Specifications Board
Supply and Services Canada

Office des Normes
Approvisionnements et Services Canada

CODASYL COBOL JOURNAL OF DEVELOPMENT

THIS REPORT HAS BEEN PREPARED BY THE PROGRAMMING LANGUAGE COMMITTEE OF THE CONFERENCE ON DATA SYSTEMS LANGUAGES (CODASYL), AN ORGANISATION COMPOSED OF VOLUNTEER REPRESENTATIVES OF COMPUTER MANUFACTURERS AND USERS IN INDUSTRY AND THE FEDERAL GOVERNMENTS OF CANADA AND THE UNITED STATES OF AMERICA.

Withdrawn/Retirée

110-GP-1d
1976

Price 7.50

Copies of this document may be ordered from:

Materiel Data Management Branch
Department of Supply and Services
5th Floor, 88 Metcalfe St.
Ottawa, Ont., Canada.
K1A 0S5.

Price is $7.50 per copy, including packing and mailing. Remittances, which should be sent with orders, may be in either Canadian or U.S. Funds and should be made payable to the Receiver General of Canada. When ordering, please quote title precisely as shown on cover.

Withdrawn/Retirée

CODASYL
COBOL

JOURNAL OF DEVELOPMENT

1976

Withdrawn/Retirée

U.S. Standard Serial Number 0591-0218

History of COBOL Specification Documents

COBOL — 60, published 1960
COBOL — 61, published 1961
COBOL — 61 Extended, published 1963
COBOL — 65, published 1965
COBOL — Journal of Development — 1968, published 1968
COBOL — Journal of Development — 1969, published 1970
COBOL — Journal of Development — 1970, published 1971
COBOL — Journal of Development — 1973, published 1973
COBOL — Journal of Development — 1976, published 1976

Withdrawn/Retiree

TABLE OF CONTENTS

PREFACE

ACKNOWLEDGMENT

Section I HISTORY OF CODASYL COBOL DEVELOPMENT

Section II PHILOSOPHY OF COBOL USE

Section III COBOL LANGUAGE SPECIFICATIONS

Section IV SUB-SCHEMA LANGUAGE SPECIFICATIONS

Withdrawn/Retirée

PREFACE

This document is a report to the COBOL community from the CODASYL Programming Language Committee. It is an official report documenting the development activities of the CODASYL Programming Language Committee as of January 1, 1976, and is subject to change, extension and further development. Neither this document nor any of its precursors (i.e. COBOL-60, COBOL-61, COBOL-61 Extended, COBOL-65, CODASYL COBOL Journal of Development-1968, etc.) are to be construed as official standards. Standardization of COBOL in the United States is in the purview of ANSI (formerly USASI) Committee X3 and it, rather than CODASYL, is responsible for the preparation and submission for approval of a COBOL standard.

COBOL, Journal of Development, is composed of four sections as follows:

Section I.	History of CODASYL COBOL Development
Section II.	Philosophy of COBOL Use
Section III.	COBOL Language Specifications
Section IV.	Sub-schema Language Specifications

It is intended that the Journal of Development be published no more often than once each year and no less often than every three years.

The objectives for the different sections are as follows:

Section I. As a chronological record, this Section is a brief history of the CODASYL organization and of the development of the COBOL language definition from 1959 through the specification and publication of the present document. Participating companies during this period are noted as well as the official documents that have been published by the CODASYL Executive Committee.

Section II. In the design and use of the COBOL language specifications, a philosophy of language use exists from both an implementor and a user point of view. This Section is included as an expression of language guidelines to the user of the language. To aid the in-depth understanding of both the implementor and user, guidelines include background discussion of extended language features whose data processing concepts are not always evident. Guidelines also include objectives that the COBOL designers considered in the need for a common language and discuss the practical experience that is known today in relation to the original objectives.

Section III. This self-contained Section is a complete, formal, semantic description of the COBOL language specifications, independent of the COBOL Sub-schema. The need for a thorough language definition, without philosophical treatment of design criteria, is the objective in the presentation of the language.

Withdrawn/Retiree

Section IV. This Section contains the formal syntatic and semantic description of the language specifications for the COBOL Sub-schema. The Sub-schema describes not only the portion of a data base which is logically made available to the COBOL program, but also describes the form in which the COBOL program will use that data. Unlike Section III, this Section is not a stand-alone document but must be used with Section III in order to define the COBOL Sub-schema language completely.

Withdrawn/Retirée

ACKNOWLEDGMENT

The COBOL Journal of Development is the product of the CODASYL Programming Language Committee. This effort has been aided significantly by the contributions of the European Computer Manufacturer's Association (ECMA), the International Organization for Standardization (ISO), the Japanese COBOL Standards Committee, the American National Standards Institute, Inc. (ANSI), and by other interested organizations and individuals. The COBOL Journal of Development was developed by the Programming Language Committee and reflects the official CODASYL COBOL language as of January 1, 1976.

The maintenance of COBOL is the function of the Programming Language Committee of CODASYL (COnference on DAta SYstems Languages). Proposed changes to the language specifications are introduced by written proposal. Although no specific proposal format is mandatory the following guidelines are applied to proposal content:

1. Proposals made to specific points must cite all specific references.

2. Proposals of a general nature should cite at least specific instances.

3. Sufficient justification and motivation should be contained in the proposal to point out what the problem appears to be and why this proposal is a solution.

4. The proposal should include recommended specification changes with specific references where necessary.

Additional information concerning the procedures for proposing changes or suggested changes should be directed to the Programming Language Committee:

Chairman, Programming Language Committee
928, Garden City Drive
Box 124
Monroeville, Pennsylvania 15146

Any organization interested in reproducing the COBOL report and specifications in whole or in part, using ideas from this report as the basis for an instruction manual or for any other purpose, is free to do so. However, all such organizations are requested to reproduce the following acknowledgment paragraphs in their entirety as part of the preface to any such publication. Any organization using a short passage from this document, such as in a book review, is requested to mention "COBOL" in acknowledgment of the source, but need not quote the acknowledgment.

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.

Withdrawn/Retirée

The authors and copyright holders of the copyrighted material used herein

> FLOW-MATIC (trademark of Sperry Rand Corporation), Programming for the Univac (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F 28-8013, 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.

Withdrawn/Retirée

SECTION I: HISTORY OF CODASYL

COBOL DEVELOPMENT

Withdrawn/Retirée

TABLE OF CONTENTS

SECTION I: HISTORY OF CODASYL COBOL DEVELOPMENT

Page

CHAPTER 1. BACKGROUND AND INTRODUCTION

1.1 Objectives I-1-1
1.2 Establishment of CODASYL I-1-1

CHAPTER 2. DEVELOPMENT OF COBOL

2.1 Organization of COBOL Effort I-2-1
2.2 Evolution of COBOL I-2-7
2.3 Standardization I-2-12
2.4 Future Developments I-2-12

Withdrawn/Retirée

CHAPTER 1

BACKGROUND and INTRODUCTION

1.1 OBJECTIVES

On May 28 and 29, 1959 a meeting was held in the Pentagon for the purpose of considering both the desirability and the feasibility of establishing a common language for the programming of electronic computers for business-type applications. Representatives from users, both in private industry and in government, computer manufacturers, and other interested parties were present. The group agreed that the project should be undertaken. The following general objective was stated:

> The current experience of users of electronic data processing equipment indicates that a major problem in the efficient utilization of such equipment lies in the inability to state the data processing application in such a way that computer programs are developed and maintained with a minimum of time and programming effort.
>
> A COmmon Business Oriented Language, independent of any make or model of computer, open ended, and stated in both an English notation and a narrative form, would do much to solve or to ameliorate this problem. Such a language would also simplify and speed up the solution of the related problem of training personnel in the design of data processing systems and in the development of computer programs for such systems.
>
> In general, the development of a common language would serve to benefit the user in the following situations.
>
> > In developing data processing systems for existing computers, it is important that these systems be capable of processing on future, more powerful computers of any manufacturer, with a minimum of conversion costs.
> >
> > Full documentation of present systems in a form conducive to making changes and additions with minimum expenditures of time and cost is necessary in order to effectively meet the rapidly changing and expanding requirements of management.
> >
> > The need to produce a large number of computer programs in a short period of time often requires the augmentation of programming staffs by the addition of relatively inexperienced programmers.

1.2 ESTABLISHMENT OF CODASYL

The concept of the three committees was agreed upon and Short Range, Intermediate Range, and Long Range committees were established. The Short Range Committee was given the task of developing an immediate language and was instructed to take the best of three existing language-compiler systems, FLOWMATIC, AIMACO, and Commercial Translator, and to produce a language superior to any of these. The COnference on DAta SYstems Languages (CODASYL) developed out of that meeting. The Short Range Committee

Withdrawn/Retiree

eventually became the official COBOL branch of CODASYL and the Intermediate and the Long Range Committees evolved into the Systems and Language Structures Committees.

Withdrawn/Retirée

CHAPTER 2

DEVELOPMENT OF COBOL

2.1 ORGANIZATION OF COBOL EFFORT

The original COBOL organization was the Short Range Committee of CODASYL. By September, 1959, this committee had specified a language which they considered superior to existing language-compiler systems. This language specification was further modified and by December, 1959, COBOL existed as a language that was not identified with any manufacturer and therefore presented advantages for both government and private industry users.

2.1.1 INITIAL ORGANIZATION

The product of Phase I of COBOL development was a report published in April of 1960 by the Government Printing Office entitled ''COBOL—A Report to the Conference on Data Systems Languages, including Initial Specifications for a Common Business Oriented Language (COBOL) for Programming Electronic Digital Computers''. The language described in this report has since become known as COBOL-60.

The organizations participating in the original COBOL development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U.S. Navy
EDP Division, Minneapolis-Honeywell Regulator Co.
Burroughs Corporation
International Business Machines Corporation
Radio Corporation of America
Sylvania Electric Products Incorporated
Univac Division of Sperry Rand Corporation

2.1.2 THE COBOL MAINTENANCE COMMITTEE

The Executive Committee recognized that the task of defining COBOL was a continuing one and that the language had to be maintained and improved. To this end, the COBOL Maintenance Committee was created and charged with the task of answering questions arising from users and implementors of the language and making definitive modifications, including additions, clarifications, and changes to the language.

The Maintenance Committee was comprised of a Users Group and a Manufacturers Group. These groups met together but voted separately on proposals.

In order to devote concentrated attention to publishing a revised and updated COBOL-60, the Executive Committee created a Special Task Group. The product of this task group was the COBOL-61 manual, which was published by the Government Printing Office in mid-1961.

Withdrawn/Retiree

The next official COBOL publication was also the product of the Maintenance Committee and was called COBOL-61 Extended, which was released in mid-1963.

Organizations participating in the Maintenance Committee or the Special Task Group were:

Air Materiel Command, United States Air Force
Allstate Insurance Company
Bendix Corporation, Computer Division
The Boeing Company
Burroughs Corporation
Chase Manhattan Bank
Control Data Corporation
David Taylor Model Basin, Bureau of Ships, U.S. Navy
DuPont Company
General Electric Company
General Motors Corporation
International Business Machines Corporation
Lockheed Aircraft Corporation
Minneapolis-Honeywell Regulator Company
National Cash Register Company
Owens-Illinois, Incorporated
Philco Corporation
Radio Corporation of America
Royal McBee Corporation
Space Technology Laboratories, Incorporated
Southern Railway Company
Standard Oil Company (N.J.)
Sylvania Electric Products, Incorporated
Univac Division of Sperry Rand Corporation
United States Steel Corporation
Westinghouse Electric Corporation

The membership of the various Subcommittees of the COBOL Maintenance Committee was composed of members of both the Users and the Manufacturers groups.

2.1.3 THE COBOL COMMITTEE

In January, 1964, the COBOL Maintenance Committee was reorganized to provide a true industry group and to broaden its scope of activities. This organization is shown in Figure 2.1-1.

