRESPONDING (CORR) phrase, 5-15
    ADD statement, 5-ADD
    MOVE statement, 5-MOVE
    SIZE ERROR phrase, 5-14
    SUBTRACT statement, 5-SUBTRACT
CR PICTURE symbol, 4-PICTURE-4
Crossfooting, 11-SUM, 11-TIME
Currency PICTURE symbol, 4-PICTURE-4
Currency sign, 3-11, 4-PICTURE-4
CURRENCY SIGN clause, 3-11, 4-PICTURE-4
Currency symbol, 4-PICTURE-4
Current record pointer
    DELETE statement, 5-DELETE
    OPEN statement, 5-OPEN
    READ statement, 5-READ
    REWRITE statement, 5-REWRITE
    START statement, 5-START
    WRITE statement, 5-WRITE

Data description entry, 4-30
    Linkage Section, 4-27, 7-15
    Working-Storage Section, 4-26
Data Division, 2-42, 4-1
    Inter-Program Communication, 7-15
    Reference format, 2-56
    Report Writer, 11-9
    Screen Handling, 6-1
    Sort-Merge, 9-7
Data-name, 4-DATA-NAME or FILLER, 11-DATA-NAME
    Identifier, 2-18
    Indexed, 2-22
    Qualified, 2-14
    Subscripted, 2-18
DATA-NAME clause, 11-DATA-NAME
DATA RECORDS clause, 4-DATA RECORDS
    Sort-Merge, 9-8
DATE, 5-ACCEPT
DATE-COMPILED paragraph, 3-4
DATE-WRITTEN paragraph, 3-2
DAY, 5-ACCEPT
DB PICTURE symbol, 4-PICTURE-4
DE (See DETAIL)
DEBUG-CONTENTS, 10-USE FOR DEBUGGING
DEBUG-ITEM, 10-USE FOR DEBUGGING
DEBUG-LINE, 10-USE FOR DEBUGGING
DEBUG-NAME, 10-USE FOR DEBUGGING
DEBUG-SUB-1, 10-USE FOR DEBUGGING
DEBUG-SUB-2, 10-USE FOR DEBUGGING
DEBUG-SUB-3, 10-USE FOR DEBUGGING

Debugging line, 10-12
Library text, 2-COPY
DEBUGGING MODE clause, 10-WITH DEBUGGING MODE
Compile time switch, 10-2
Debugging lines, 10-12
Debugging section, 10-4
Decimal point
Actual, 4-PICTURE-7
Alignment, 4-5
Assumed, 4-PICTURE-3
DECIMAL POINT IS COMMA clause, 3-10
Declarative-sentence, 2-44
Declaratives, 2-44
Reference format, 2-45
Segmentation, 8-1
USE BEFORE REPORTING stmnt, 11-USE BEFORE REPORTING
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
USE statement, 7-USE
Definitions, B-1
DELETE statement, 5-DELETE
OPEN mode, 2-27, 5-OPEN
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
Delimiters
Character-string, 2-2
Pseudo-text, 2-1
DEPENDING phrase
GO TO statement, 5-GO TO
OCCURS clause, 4-OCCURS
DESCENDING KEY phrase
MERGE statement, 9-MERGE
OCCURS clause, 4-OCCURS
SORT statement, 9-SORT
DETAIL (DE), 11-TYPE
DISPLAY in USAGE clause, 4-USAGE
DISPLAY statement, 5-DISPLAY
Figurative constant, 2-8
Imperative statement, 2-45
Mnemonic-name, 2-4
Screen Handling, 6-DISPLAY
SPECIAL-NAMES paragraph, 3-10
DIVIDE statement, 5-DIVIDE
Composite of operands, 5-16
COMPUTE statement, 5-COMPUTE
Conditional statement, 2-46
Data conversion, 5-16
Decimal alignment, 5-16
Imperative statement, 2-47
Maximum operand size, 5-16
Multiple results, 5-17
SIZE ERROR phrase, 5-14
Division Formats, 2-39
Division header, 2-55
DOWN BY, 5-SET
DUPLICATES phrase, 3-21

Editing characters, B-7
Editing rules, 4-PICTURE-4
Editing sign, 4-5
Elementary item, 4-3
Noncontiguous, 4-26
Elements, 1-5
Ellipsis, 1-6
ELSE clause, 5-IF
END DECLARATIVES, 2-44, 5-1
END-OF-PAGE phrase, 5-WRITE
Entry, B-8
Environment Division, 2-40, 3-5
Debug, 10-2
Report Writer, 11-5
Segmentation, 8-3
Sort-Merge, 9-3
EQUAL TO relation, 5-5
EOP phrase, 5-WRITE
Execution, 2-44
EXIT statement, 5-EXIT
Imperative statement, 2-47
EXIT PROGRAM statement, 7-EXIT PROGRAM
CALL statement, 7-CALL
CANCEL statement, 7-CANCEL
Explicit, 2-50
Exponentiation, 5-3

FD level indicator, 2-56, 4-9, 7-17
Report Writer, 11-14
Figurative constant, 2-9
DISPLAY statement, 5-DISPLAY
INSPECT statement, 5-INSPECT
STOP statement, 5-STOP
STRING statement, 5-STRING
UNSTRING statement, 5-UNSTRING
VALUE clause, 4-VALUE, 6-VALUE
VALUE OF clause, 4-VALUE OF
File control entry, 3-15
Sort-Merge, 9-3
FILE-CONTROL paragraph, 3-14
Sort-Merge, 9-3
File description entry, 4-8
Report Writer, 11-10
File-name, 2-3
File Section, 4-7
Report Writer, 11-9
Sort-Merge, 9-7
FILE STATUS clause, 3-FILE STATUS
FILE STATUS data item, 3-FILE STATUS
FILLER, 4-DATA-NAME or FILLER
FILLER clause, 6-SCREEN-NAME or FILLER

FINAL
    CONTROL clause, 11-CONTROL
    SUM clause, 11-SUM
    TYPE clause, 11-TYPE
FIRST, 5-INSPECT
FIRST DETAIL, 11-PAGE
Floating insertion editing, 4-PICTURE-6
FOOTING, 11-PAGE, 11-TYPE
FOR, 5-INSPECT
Format punctuation, 1-7
FROM phrase
    ACCEPT statement, 5-ACCEPT
    PERFORM VARYING statement, 5-PERFORM
    RELEASE statement, 9-RELEASE
    SUBTRACT statement, 5-SUBTRACT
    WRITE statement, 5-WRITE
FULL clause, 6-FULL

General format, 1-5
General rules, 1-8
GENERATE statement, 11-GENERATE
    CONTROL clause, 11-CONTROL
    Data-name, 4-DATA-NAME or FILLER
    Imperative statement, 2-47
    SUM clause, 11-SUM
    TERMINATE statement, 11-TERMINATE
Generic terms, 1-5
GIVING phrase
    ADD statement, 5-ADD
    DIVIDE statement, 5-DIVIDE
    MERGE statement, 9-MERGE
    MULTIPLY statement, 5-MULTIPLY
    SORT statement, 9-SORT
    SUBTRACT statement, 5-SUBTRACT
Glossary of COBOL terms, Appendix B
GO TO statement, 5-GO TO
    Imperative statement, 2-45
    MERGE statement, 9-MERGE
    PERFORM statement, 5-PERFORM
    SEARCH statement, 5-SEARCH
    Segmentation, 8-1
    SORT statement, 9-SORT
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
GREATER THAN relation, 5-5
Group, 4-3
GROUP INDICATE clause, 11-GROUP INDICATE
    VALUE clause, 11-VALUE

HEADING, 11-PAGE, 11-TYPE
HEBREW clause, 6-HEBREW
HIGH-VALUE/HIGH-VALUES, 2-9
HIGHLIGHT clause, 6-HIGHLIGHT

Identification Division, 2-39, 3-1
Identifier, 2-18
IF statement, 5-IF
    Conditional statement, 2-46
    Imperative statement, 2-47
Imperative sentence, 2-47
Imperative statement, 2-47
Implementation of the standard, 1-3
Implementor-name
    Alphabet-name clause, 3-10
    ASSIGN clause, 3-ASSIGN
    SPECIAL-NAMES paragraph, 3-10
    VALUE OF clause, 4-VALUE OF
Implicit, 2-50
Implied relational operator, 5-11
Implied subject, 5-11
IN qualifier, 2-15
Incompatible data, 5-17
Indentation, 2-56
Index, 2-22
Index data item, 2-22
    Condition-name, 2-19
    CONTROL clause, 11-CONTROL
    Initial value, 4-2
    MOVE statement, 5-MOVE
Index-name, 2-3
    OCCURS clause, 4-OCCURS
    PERFORM statement, 5-PERFORM
    Relation condition, 5-5
    SEARCH statement, 5-SEARCH
    SET statement, 5-SET
Indexed file, 2-25
Indexing, 2-22
    Condition-name, 2-19
    Conditional variable, 5-8
    CONTROL clause, 11-CONTROL
    DEBUG-NAME, 10-USE FOR DEBUGGING
    MOVE statement, 5-MOVE
    OCCURS clause, 4-OCCURS
    Qualification, 2-14
    RETURN statement, 9-RETURN
    Subscripting, 2-17
Indicator area
    COPY statement, 2-COPY
    Debugging line, 10-12

INITIATE statement, 11-INITIATE
    GENERATE statement, 11-GENERATE
    Imperative statement, 2-47
    OPEN statement, 5-OPEN
    SUM clause, 11-SUM
    TERMINATE statement, 11-TERMINATE
    USE BEFORE REPORTING statement, 11-USE BEFORE REPORTING
Input-Output Section, 3-14
    Sort-Merge, 9-3
INPUT PROCEDURE phrase, 9-15
INSPECT statement, 5-INSPECT
    Imperative statement, 2-47
INSTALLATION paragraph, 3-2
Integer, B-11
INTO
    DIVIDE statement, 5-DIVIDE
    READ statement, 5-READ
    RETURN statement, 9-RETURN
    STRING statement, 5-STRING
INVALID KEY condition, 2-30
    DELETE statement, 5-DELETE
    READ statement, 5-READ
    REWRITE statement, 5-REWRITE
    START statement, 5-START
    WRITE statement, 5-WRITE
I-O CONTROL paragraph, 3-14, 3-28
    Sort-Merge, 9-1
I-O Status values, Appendix C

JUST
JUSTIFIED (JUST) clause, 4-JUSTIFIED
    Condition-name, 2-19
    Figurative constant, 2-9
    Screen Handling, 6-JUSTIFIED
    Standard alignment, 4-5
    USAGE IS INDEX clause, 4-USAGE
    VALUE clause, 4-VALUE

KEY data-names
    MERGE statement, 9-MERGE
    SORT statement, 9-SORT
KEY phrase
    OCCURS clause, 4-OCCURS
    READ statement, 5-READ
    SEARCH statement, 5-SEARCH
    START statement, 5-START
Key word, 2-5

LABEL RECORDS clause, 4-LABEL RECORDS
    Report Writer, 11-10
LAST DETAIL, 11-PAGE
LEADING
    INSPECT statement, 5-INSPECT
    SIGN clause, 4-SIGN
LEFT, 4-SYNCHRONIZED
LESS THAN relation, 5-5
Level indicator, 2-56
Level-number, 1-5, 2-13
    Data description entry, 4-3
    Qualifier, 2-14
    Reference format, 2-53
    Report group description entry, 11-LEVEL NUMBER
    Screen Handling, 6-LEVEL NUMBER
Library facility, 2-33
LINAGE clause, 4-LINAGE
LINAGE-COUNTER, 2-28
LINE-COUNTER, 11-16
    Final setting rules, 11-33, 11-35
    Special register, 2-28
    Subscripting, 2-17
LINE clause, 6-LINE
LINE NUMBER clause, 11-LINE NUMBER
    Notation, 11-29
    Sequence substitution, 11-30
Linkage Section, 4-27, 7-15
    VALUE clause, 4-VALUE
Literal
    CURRENCY SIGN clause, 3-10
    STOP statement, 5-STOP
Logical operator, 5-10
LOW-VALUE/LOW-VALUES, 2-9

MEMORY SIZE clause, 3-5
MERGE statement, 9-MERGE
    Imperative statement, 2-47
    OPEN statement, 5-OPEN
    Segmentation, 8-1
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
Minus(-) PICTURE symbol, 4-PICTURE-4
Mnemonic-name, 2-4
    ACCEPT statement, 5-ACCEPT
    DISPLAY statement, 5-DISPLAY
    SPECIAL-NAMES paragraph, 3-10
    WRITE statement, 5-WRITE
MOVE statement, 5-MOVE
    CORRESPONDING (CORR), 5-14
    Imperative statement, 2-47
    Index data item, 4-USAGE
    Overlapping operands, 5-16
MULTIPLE FILE clause, 3-28
Multiple results in arithmetics, 5-17
MULTIPLY statement, 5-MULTIPLY
    Composite of operands, 5-16
    COMPUTE statement, 5-COMPUTE
    Conditional statement, 2-46
    Data conversion, 5-16
    Imperative statement, 2-47

Maximum operand size, 5-16
Multiple results, 5-17
SIZE ERROR phrase, 5-14

NATIVE phrase, 3-10
Native character set, B-13
Native collating sequence, B-13
Negated condition, 5-10
NEXT phrase, 5-READ
NEXT GROUP clause, 11-NEXT GROUP
Body group presentation rules, 11-36
PAGE FOOTING presentation rules, 11-41
REPORT HEADING group presentation rules, 11-31
Saved next group integer, 11-30
NEXT PAGE phrase, 11-NEXT PAGE
LINE NUMBER clause, 11-LINE NUMBER
NEXT GROUP clause, 11-NEXT GROUP
NEXT SENTENCE phrase
IF statement, 5-IF
SEARCH statement, 5-SEARCH
Noncontiguous elementary item, 4-26
Level-number "77", 4-3
Nonnumeric comparison, 5-6
Nonnumeric literal, 2-7
Continuation, 2-54
NOT
Logical operator, 5-9
Relational operator, 5-5
Notation rules, 1-5
Numeric category, 4-PICTURE-1
Numeric character, B-14
Numeric class, 5-7
Numeric comparison, 5-6
Numeric edited category, 4-PICTURE-2
Numeric literal, 2-8

OBJECT-COMPUTER paragraph, 3-8
Occurrence number, 2-21, 4-OCCURS
OCCURS clause, 4-OCCURS
CORRESPONDING PHRASE, 5-14
MOVE statement, 5-MOVE
REDEFINES clause, 4-REDEFINES
RENAMES clause, 4-RENAMES
SEARCH statement, 5-SEARCH
SYNCHRONIZED clause, 4-SYNCHRONIZED
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
VALUE clause, 4-VALUE
OF qualifier, 2-15
OFF STATUS phrase, 3-10
ON SIZE ERROR phrase, 5-14
ON STATUS phrase, 3-10
OPEN statement, 5-OPEN
CLOSE statement, 5-CLOSE
Imperative statement, 2-47
INITIATE statement, 11-INITIATE
I-O status, 2-32
LINAGE clause, 4-LINAGE
READ statement, 5-READ
REPORT clause, 11-REPORT
Report Writer, 11-OPEN
REWRITE statement, 5-REWRITE
START statement, 5-START
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
WRITE statement, 5-WRITE
Operands, 5-6
Operational sign, 4-5
Operator
Arithmetic, 5-3
Logical, 5-9
Relational, 5-5
OPTIONAL phrase
CLOSE statement, 5-CLOSE
READ statement, 5-READ
Optional word, 2-6
OR phrase, 5-UNSTRING
OR
Logical operator, 5-10
Abbreviated combined relation condition, 5-11
Hierarchy, 5-12
ORGANIZATION IS clause, 3-ORGANIZATION
OUTPUT PROCEDURE phrase, 9-MERGE
OVERFLOW phrase
CALL statement, 7-CALL
STRING statement, 5-STRING
UNSTRING statement, 5-UNSTRING
Overlapping operands, 5-16
Overlays, 8-1

"P" PICTURE symbol, 4-PICTURE-3
PAGE
WRITE statement, 5-WRITE
PAGE clause, 11-PAGE
PAGE-COUNTER, 11-15
special register, 11-3
Subscripting, 2-17
PAGE FOOTING (PF)
PAGE FOOTING presentation rules, 11-41
PAGE HEADING (PH)
PAGE HEADING group presentation rules, 11-34

Paragraph, 2-54
Paragraph header, 2-55
Paragraph-name, 2-55
    Qualified, 2-14
Parentheses, 5-3
    Condition, 5-10
    Indices, 2-22
    PICTURE clause, 4-PICTURE-2
    Separators, 2-1
    Subscripts, 2-17, 2-21
PERFORM statement, 5-PERFORM
    Imperative statement, 2-47
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
    USE statement, 7-USE
Period
    Separator, 2-1
Period (.) PICTURE symbol, 4-PICTURE-4
PF (See PAGE FOOTING)
PH (See PAGE HEADING)
Phrase, 1-5
PIC clause, 4-PICTURE
PICTURE character-string, 2-11
PICTURE (PIC) clause, 4-PICTURE
    BLANK WHEN ZERO clause, 4-BLANK WHEN ZERO
    COMPUTATIONAL clause, 4-USAGE
    CURRENCY SIGN clause, 3-10
    DECIMAL POINT IS COMMA clse, 3-10
    Linkage Section, 4-27, 7-15
    Screen Section, 6-PICTURE
    SYNCHRONIZED clause, 4-SYNCHRONIZED
    USAGE IS INDEX clause, 4-USAGE
    Working-Storage Section, 4-26
Plus (+) PICTURE symbol, 4-PICTURE-4
POINTER phrase
    STRING statement, 5-STRING
    UNSTRING statement, 5-UNSTRING
Precedence rules for PICTURE
    character string, 4-PICTURE-8
Presentation rules tables, 11-28
Procedure, 2-44
Procedure Division, 2-45, 5-1
    Debug, 10-4
    Inter-Program Communication, 7-34
    Report Writer, 11-65
    Screen Handling, 6-29
    Sort-Merge, 9-9
Procedure Division header, 2-45
Procedure-name, B-16
    Qualifier, 2-14
PROGRAM-ID paragraph, 3-3
Program-name, 2-3, 3-3
    CALL statement, 7-CALL
    CANCEL statement, 7-CANCEL
PROGRAM COLLATING SEQUENCE clause, 3-8
Pseudo-text delimiters, 2-55
Punctuation characters, B-17
    Format punctuation, 1-7
    Separators, 2-1

Qualification, 2-14
    CONTROL clause, 11-CONTROL
    COPY statement, 2-COPY
    CORRESPONDING phrase, 5-15
    DEBUG-NAME, 10-USE FOR DEBUGGING
    LINE-COUNTER, 11-4
    Linkage Section, 4-27, 7-15
    MERGE statement, 9-MERGE
    OCCURS Clause, 4-OCCURS
    PAGE-COUNTER, 11-3
    READ statement, 5-READ
    RELEASE statement, 9-RELEASE
    RENAMES clause, 4-RENAMES
    REWRITE statement, 5-REWRITE
    SORT statement, 9-SORT
    START statement, 5-START
    VALUE OF clause, 4-VALUE OF
    Working-Storage Section, 4-26
    WRITE statement, 5-WRITE
Quotation mark separator, 2-1
QUOTE/QUOTES, 2-9

RD entry, 4-29, 11-13
RD level indicator, 2-54, 11-14
READ statement, 5-READ
    CLOSE statement, 5-CLOSE
    DELETE statement, 5-DELETE
    OPEN statement, 5-OPEN
    REWRITE statement, 5-REWRITE
    USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
Record
    Logical, 2-12
    Physical, 2-12
RECORD CONTAINS clause, 4-RECORD CONTAINS
    Report Writer, 11-10
    Sort-Merge, 9-8
Record description entry, 4-9
RECORD KEY clause, 3-RECORD KEY
Record-name, 2-3
REDEFINES clause, 4-REDEFINES
REEL, 5-CLOSE
Reference format, 2-56
    Text-words, 2-COPY
Relation character, B-18
Relation condition, 5-5
    Abbreviated combined, 5-11
    Index data item, 2-3, 5-7
    Index-name, 2-3, 5-7
    MERGE statement, 9-MERGE