Withdrawn/Retirée

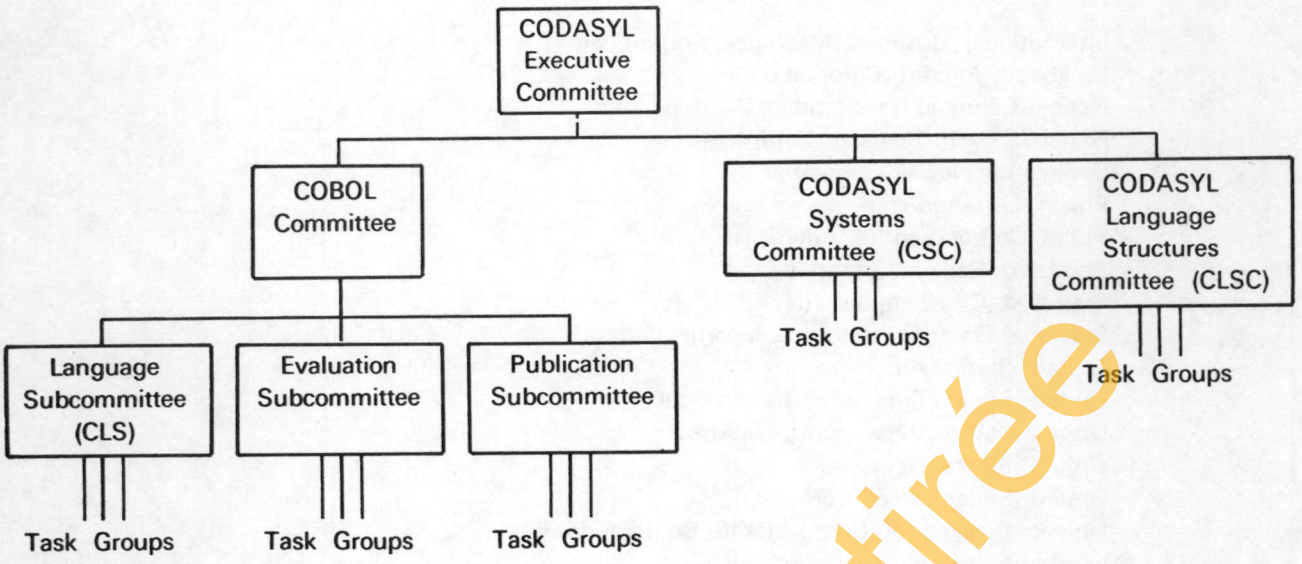


Figure 2.1–1 Organization of CODASYL (January 1964 — July 1968)

The Language Subcommittee's function was much the same as was that of the former COBOL Maintenance Committee, namely, the maintenance and further development of COBOL. In addition it carried on liaison with the United States of America Standards Institute (USASI: formerly the American Standards Association — ASA) and the International Organization for Standardization (ISO) in their work concerning the development of proposed COBOL Standards.

The Publication Subcommittee was charged with the production of official COBOL publications and liaison with USASI as to the content of the COBOL Information Bulletin (CIB). The CIB is a collection of material relating to COBOL, distributed to the COBOL community by USASI.

The Evaluation Subcommittee's task was the analysis and evaluation of compiler implementations and user surveys. This Subcommittee provided information to the COBOL Committee regarding the use of COBOL.

The product of the COBOL Committee was the manual, "COBOL, Edition 1965".

Organizations participating in the work of the COBOL Committee were:

Allstate Insurance Company
American Telephone and Telegraph Company
The Boeing Company
Burroughs Corporation
Canadian Federal Government
Commercial Credit Corporation
Collins Radio Company
Control Data Corporation
General Electric Company
General Motors Corporation
Honeywell, Incorporated

Withdrawn/Retirée

International Business Machines Corporation
Lockheed Aircraft Corporation
National Bureau of Standards
National Cash Register Company
Owens–Illinois, Incorporated
Philco Corporation
Radio Corporation of America
Southern Railway Company
Standard Oil Company (N. J.)
Sylvania Electric Products, Incorporated
United States Air Force
United States Bureau of the Budget
United States Department of Army
United States Navy
United States Steel Corporation
Univac Division of Sperry Rand Corporation
Westinghouse Electric Corporation

2.1.4 PROGRAMMING LANGUAGE COMMITTEE

In July, 1968 the CODASYL Executive Committee adopted a revised constitution which accomplished certain needed organization changes in an effort to stabilize and improve the methods of achieving CODASYL objectives. This revised organization is shown in Figure 2.1-2.

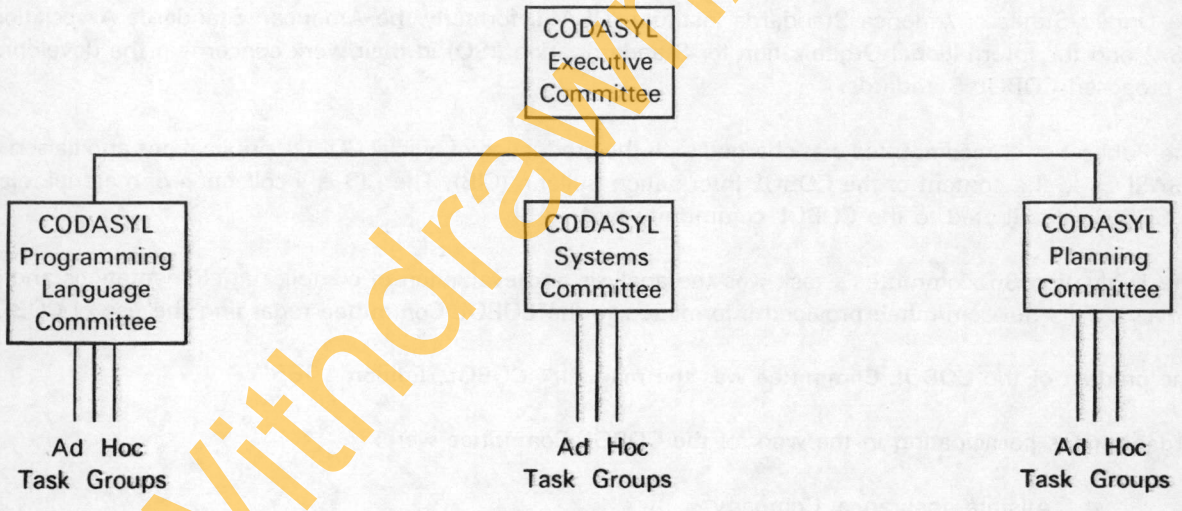


Figure 2.1-2 Organization of CODASYL (July 1968 — August 1971)

With the formation of the Programming Language Committee (PLC) the former COBOL Language Subcommittee has been elevated to full committee status, and its chairman becomes a member of the Executive Committee.

The purpose and objectives of PLC include and extend those of the former CLS. The objectives are to make possible: compatible, uniform, source programs and object results, with continued reduction in the number

Withdrawn/Retirée

of changes necessary for conversion or interchange of source programs and data. The PLC concentrates its efforts in the area of tools, techniques and ideas aimed at the programmer.

The Programming Language Committee is responsible for the presentation of the COBOL Journal of Development.

Member organizations of the COBOL Programming Language Committee are:

American Telephone and Telegraph Company
Applied Data Research, Inc.
Burroughs Corporation
Canadian Federal Government
Cincom Systems, Incorporated
Computer Sciences Corporation
Control Data Corporation
Defense Supply Agency
Digital Equipment Corporation
Honeywell Information Systems, Incorporated
International Business Machines Corporation
International Computers Limited
National Bureau of Standards
NCR Corporation
Southern Railway Company
Sperry Univac, Division of Sperry Rand Corporation
United States Air Force
United States Army
United States Navy
Westinghouse Electric Corporation

2.1.5 OTHER CODASYL ACTIVITIES

In 1961 a portion of the Intermediate Range Committee was combined with the Long Range Committee to form the Development Committee. This committee was comprised of a Systems Group and a Language Structures Group. In April 1965 the Development Committee was reorganized as the CODASYL Systems Committee (CSC) and CODASYL Languages Structures Committee (CLSC) — as shown in Figure 2.1-1.

Before its reorganization, the Development Committee had produced:

a. In 1962, through the Systems Group, a "Decision Table Structured Language", identified as "Detab-X".

b. In 1962, through the Language Structures Group, a "non-procedural approach to problem statement", identified as "Information Algebra".

Under this revised committee structure, work continued on the further development of these and similar languages and techniques.

In July 1968 the CODASYL Executive Committee reorganized these committees resulting in the formation of the Planning Committee (replacing the CLSC) and redefinition of the work of the Systems Committee. This structure is shown in Figure 2.1-2. These two committees have the following purposes:

Withdrawn/Retirée

a. The CODASYL Planning Committee is to aid in CODASYL planning by gathering, assimilating and disseminating information from implementors and users pertaining to the goals of CODASYL.

b. The CODASYL Systems Committee is to build an expertise in, and to develop advanced languages and techniques for data processing, with the aim of automating as much as possible of the processes currently thought of as systems analysis, design and implementation.

In August, 1971, the CODASYL Executive Committee announced the formation of a new standing committee, the Data Description Language Committee. This committee, using the foundation established by the April, 1971 report of the Data Base Task Group (DBTG) of the Programming Language Committee, was charged with finalizing the specifications for a common Data Description Language, independent of but common to many other higher-level programming languages.

This organization is shown in Figure 2.1-3.

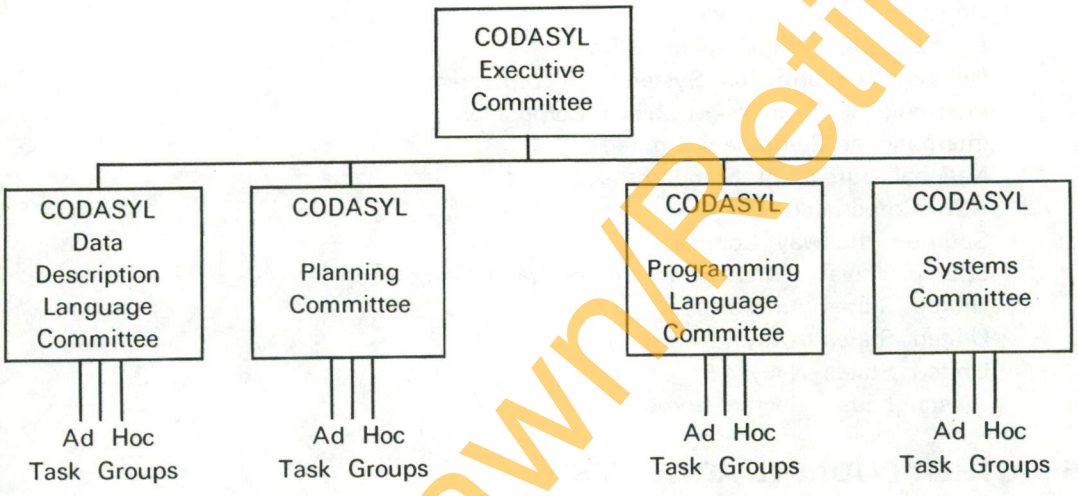


Figure 2.1-3 Organization of CODASYL (August 1971 to 1974)

In late 1974, the CODASYL Executive Committee announced the formation of the CODASYL FORTRAN Data Manipulation Language Committee. This committee, using the CODASYL Data Description Language Journal of Development as its basic specification, was charged to develop a data manipulation language specification for FORTRAN within the confines of that language as specified in the American National FORTRAN Standards. Also, earlier in 1974, the CODASYL Executive Committee announced that the CODASYL Planning Committee had been disbanded.

Withdrawn/Retirée

The revised organization of CODASYL is shown in Figure 2.1-4.

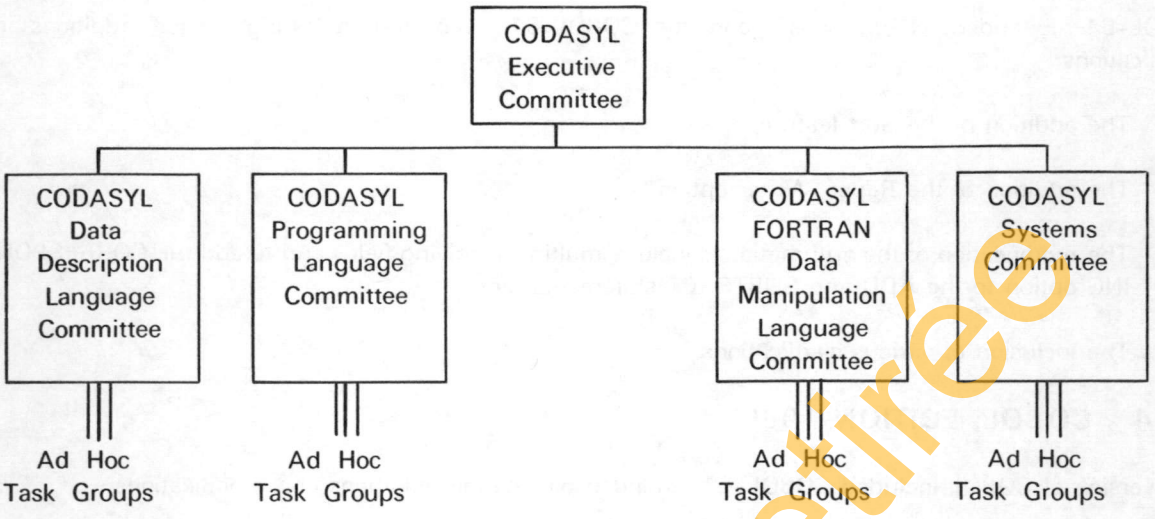


Figure 2.1-4 Organization of CODASYL (Since 1974)

2.2 EVOLUTION OF COBOL

2.2.1 COBOL-60

COBOL-60, the first version of the language published, proved that the concept of a common business oriented language was indeed practical.

2.2.2 COBOL-61

COBOL-61, the second official version of COBOL, was not completely compatible with COBOL-60. The changes were in areas such as organization of the Procedure Division rather than the addition of any major functions. The avowed goal of CODASYL in terms of successive versions of the language was to make changes of an evolutionary rather than revolutionary nature. This version was generally implemented and was the basis for many COBOL compilers.

The terms ''Required'' and ''Elective'' COBOL were adopted by the Executive Committee at this time in order to provide a uniform basis for a minimum implementation of COBOL. Those elements that were required to be implemented were designated as ''Required' elements. Those elements of the language, the implementation of which it was permissible for the manufacturer to delay temporarily, were designated as ''Elective'' elements.

2.2.3 COBOL-61 EXTENDED

This version of COBOL was generally compatible with COBOL-61. The term ''generally'' must be used, not because of any basic changes in the philosophy or organization of the language, but because certain arithmetic

Withdrawn/Retiree

extensions and general clarifications did make the syntax for certain statements and entries different from those in COBOL-61.

COBOL-61 Extended, then, was generally COBOL-61 with the following major additions and modifications:

a. The addition of the Sort feature,

b. The addition of the Report Writer option,

c. The modification of the arithmetics to include multiple receiving fields and to add the CORRESPONDING option to the ADD and SUBTRACT statements, and

d. The inclusion of various clarifications.

2.2.4 COBOL, EDITION 1965

This version of COBOL included COBOL-61 Extended plus certain additions and modifications.

The major changes incorporated in COBOL, Edition 1965 were:

a. The inclusion of a series of options to provide for the reading, writing and processing of Mass Storage files,

b. The addition of the Table Handling feature which includes indexing and search options,

c. The modification of the specifications to delete the requirement for specific error diagnostic messages,

d. The deletion of the terms ''Required'' and ''Elective'', and

e. The inclusion of various clarifications.

2.2.5 COBOL, 1968

This version of COBOL, published in the Journal of Development, was based on COBOL, Edition 1965 with certain additions, and deletions.

The major changes incorporated in COBOL, 1968 were:

a. The inclusion of inter-program communication and the concept of a run unit,

b. The elimination of redundant editing clauses and certain data clauses more succinctly expressed by the PICTURE clause,

c. An improved COPY specification for all divisions except the Identification Division and the elimination of the INCLUDE statement,

d. The inclusion of a hardware independent means of specifying and testing for page overflow conditions,

Withdrawn/Retirée

e. The elimination of type 4 abbreviations,

f. The elimination of the DEFINE statement,

g. The inclusion of a remainder option for the DIVIDE statement,

h. The deletion of NOTE and REMARKS in favor of a general comment capability for all divisions,

i. The inclusion of the SUSPEND statement as additional means of controlling graphic display devices,

j. The inclusion of additional abbreviations,

k. A revision of the EXAMINE statement to allow the specification of dynamic parameter values, and

l. The inclusion of various clarifications.

2.2.6 COBOL, 1969

This version of COBOL, published in the Journal of Development, is based on COBOL, 1968 with certain additions and deletions.

The major changes incorporated in COBOL, 1969 were:

a. The deletion of the EXAMINE statement and the inclusion of a more powerful statement, INSPECT, in its place,

b. The inclusion of a communications facility to permit input and output with communications devices,

c. The inclusion of the STRING and UNSTRING statements, to facilitate character string manipulation,

d. Deletion of the CONSTANT SECTION of the Data Division,

e. The inclusion of a compile-time page ejection facility,

f. The inclusion of a facility to access the system's date and time,

g. The inclusion of a SIGN clause as a means of specifying the position and mode of representation of the operational sign, and

h. The inclusion of various clarifications.

2.2.7 COBOL, 1970

This version of COBOL, published in the Journal of Development, is based on COBOL, 1969 with certain additions, deletions, and modifications.

The major changes incorporated in COBOL, 1970 are:

Withdrawn/Retirée

a. The deletion of the RANGE clause,

b. The inclusion of the INITIALIZE statement, to facilitate setting data items to values consistent with their data descriptions,

c. The inclusion of a debugging facility,

d. The inclusion of a merge facility,

e. A complete revision to the Report Writer function, and

f. The inclusion of various clarifications.

2.2.8 COBOL, 1973

This version of COBOL, published in the Journal of Development, is based on COBOL, 1970 with certain additions, deletions, and modifications.

The major changes incorporated in COBOL, 1973 are:

a. A revision and extension to the mass storage facility,

b. A clarification and extension to the COBOL library facility,

c. An enhancement of the INSPECT statement,

d. A revision to the SELECT statement for a sort or merge file which included the deletion of Format 3,

e. A revision to the RERUN facility,

f. The removal of the restriction on 77 level-numbers that they must precede 01 level-numbers,

g. The inclusion of a page advancing feature as part of the WRITE statement,

h. A clarification and enhancement of the COBOL language structure,

i. An enhancement of the LINAGE clause to permit specification of margins, and

j. The inclusion of various clarifications.

2.2.9 COBOL, 1976

This version of COBOL, published in the Journal of Development, is based on COBOL, 1973 with certain additions, deletions and modifications.

The major changes incorporated in COBOL, 1976 are:

a. The inclusion of a data base facility which interfaces with the CODASYL Data Description Language Journal of Development,

Withdrawn/Retirée

b. The inclusion of collating sequence and character set declarations,

c. The inclusion of a boolean (bit) manipulation facility,

d. The inclusion of a de-editing facility,

e. The inclusion of a reference modification facility,

f. The inclusion of extensions to the file processing capabilities in the Environment and Data Divisions,

g. The inclusion of the DELETE FILE statement,

h. The inclusion of the PURGE statement which prevents partial messages from being communicated to a communications device,

i. The inclusion of a variable-length record facility,

j. The removal of random processing specifications,

k. The removal of the ALTER statement,

l. The removal of all numeric paragraph and section names,

m. The removal of the OPEN REVERSED facility,

n. The removal of level-number 77,

o. Realignment of clauses between the Environment and Data Divisions,

p. An option to omit the FILLER clause,

q. An enhancement to the table handling facility to allow specification of tables having more than three dimensions,

r. An enhancement to the DISPLAY statement to allow NO ADVANCING,

s. An enhancement to the INSPECT statement to simplify data transformation,

t. The extension of the SORT and MERGE statements to permit multiple file specifications in the GIVING phrase,

u. The extension of the SORT and MERGE statements to relative and indexed files,

v. The extension of the use of OPTIONAL to all file organizations, and

w. The inclusion of various clarifications.

Withdrawn/Retirée

2.3 STANDARDIZATION

The effort to define an International COBOL standard was begun in October of 1962 by the ISO Technical Committee 97, Computers and Information Processing, Working-Group E, Programming Languages. Working-Group E is now Subcommittee 5. The effort to define a United States COBOL standard was begun in September of 1962 by the ASA (now ANSI) X3.4.4 Task Group (now X3J4 Technical Committee) on Processor Specifications and COBOL. This latter effort resulted in the American National Standard COBOL X3.23-1974 which was approved in May 1974. These standardization efforts are based on the technical content of COBOL as defined by CODASYL.

2.4 FUTURE DEVELOPMENTS

Further development and modification of COBOL are the responsibility of the Programming Language Committee of CODASYL. The development effort continues with emphasis on:

a facility to define functions
a facility for a generalized error reporting facility
an extension to the inter-program communication facility
an asynchronous processing facility
a clarification and extension of file label processing facilities
a facility to permit COBOL shorthand
additional facilities to support structured programming
a facility to control compiler capabilities for presentation of the source program

Withdrawn/Retirée

SECTION II: PHILOSOPHY OF COBOL USE

Withdrawn/Retirée

TABLE OF CONTENTS

SECTION II: PHILOSOPHY OF COBOL USE

Page

CHAPTER 1. BACKGROUND II-1-1

CHAPTER 2. CONCEPTS

2.1 Files II-2-1
2.2 Report Writer II-2-8
2.3 Table Handling II-2-12
2.4 Rerun II-2-18
2.5 Shared Memory Area II-2-19
2.6 Program Communication II-2-20
2.7 Communications Facility II-2-23
2.8 Data Base Facility II-2-31

Withdrawn/Retirée

CHAPTER 1

BACKGROUND

1.1 GENERAL

The task of specifying COBOL was undertaken because of the computer users' need for a problem-oriented, machine-independent language for business applications. Such a language, in order to be successful, should have certain characteristics. Programs written in the language should be capable of easy conversion from one machine to another, the language should be easily understandable in format and notation, it should provide good program documentation, and it should be capable of specifying data processing problems in such a way that an implementation of the language by means of a compiler can produce efficient object code.

COBOL is a language for programmers. It is not intended to be used by people unfamiliar with computers. The attributes of the language make it more feasible for use by applications oriented personnel and make it easier to teach to novices, but an understanding of programming and data processing concepts is prerequisite to the writing of efficient COBOL programs.

Although COBOL is a programmer's language, major benefits of the use of COBOL accrue to management by improving the over-all efficiency of the data processing function.

In describing a data processing problem, there are two elements involved. One is the set of procedures which specify how the data is to be manipulated, and the other is a description of the data involved. Furthermore, it was recognized that certain information pertaining to the specific computer on which the problem is to be run and some information identifying the program were also a necessary part of the description of the problem. The information pertaining to the computer itself would never carry over from one computer to another. However, it was felt that the advantages of having a common means of expression were sufficiently great to warrant the development of a standard form for even those items which clearly changed from computer to computer.

Withdrawn/Retiree

COBOL SYSTEM ELEMENTS

The COBOL system is composed of two elements, the source program written in COBOL and the compiler which translates this source program into an object program capable of running on a computer. This report, in general, considers only the source program and does not consider the compiler directly. However, the specifications of a language obviously determine, to a large extent, the boundaries of a compiler. Therefore, the compiler is mentioned in certain cases to facilitate the explanation of the language.

A source program is used to specify the solution of a business data processing problem. The four elements of this specification are the:

1. Identification of the program.
2. Description of the equipment being used in the processing.
3. Description of the data being processed.
4. Set of procedures which determine how the data is to be processed.

The COBOL system consists of two separate and distinct types of source programs: the COBOL source program and the COBOL Sub-schema source program.

The COBOL source program has a separate division for each of the elements listed above. The names of these divisions respectively are:

IDENTIFICATION
ENVIRONMENT
DATA
PROCEDURE

The COBOL Sub-schema source program consists of descriptions of data to be processed that is stored in a data base. It consists of the following divisions:

TITLE
MAPPING
STRUCTURE

IDENTIFICATION DIVISION

The purpose of the Identification Division is to identify the source program and the outputs of a compilation. In addition, the user may include the date that the program was written, the date that the compilation was accomplished, and any other information which is desired.

Withdrawn/Retired

ENVIRONMENT DIVISION

The Environment Division is that part of the source program which specifies the equipment being used. It contains descriptions of the computers to be used both for compiling the source program and for running the object program. Problem-oriented names may be assigned to particular equipment. Those aspects of a file which relate directly to hardware are described here. Because this division deals entirely with the specifications of the equipment being used, it is largely computer dependent.

DATA DIVISION

The Data Division uses file and record descriptions to describe the files of data that the object program is to manipulate or create and the individual logical records which comprise these files. The characteristics or properties of the data are described in relation to a standard data format rather than an equipment-oriented format. Therefore, this division is to a large extent computer independent. While compatibility among computers cannot, in general, be absolutely assured, careful planning in the data layout will permit the same data description, with minor modifications, to apply to more than one computer.

The Data Division also specifies the Sub-schema to be used in accessing data stored in a data base.