Nonnumeric operands, 5-6
Numeric operands, 5-6
SORT statement, 9-SORT
Relational operator, B-18
Relative file, 2-25
Relative indexing, 2-22
RELATIVE KEY phrase
READ statement, 5-READ
REWRITE statement, 5-REWRITE
START statement, 5-START
WRITE statement, 5-WRITE
Relative record number, 3-14
RELEASE statement, 9-RELEASE
Imperative statement, 2-47
REMAINDER phrase, 5-DIVIDE
RENAMES clause, 4-RENAMES
Level-number, 4-3
REPLACING phrase, 2-COPY, 5-INSPECT
REPORT clause, 11-REPORT
Report description entry, 11-14
REPORT FOOTING (RF), 11-43
REPORT FOOTING presentation rules, 11-43
Report group description entry, 11-23
REPORT HEADING (RH)
REPORT HEADING group presentation rules, 11-31
Report-name, 2-3
Report Section, 11-13
REQUIRED clause, 6-REQUIRED
RESERVE clause, 3-RESERVE
Reserved word, 2-5
Reserved word list, Appendix A
RESET phrase, 11-SUM
RETURN statement, 9-RETURN
Conditional statement, 2-46
REVERSE clause, 6-REVERSE
REWRITE statement, 5-REWRITE
OPEN statement, 5-OPEN
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
RF (See REPORT FOOTING)
RH (See REPORT HEADING)
RIGHT, 4-JUSTIFIED, 4-SYNCHRONIZED
Rolling forward, 11-SUM
ROUNDED phrase, 5-14
RUN, 5-STOP

"S" PICTURE symbol, 4-PICTURE-3
SAME AREA clause, 3-SAME AREA
SAME RECORD AREA clause, 3-SAME AREA
Sort-Merge, 9-SAME AREA
SAME SORT AREA clause, 9-SAME AREA
SAME SORT-MERGE AREA clause, 9-SAME AREA
SD level indicator, 2-56
SCREEN-NAME, 6-SCREEN-NAME or FILLER
Screen Section, 4-28, 6-1
Scope of statements, 2-49
SEARCH statement, 5-SEARCH
Conditional statement, 2-46
USAGE IS INDEX clause, 4-USAGE
Section, 2-55
Section header, 2-45, 2-55
Section-name, 2-3
SECURE clause, 6-SECURE
SECURITY paragraph, 3-1
Segment, 8-1
SEGMENT-LIMIT clause, 8-4
Segment-number, 2-3, 8-6
Segmentation
CALL statement, 7-CALL
MERGE statement, 9-MERGE
SORT statement, 9-SORT
SELECT clause, 3-SELECT
Sort-Merge, 9-3
Semicolon
Interchangeable with comma, 1-7
Library text-word, 2-COPY
Punctuation character, B-17
Sentence, 2-46
Separator, 2-1
SEQUENCE clause, 3-8
Sequence number, 2-54
Sequential file, 2-24
SET statement, 5-SET
Imperative statement, 2-47
Overlapping operands, 5-16
SEARCH statement, 5-SEARCH
USAGE IS INDEX clause, 4-USAGE
SIGN clause, 4-SIGN
Class condition, 5-7
MOVE statement, 5-MOVE
Operational sign, 4-5
PICTURE clause, 4-PICTURE
Report Writer, 11-SIGN
Sign condition, 5-9
Simple condition, 5-5
SIZE ERROR phrase, 5-15
Conditional statement, 2-46
SORT statement, 9-SORT
Imperative statement, 2-47
OPEN statement, 5-OPEN
Segmentation, 8-1
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING

Sort-Merge file description entry, 9-8
SOURCE clause, 11-SOURCE
SOURCE-COMPUTER paragraph, 3-7
WITH DEBUGGING MODE phrase, 10-WITH DEBUGGING MODE
Source program, 2-53
COPY statement, 2-COPY
Space
Library text-word, 2-COPY
Separator, 2-1
SPACE/SPACES, 2-8
Special character, B-22
Special-character words, 1-7
SPECIAL-NAMES paragraph, 3-10
ACCEPT statement, 5-ACCEPT
Condition-name, 2-4
DISPLAY statement, 5-DISPLAY
Mnemonic-name, 2-4
Switch-status condition, 5-9
WRITE statement, 5-WRITE
Special registers, 2-6
DEBUG-ITEM, 10-USE FOR DEBUGGING
LINAGE-COUNTER, 2-28
LINE-COUNTER, 11-4
PAGE-COUNTER, 11-3
Standard alignment rules, 4-5
Standard data format, 2-12
STANDARD-1 phrase, 3-10
START statement, 5-START
OPEN statement, 5-OPEN
READ statement, 5-READ
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
Statement, 2-46
Status key, 2-32
STOP statement, 5-STOP
Figurative constant, 2-9
Imperative statement, 2-47
STRING statement, 5-STRING
Figurative constant, 2-9
Imperative statement, 2-47
Overlapping operands, 5-16
Subscripting, 2-17
Condition-name, 2-4
Conditional variable, 5-8
CONTROL clause, 11-CONTROL
Qualification, 2-14
Subtotalling, 11-SUM
SUBTRACT statement, 5-SUBTRACT
Composite of operands, 5-16
COMPUTE statement, 5-COMPUTE
Conditional statement, 2-46
CORRESPONDING (CORR), 5-14
Data conversion, 5-16
Decimal alignment, 5-16
Imperative statement, 2-47
Maximum operand size, 5-16
Multiple results, 5-17
SUM clause, 11-SUM
Sum counter, 11-54
INITIATE statement, 11-INITIATE
USE FOR DEBUGGING statement, 10-USE FOR DEBUGGING
SUPPRESS statement, 11-SUPPRESS
Imperative statement, 2-47
Switch-status condition, 5-9
SYNC clause, 4-SYNCHRONIZED
SYNCHRONIZED (SYNC) clause, 4-SYNCHRONIZED
Elementary data item, 4-33
USAGE IS INDEX clause, 4-USAGE
Syntax rules, 1-8
System-name, 2-5, A-4

Table Handling, 2-19
TALLYING phrase
INSPECT statement, 5-INSPECT
UNSTRING statement, 5-UNSTRING
TERMINATE statement, 11-TERMINATE
GENERATE statement, 11-GENERATE
Imperative statement, 2-47
INITIATE statement, 11-INITIATE
SUM clause, 11-SUM
TYPE clause, 11-TYPE
USE statement, 7-USE
Text-name, 2-3
Qualified, 2-14
THROUGH (THRU)
MERGE statement, 9-MERGE
PERFORM statement, 5-PERFORM
RENAMES clause, 4-RENAMES
SORT statement, 9-SORT
VALUE clause, 4-VALUE
THRU (See THROUGH)
TIME, 5-ACCEPT
TIMES, 5-PERFORM
TRAILING, 4-SIGN
TYPE clause, 11-TYPE

Unary arithmetic operator, 5-3
Unary minus, 5-3
Unary plus, 5-3
UNIT, 5-CLOSE
UNSTRING statement, 5-UNSTRING
Figurative constant, 2-9
Imperative statement, 2-47
Overlapping operands, 5-16

UNTIL phrase, 5-PERFORM
UP BY, 5-SET
UPON phrase
    DISPLAY statement, 5-DISPLAY
    SUM clause, 11-SUM
USAGE clause, 4-USAGE
    Class condition, 5-7
    INSPECT statement, 5-INSPECT
    Relation condition, 5-5
    SIGN clause, 5-9
    STRING statement, 5-STRING
    UNSTRING statement, 5-UNSTRING
USAGE IS INDEX clause, 4-USAGE
    CORRESPONDING phrase, 5-15
    MOVE statement, 5-MOVE
    SEARCH statement, 5-SEARCH
    Report Writer, 11-USAGE
    Working-Storage Section, 2-45
USE statement, 7-USE
    Compiler directing statement, 2-46
    Declarative statement, 2-44
    DELETE statement, 5-DELETE
    INVALID KEY condition, 2-30
    READ statement, 5-READ
    REWRITE statement, 5-REWRITE
    START statement, 5-START
    WRITE statement, 5-WRITE
USE BEFORE REPORTING statement,
        11-USE BEFORE REPORTING
USE FOR DEBUGGING statement,
        10-USE FOR DEBUGGING
User-defined words, 2-3
USING phrase
    CALL statement, 7-CALL
    Linkage Section, 4-27, 7-15
    MERGE statement, 9-MERGE
    Procedure Division header, 2-45
    SORT statement, 9-SORT

"V" PICTURE symbol, 4-PICTURE-3
VALUE clause, 4-VALUE, 6-VALUE
    Report Writer, 11-VALUE
VALUE OF clause, 4-VALUE OF
VARYING phrase
    PERFORM statement, 5-PERFORM
    SEARCH statement, 5-SEARCH
Verbs, 2-47

WHEN, 5-SEARCH
Word, 1-5
Working-Storage Section, 4-26
WRITE statement, 5-WRITE
    Conditional statement, 2-46
    Imperative statement, 2-47
    OPEN statement, 5-OPEN
    SPECIAL-NAMES paragraph, 3-10
    USE FOR DEBUGGING statement,
        10-USE FOR DEBUGGING

"X" PICTURE symbol, 4-PICTURE-3

ZERO/ZEROS/ZEROES, 2-9

"0" PICTURE symbol, 4-PICTURE-4
"9" PICTURE symbol, 4-PICTURE-4
"66" RENAMES data description
    entry, 4-RENAMES
"77" item description entry, 4-26, 4-33
"88" condition-name data description
    entry, 4-33

> relation, 5-5
< relation, 5-5
= relation, 5-5
+ operator, 5-3
+ PICTURE symbol, 4-PICTURE-4
- continuation line, 2-54
- operator, 5-3
- PICTURE symbol, 4-PICTURE-4
* comment line, 2-54
* operator, 5-3
* PICTURE symbol, 4-PICTURE-4
/ comment line, 2-54
/ operator, 5-3
/ PICTURE symbol, 4-PICTURE-4
** operator, 5-3
== pseudo-text delimiter, 2-COPY

INDEX

In the manual all Statement and Clause are numbered using the names of Statements/Clauses, but references to those pages in the index are given in the following format:

STATEMENT NAME Statement, "chapter number" - "number"

Where for example:

```
 PICTURE Clause,
    Precedence Rules,   4-3
```

refers to 'Precedence Rules' to be found on the page PICTURE-3 of the PICTURE Clause in chapter 4.

INDEX

A

A PICTURE Symbol, 4-3
Abbreviated Combined Relation Conditions, 5-12, B-1
ACCEPT Statement, 5-1, 6-1
ACCESS MODE Clause
  FILE-CONTROL Paragraph, 3-1, 11-6
ADD Statement, 5-1
  Arithmetic Statements, 5-16
  CORRESPONDING Phrase, 5-15
  Multiple Results, 5-17
  ROUNDED Phrase, 5-14
  SIZE ERROR Phrase, 5-14
ADVANCING PAGE Phrase
  See Also LINAGE Clause,
  WRITE Statement, 5-4
ADVANCING Phrase
  DISPLAY Statement, 5-2
  WRITE Statement, 5-4
AFTER Phrase
  INSPECT Statement, 5-4
  PERFORM Statement, 5-2
  USE Statement, 7-1
  WRITE Statement, 5-4
Algebraic Signs, 4-5
Alignment of Data
  ACCEPT Statement, 5-2
  MOVE Statement, 5-3
  Rules, 4-5
  SYNCHRONIZED Clause, 4-1
ALL Literal, 2-10
  DISPLAY Statement, 5-1
  INSPECT Statement, 5-4
  STOP Statement, 5-1
  STRING Statement, 5-1
  UNSTRING Statement, 5-2
ALL Phrase
  INSPECT Statement, 5-7
  SEARCH Statement, 5-3
  USE FOR DEBUGGING Statement, 10-1
ALL PROCEDURES Phrase
  USE FOR DEBUGGING Statement, 10-1
ALL REFERENCES Phrase
  USE FOR DEBUGGING Statement, 10-2
ALLOC-UNIT Phrase
  VALUE OF Clause, 4-3
ALPHABET Clause
  SPECIAL-NAMES Paragraph, 3-11
Alphabet-name, B-1
  CODE-SET Clause, 4-1
  MERGE Statement, 9-3
  SORT Statement, 9-3
ALPHABETIC
  Class Condition, 5-8
Alphabetic Data Category, 4-1
Alphanumeric Data Category, 4-2
Alphanumeric Edited Data Category, 4-2
ALSO Phrase
  EVALUATE Statement, 5-2
  SPECIAL-NAMES Paragraph, 3-13
ALTERNATE Phrase
  LOCK Statement, 5-1
  UNLOCK Statement, 5-2
ALTERNATE RECORD KEY Clause
  FILE-CONTROL Paragraph, 3-1
AND
  Combined Conditions, 5-10
  Complex Conditions, 5-10
  Logical Operator, 5-10
  SEARCH Statement, 5-3
ANY in the WHEN Phrase
  EVALUATE Statement, 5-2
AREA
  RESERVE Clause, 3-1
  SAME AREA Clause, 3-1, 9-1
Arithmetic Expressions, 5-3, B-1
  COMPUTE Statement, 5-1
  Sign Condition, 5-9
Arithmetic Operators, B-2
  Arithmetic Expressions, 5-3
Arithmetic Statements, 2-48, 5-16
  Overlapping Operands, 5-16
ASCENDING KEY Phrase
  MERGE Statement, 9-2
  OCCURS Clause, 4-2
  See Also SEARCH

Statement,
SORT Statement, 9-2
ASSIGN Clause
FILE-CONTROL Paragraph, 3-1, 9-3, 11-7
Asterisk (*)
Arithmetic Operator, 5-3
Comment Line, 2-54
PICTURE Symbol, 4-4
AT END Condition, 2-31, B-2
I-O Status, 2-32
USE Statement, 5-2
AT END Phrase
READ Statement, 5-2, 5-4
RETURN Statement, 9-1
SEARCH Statement, 5-3
AUTHOR Paragraph
IDENTIFICATION DIVISION, 3-2
AUTO Clause
SCREEN SECTION, 6-1, 7-33

B

B PICTURE Symbol, 4-3
BEFORE Phrase
INSPECT Statement, 5-4
WRITE Statement, 5-4
BELL Clause
SCREEN SECTION, 6-1, 7-33
Binary Operator
Arithmetic Expressions, 5-3
BINARY see COMPUTATIONAL,
BLANK Clause
SCREEN SECTION, 6-1
Blank Line, 2-54
COPY Statement, 2-36
BLANK SCREEN Clause
SCREEN SECTION, 7-33
BLANK WHEN ZERO Clause
Data Description Entry, 4-1, 7-28
SCREEN SECTION, 6-1
See Also PICTURE Clause,
See Also USAGE IS INDEX Clause,
See Also VALUE Clause,
BLINK Clause
SCREEN SECTION, 6-1, 7-33
BLOCK CONTAINS Clause
FILE SECTION, 4-1, 7-25, 11-11
BOTTOM Phrase
LINAGE Clause, 4-1
BOX Clause
SCREEN SECTION, 6-1, 7-33
Braces { }, 1-6
Brackets [ ], 1-6
BY Phrase
COPY Statement, 2-35
DIVIDE Statement, 5-3
INSPECT Statement, 5-7
MULTIPLY Statement, 5-1
PERFORM Statement, 5-2

C

CALL Statement, 7-1
See Also EXIT PROGRAM Statement,
See Also MERGE Statement,
See Also SORT Statement,
CALL Statement Scope, 7-10
Called Program
CALL Statement, 7-1
Calling
Link Units, 7-2
Nested Programs, 7-2
Separately Compiled Programs, 7-2
CANCEL Statement, 7-1
See Also CALL Statement,
See Also EXIT PROGRAM Statement,
Categories of Statements, 2-48
Category of Data, 4-4
MOVE Statement, 5-2
PICTURE Clause, 4-1
VALUE Clause, 4-1
CF see CONTROL FOOTING Phrase,
CH see CONTROL HEADING Phrase, 11-3
Character Representation, 4-4
Character set, 2-1
Character-string, 2-2
PICTURE, 2-11
CHARACTERS Phrase
BLOCK CONTAINS Clause, 4-1
INSPECT Statement, 5-4
RECORD CONTAINS Clause, 4-1
Choice Indicators {| |}, 1-6
CLASS Clause
SPECIAL-NAMES Paragraph, 3-14
Class Condition, 5-7, B-3
Class of Data, 4-4
Class-name
Class Condition, 5-8
Clause, B-3

CLOSE Statement, 5-1, 11-1
See Also CANCEL Statement,
See Also OPEN Statement,
See Also READ Statement,
See Also TERMINATE Statement,
COBOL Character Set, B-3
COBOL Source Program
DATA Description Aspects, 2-42
Data Division Entries, 2-56
Declaratives, 2-56
DIVISION descriptions, 2-39
Example Program, 2-57
Organisation, 2-38
Reference Format, 2-53
Statement types, 2-46
Statements Formats, 2-55
COBOL Word, 2-3, B-3
CODE Clause
REPORT SECTION, 7-35, 11-1
CODE-SET Clause
FILE SECTION, 4-1, 7-25, 11-11
See Also SIGN Clause,
Collating Sequence, B-3
COLLATING SEQUENCE Clause
OBJECT-COMPUTER Paragraph, 3-8
COLLATING SEQUENCE Phrase
MERGE Statement, 9-3
SORT Statement, 9-3
COLUMN Clause
SCREEN SECTION, 6-1
See Also LINE Clause,
COLUMN NUMBER Clause
REPORT SECTION, 11-1
SCREEN SECTION, 7-33
Combined Conditions, 5-10
Comma (,)
Library Text, 2-35
PICTURE Symbol, 4-4
Separator, 2-1
Comment Entries, 2-11
Comment Line, 2-54, B-4
COPY Statement, 2-36
Debugging Line, 10-8
COMMON Clause
PROGRAM-ID Paragraph, 7-17
Common Options for Statements, 5-14
Common Programs, 7-5
COMP see COMPUTATIONAL,
COMP-1 see COMPUTATIONAL-1,
COMP-2 see COMPUTATIONAL-2,
COMP-3 see COMPUTATIONAL-3,
COMP-4 see COMPUTATIONAL-4,
Comparison
Involving Index-data-items, 5-7
Involving Index-names, 5-7
Nonnumeric Operands, 5-6
Numeric Operands, 5-6
Comparison Rules
EVALUATE Statement, 5-4
Compiler Directing Statements, 2-46, 2-48
Complex Conditions, 5-10
Composite of Operands
ADD Statement, 5-2
COMPUTE Statement, 5-1
DIVIDE Statement, 5-2
MULTIPLY Statement, 5-1
SUBTRACT Statement, 5-2
COMPUTATIONAL Keyword
USAGE Clause, 4-2, 7-28
COMPUTATIONAL-1 Keyword
USAGE Clause, 4-2, 7-28
COMPUTATIONAL-2 Keyword
USAGE Clause, 4-2, 7-28
COMPUTATIONAL-3 Keyword
USAGE Clause, 4-3, 7-28
COMPUTATIONAL-4 Keyword
USAGE Clause, 4-2, 7-28
COMPUTE Statement, 5-1
Arithmetic Statements, 5-16
Multiple Results, 5-17
ROUNDED Phrase, 5-14
SIZE ERROR Phrase, 5-14
Computer-name
OBJECT-COMPUTER Paragraph, 3-8
SOURCE-COMPUTER Paragraph, 3-7
Condition, B-4
AT END, 2-31
Class, 5-7
Condition-name, 5-9
File Attribute Conflict, 2-31
INVALID KEY, 2-30
Sign, 5-9
SIZE ERROR, 5-14
Switch-Status, 5-9
Condition-name, 2-4, 2-19
Indexing, 2-22
Qualification, 2-19
Subscripting, 2-21
Condition-name Condition, 5-9
Conditional Expressions, 5-5
Relation Conditions, 5-5

Simple Conditions, 5-5
Conditional Statements, 2-46, 2-48
Conditions
Abbreviated Combined Relations, 5-12
Combined, 5-10
Complex, 5-10
Negated, 5-10
Order of Evaluation, 5-13
Relation, 5-5
CONFIGURATION SECTION
ENVIRONMENT DIVISION, 3-6
CONSOLE Clause
SPECIAL-NAMES Paragraph, 3-11
CONSOLE Phrase
ACCEPT Statement, 5-2, 6-2
DISPLAY Statement, 5-2, 6-1
EDIT Statement, 6-1
CONTAINS
BLOCK CONTAINS Clause, 4-1
MULTIPLE FILE Clause, 3-2, 11-8
RECORD CONTAINS Clause, 4-1
Continuation Line, 2-54
Control Break, 11-3
CONTROL Clause
REPORT SECTION, 7-35, 11-1
CONTROL FOOTING Phrase
TYPE Clause, 11-3
CONTROL HEADING Phrase
TYPE Clause, 11-3
Conversion of Data
MOVE Statement, 5-2
CONVERTING Phrase
INSPECT Statement, 5-4
COPY Statement, 2-34
CORR see CORRESPONDING Phrase,
CORRESPONDING Phrase, 5-15
ADD Statement, 5-1
MOVE Statement, 5-1
SUBTRACT Statement, 5-2
COUNT Phrase
UNSTRING Statement, 5-2
CR PICTURE Symbol, 4-4
Crossfooting, 11-4
SUM Clause, 11-2
cs PICTURE Symbol, 4-4
CURRENCY SIGN Clause
SPECIAL-NAMES Paragraph, 3-14
Currency Symbol, B-6