PROCEDURE DIVISION

The Procedure Division specifies the steps that the user wishes the computer to follow. These steps are expressed in terms of meaningful English words, statements, sentences, and paragraphs. This aspect of the over-all system is often referred to as the "program"; in reality it is only part of the total specification of the problem solution (i.e. the program) and is insufficient, by itself, to describe the entire problem. This is true because repeated references must be made, either explicitly or implicitly, to information appearing in the other divisions. This division, more than any other, allows the user to express his thoughts in meaningful English. Concepts of verbs to denote actions and sentences to describe procedures are basic, as is the use of conditional statements to provide alternative paths of action. The Procedure Division is essentially computer independent. That is, any user of COBOL can understand the information appearing in this division without regard to any particular computer. Furthermore, every COBOL compiler should interpret this information in the same way if the source program is syntactically correct.

As explanation of the divisions of the COBOL Sub-schema source program is contained in the appropriate section. (See Section IV, The COBOL Sub-schema Language Specifications.)

1.1.1 DOCUMENTATION

The completeness of program documentation produced by a COBOL compilation is dependent upon the particular implementation (compiler) and upon the amount of thought and effort expended by the programmer. Because of the narrative nature and English-like syntax of the language, however, a certain amount of documentation is inherent. If the user establishes programmer guidelines so that data-names and procedure-names are consistent and meaningful, the listings produced by a compilation will constitute a good basis for a program documentation package. Such techniques as the use of the PICTURE clause for all data description at the elementary item level and the use of the COPY clause to call standard data descriptions or

Withdrawn/Retirée

procedures from a library further enhance documentation. Generally, the quality and quantity of documentation produced by the use of COBOL is superior to that of other languages. This superiority is very important for the maintenance and revision of production programs.

1.1.2 COMPREHENSION

This feature of COBOL is in one respect closely allied with the previous topic of documentation. There is a good deal more to be said, however, for the relative ease of understanding a COBOL program.

The organization of the language, both as to data hierarchies and procedure statements, result in COBOL being much easier to teach and understand than machine-oriented languages or assembly systems. This attribute makes COBOL very useful for man-to-man as well as man-to-machine communications. However, it is necessary to keep in mind the importance of comprehensive training, including basic computer logic and programming fundamentals, as well as COBOL training itself.

In general, it is wise to keep the Procedure Division as simple as possible by avoiding the use of complex, nested, or compound sentences where practicable.

1.1.3 COMPATIBILITY

While COBOL is not a completely machine-independent language, it closely approaches this goal. The user must exercise some degree of care if he wishes to produce highly compatible programs. It is appropriate here to examine, division by division, the degree of machine-independence in a COBOL program.

The Identification Division is totally machine-independent. However, since the Identification Division does not produce object code, this machine-independence is not of great consequence to the user.

The Environment Division is almost completely machine-dependent. Those aspects of the data processing problem that are related to a particular hardware system are specified here. When a COBOL program is converted from one computer to another, this division normally would require at least a partial rewrite. The Environment Division usually consists of only three or four paragraphs, so that the effort of rewriting it is not large.

The Data Division is compatible across machine lines in so far as the data being described are compatible. For example, changes will be required in the Data Division if the user is converting from a BCD, variable word length machine, to a fixed word length machine with more than one internal representation.

The Procedure Division is compatible across machine lines to the extent that the language can separate the problem from the hardware. If, for instance, the collating sequence of a new machine is different from that of the machine currently in use, certain statements in the Procedure Division may have to be rewritten. Aside from this type of ''language independent'' circumstance, the Procedure Division should convert from one machine to another with no change. The user can create incompatibilities if non-COBOL coding is introduced with the ENTER statement; if the COMPUTE statement is used to combine arithmetic operations, and the number of digits in the intermediate results of these arithmetic operations must be carefully controlled; or if implementor-dependent techniques are used.

Withdrawn/Retired

This discussion of compatibility across machine lines assumes that features used in writing COBOL programs are present in both compilers.

1.1.4 EFFICIENCY

User experience has shown that there are many ways to achieve highly efficient COBOL object programs. In discussing the efficiency of COBOL, a distinction must be made between efficiency of compilation and efficiency of the object program.

As far as the speed of compilation is concerned, experience has shown that the most important factor here is the design of the compiler. Though the complexity and notational form of COBOL have some effect on compilation speed, examination of current implementations shows that compiling can be quite fast. Because the diagnostics included in the compiler and the documentation produced by the compiler significantly affect the measurement of compiling efficiency, they, as well as the factor of compilation speed, must be considered in this measurement.

In general, the efficiency of the object code produced is not limited by the structure, notation, or format of COBOL. Object program efficiency is mainly dependent on the compiler and the source programmer. Again, using current implementations as a guide, COBOL compilers are capable of producing efficient object programs, both in terms of running speed and in terms of memory usage.

One point of confusion that has arisen periodically during the history of COBOL use is language problems versus compiler problems. While it is true that language notation or structure can contribute to inefficient implementations, COBOL can, and has been, implemented efficiently.

With the dependence that exists between highly efficient object programs and programmer understanding of compiler idiosyncrasies, the implementor should supply his user with a manual containing specific guidelines outlining peculiarities of the compiler and methods for achieving most efficient programs. The user must exercise care in establishing the content of training courses and installation standards which result in compatible and efficient programs.

Withdrawn/Retirée

CHAPTER 2

CONCEPTS

COBOL offers many features which allow the user to obtain a necessary function without programming the function in detail. In this chapter we will discuss each of these features, considering the reason for its inclusion in the language and the concept of its use and organization.

2.1 FILES

A file is a collection of records which may be placed into or retrieved from a storage media. 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.

2.1.1 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. Some of the other fixed attributes of the file are primary key, the maximum logical record size, the labelling conventions, the record type (fixed or variable) and the block type and factor.

2.1.1.1 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.

2.1.1.2 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.

Withdrawn/Retirée

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.

2.1.1.3 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 Description 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 Description entry. The value of a particular Alternate Record Key in each record need not be unique. When these values may not be unique, the DUPLICATES phrase is specified in the ALTERNATE RECORD KEY clause.

2.1.1.4 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. There are two types of records: fixed-length and variable-length. When a file is created, it is declared to contain either fixed-length or variable-length records. In any case, the contents of the record area do not reflect any information the implementor may add 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.

2.1.1.4.1 Fixed-Length Records

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. Even though more than one record description may be given for the file and the record descriptions may describe different numbers of character positions, the records are still fixed-length and the size of each record is constant.

2.1.1.4.2 Variable–Length Records

Variable-length records may contain differing numbers of character positions among the records in the file. There are three ways to specify variable-length records. These are the RECORD IS VARYING DEPENDING ON clause, the OCCURS DEPENDING ON clause, and by specifying different-length record areas. In all cases, the VARYING phrase must be specified in the File Description or Sort-Merge Description entry for the file.

Withdrawn/Retirée

2.1.2 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.

2.1.2.1 Record Operations

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 may be sequential, random or dynamic.

For files that are organized as relative or indexed, any of the three access modes can be used to access the file regardless of the access mode used to create the file. A file with sequential organization may only be accessed in sequential mode.

The organization, format and contents of an output report may be specified using the Report Writer feature. (See Paragraph 2.2, Report Writer.)

2.1.2.1.1 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. 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 that set. The START statement may be used to establish a starting point within an indexed file for a series of subsequent sequential retrievals.

2.1.2.1.2 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.

Withdrawn/Retirée

2.1.2.1.3 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.

2.1.2.1.4 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.

A file in the open INPUT mode 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.

The EXTEND mode applies only to sequential organization. In this case, new records are added to the end of a file by issuing WRITE 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 from the file logically; the WRITE statement may be used in random or dynamic access modes to insert a new record into the file.

2.1.2.1.5 Next Record Pointer

The Next Record Pointer is a conceptual entity used in this document to facilitate exact specification of the next record to be accessed within a given file under combinations of operations. The setting of the Next Record Pointer is affected only by the DELETE, OPEN, READ and START statements. The concept of a Next Record Pointer has no meaning for a file opened in the OUTPUT mode.

Withdrawn/Retirée

2.1.2.1.6 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.

2.1.2.2 File Operations

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

2.1.2.2.1 File Deletion

Many data processing applications are cyclic in nature, involving the creation of new files which obsolete older files. A method for releasing the storage media, on which an obsolete file is recorded, is required in order that the storage media may be reused. The DELETE FILE statement supplies the required method.

2.1.2.2.2 Sorting and Merging

2.1.2.2.2.1 Sorting

In many sort applications it is necessary to apply some special processing to the contents of a sort file. The special processing may consist of addition, deletion, creation, altering, editing, or other modification of the individual records in the file. It may be necessary to apply the special processing before or after the records are reordered by the sort, or special processing may be required in both places. The COBOL sort feature allows the user to express these procedures and to specify at which point, before or after the sort, they are to be executed. A COBOL program may contain any number of sorts, and each of them may have its own input and output procedures. The sort feature automatically causes execution of these procedures at the specified point in such a way that extra passes over the sort file are not required.

Within an input procedure, the RELEASE statement is used to create the sort file. That is, at the completion of execution of the input procedure those records that have been processed by use of the RELEASE statement (rather than the WRITE statement) comprise the sort file, and this file is available only to the SORT statement. Execution of the SORT statement arranges the entire set of records in the sort file according to the keys specified in the SORT statement. The sorted records are made available from the sort file by use of the RETURN statement during execution of the output procedure.

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 of 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. The sort file itself is referred to and accessed only by the SORT statement. A Sort-Merge File Description can be considered to be a particular type of File Description. That is, a sort file, like any file, is a set of records.

Withdrawn/Retirée

2.1.2.2.2.2 Merging

In some applications it is necessary to apply some special processing to the contents of a merged file. The special processing may consist of addition, deletion, altering, editing, or other modification of the individual records in the file. The COBOL merge feature allows the user to express an output procedure to be executed as the merged output is created. The merged records are made available from the merge file by use of the RETURN statement in the output procedure.

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. The merge file itself is referred to and accessed only by the MERGE statement. A Sort-Merge File Description can be considered to be a particular type of File Description. That is, a merge file, like any file, is a set of records.

2.1.2.3 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.

2.1.2.3.1 Status Keys

Status keys are elementary alphanumeric data items which are referenced in the File Description entry for the file. A numeric value is placed into the status keys as a result of the execution of CLOSE, DELETE, OPEN, READ, REWRITE, START and WRITE statements. This value indicates status of the input or output operation. This value is placed into the data item during the execution of the statement and prior to the execution of any optional phrase or phrases associated with that statement and prior to the execution of any exception declarative.

2.1.2.3.2 Exception Declaratives

A USE AFTER STANDARD EXCEPTION procedure, when one is specified for the file, is executed whenever an input or output condition arises which would result in a non-zero value for the status key (whether one is specified or not). 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.

2.1.2.3.3 Optional Phrases

The INVALID KEY phrase may be associated with the DELETE, READ, REWRITE, START or WRITE statements. An invalid key condition occurs when a requested key does not exist in the file (DELETE, READ or START statements), or when a key is already in a file and duplicates are not allowed (WRITE statement),

Withdrawn/Retirée

or when a key does not exist in the file or that 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 phrase has been specified, the statement identified by that INVALID KEY phrase is executed.

The AT END phrase may be associated with a READ statement. The at end condition occurs in a sequentially accessed file when no next logical record 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.

Withdrawn/Retirée

2.2 REPORT WRITER

The Report Writer is a special purpose feature which places its emphasis on the organization, format and contents of an output report. Although a report can be produced using the standard COBOL language, the Report Writer language features provide a more concise facility for report structuring and report production. Much of the Procedure Division programming which would normally be supplied by the programmer is instead provided automatically by the Report Writer Control System (RWCS). Thus the programmer is relieved of writing procedures for moving data, constructing print lines, counting lines on a page, numbering pages, producing heading and footing lines, recognizing the end of logical data subdivisions, updating sum counters, etc. All these operations are accomplished by the Report Writer Control System as a consequence of source language statements that appear primarily in the Report Section of the Data Division of the source program.

2.2.1 REPORT SECTION

The Report Section of a COBOL Data Division contains one or more Report Description (RD) entries, each of which forms the complete description of a report.

The report named in the RD entry is not assigned directly to an output device. Instead, it is associated with a file in the File Section and the file is assigned to an output device in the normal manner. More than one report may be associated with the same file. The File Description entry of a file 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 an RD entry for each report associated with that file.

The RD entry contains a set of clauses that names the report and supply 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 RD entry so that each report may be identified separately in an intermediate output file.

Following each RD entry are one or more 01 level-number entries, each followed by a hierarchic 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.

2.2.2 REPORT STRUCTURE

When structuring a report, major consideration must be given to vertical and horizontal spacing requirements, manipulation of data, and the physical and logical subdivisions of a report.

2.2.2.1 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.

Withdrawn/Retired

2.2.2.2 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 supplies space-fill between all defined fields.

2.2.2.3 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. The Report Section clauses which effect the manipulation of data are the SOURCE, SUM, and VALUE clauses.

The SOURCE clause specifies the sending data item of an implicit MOVE statement. The receiving printable item is defined by the description of the report group item in which the SOURCE clause appears.

The SUM clause automatically causes the establishment of a sum counter. The object of the SUM clause names the data item(s) which are added to the sum counter when a GENERATE statement is executed. The move of the sum counter contents to the receiving printable item, defined by the description of the report group item in which the SUM clause appears, is accomplished automatically when that report group is presented.

The VALUE clause defines a literal that appears in the printable item of a report group each time that report group is to be presented.

In summary, 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.

2.2.2.4 Report Subdivisions

The physical and logical organization of a report interact to determine what is presented on a page.

2.2.2.4.1 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.

2.2.2.4.2 Logical Subdivisions 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.

Withdrawn/Retirée

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 automatically 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.

2.2.3 PROCEDURE DIVISION REPORT WRITER STATEMENTS

The Report Writer statements that appear in the Procedure Division are:

1. INITIATE
2. GENERATE
3. TERMINATE
4. SUPPRESS
5. USE BEFORE REPORTING

The INITIATE statement causes the Report Writer Control System to perform automatically a number of initialization functions. A report must be initiated before any detail processing may take place.

The GENERATE statement which specifies a data-name causes the named DETAIL report group to be formatted and written to the output device. In addition, it triggers the Report Writer Control System to perform the many implicit actions described in the preceding section.

The GENERATE statement which specifies a report-name provides a means of summary reporting. A report produced by this type of statement has all detail print lines suppressed automatically and consists of only the summary totals accumulated during the processing of the DETAIL report group. The RWCS processing for a GENERATE report-name statement is identical to that which occurs for a GENERATE data-name statement, except that the former results in the suppression of detail print lines.

The TERMINATE statement causes the Report Writer Control System to perform all of the automatic functions associated with the termination of a report. The TERMINATE statement must be executed before the file containing the report is closed.

The SUPPRESS statement provides the object time facility to suppress the printing of an entire report group.

The BEFORE REPORTING phrase of the USE statement provides a mechanism whereby Procedure Division statements may be executed at specific instances within the automatic procedures performed by the Report Writer Control System. The statements in the USE BEFORE REPORTING procedure may alter the contents

Withdrawn/Retired

of the data items that are referenced by SOURCE clauses. Thus control is possible over the contents of data items referenced within report groups that are produced automatically.

Withdrawn/Retirée

2.3 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, and this technique of specifying individual table elements is called subscripting.

In order to facilitate such operations as table searching and manipulating specific items, a technique called indexing is also available. Both subscripting and indexing are discussed below.

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

2.3.1 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, but DOG does not since there is an OCCURS clause in the description of the group item (TABLE-ELEMENT) which contains DOG.

Withdrawn/Retired

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 have a group name.