D

D Indicator, 10-8
Data Alignment, 4-5
ACCEPT Statement, 5-2
EDIT Statement, 6-2
MOVE Statement, 5-3
Data Category, 4-4
Data Class, 4-4
Data Description
Concept of Levels, 4-3
Initial Value, 4-2
Noncontiguous Items, 4-2
Record Description, 4-2
Data Description Entry, 4-8, 7-26, B-6
66 RENAMES, 4-11
77 Noncontiguous Data Item, 4-11
88 Condition-name, 4-11
Level-number, 4-11
Data Display
EDIT Statement, 6-3
DATA DIVISION, 4-1
Data Item Sizes
DISPLAY Statement, 5-1
DATA Phrase
ACCEPT Statement, 5-2
DATA RECORDS Clause
FILE SECTION, 4-1, 7-25, 9-3, 11-11
Data-name Clause
REPORT SECTION, 11-1
Data-name or FILLER Clause
Data Description Entry, 4-1
DATE-COMPILED Paragraph
IDENTIFICATION DIVISION, 3-4
DATE-WRITTEN Paragraph
IDENTIFICATION DIVISION, 3-2
DAY Phrase
ACCEPT Statement, 5-2
DB PICTURE Symbol, 4-4
DE see DETAIL Phrase,
Debug, 10-1
DEBUG-CONTENTS Data Item, 10-5
DEBUG-ITEM
Implicit Description, 10-4
Special Register, 2-6, 10-1
DEBUG-LINE Data Item, 10-5
DEBUG-NAME Data Item, 10-5
DEBUG-SUB-1 Data Item, 10-5
DEBUG-SUB-2 Data Item, 10-5
DEBUG-SUB-3 Data Item, 10-5
Debugging Line
COPY Statement, 2-36
Debugging Lines, 10-8

DEBUGGING MODE Clause
Compile Time Switch, 10-2
SOURCE-COMPUTER
Paragraph, 10-1
DECIMAL-POINT IS COMMA Clause
SPECIAL-NAMES
Paragraph, 3-14
DECLARATIVES, 2-30, B-6
USE BEFORE REPORTING
Statement, 11-1
USE FOR DEBUGGING
Statement, 10-1
USE Statement, 5-1
DECLARATIVES SECTION
PROCEDURE DIVISION, 5-2
Default Value
ALLOC-UNIT Phrase, 4-3
FILE-SIZE Phrase, 4-3
Line-Param ACCEPT
Statement, 5-3
Literal-2 COPY
Statement, 2-35
PAGE-SIZE Phrase, 4-4
SPLIT-STRATEGY Phrase, 4-4
SYNCHRONIZED Clause, 4-1
TIME-OUT Phrase, 4-3
DELETE FILE Statement, 5-1
DELETE Statement, 5-1
DELIMITED Phrase
STRING Statement, 5-2
UNSTRING Statement, 5-2
Delimited Scope Statement, 2-48
DELIMITER Phrase
UNSTRING Statement, 5-2
DEPENDING ON Phrase
GO TO Statement, 5-1
OCCURS Clause, 4-3
DESCENDING KEY Phrase
MERGE Statement, 9-2
OCCURS Clause, 4-2
SORT Statement, 9-2
DETAIL Phrase
TYPE Clause, 11-3
DISPLAY Keyword
USAGE Clause, 4-3
Display of Prompt (?), 5-2
DISPLAY Phrase
USAGE Clause, 11-1
DISPLAY Statement, 5-1, 6-1
DIVIDE Statement, 5-1
Arithmetic Statements, 5-16
Multiple Results, 5-17
ROUNDED Phrase, 5-14
SIZE ERROR Phrase, 5-14
Division Header, B-7
DOUBLE-REAL see COMPUTATIONAL-2
DOWN BY Phrase
SET Statement, 5-2
DUPLICATES Phrase
ALTERNATE RECORD KEY
Clause, 3-1
SORT Statement, 9-2
DYNAMIC Access Mode, 2-27, B-7
DYNAMIC Phrase
ACCESS MODE Clause, 3-2
Dynamically Loadable Link
Unit, 7-9, B-7

E

EDIT Statement, 6-1
Editing Character, B-7
Elementary Item Alignment
SYNCHRONIZED Clause, 4-1
Elementary Items
Record Concept, 2-13
Ellipsis (...), 1-6
ELSE NEXT SENTENCE Phrase
IF Statement, 5-1
ELSE Phrase
IF Statement, 5-1
END DECLARATIVES Key Words
DECLARATIVES SECTION, 5-2
END PROGRAM Header, 7-16
END-OF-PAGE Phrase
WRITE Statement, 5-4
END-statement Scope Terminator
Statements, 2-52
ENVIRONMENT DIVISION, 3-5
EOP Phrase
WRITE Statement, 5-4
EQUAL TO (-)
Relation Condition, 5-5
EQUAL TO (-)
SEARCH Statement, 5-3
ERROR See EXCEPTION Phrase,
ESCAPE Phrase
ACCEPT Statement, 6-3
EDIT Statement, 6-5
EVALUATE Statement, 5-1
Example, 5-5
EXCEPTION Phrase
ACCEPT Statement, 6-3
CALL Statement, 7-1
EDIT Statement, 6-5
USE Statement, 5-1, 7-1
Exeption Handling, 2-30
EXIT PROGRAM Statement, 7-1
See Also CANCEL
Statement,

EXIT Statement, 5-1
Explicit and Implicit
Specifications, 2-50
  Attributes, 2-52
  Procedure Division
  References, 2-50
  Scope Terminators, 2-52
  Transfers of Control, 2-50
Exponentiation (**)
  Arithmetic Operator, 5-3
EXTEND Open Mode, 2-27
  OPEN Statement, 5-2, 11-1
  USE Statement, 5-1, 7-1
EXTERNAL Clause
  File Description Entry, 7-1
External Objects, 7-3
External Switch, 2-52

F

FALSE before the WHEN Phrase
  EVALUATE Statement, 5-2
FALSE in the WHEN Phrase
  EVALUATE Statement, 5-2
FD Level Indicator
  FILE SECTION, 4-8, 7-24,
  11-10
Field Termination Key Codes
  ACCEPT Statement, 6-3
Figurative Constant, 2-6
Figurative Constant Values, 2-9
Figurative Constants
Rules, 2-10
File Access Mode
  DYNAMIC, 2-27
  RANDOM, 2-26
  SEQUENTIAL, 2-26
File Attributes
  Conflict, 2-31
  File Open Mode, 2-27
  Record Access Mode, 2-26
  Record Organization, 2-24
File Concept
  Characteristics, 2-12
  LINAGE Clause, 2-28
  Physical, 2-12
  Position Indicator, 2-28
  Sorting and Merging, 2-28
File Control Entry, 3-16, 9-3,
11-5
File Description Entry, B-9
  Indexed Organization, 4-11
  Inter-Program
  Communication, 7-20
  Relative Organization, 4-10
  Report Writer, 11-10
  Sequential Organization, 4-9
  Sort-Merge, 9-2
File Open Mode, B-14
  EXTEND, 2-27
  I-O, 2-27
  INPUT, 2-27
  OUTPUT, 2-27
File Organization
  INDEXED, 2-25
  RELATIVE, 2-25
  SEQUENTIAL, 2-24
File Path Name, 9-3
  CALL Statement, 7-2
  CANCEL Statement, 7-1
  COPY Statement, 2-34
  SELECT .. ASSIGN Clause, 3-2
FILE SECTION
  DATA DIVISION, 4-7, 9-2,
  11-9
File Sharing, 7-5
FILE STATUS Clause
  FILE-CONTROL Paragraph, 3-1,
  11-6
FILE-CONTROL Paragraph
  INPUT-OUTPUT SECTION, 3-15
FILE-SIZE Phrase
  VALUE OF Clause, 4-3
FILLER Clause
  Data Description Entry, 4-1
  SCREEN SECTION, 6-1, 6-6
FINAL Phrase
  CONTROL Clause, 11-1
  SUM Clause, 11-1
  TYPE Clause, 11-2
FIRST DETAIL Phrase
  PAGE Clause, 11-2
FIRST Phrase
  INSPECT Statement, 5-5
FOOTING Phrase
  LINAGE Clause, 4-1
  PAGE Clause, 11-2
Format Punctuation, 1-7
Formation and Evaluation Rules
  Arithmetic Expressions, 5-3
FROM ESCAPE KEY Phrase
  ACCEPT Statement, 6-5
FROM Phrase
  ACCEPT Statement, 6-2
  PERFORM Statement, 5-2
  PICTURE Clause, 6-2
  RELEASE Statement, 9-1
  REWRITE Statement, 5-2
  SUBTRACT Statement, 5-2
  WRITE Statement, 5-2
FULL Clause

SCREEN SECTION, 6-1, 7-33
Function Key Codes
ACCEPT Statement, 6-3

G

General Rules, 1-8
GENERATE Statement, 11-1
Control Break, 11-3
See Also SUM Clause,
GIVING Phrase
ADD Statement, 5-1
DIVIDE Statement, 5-2
MERGE Statement, 9-4
MULTIPLY Statement, 5-1
SORT Statement, 9-5
SUBTRACT Statement, 5-1
GLOBAL Clause
File Description Entry, 7-1
See Also EXTERNAL Clause,
Global Name, 7-2, B-9
GLOBAL Phrase
USE Statement, 7-1
Glossary, B-1
GO TO Statement, 5-1
See Also MERGE Statement,
See Also PERFORM Statement,
See Also SORT Statement,
GREATER THAN (>)
Relation Condition, 5-5
GROUP INDICATE Clause
REPORT SECTION, 11-1
VALUE Clause, 11-1

H

HEADING Phrase
PAGE Clause, 11-2
HEBREW Clause
SCREEN SECTION, 6-1, 7-33
HIGH-VALUE/HIGH-VALUES, 2-9
HIGHLIGHT Clause
SCREEN SECTION, 6-1, 7-33
Hyphen (-)
Line Continuation, 2-54

I

I-O Open Mode, 2-27, B-10
OPEN Statement, 5-2
USE Statement, 5-1, 7-1
I-O Status, 2-32, B-10
CLOSE Statement, 5-2
Data Item, 3-1
DELETE FILE Statement, 5-1
DELETE Statement, 5-1
LOCK Statement, 5-2
OPEN Statement, 5-5
READ Statement, 5-3
REWRITE Statement, 5-3
START Statement, 5-2
WRITE Statement, 5-7
I-O Status Conditions, C-1
I-O Status Values, C-1
I-O-CONTROL Paragraph
INPUT-OUTPUT SECTION, 3-2, 9-5, 11-8
IDENTIFICATION DIVISION, 3-1
Identifier, 2-18
IF Statement, 5-1
Imperative Statements, 2-47
Implementation of the Standard, 1-3
Implementor-name
ALPHABET Clause, 3-11
Implicit FILLER
SYNCHRONIZED Clause, 4-2
IN Phrase
COPY Statement, 2-35
IN Qualifier, 2-15
In-Line PERFORM Statement, 5-3
Incompatibe Data, 5-17
INDEX Keyword
USAGE Clause, 4-3
Index-name Conventions, 7-8
INDEXED BY Phrase
OCCURS Clause, 4-2
INDEXED Organization, 2-25
INDEXED Phrase
ORGANIZATION Clause, 3-1
INITIAL Clause
PROGRAM-ID Paragraph, 7-17
INITIAL Phrase
INSPECT Statement, 5-4
Initial Programs, 7-5
Initial State of a Program, 7-15
INITIATE Statement, 11-1
See Also GENERATE Statement,
See Also OPEN Statement,
See Also SUM Clause,
INPUT Open Mode, 2-27
OPEN Statement, 5-2
USE Statement, 5-1, 7-1

INPUT PROCEDURE Phrase
SORT Statement, 9-3
INPUT-OUTPUT SECTION
ENVIRONMENT DIVISION, 3-15, 11-5
Input-Output Statements, 2-48
INSPECT Statement, 5-1
Examples, 5-9
Overlapping Operands, 5-16
INSTALLATION Paragraph
IDENTIFICATION DIVISION, 3-2
Inter-Program
Communication, 7-1
Internal Objects, 7-3
INTO Phrase
DIVIDE Statement, 5-3
READ Statement, 5-4
RETURN Statement, 9-1
STRING Statement, 5-2
UNSTRING Statement, 5-1
INVALID KEY Condition, 2-30, B-11
I-O Status, 2-32
USE Statement, 5-2
INVALID KEY Phrase
DELETE Statement, 5-1
READ Statement, 5-2
REWRITE Statement, 5-3
START Statement, 5-1
WRITE Statement, 5-2
Item Alignment for
Efficiency, 4-6

J

JUST see JUSTIFIED,
JUSTIFIED Clause
Data Description Entry, 4-1, 7-28
SCREEN SECTION, 6-1
See Also USAGE IS INDEX Clause,
See Also VALUE Clause,

K

KEY Data-name
MERGE Statement, 9-2
SORT Statement, 9-2
KEY Phrase
LOCK Statement, 5-2
OCCURS Clause, 4-2
READ Statement, 5-2
See Also SEARCH
Statement,
START Statement, 5-1
UNLOCK Statement, 5-2

L

LABEL RECORDS Clause
FILE SECTION, 4-1, 7-25, 11-11
Language Concepts
Debug, 10-1
Inter-Program
Communication, 7-1
Report Writer, 11-1
LAST DETAIL Phrase
PAGE Clause, 11-2
LEADING Phrase
INSPECT Statement, 5-5
SIGN Clause, 4-1, 6-1, 11-1
SIGN IS Clause, 4-2
LEFT Phrase
SYNCHRONIZED Clause, 4-2
LESS THAN (<)
Relation Condition, 5-5
Level Indicator, B-12
Level-number, 4-1, 4-3, B-12
01 First Entry of
Record, 4-1
66 RENAMES, 4-1
77 Noncontiguous Data
Items, 4-1
88 Condition-names, 4-1
Data Description Entry, 4-1, 4-11
REPORT SECTION, 11-1
SCREEN SECTION, 6-1, 6-6
Library Facility, 2-33
COPY Statement, 2-34
LIMIT Phrase
PAGE Clause, 11-1
LINAGE Clause
FILE SECTION, 4-1, 7-25
LINAGE-COUNTER
Qualification, 2-16
Special Register, 2-6, 2-28, 4-3
LINE Clause
SCREEN SECTION, 6-1
See Also COLUMN Clause,
SPECIAL-NAMES
Paragraph, 3-11
LINE NUMBER Clause
Notation, 11-10
REPORT SECTION, 11-1
SCREEN SECTION, 7-33

LINE Phrase
BLANK Clause, 6-1
PAGE Clause, 11-1
LINE-COUNTER
Qualification, 2-16
Rules, 11-5
Special Register, 2-6, 11-4
LINE-PARAM Phrase
ACCEPT Statement, 5-3
LINES AT BOTTOM Phrase
LINAGE Clause, 4-1
LINES AT TOP Phrase
LINAGE Clause, 4-1
Link Unit, 7-8, B-12
CALL Statement, 7-1
LINKAGE SECTION
CALL Statement, 7-4
DATA DIVISION, 4-6, 7-18
Literal, 2-6
CURRENCY SIGN Clause, 3-14
STOP Statement, 5-1
Local Name, 7-2
LOCK Phrase
CLOSE Statement, 5-2
OPEN Statement, 5-7
READ Statement, 5-7
REWRITE Statement, 5-3
WRITE Statement, 5-7
LOCK Statement, 5-1
Logical Operators, B-12
Conditional Expressions, 5-10
Precedence, 5-11
LOW-VALUE/LOW-VALUES, 2-9

M

Maximum Limit
Arithmetic Statement Operands, 5-16
Character Transfer ACCEPT Statement, 5-2
Characters in a Record, 4-1
Data Transfer DISPLAY Statement, 5-1
DECLARATIVE Sections, 5-2
EVALUATE Statement, 5-3
File Allocation Size, 4-2
File Path Name, 9-4
Merge KEY Data-names, 9-2
Number of Screen Items, 6-1
PAGE-SIZE, 4-2
Procedure-names GO TO Statement, 5-1
Pseudo-text Length, 2-34
Record Size, 4-8
Sort KEY Data-names, 9-2
SPLIT-STRATEGY, 4-2
Time Out ACCEPT Statement, 6-2
TIME-OUT Period, 4-2
Value of Data Item EDIT Statement, 6-2
Merge File, 9-1, B-13
MERGE Statement, 9-1
See Also OPEN Statement,
MESSAGE
Special Register, 2-6, 7-9
Values, 7-9
Minus (-)
Arithmetic Operator, 5-3
PICTURE Symbol, 4-4
Mnemonic-name, 2-4, B-13
ACCEPT Statement, 5-1, 6-1
DISPLAY Statement, 5-1, 6-1
EDIT Statement, 6-1
SPECIAL-NAMES Paragraph, 3-11
WRITE Statement, 5-2
MOVE Statement, 5-1
CORRESPONDING Phrase, 5-15
Overlapping Operands, 5-16
See Also READ Statement,
See Also SUM Clause,
Multiple Conditions
EVALUATE Statement, 5-3
MULTIPLE FILE Clause
I-O-CONTROL Paragraph, 3-2, 11-8
Multiple Results, 5-17
ADD Statement, 5-2
MULTIPLY Statement, 5-2
SUBTRACT Statement, 5-2
MULTIPLY Statement, 5-1
Arithmetic Statements, 5-16
Multiple Results, 5-17
ROUNDED Phrase, 5-14
SIZE ERROR Phrase, 5-14

N

NATIVE Phrase
SPECIAL-NAMES Paragraph, 3-12
Negated Conditions, 5-10
NEGATIVE
Sign Condition, 5-9
Nested Source Programs, 7-13
NEXT GROUP Clause
REPORT SECTION, 11-1

NEXT PAGE Phrase
LINE NUMBER Clause, 11-1
NEXT GROUP Clause, 11-1
NEXT Phrase
READ Statement, 5-2
NEXT SENTENCE Phrase
IF Statement, 5-1
SEARCH Statement, 5-3
NO ADVANCING Phrase
DISPLAY Statement, 5-2
NO LOCK Phrase
OPEN Statement, 5-1
REWRITE Statement, 5-3
NO REWIND Phrase
CLOSE Statement, 5-2
OPEN Statement, 5-5, 11-1
Nonnumeric Literal, 2-7, B-13
NOT
Abbreviated Combined Relation Conditions, 5-12
Class Condition, 5-8
Logical Operator, 5-10
Relation Condition, 5-5
Sign Condition, 5-9
NOT Phrase
EVALUATE Statement, 5-3
Notation used in Formats, 1-5
NUMERIC
Class Condition, 5-8
Numeric Data Category, 4-1
Numeric Edited Data Category, 4-2
Numeric Literal, 2-8

O

OBJECT-COMPUTER Paragraph
CONFIGURATION SECTION, 3-8, 8-3
OCCURS Clause
Data Description Entry, 4-1
See Also REDEFINES Clause,
See Also RENAMES Clause,
See Also SEARCH Statement,
See Also SYNCHRONIZED Clause,
See Also VALUE Clause,
OF Phrase
COPY Statement, 2-35
OF Qualifier, 2-15
OFF Phrase
SET Statement, 5-3
SPECIAL-NAMES
Paragraph, 3-12
OFF STATUS Phrase
SPECIAL-NAMES
Paragraph, 3-12
OMITTED Phrase
LABEL RECORDS Clause, 4-1
ON Phrase
SET Statement, 5-3
SPECIAL-NAMES
Paragraph, 3-12
ON SIZE ERROR see SIZE ERROR Phrase,
ON STATUS Phrase
SPECIAL-NAMES
Paragraph, 3-12
OPEN Statement, 5-1, 11-1
See Also INITIATE Statement,
See Also READ Statement,
See Also REWRITE Statement,
See Also START Statement,
See Also WRITE Statement,
Optional File, B-15
OPTIONAL Phrase
SELECT .. ASSIGN Clause, 3-2, 11-7
OR
Combined Conditions, 5-10
Complex Conditions, 5-10
Logical Operator, 5-10
UNSTRING Statement, 5-3
Order of Evaluation of Conditions, 5-13
ORGANIZATION Clause
FILE-CONTROL Paragraph, 3-1, 11-6
OUTPUT Open Mode, 2-27
OPEN Statement, 5-1, 11-1
See Also LINAGE Clause,
USE Statement, 5-1, 7-1
OUTPUT PROCEDURE Phrase
MERGE Statement, 9-3
SORT Statement, 9-4
OVERFLOW Phrase
CALL Statement, 7-1
STRING Statement, 5-3
UNSTRING Statement, 5-4
Overlapping Operands, 5-16
ADD Statement, 5-2
COMPUTE Statement, 5-1
INSPECT Statement, 5-9
MULTIPLY Statement, 5-2
STRING Statement, 5-3