Example 3.

```
01  ABLE.
    02  BAKER;  . . .
    02  CHARLIE;  OCCURS 20 TIMES;  . . .
    02  DOG;  . . .
```

Example 4.

```
01  ABLE.
    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.

Withdrawn/Retirée

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.

2.3.2 INITIAL VALUES OF TABLES

In the Working-Storage section, initial values of elements within tables are specified in one of the following ways:

1. The table may be described as a series of separate Data Description entries all subordinate to the same group item, each of which specifies the VALUE of an element, or part of an element, of the table. In defining the record and its elements, any data description clause (USAGE, PICTURE, etc.) may be used to complete the definition, where required. This form is required when the elements of the table require separate handling due to synchronization, USAGE, etc. The hierarchical structure of the table is then shown by use of the REDEFINES entry and its associated subordinate entries. The subordinate entries, following the REDEFINES entry, which are repeated due to OCCURS clauses, must not contain VALUE clauses.

2. When the elements of the table do not require separate handling, the VALUE of the entire table may be given in the entry defining the entire table. The lower level entries will show the hierarchical structure of the table; lower level entries must not contain VALUE clauses.

2.3.3 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 4th BAKER or the 4th CHARLIE would be complete, whereas a reference to the 4th DOG would not. To reference DOG, which is an element of a two-dimensional table, the user must reference, for example, the 4th DOG in the 5th BAKER.

2.3.4 SUBSCRIPTING

One method by which occurrence numbers may be specified is to append one or more subscripts to the data-name. A subscript is an integer whose value specifies the occurrence number of an element within the

Withdrawn/Retirée

group item that has the next lower level-number. The subscript can be represented either by a literal which is an integer or by a data-name which is defined elsewhere as a numeric elementary item with no character positions to the right of the assumed decimal point. In either case, the subscript enclosed in parentheses is written immediately following any required qualification for the name of the table element. The number of literals and data-names 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.

Example 6.

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

In Example 6, 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.

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, then the subscripts are written from left to right in the order major, intermediate, and minor. Thus, in Example 6, a reference to HARRY (18, 2, 7) means the HARRY in the 7th GEORGE, in the 2nd DOG, in the 18th BAKER.

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

When a data-name is used as a subscript, it may be used to reference items within many different tables. These tables need not have elements of the same size. The data-name may also appear as the only subscript

Withdrawn/Retirée

with one item and as one of two or more subscripts with another item. Also, it is permissible to mix literal and data-name subscripts, for example, HARRY (12, NEWKEY, 2).

2.3.5 INDEXING

Another method of referencing items in a table is indexing. To use this technique, the programmer assigns one or more index-names to an item whose Data Description entry contains an OCCURS clause. The INDEXED BY clause, 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-name since its definition is completely hardware-oriented and it is not considered data per se. At object time the contents of the index-name will correspond to an occurrence number for that specific dimension of the table to which the index-name was assigned; however, the manner of correspondence will be determined by the implementor. The initial value of an index-name at object time is not determinable and the index-name must be initialized by the SET statement before use. The initial value of an index-name is also specified in the PERFORM statement with the VARYING phrase and the SEARCH statement with the ALL phrase.

An index-name cannot be described with a Data Description entry, and therefore cannot be described as data within a COBOL program. A data item described as USAGE IS INDEX is somewhat analagous to COMPUTATIONAL items, in that the data format of such items is subject to implementor definition. These items should be described within File Section Record Description entries only with extreme caution since data compatibility may be impaired.

When a reference is made to a table element, or to an item within a table element, and the name of the item is followed by its related index-name or names in parentheses, then each occurrence number required to complete the reference will be obtained from the respective index-name. The index-name thus acts as a subscript whose value is used in any table reference that specifies indexing.

When a reference requires more than one occurrence number for completeness, the programmer must not use a data-name subscript to indicate one occurrence number and an index-name for another. Therefore, if indexing is to be used, each OCCURS clause within the hierarchy (each dimension of the table) must contain an INDEXED BY clause. The programmer may, however, mix literals and index-names within one reference. The number of literals and index-names in such a reference must equal the number of dimensions in the table whose element is being referenced.

At the time of execution of a statement which references an indexed table element, the value of the index-name associated with the table element must not correspond to a value less than one (1) nor to a value greater than the highest permissible subscript value for the table element.

The use of subscripting in a reference to a table element, or to an item within a table element, will not cause the alteration of any index-names associated with that table.

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 the table element is followed by an index in the form (index-name + integer-1), the occurrence number required to complete the reference will be the same as if integer-1 were added to the occurrence number to which the current setting at object time of the index-name corresponds. Similarly, when the form (index-name - integer-2) is used, the occurrence number obtained will be the same as if

Withdrawn/Retired

integer-2 were subtracted from the occurrence number to which the current setting of index-name corresponds.

It should be noted that the use of relative indexing will not cause the object program to alter the value of the index-name.

A reference to an item must not be indexed by an index-name that is not associated (through the use of the INDEXED BY clause) with the table of which that item is an element.

Data that has been arranged in the form of a table is very often searched. In COBOL the SEARCH statement provides facilities, through its two options, for producing serial and nonserial (for example, binary) searches. In using the SEARCH statement the programmer may vary an associated index-name or an associated data-name. This statement also provides facilities for execution of imperative statements when certain conditions are true and includes an AT END phrase.

Withdrawn/Retirée

2.4 RERUN

The RERUN feature of COBOL provides a facility for check restart. That is, executing a RERUN takes a snapshot of the program status and stores the information. It is then possible to restart the program from the point of the most recent RERUN. The use of the RERUN clause protects the user from having to start a program over from the beginning in the event of a hardware failure while the job is running.

There are two basic parts to the RERUN clause. The user must designate a medium to receive the data and a criterion from which the frequency of checkpoints may be determined. The receiving medium may be specified by designating a file name or a separate hardware device. The determination of frequency of the dump may be made on the basis of a number of records of a particular file having been processed, of the end of a reel of a particular file having been reached, of the setting of a hardware switch or of a specified number of units of an internal clock having been counted.

Withdrawn/Retirée

2.5 SHARED MEMORY AREA

This feature is basically 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.

When the RECORD option of the SAME clause is used, only 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. In implementations that include the RECORD option, 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 the output 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.

When the SAME clause is used without the RECORD option not only the file areas but the alternate areas as well are shared.

As a result, only one of the files sharing the same set of areas is permitted to be active at one time. This form of the clause is designed for the application in which a series of files are used during different phases of the object program. In these cases, the SAME clause allows the programmer to save memory space.

Withdrawn/Retired

2.6 PROGRAM COMMUNICATION

Complex data processing problems are frequently solved by the use of separately compiled but logically coordinated programs, which, at execution time, form logical and physical subdivisions of a single run unit. This approach lends itself to dividing a large problem into smaller, more manageable segments, which can be programmed and debugged independently. At execute time, control is transferred from program to program by the use of CALL and EXIT PROGRAM statements. (See Paragraph 2.6.1, Inter-Program Communication.)

Under certain circumstances, e.g., a shortage of computer storage, it is desirable to subdivide a single program into physical segments, so that, at execute time, it is not necessary to load the entire program into computer storage at one time. This approach would permit the overlaying of some segments, with a corresponding saving in total computer storage required to execute the program. This facility is called Segmentation.

There are no special statements in COBOL for communication between segments of such a program. There are, however, some special clauses used by the COBOL programmer to specify how the object program is to be segmented. (See Paragraph 2.6.2, Segmentation.)

2.6.1 INTER-PROGRAM COMMUNICATION

In COBOL terminology, a program is either a source program or an object program depending on context; a COBOL source program is a syntactically correct set of COBOL statements as specified in Section III of this manual; an object program is the set of instructions, constants, and other machine-oriented data resulting from the operation of a compiler on a source program; the compilation of a source program not written in COBOL also results in an object program; and a run unit is the total machine language necessary to solve a data processing problem. It includes one or more object programs as defined above, and it may include machine language from sources other than a COBOL compiler.

When the statement of a problem is subdivided into more than one program, the constituent programs must be able to communicate with each other. This communication may take two forms: transfer of control and reference to common data.

2.6.1.1 Transfer of Control

The CALL statement provides the means whereby control can be passed from one program to another within a run unit. A program that is activated by a CALL statement may itself contain CALL statements. However, results are unpredictable where circularity of control is initiated; i.e., where program A calls program B, then program B calls program A or another program that calls program A.

When control is passed to a called program, execution proceeds in the normal way from procedure statement to procedure 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 point immediately following the CALL statement in the calling program. Stated briefly, the EXIT PROGRAM statement terminates only the program in which it occurs, and the STOP RUN statement terminates the entire run unit.

Withdrawn/Retired

If either the called or calling program is not written in COBOL, then the calling mechanism, return mechanism and inter-program data communication between the COBOL and non-COBOL object program is implementor defined.

2.6.1.2 Inter-Program Data Storage

Program interaction requires that both programs have access to the same data items. In the calling program the common data items are described along with all other data items in the File Section, Working-Storage Section, Communication Section, or Linkage Section. At object time memory is allocated for the entire Data Division. In the called program, common data items are described in the Linkage Section. At object time memory space is not allocated for this section. Communication between the called program and the common data items stored in the calling program is effected through USING clauses contained in both programs. The USING clause in the calling program is contained in the CALL statement and the operands are a list of common data-identifiers described in its Data Division. The USING clause in the called program follows the Procedure Division header and the operands are a list of common data identifiers described in its Linkage Section. The identifiers specified by the USING clause of the CALL statement indicate those data items available to a calling program that may be referred to in the called program. The sequence of appearance of the identifiers in the USING clause of the CALL statement and the USING clause in the Procedure Division header is significant. Corresponding identifiers reference a single data item which is available to the called and calling program. The correspondence is positional and not by name. During execution of the called program, every Procedure Division reference to a shared data item is resolved by equating the reference in the called program to the data item defined in the calling program.

A called program which shares a data item with the program to which it is subordinate can, in turn, share this data item with a program subordinate to itself.

Once control leaves a called program its state is maintained until a CANCEL is executed naming that program. Therefore, initialization of the program in case of repetitive calls is not necessary.

Execution of the CANCEL statement allows the user to indicate that the memory areas occupied by the called program(s) may be released. In addition, the CANCEL guarantees that the program cancelled will be in its initial state when called by a subsequent CALL statement.

2.6.2 SEGMENTATION

The Segmentation facility permits the user to subdivide physically the Procedure Division of a COBOL object program. All source paragraphs which contain the same segment-number in their paragraph headers will be considered at object time to be one segment. Since segment-numbers can range from 00 through 49, it is possible to subdivide any object program into a maximum of 50 segments.

Program segments may be of two types: non-overlayable and overlayable.

Non-overlayable segments are in computer storage during the execution of the entire program, i.e., they cannot be overlayed except when the system is executing another program, in which case any segment may be 'rolled out' temporarily.

Withdrawn/Retired

Overlayable segments may be overlayed during program execution, but any such overlaying is transparent to the user, i.e., they are logically identical to non-overlayable segments, but physically different from them.

2.6.2.1 Segment Limit

Unless the COBOL user specifies otherwise, all segments are non-overlayable. If, however, the user needs overlayable segments, they are numbered from a user-specified value (00 through 49) through 49. In the SEGMENT-LIMIT clause of the Environment Division the user specifies the lowest numbered segment which is to be overlayable. Therefore, the more overlayable segments there are, the fewer non-overlayable segments there can be.

The logical relationship between all segments is always the same, regardless of the Segment Limit. Intervening overlays of a segment will not result in initialization of that segment. Therefore, COBOL paragraphs can be written as if all such paragraphs were always fixed in computer storage, and subsequent changes in the Segment Limit will have no effect on program logic.

2.6.2.2 Transfer of Control

There are three methods of transferring control within a program:

1. A GO TO statement.
2. A PERFORM statement.
3. An input procedure associated with a SORT statement, or an output procedure associated with either a SORT or a MERGE statement.

An input/output procedure can be considered an implicit PERFORM which is executed in conjunction with a SORT or MERGE statement, and, for that reason, the restrictions on the PERFORM statement apply equally to input/output procedures.

The CALL statement transfers control from a calling program to a called program, and is not, therefore, relevant to a discussion of segmented programs.