UNSTRING Statement, 5-5
Overlays, 7-8

P

P PICTURE Symbol, 4-3
PACKED-DECIMAL see COMPUTATIONAL-3,
PAGE Clause
REPORT SECTION, 7-35, 11-1
PAGE FOOTING Phrase
TYPE Clause, 11-3
PAGE HEADING Phrase
TYPE Clause, 11-2
PAGE Phrase
WRITE Statement, 5-4
Page Regions, 11-3
Page Splitting Strategy, B-15
PAGE-COUNTER
Qualification, 2-16
Rules, 11-4
Special Register, 2-6, 11-3
PAGE-SIZE Phrase
VALUE OF Clause, 4-3
Paragraph, B-15
Paragraph-name, 2-4
Parentheses in Conditional Expressions, 5-11
Passing Data, 7-11
PERFORM Statement, 5-1
Declarative References, 5-3
Example, 5-10
See Also MERGE Statement,
See Also SORT Statement,
See Also USE BEFORE REPORTING Procedure,
See Also USE Statement,
Period (.)
PICTURE Symbol, 4-4
Separator, 1-7
PF see PAGE FOOTING Phrase,
PH see PAGE HEADING Phrase,
Phrase, B-16
PIC see PICTURE Clause,
PICTURE Clause
Data Description Entry, 4-1, 7-28
Editing Precedence Rules, 4-8
Fixed Insertion, 4-6
Floating Insertion, 4-6
SCREEN SECTION, 6-1, 7-33
See Also USAGE IS INDEX Clause,
Simple Insertion, 4-5
Special Insertion, 4-5
Types of Editing, 4-4, 4-5
Zero Suppression, 4-7
Plus (+)
Arithmetic Operator, 5-3
PICTURE Symbol, 4-4
PLUS Phrase
COLUMN Clause, 6-1
LINE Clause, 6-1
LINE NUMBER Clause, 11-1
NEXT GROUP Clause, 11-1
POINTER Phrase
STRING Statement, 5-2
UNSTRING Statement, 5-3
POSITION Phrase
MULTIPLE FILE Clause, 3-2
POSITIVE
Sign Condition, 5-9
Precedence of Logical Operators, 5-11
PREVIOUS Phrase
READ Statement, 5-2
PRIMARY Phrase
LOCK Statement, 5-1
UNLOCK Statement, 5-2
PRINTING Keyword
SUPPRESS Statement, 11-1
PROCEDURE DIVISION, 5-1, 8-5, 11-2
Segmentation Organization, 8-1
Procedure Division Header, 5-1
Inter-Program Communication, 7-2
PROCEDURE ON Phrase
USE Statement, 5-1
PROCEDURE Phrase
USE Statement, 7-1
PROGRAM COLLATING SEQUENCE Clause
OBJECT-COMPUTER Paragraph, 3-8
Program Organization
Inter-Program Communication, 7-1
PROGRAM-ID Paragraph
IDENTIFICATION DIVISION, 3-3, 7-17
Program-name Conventions, 7-7
Pseudo-text, 2-55, B-16
COPY Statement, 2-34
Punctuation Character, B-17
See Also Separator,

Q

Qualification, 2-14
   LINAGE-COUNTER, 2-16
   LINE-COUNTER, 2-16
   PAGE-COUNTER, 2-16
Qualifier, B-17
Quotation Mark (')
   Line Continuation, 2-54
QUOTE/QUOTES, 2-10

R

Radix, 4-4
RANDOM Access Mode, 2-26, B-17
RANDOM Phrase
   ACCESS MODE Clause, 3-2
RD Level Indicator, 11-3
READ Statement, 5-1
   See Also OPEN Statement,
   See Also REWRITE
    Statement,
   See Also START
    Statement,
Record Concept, 2-13
   Levels, 2-13
   Logical, 2-12
   Shared Memory Area, 2-29
RECORD CONTAINS Clause
   FILE SECTION, 4-1, 7-25,
    9-3, 11-11
Record Description Entry, 4-8,
    9-2, B-17
   Structure, 4-8
Record Key, B-17
RECORD KEY Clause
   FILE-CONTROL Paragraph, 3-1
RECORD Phrase
   DELETE Statement, 5-1
   SAME AREA Clause, 9-1
   UNLOCK Statement, 5-2
RECORDS Phrase
   BLOCK CONTAINS Clause, 4-1
   DATA RECORDS Clause, 4-1
   LABEL RECORDS Clause, 4-1
REDEFINES Clause
   Data Description Entry, 4-1,
    7-28
   See Also EXTERNAL
    Clause,
   See Also GLOBAL Clause,
   See Also VALUE Clause,
REEL Phrase
   CLOSE Statement, 5-1
Reference Format, 2-53
 Relation Character, B-18
Relation Conditions, 5-5, B-18
Relational Operator, B-18
   START Statement, 5-2
Relative Key, B-18
RELATIVE KEY Phrase
   ACCESS MODE Clause, 3-2
   READ Statement, 5-6
   START Statement, 5-1
RELATIVE Organization, 2-25
RELATIVE Phrase
   ORGANIZATION Clause, 3-1
 RELEASE Statement, 9-1
REMAINDER Phrase
   DIVIDE Statement, 5-2
REMOVAL Phrase
   CLOSE Statement, 5-2
RENAMES Clause
   Data Description Entry, 4-1
REPLACING Phrase
   COPY Statement, 2-35
   INSPECT Statement, 5-5
REPORT Clause
   File Description Entry, 11-1
   REPORT SECTION, 7-25
Report Description Entry, 11-2
   REPORT SECTION, 7-34
Report File, 11-3
REPORT FOOTING Phrase
   TYPE Clause, 11-3
Report Group Description
    Entry, 11-2
   REPORT SECTION, 11-4
REPORT HEADING Phrase
   TYPE Clause, 11-2
Report Presentation Rules, 11-9
   Body Group, 11-17
   Page Footing, 11-22
   Page Heading Group, 11-15
   Report Footing, 11-24
   Report Heading Group, 11-12
REPORT SECTION
   DATA DIVISION, 4-7, 11-2
Report Structure, 11-2
Report Writer, 11-1
REQUIRED Clause
   SCREEN SECTION, 6-1, 7-33
Required Words, 2-6
RESERVE Clause
   FILE-CONTROL Paragraph, 3-1
Reserved Words, 2-5
   Implementor-names, A-4
   System Names, A-4
Reserved Words List, A-1
RESET Phrase
   SUM Clause, 11-1

RETURN Statement, 9-1
REVERSE Clause
SCREEN SECTION, 6-1, 7-33
REVERSED Phrase
OPEN Statement, 5-1
REWIND Phrase
CLOSE Statement, 5-2
REWRITE Statement, 5-1
See Also OPEN Statement,
See Also READ Statement,
RF see REPORT FOOTING Phrase,
RH see REPORT HEADING Phrase,
RIGHT Phrase
SYNCHRONIZED Clause, 4-2
Root Link Unit, 7-9, B-19
ROUNDED Phrase, 5-14
ADD Statement, 5-2
COMPUTE Statement, 5-1
DIVIDE Statement, 5-2
MULTIPLY Statement, 5-2
SUBTRACT Statement, 5-2
RUN Phrase
STOP Statement, 5-1
Run Unit, 7-8, B-20
Run Unit Organization, 7-1

S

S PICTURE Symbol, 4-3
SAME AREA Clause
I-O-CONTROL Paragraph, 3-1, 9-1
SAME Clause
I-O-CONTROL Paragraph, 11-8
SAME RECORD AREA Clause
I-O-CONTROL Paragraph, 3-1, 9-1
SAME SORT AREA Clause
I-O-CONTROL Paragraph, 9-1
SAME SORT-MERGE AREA Clause
I-O-CONTROL Paragraph, 9-1
Scope of CALL Statement, 7-10
Scope of Names, 7-6
Scope of Statements, 2-49
Scope Terminator Statements, 2-52
Screen Description Entry
SCREEN SECTION, 6-2, 7-29
SCREEN Phrase
BLANK Clause, 6-1
SCREEN SECTION
DATA DIVISION, 4-7, 6-1
Screen-name or FILLER Clause
SCREEN SECTION, 6-1
SD Level Indicator, 9-3
SEARCH ALL Statement
SEARCH Statement, 5-3
SEARCH Statement, 5-1
See Also USAGE IS INDEX Clause,
SECTION
PROCEDURE DIVISION, 5-2
Section-name, 2-4
SECURE Clause
SCREEN SECTION, 6-1, 7-33
SECURITY Paragraph
IDENTIFICATION DIVISION, 3-2
Segment Classification, 8-2
Segment Number, 8-6, B-21
SEGMENT-LIMIT Clause
OBJECT-COMPUTER Paragraph, 8-4
Segmentation, 8-1
Segmentation Control, 8-2
SELECT .. ASSIGN Clause
FILE-CONTROL Paragraph, 3-1, 3-16, 9-3, 11-6
Semicolon (;)
Library Text, 2-35
Separator, 2-1
SEPARATE CHARACTER Phrase
SIGN Clause, 4-2, 6-1, 11-1
SEPARATE Phrase
SIGN Clause, 4-2
Separator, 2-1
Comma (,), 2-1
Parentheses (), 2-1
Period (.), 2-1
Pseudo-text Delimiter (--), 2-2
Quotation Mark ('), 2-2
Semicolon (;), 2-1
Space ( ), 2-1
SEQUENCE Clause
OBJECT-COMPUTER Paragraph, 3-8
Sequence Number, 2-54
SEQUENTIAL Access Mode, 2-26, B-21
SEQUENTIAL Organization, 2-24
SEQUENTIAL Phrase
ACCESS MODE Clause, 3-2
ORGANIZATION Clause, 3-1
SET Statement, 5-1
Overlapping Operands, 5-16
See Also SWITCH Clause,
See Also USAGE IS INDEX Clause,
Shared Memory Area, 2-29

Sharing Data
Inter-Program
Communication, 7-5
SIGN Clause
Data Description Entry, 4-1,
7-28
REPORT SECTION, 11-1
SCREEN SECTION, 6-1
Sign Condition, 5-9
Simple Conditions, 5-5
SINGLE-REAL see COMPUTATIONAL-
1,
SIZE ERROR Condition, 5-14
SIZE ERROR Phrase, 5-14
ADD Statement, 5-2
COMPUTE Statement, 5-1
DIVIDE Statement, 5-3
MULTIPLY Statement, 5-2
SUBTRACT Statement, 5-2
SIZE Phrase
STRING Statement, 5-2
Slant (/)
Arithmetic Operator, 5-3
Comment Line, 2-54
Path Name Indicator, 2-34
PICTURE Symbol, 4-4
Sort File, 9-1, B-21
SORT Phrase
SAME AREA Clause, 9-1
SORT Statement, 9-1
See Also OPEN Statement,
SORT-MERGE Phrase
SAME AREA Clause, 9-1
SOURCE Clause
REPORT SECTION, 11-1
See Also TYPE Clause,
SOURCE-COMPUTER Paragraph
CONFIGURATION SECTION, 3-7
Space ( )
Separator, 2-1
SPACE/SPACES, 2-9
Special Character, B-22
Special Character Words, 1-7
Special Register, 2-6
DEBUG-ITEM, 10-1
LINAGE-COUNTER, 2-28, 4-3
LINE-COUNTER, 11-4
MESSAGE, 7-9
PAGE-COUNTER, 11-3
SPECIAL-NAMES, B-22
SPECIAL-NAMES Paragraph
CONFIGURATION SECTION, 3-10
See Also CODE-SET
Clause,
SPLIT-STRATEGY Phrase
VALUE OF Clause, 4-4
STANDARD Phrase
LABEL RECORDS Clause, 4-1
STANDARD-1 Phrase
SPECIAL-NAMES
Paragraph, 3-12
START Statement, 5-1
See Also OPEN Statement,
See Also READ Statement,
Statement, B-23
STATUS Clause
FILE-CONTROL Paragraph, 3-1
STOP Statement, 5-1
Storage Allocation for Numeric
Items, 4-3
STRING Statement, 5-1
Overlapping Operands, 5-16
Subscript, B-23
Subscripting, 2-17
REPORT SECTION, 11-4
Subtotalling, 11-4
SUM Clause, 11-3
SUBTRACT Statement, 5-1
Arithmetic Statements, 5-16
CORRESPONDING Phrase, 5-15
Multiple Results, 5-17
ROUNDED Phrase, 5-14
SIZE ERROR Phrase, 5-14
SUM Clause
REPORT SECTION, 11-1
See Also TYPE Clause,
Sum Counter, 11-1
SUPPRESS Statement, 11-1
See Also TYPE Clause,
See Also USE BEFORE
REPORTING Procedure
SWITCH Clause
SPECIAL-NAMES
Paragraph, 3-11
Switch-Status Condition, 5-9
Symbols used to Describe Data
Items, 4-3
SYNC see SYNCHRONIZED,
SYNCHRONIZED Clause
Data Description Entry, 4-1,
7-28
Implicit FILLER, 4-2
SYNCHRONIZED Phrase
SELECT .. ASSIGN
Clause, 3-2, 9-4, 11-7
Syntax Rules, 1-8
SYSPRINT1 Keyword
ASSIGN Clause, 3-2, 11-7
SYSPRINT2 Keyword
ASSIGN Clause, 3-2, 11-7
SYSPRINT3 Keyword
ASSIGN Clause, 3-2, 11-7

SYSPRINT4 Keyword
ASSIGN Clause, 3-2, 11-7
System Names, 2-5, A-4

T

Table Handling, 2-19
Data-name Subscripting, 2-21
References to Table Items, 2-21
Subscripting Using Index-names, 2-22
Subscripting Using Integers or Data-names, 2-22
Table Definition, 2-20
TALLYING Phrase
INSPECT Statement, 5-5
UNSTRING Statement, 5-4
TERMINATE Statement, 11-1
See Also CLOSE Statement,
See Also GENERATE Statement,
See Also SUM Clause,
See Also TYPE Clause,
TEXT-FILE Keyword
ASSIGN Clause, 3-2, 9-3
OPEN Statement, 5-1
READ Statement, 5-5
THROUGH Phrase
ALPHABET Clause, 3-13
CLASS Clause, 3-14
EVALUATE Statement, 5-3
MERGE Statement, 9-3
PERFORM Statement, 5-3
RENAMES Clause, 4-2
SORT Statement, 9-2
VALUE Clause, 4-1
THRU see THROUGH,
TIME Phrase
ACCEPT Statement, 5-2
TIME-OUT Phrase
ACCEPT Statement, 6-3
EDIT Statement, 6-5
VALUE OF Clause, 4-3
TIMES Phrase
PERFORM Statement, 5-4
TO Phrase
MOVE Statement, 5-1
PICTURE Clause, 6-3
SET Statement, 5-2
TOP Phrase
LINAGE Clause, 4-1
TRAILING Phrase
SIGN Clause, 4-1, 6-1, 11-1
SIGN IS Clause, 4-2
Transfer of Control
GO TO Statement, 5-1
IF Statement, 5-1
Inter-Program Communication, 7-10
STOP Statement, 5-1
USE Statement, 5-2
TRUE before the WHEN Phrase
EVALUATE Statement, 5-2
TRUE in the WHEN Phrase
EVALUATE Statement, 5-2
TRUE Phrase
SET Statement, 5-3
TYPE Clause
REPORT SECTION, 11-1
See Also GENERATE Statement,
See Also SOURCE Clause,
See Also SUM Clause,

U

Unary Operator, B-24
Arithmetic Expressions, 5-3
UNDERLINE Clause
SCREEN SECTION, 6-1, 7-33
Uniqueness of Reference, 2-14
Qualification, 2-14
Subscripting, 2-17
UNIT Phrase
CLOSE Statement, 5-1
UNLOCK Statement, 5-1
UNSTRING Statement, 5-1
Overlapping Operands, 5-16
Unsuccessful Execution, B-24
UNTIL Phrase
PERFORM Statement, 5-5
UP BY Phrase
SET Statement, 5-2
UPON Phrase
DISPLAY Statement, 5-2, 6-1
SUM Clause, 11-1
USAGE Clause
Data Description Entry, 4-1, 7-28
REPORT SECTION, 11-1
See Also INSPECT Statement,
See Also STRING Statement,
See Also UNSTRING Statement,
USAGE IS INDEX Clause, 4-3
See Also MOVE Statement,

See Also SEARCH Statement,
USE BEFORE REPORTING Procedure
See Also SUPPRESS Statement,
See Also TYPE Clause,
USE BEFORE REPORTING Statement, 11-1
USE FOR DEBUGGING Statement, 10-1
USE Statement
DECLARATIVES SECTION, 5-1
Inter-Program Communication, 7-1
See Also WRITE Statement,
User Defined Words, 2-3
USING Option
PROCEDURE DIVISION, 5-2, 7-2
USING Phrase
CALL Statement, 7-1
MERGE Statement, 9-2
PICTURE Clause, 6-3
See Also USAGE IS INDEX Clause,
SORT Statement, 9-4

V

V PICTURE Symbol, 4-3
VALUE Clause
Data Description Entry, 4-1, 7-28
REPORT SECTION, 11-1
SCREEN SECTION, 6-1
See Also EXTERNAL Clause,
See Also USAGE IS INDEX Clause,
VALUE OF Clause
FILE SECTION, 4-1, 7-25, 11-11
VARYING Phrase
PERFORM Statement, 5-2
SEARCH Statement, 5-4

W

WHEN OTHER Phrase
EVALUATE Statement, 5-5
WHEN Phrase
EVALUATE Statement, 5-2
SEARCH Statement, 5-3
WORD-BINARY see COMPUTATIONAL-4
Words, 1-5
WORKING-STORAGE SECTION
DATA DIVISION, 4-5
WRITE Statement, 5-1
See Also LINAGE-COUNTER,
See Also OPEN Statement,
See Also READ Statement,
WS-PRINTER Keyword
ASSIGN Clause, 3-2, 11-7

X

X PICTURE Symbol, 4-3

Z

Z PICTURE Symbol, 4-3
ZERO/ZEROS/ZEROES, 2-9

0 (Zero) PICTURE Symbol, 4-4
9 PICTURE Symbol, 4-4
66 RENAMES Data Description Entry, 4-1, 4-11
77 Data Item Description Entry, 4-5, 4-6, 4-11
88 Condition-name Data Description Entry, 4-11
(Space)
Separator, 2-1
' (Quotation Mark)
Line Continuation, 2-54
Separator, 2-2
() (Parentheses)
Separator, 2-1
* (Asterisk)
Arithmetic Operator, 5-3
Comment Line, 2-54
PICTURE Symbol, 4-4
** (Exponentiation)
Arithmetic Operator, 5-3
+ (Plus)
Arithmetic Operator, 5-3
PICTURE Symbol, 4-4
, (Comma)
PICTURE Symbol, 4-4
Separator, 2-1
- (Hyphen)
Line Continuation, 2-54

- (Minus)
  Arithmetic Operator, 5-3
  PICTURE Symbol, 4-4
. (Period)
  PICTURE Symbol, 4-4
  Separator, 2-1
/ (Slant)
  Arithmetic Operator, 5-3
  Comment Line, 2-54
  Path Name Indicator, 2-34
  PICTURE Symbol, 4-4
; (Semicolon)
  Separator, 2-1
< (LESS THAN)
  Relation Condition, 5-5
= (EQUAL TO)
  Relation Condition, 5-5
  SEARCH Statement, 5-3
-- (Pseudo-text Delimiter)
  Separator, 2-2
> (GREATER THAN)
  Relation Condition, 5-5
? (Prompt) Display, 5-2

Code 4004290 H (4)

Printed in Italy