The Segmentation facility imposes no restrictions on transfer of control. All segments are logically identical and can be treated by the user as though the program were not segmented.

Withdrawn/Retiree

2.7 COMMUNICATIONS FACILITY

The communications facility provides the ability to access, process, and create messages or portions thereof. It provides the ability to communicate through a Message Control System with local and remote communication devices.

2.7.1 THE MESSAGE CONTROL SYSTEM

The implementation of the communications facility requires that a Message Control System (MCS) be present in the COBOL object program's operating environment.

The MCS is the logical interface to the operating system under which the COBOL object program operates. The primary functions of the MCS are the following:

1. To act as an interface between the COBOL object program and the network of communications devices, in much the same manner as an operating system acts as an interface between the COBOL object program and such devices as card readers, magnetic tape and mass storage devices, and printers.

2. To perform line discipline, including such tasks as dial-up, polling, and synchronization.

3. To perform device-dependent tasks, such as character translation and insertion of control characters, so that the COBOL user can create device-independent programs.

The first function, that of interfacing the COBOL object program with the communications devices, is the most obvious to the COBOL user. In fact, the COBOL user may be totally unaware that the other two functions exist. Messages from communications devices are placed in input queues by the MCS while awaiting disposition by the COBOL object program. Output messages from the COBOL object program are placed in output queues by the MCS while awaiting transmission to communications devices. The structures, formats, and symbolic names of the queues are defined by the user to the MCS at some time prior to the execution of the COBOL object program. Symbolic names for message sources and destinations are also defined at that time. The COBOL user must specify in his COBOL program symbolic names which are known to the MCS.

During execution of a COBOL object program, the MCS performs all necessary actions to update the various queues as required.

2.7.2 THE COBOL OBJECT PROGRAM

The COBOL object program interfaces with the MCS when it is necessary to send data, receive data, or to interrogate the status of the various queues which are created and maintained by the MCS. In addition, the COBOL object program may direct the MCS to establish or break the logical connection between the communications device and a specified portion of the MCS queue structure. The method of handling the physical connection is a function of the MCS.

Withdrawn/Retirée

2.7.3 RELATIONSHIP OF THE COBOL PROGRAM TO THE MESSAGE CONTROL SYSTEM AND COMMUNICATION DEVICES

The interfaces which exist in a COBOL communications environment are established by the use of a CD and associated clauses in the Communication Section of the Data Division. There are two such interfaces:

1. The interface between the COBOL object program and the MCS, and;

2. The interface between the MCS and the communications devices.

The COBOL source program uses three statements to control the interface with the MCS:

1. The RECEIVE statement, which causes data in a queue to be passed to the COBOL object program,

2. The SEND statement, which causes data associated with the COBOL object program to be passed to one or more queues, and;

3. The ACCEPT statement with the COUNT phrase, which causes the MCS to indicate to the COBOL object program the number of complete messages in the specified queue structure.

The COBOL source program uses two statements to control the interface between the MCS and the communications devices:

1. The ENABLE statement, which establishes logical connection between the MCS and one or more given communications devices, and;

2. The DISABLE statement, which breaks a logical connection between the MCS and one or more given communications devices.

These relationships are shown in Figure 2.7-1, COBOL Communications Environment, and are additionally explained in the appropriate paragraph. (See 2.7.5.2, Enabling and Disabling Queues.)

2.7.3.1 Invoking the COBOL Object Program

There are two methods of invoking a COBOL communications object program:

1. Scheduled initiation

2. MCS invocation

Regardless of the method of invocation, the only operating difference between the two methods is that MCS invocation causes the areas referenced by the symbolic queue and sub-queue names in the specified CD to be filled.

Withdrawn/Retirée

2.7.3.1.1 Scheduled Initiation of the COBOL Object Program

A COBOL object program using the communications facility may be scheduled for execution through the normal means available in the program's operating environment, such as job control language. In that case, the COBOL program can use three methods to determine what messages, if any, are available in the input queues:

1. The ACCEPT statement with the COUNT phrase,

2. The RECEIVE statement with a NO DATA phrase, and

3. The RECEIVE statement without a NO DATA phrase (in which case a program wait is implied if no data is available).

2.7.3.1.2 Invocation of the COBOL Object Program by the MCS

It is sometimes desirable to schedule a COBOL object communications program only when there is work available for it to do. Such scheduling occurs if the MCS determines what COBOL object program is required to process the available message and subsequently causes that program to be scheduled for execution. Prior to the execution of the COBOL object program, the MCS places symbolic queue and sub-queue names in the data items of the CD that specifies the FOR INITIAL INPUT clause.

A subsequent RECEIVE statement directed to that CD will result in the available message being passed to the COBOL object program.

Withdrawn/Retirée

COBOL COMMUNICATIONS ENVIRONMENT

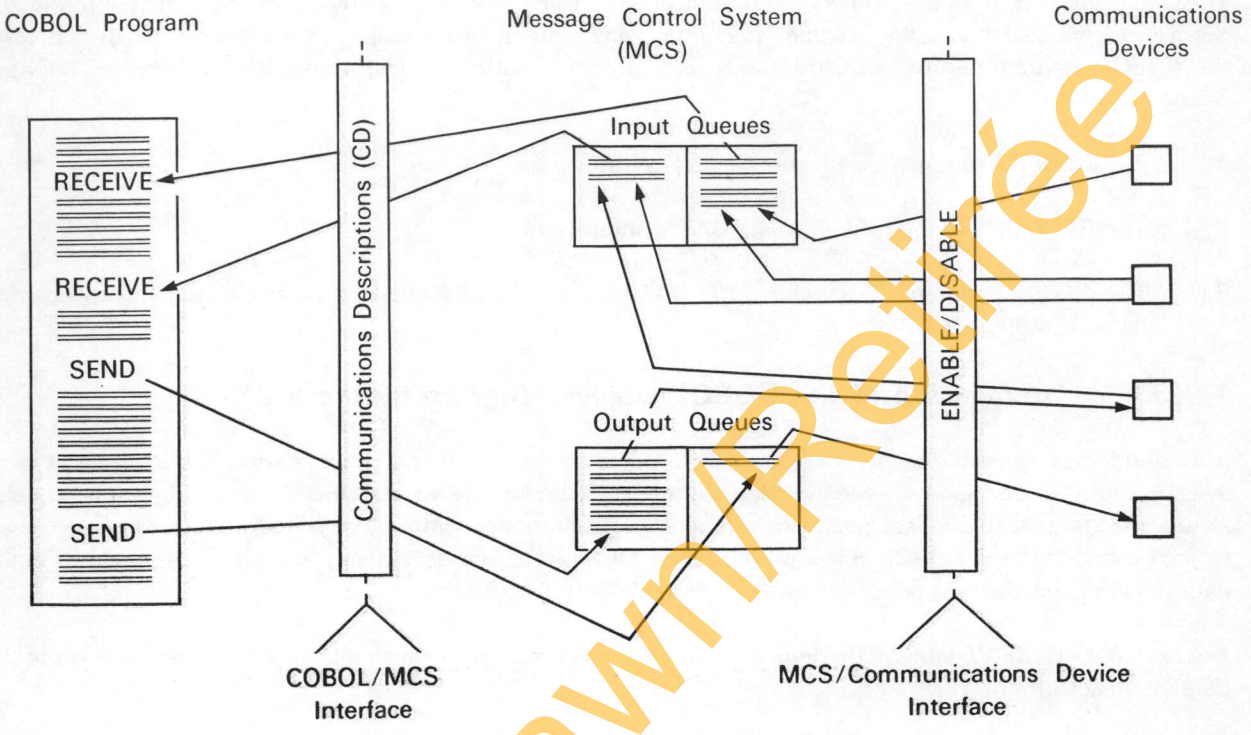


Figure 2.7-1 COBOL Communications Environment

2.7.3.1.3 Determining the Method of Scheduling

A COBOL source program can be written so that its object program can operate with either of the above two modes of scheduling. In order to determine which method was used to load the COBOL object program, the following is one technique that may be used:

1. One CD must contain a FOR INITIAL INPUT clause.

2. The Procedure Division may contain statements to test the initial value of the symbolic queue name in that CD. If it is space-filled, job control statements were used to schedule the COBOL object program. If not space-filled, the MCS has invoked the COBOL object program and replaced the spaces with the symbolic name of the queue containing the message to be processed.

Withdrawn/Retirée

2.7.4 THE CONCEPT OF MESSAGES AND MESSAGE SEGMENTS

A message consists of some arbitrary amount of information, usually character data, whose beginning and end are defined or implied. As such, messages comprise the fundamental but not necessarily the most elementary unit of data to be processed in a COBOL communications environment.

Messages may be logically subdivided into smaller units of data called message segments which are delimited within a message by means of End of Segment Indicators (ESI). A message consisting of one or more segments is delimited from the next message by means of an End of Message Indicator (EMI). In a similar manner, a group of several messages may be logically separated from succeeding messages by means of an End of Group Indicator (EGI). When a message or message segment is received by the COBOL program, a communication description interface area is updated by the MCS to indicate which, if any, delimiter was associated with the text transferred during the execution of that RECEIVE statement. On output the delimiter, if any, to be associated with the text released to the MCS during execution of a SEND statement is specified or referenced in the SEND statement. Thus the presence of these logical indicators is recognized and specified both by the MCS and by the COBOL object program; however, no indicators are included in the message text processed by COBOL programs.

A precedence relationship exists between the indicators EGI, EMI and ESI. EGI is the most inclusive indicator and ESI is the least inclusive indicator. The existence of an indicator associated with message text implies the association of all less inclusive indicators with that text. For example, the existence of the EGI implies the existence of EMI and ESI.

2.7.5 THE CONCEPT OF QUEUES

Queues consist of one or more messages from or to one or more communications devices, and as such, form the data buffers between the COBOL object program and the MCS. Input queues are logically separate from output queues.

The MCS logically places in queues or removes from queues only complete messages. Portions of messages are not logically placed in queues until the entire message is available to the MCS. That is, the MCS will not pass a message segment to a COBOL object program unless all segments of that message are in the input queue, even though the COBOL source program uses the SEGMENT phrase of the RECEIVE statement. For output messages, the MCS will not transmit any segment of a message until all its segments are in the output queue. Interrogation of the queue depth, or number of messages that exist in a given queue, reflects only the number of complete messages that exist in the queue.

The process by which messages are placed into a queue is called enqueueing. The process by which messages are removed from a queue is called dequeueing.

2.7.5.1 Independent Enqueueing and Dequeueing

It is possible that a message may be received by the MCS from a communications device prior to the execution of the COBOL object program. As a result, the MCS enqueues the message in the proper input queue (provided that input queue is enabled) until the COBOL object program requests dequeueing with the

Withdrawn/Retirée

RECEIVE statement. It is also possible that a COBOL object program will cause the enqueueing of messages in an output queue which are not transmitted to a communications device until after the COBOL object program has terminated. Two common reasons for this occurrence are:

1. When the output queue is disabled.

2. When the COBOL object program creates output messages at a speed faster than the destination can receive them.

2.7.5.2 Enabling and Disabling Queues

Usually, the MCS will enable and disable queues based on time of day, message activity, or other factors unrelated to the COBOL program. However, the COBOL program has the ability to enable and disable queues itself through use of the ENABLE and DISABLE statements.

A key is required in both statements in order to prevent indiscriminate use of the facility by a COBOL user who is not aware of the total network environment, and who may therefore disrupt system functions by the untimely enabling and/or disabling of a queue. However, this action never interrupts a transmission. (See 7.15, The DISABLE Statement.)

2.7.5.3 Enqueueing and Dequeueing Methods

In systems that allow the user to specify certain MCS functions, it may be necessary that the user specify to the MCS, prior to execution of programs which reference these facilities, the selection algorithm and other designated MCS functions to be used by the MCS in placing messages in the various queues. A typical selection algorithm, for example, would specify that all messages from a given source be placed in a given input queue, or that all messages to be sent to a given destination be placed in a given output queue.

Dequeueing is often done on a first in, first out basis. Thus, messages dequeued from either an input or output queue are those messages which have been in the queue for the longest period of time. However, the MCS can, upon prior specification by the user, dequeue on some other basis, i.e., priority queueing can be employed.

2.7.5.4 Queue Hierarchy

In order to control more explicitly the messages being enqueued and dequeued, it is possible to define in the MCS a hierarchy of input queues, i.e., queues comprising queues. In COBOL, four levels of queues are available to the user. In order of decreasing significance, the queue levels are named queue, sub-queue-1, sub-queue-2, and sub-queue-3. The full queue structure is depicted in Figure 2.7-2, Hierarchy of Queues, where queues and sub-queues have been named with the letters A through O. Messages have been named with a letter according to their source (X, Y, or Z) and with a sequential number.

Withdrawn/Retirée

Let us assume that the MCS is operating under the following queueing algorithm:

1. Messages are placed in queues according to the contents of some specified data field in each message.

2. With the RECEIVE statement, if the user does not specify a given sub-queue level, the MCS will choose the sub-queue from that level in alphabetical order, e.g., if sub-queue-1 is not specified by the user, the MCS will dequeue from sub-queue-1 B.

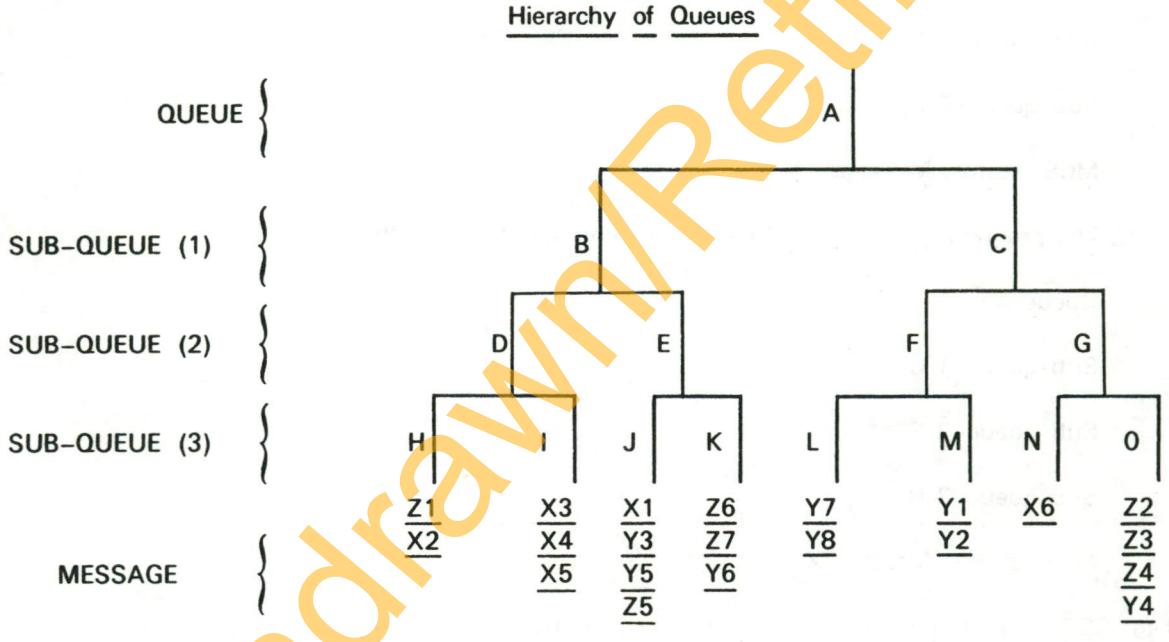


Figure 2.7-2 Hierarchy of Queues

The following examples illustrate the effect of the above algorithms (See Figure 2.7-2, Hierarchy of Queues):

1. The program executes a RECEIVE statement, specifying via the CD:

   Queue A

   MCS returns: Message Z1

Withdrawn/Retiree

2. The program executes a RECEIVE statement, specifying via the CD:

   Queue A

   Sub-queue-1 C

   MCS returns: Message Y7

3. The program executes a RECEIVE statement, specifying via the CD:

   Queue A

   Sub-queue-1 B

   Sub-queue-2 E

   MCS returns: Message X1

4. The program executes a RECEIVE statement, specifying via the CD:

   Queue A

   Sub-queue-1 C

   Sub-queue-2 G

   Sub-queue-3 N

   MCS returns: Message X6

If the COBOL programmer wishes to access the next message in a queue, regardless of which sub-queue that message may be in, he specifies the queue name only. The MCS, when supplying the message, will return to the COBOL object program, any applicable sub-queue names via the data items in the associated CD. If, however, he desires the next message in a given sub-queue, he must specify both the queue name and any applicable sub-queue names.

For output, the COBOL user specifies only the destination(s) of the message, and the MCS places the message in the proper output queue structure.

There is no one-to-one relationship between a communications device and a source/destination. A source or destination may consist of one or more physical devices. The device or devices which comprise a source/destination are defined to the MCS.

Withdrawn/Retirée

2.8 DATA BASE FACILITY

The concept of data control and access has evolved from the earlier notion that each program creates and manipulates its own data independent of other data that exists in the computer system. The concept of a data base, shared among several programs simultaneously, places additional requirements on the computer system which has access to that data base:

1. Data therein must be accessible in a form and order which is best suited to each individual program.

2. Data therein must be simultaneously accessible to all executing run units.

3. Data integrity must be protected from untoward results that could occur during simultaneous access by different run units.

4. Data must be protected from unauthorized access.

The COBOL Data Base Facility provides access and control of a data base in accordance with the above requirements.

2.8.1 DATA BASE ENVIRONMENT

The environment in which a COBOL program accesses a data base consists of at least the following:

1. The data base itself, consisting of all the data that is to be accessed,

2. A Data Base Control System (DBCS) that provides the interface to the data base,

3. A Schema, which describes the structure of the data base,

4. A Sub-schema, which describes that portion of the data base that is of interest to the COBOL program and describes the form in which the COBOL program will use that data, and

5. A COBOL program containing statements and clauses pertaining to the data base.

2.8.1.1 Data Base Control System (DBCS)

The DBCS is that collection of functions which, in conjunction with the operating system, accesses the data base.

2.8.1.2 Schema

The description of the entire data base is known as the Schema. The Schema is written in the language defined in the CODASYL Data Description Language Journal of Development.

Withdrawn/Retirée

2.8.1.3 Sub-schema

The description of that portion of the data base that a given COBOL program is to access is known as the Sub-schema. The Sub-schema is written in the COBOL Sub-schema Language. (See Section IV.)

2.8.1.4 COBOL Program

A COBOL program which accesses the data base must specify that a given Sub-schema is to be used for that access, and must specify data base manipulation statements to cause such access to take place. (See Section III.)

2.8.2 RELATIONSHIPS AMONG SCHEMA, SUB-SCHEMAS, AND COBOL PROGRAMS

The data base has only one Schema, but it can have as many Sub-schemas as necessary. Each Sub-schema describes a portion of the data base as it is to be accessed by a COBOL program. Sub-schema descriptions can overlap; that is, the same data can be described in more than one Sub-schema.

Many characteristics of the data in the data base are fixed by the Schema and cannot be changed by the Sub-schema; that is, a COBOL program accessing the data must accept that data with its characteristics as defined by the Schema. Other characteristics, however, can be specified in the Sub-schema to be different from those described in the Schema. For example, the usage and radix of a data item can be described in a Sub-schema differently from the way they are described in the Schema. The Schema describes the data as it exists in the data base, the Sub-schema describes the data as it is known to the COBOL program. The DBCS performs any necessary transformations of data during retrieval and storage.

The COBOL Sub-schemas are stored in the COBOL Sub-schema Library. A COBOL program uses a given Sub-schema by specifying its name in the Sub-schema Section of the Data Division. A COBOL program can specify only one Sub-schema.

A Sub-schema is associated with the given Schema whose name is referenced in the Sub-schema Description entry of the Title Division. A Sub-schema can be associated with only one Schema. Therefore, in the data base environment, there is a unique triplet of a Schema, a Sub-schema, and a COBOL program which, when bound together, performs the user specified program functions.

The compilation of the Schema, the Sub-schema, and the COBOL program may each occur at different times. In addition, the binding of these three may also occur at different times.

The following specifies some possible times when binding may take place. However, the control and availability of these options are implementor-defined.

1. COBOL compile time.

Withdrawn/Retirée

2. Sub-schema compile time.

3. Execution time.

4. A mixture of 1, 2, 3, above.

5. Any other implementor-defined times.

2.8.3 RECORDS IN THE DATA BASE

A record in the data base is conceptually like a record in a file, in that it is a collection of data items described by a level-number 01 Data Description entry. However, the data base record is significantly different from a record in a file in a number of ways.

A data base record is not necessarily a physically contiguous collection of data items. In fact, data items that make up the record need not be stored in the data base at all, if they are described as virtual derived data items. Regardless of the method of storage in the data base, however, the DBCS assembles the record as described in the Sub-schema and makes it available to the COBOL program, or stores the record generated by the COBOL program.

Each Record Description entry in the Sub-schema describes one record type. There may be an arbitrary number of records of that record type stored in the data base.

2.8.4 DATA BASE KEYS

A data base key is a conceptual entity whose value uniquely identifies a data base record. A data base key is not part of any data base record, but its value may be accessed by a COBOL program.

When a record is initially stored in the data base, the DBCS assigns a unique data base key value to the record. The value thus assigned remains invariable for that record until the record is removed from the data base. After that time, the DBCS can reassign that data base key value to a new record.

2.8.5 REALMS

A realm is a named logical subdivision of the data base. It is that portion of the data base that contains all records having specified data base key values. Realms are defined in the Schema.

A realm is somewhat similar to a COBOL file; it is the object of the READY statement, which is the data base counterpart of the OPEN statement.

2.8.6 SETS

The set is the basic structural unit of the data base. A set type is a named collection of sets, and is defined in the Schema. One record type is defined to be the owner of the set type, and one or more other record types are defined to be the members of the set type. In the data base, each owner record may have member records

Withdrawn/Retirée

associated with it. Each of these member records, in turn, can be defined to be the owner of other sets of different types. In the data base a set, consisting of an owner record and its member records, is identified by identifying the owner record.

Each set type has an order specified for its member records, to permit the DBCS to control the sequence of retrieval of the records and to control the storage of new member records into the sets of that set type. Each set type also has associated with it set selection criteria to aid the DBCS in selecting and identifying a specific owner record and thus a specific set of that set type.

2.8.6.1 Set Ordering

The set ordering criteria defined in the Schema for each set type determine the logical ordering of the member records in each set of that set type. The physical order of the member records in the data base is independent of the logical ordering specified by the set ordering criteria. In addition, the set ordering criteria for each set type can be different, so that a record that is a member of two different set types could be subject to retrieval in two entirely different orders. The set ordering criteria of a set type can be any of the following:

1. In an order based upon the values of data items appearing in the records.

2. In the order of storage of records, either last in-first out or first in-first out.

3. Procedurally selected, either immediately prior to or after a record selected by the COBOL program.

4. Any other criteria as specified in the Schema.

2.8.6.2 Set Membership

There are two classes of set membership that a record type can have, and two types of set membership within each of the two classes.

2.8.6.2.1 Storage Class of Membership

A record type's storage class of membership in a given set type determines the conditions under which records of that record type become members of sets of that set type. A record's storage class of membership can be either automatic or manual. If it is automatic, that record becomes a member of that set when that record is initially stored in the data base. If manual, that record becomes a member of that set only if it is stored in the data base and subsequently placed into set membership by the execution of a CONNECT statement.

2.8.6.2.2 Removal Class of Membership

A record type's removal class of membership in a given set type determines whether or not a record of that record type can be removed from membership in a set of that set type once a member. A record's removal class of membership is independent of its storage class of membership. A record's removal class of membership can be either permanent or transient. If it is permanent, that record cannot be removed from

Withdrawn/Retired

membership in that set; if transient, that record can be removed from membership in that set by execution of a DISCONNECT or MODIFY statement.

2.8.7 RECORD PLACEMENT AND SELECTION

When a record is initially stored in the data base, its logical placement is determined by several means. When the record is subsequently retrieved, it is selected and identified for retrieval by several means.

2.8.7.1 Currency Indicators

Currency indicators are conceptual entities that are maintained by the DBCS to keep track of record retrieval and storage. There is one currency indicator for each run unit, for each realm available to that run unit, for each set type available to that run unit, and for each record type available to that run unit. Unless the action is specifically suppressed, the DBCS automatically updates its currency indicators when a record of a given type is retrieved from a set of a given type from a given realm. The contents of a currency indicator consist of a data base key value that identifies the record.

2.8.7.2 Location Modes

Each record type has associated with it a location mode that is used by the DBCS in determining how to assign a data base key value to each record of that type when it is initially stored in the data base. The location mode of a record type can specify that the DBCS is to assign a specific data base key value, or a data base key value based on the value of data items in the record, or a data base key value that is closely related to the data base key values of members of a given set. Location modes are defined in the Schema.

2.8.7.3 Set Selection

Each set type has associated with it set selection criteria, as defined in the Schema. The set selection criteria specifies the method to be used by the DBCS to select and identify a specific owner record of a set of that set type, and thus select and identify the member records of that set. The Sub-schema may specify alternate criteria.

2.8.7.4 Record Selection Expressions

Record selection expressions are specified in the FIND statement of a COBOL program for the purpose of selecting and identifying a specific record in the data base for subsequent use, such as retrieval. Record selection expressions can specify that a record is to be retrieved based on its data base key value, or based on values of data items contained in the record, or based on its position in the data base relative to other records in a realm or set. If the record selection expression specifies that the record is to be selected based on its relative position in a realm, the first, last, prior, or nth record in the realm can be selected. If the record selection expression specifies that the record is to be selected based on its relative position in a set, the record can be selected based on the set ordering criteria for the set, or the first, last, nth, or owner record can be selected.

Withdrawn/Retiree

2.8.8 SECURITY

In both the Schema and the Sub-schema, privacy locks can be specified to be associated with the use of the Schema and Sub-schema themselves as well as the realms, set types, record types, or data items. In addition, privacy locks can be associated with specific statements. By associating privacy locks with the data itself and with the statements used to access the data, a security system is established to prevent unauthorized access to the data base. Privacy locks specified in the Sub-schema for a given resource take precedence over privacy locks specified in the Schema for the same resource. This enables control over the use of the Sub-schema even though the Schema privacy locks may be known to unauthorized processes.

2.8.9 INTEGRITY

To protect the accuracy and completeness of the data in the data base from the results of undesirable interaction with concurrently executing run units, the facility to monitor and lock records is provided. Each run unit can specify that a given record be placed in a monitored mode such that, if any other run unit affects the record or its set relationships, the run unit is notified. Thus, during the time that the run unit has retrieved a record and is in the process of altering it or its set relationships, it can ensure that no covert action takes place that would jeopardize the process and affect the integrity of the data base. Records are locked and made unavailable to concurrent run units only during the time that the DBCS is altering them or their record relationships.

A run unit can specify that a given realm is to be placed under its exclusive control such that no record stored in that realm is made available to any other run unit until the exclusive control has been relinquished. A run unit can also specify that records stored in a given realm can be altered by that run unit only, but that other run units can access records stored in that realm for the purpose of retrieving them only.

2.8.10 DATA BASE PROCEDURES

Some of the functions of the DBCS can be controlled procedurally during the execution of an object program, in addition to the more common declarative control. This procedural control is achieved through the use of data base procedures.

A data base procedure is a routine, referenced by a routine-name, to which control is transferred during the execution of an object program, as defined in the Schema or in the Sub-schema.

Withdrawn/Retirée

SECTION III: COBOL

LANGUAGE SPECIFICATIONS

Withdrawn/Retirée

TABLE OF CONTENTS

SECTION III: COBOL LANGUAGE SPECIFICATIONS

Page

CHAPTER 1. INTRODUCTION

1.1 Objectives of Section III III-1-1
1.2 History of COBOL Specification Documents III-1-1
1.3 Organization of Section III III-1-2
1.4 Notation Used in Formats and Rules III-1-2

CHAPTER 2. GLOSSARY

2.1 Introduction III-2-1
2.2 Definitions III-2-1

CHAPTER 3. LANGUAGE CONCEPTS

3.1 Character Set III-3-1
3.2 Language Structure III-3-2
3.2.1 Separators III-3-2
3.2.2 Character-Strings III-3-3
3.2.2.1 COBOL Words III-3-3
3.2.2.2 Literals III-3-8
3.2.2.3 PICTURE Character-Strings III-3-11
3.2.2.4 Comment-Entries III-3-11
3.3 Concept of Computer-Independent Data Description III-3-12
3.3.1 Logical Record Concept III-3-12
3.3.2 Concept of Levels III-3-13
3.3.3 Concept of Classes of Data III-3-14
3.3.4 Selection of Character Representation and Radix III-3-15
3.3.5 Algebraic Signs III-3-16
3.3.6 Standard Alignment Rules III-3-16
3.3.7 Item Alignment for Increased Object-Code Efficiency III-3-17
3.3.8 Uniqueness of Reference III-3-18
3.3.9 Data Base Keys III-3-25
3.4 Explicit and Implicit Specifications III-3-27
3.4.1 Explicit and Implicit Procedure Division References III-3-27
3.4.2 Explicit and Implicit Transfers of Control III-3-27

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

		Page
3.4.3	Explicit and Implicit Attributes	III-3-28
3.5	External Switch	III-3-29

CHAPTER 4. IDENTIFICATION DIVISION

4.1	General Description	III-4-1
4.2	Organization	III-4-1
4.2.1	Structure	III-4-1
4.3	The PROGRAM-ID Paragraph	III-4-2
4.4	The DATE-COMPILED Paragraph	III-4-3

CHAPTER 5. ENVIRONMENT DIVISION

5.1	General Description	III-5-1
5.2	Organization	III-5-1
5.2.1	Structure	III-5-1
5.3	Configuration Section	III-5-3
5.3.1	The SOURCE-COMPUTER Paragraph	III-5-3
5.3.2	The OBJECT-COMPUTER Paragraph	III-5-5
5.3.3	The SPECIAL-NAMES Paragraph	III-5-7
5.4	Input-Output Section	III-5-14
5.4.1	The FILE-CONTROL Paragraph	III-5-14
5.4.2	The I-O-CONTROL Paragraph	III-5-18

CHAPTER 6. THE DATA DIVISION

6.1	General Description	III-6-1
6.2	Sub-schema Section	III-6-5
6.3	File Section	III-6-6
6.4	Working-Storage Section	III-6-7
6.5	Linkage Section	III-6-8
6.6	Communication Section	III-6-9
6.7	Report Section	III-6-10
6.8	Record Description-Structure	III-6-11
6.9	The Sub-schema Entry — Complete Entry Skeleton	III-6-12
6.10	The File Description — Complete Entry Skeleton	III-6-13
6.11	The Sort-Merge File Description — Complete Entry Skeleton	III-6-17

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

		Page
6.12	The Communication Description — Complete Entry Skeleton	III-6-18
6.13	The Report Description — Complete Entry Skeleton	III-6-28
6.14	The Data Description — Complete Entry Skeleton	III-6-30
6.15	The Report Group Description — Complete Entry Skeleton	III-6-34
6.16	The ACCESS MODE Clause	III-6-55
6.17	The ALTERNATE RECORD KEY Clause	III-6-57
6.18	The BLANK WHEN ZERO Clause	III-6-58
6.19	The CHECK Clause	III-6-59
6.20	The CODE Clause	III-6-61
6.21	The COLUMN NUMBER Clause	III-6-62
6.22	The CONTROL Clause	III-6-63
6.23	The Data-Name or FILLER Clause	III-6-65
6.24	The DATA RECORDS Clause	III-6-67
6.25	The FILE STATUS Clause	III-6-68
6.26	The GROUP INDICATE Clause	III-6-69
6.27	The JUSTIFIED Clause	III-6-70
6.28	The LABEL RECORDS Clause	III-6-71
6.29	Level-Number	III-6-72
6.30	The LINAGE Clause	III-6-74
6.31	The LINE NUMBER Clause	III-6-77
6.32	The NEXT GROUP Clause	III-6-79
6.33	The OCCURS Clause	III-6-81
6.34	The PAGE Clause	III-6-85
6.35	The PICTURE Clause	III-6-89
6.36	The PRIVACY LOCK Clause	III-6-100
6.37	The RECORD Clause	III-6-101
6.38	The RECORD KEY Clause	III-6-104
6.39	The REDEFINES Clause	III-6-105
6.40	The RENAMES Clause	III-6-107
6.41	The REPORT Clause	III-6-109
6.42	The SIGN Clause	III-6-110
6.43	The SOURCE Clause	III-6-112
6.44	The SUM Clause	III-6-113
6.45	The SYNCHRONIZED Clause	III-6-116
6.46	The TYPE Clause	III-6-118
6.47	The USAGE Clause	III-6-123
6.48	The VALUE Clause	III-6-126
6.49	The VALUE OF Clause	III-6-129

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

Page

CHAPTER 7. THE PROCEDURE DIVISION

7.1	General Description	III-7-1
7.1.1	Declaratives	III-7-1
7.1.2	Procedures	III-7-1
7.1.3	Execution	III-7-1
7.1.4	Procedure Division Structure	III-7-2
7.2	Statements and Sentences	III-7-4
7.2.1	Conditional Statements and Conditional Sentences	III-7-4
7.2.2	Compiler Directing Statements and Compiler Directing Sentences	III-7-5
7.2.3	Imperative Statements and Imperative Sentences	III-7-5
7.3	Arithmetic Expressions	III-7-7
7.3.1	Definition of an Arithmetic Expression	III-7-7
7.3.2	Arithmetic Operators	III-7-7
7.3.3	Formation and Evaluation Rules	III-7-7
7.4	Boolean Expressions	III-7-10
7.4.1	Definition of a Boolean Expression	III-7-10
7.4.2	Boolean Operators	III-7-10
7.4.3	Boolean Formation and Evaluation Rules	III-7-10
7.5	Conditional Expressions	III-7-13
7.5.1	Simple Conditions	III-7-13
7.5.1.1	Relation Condition	III-7-13
7.5.1.2	Class Condition	III-7-16
7.5.1.3	Condition-Name Condition (Conditional Variable)	III-7-17
7.5.1.4	Switch-Status Condition	III-7-17
7.5.1.5	Sign Condition	III-7-17
7.5.1.6	Tenancy Condition	III-7-18
7.5.1.7	Member Condition	III-7-18
7.5.1.8	Nullity Condition	III-7-18
7.5.2	Complex Conditions	III-7-18
7.5.2.1	Negated Simple Conditions	III-7-19
7.5.2.2	Combined and Negated Combined Conditions	III-7-19
7.5.3	Abbreviated Combined Relation Conditions	III-7-20
7.5.4	Condition Evaluation Rules	III-7-20
7.6	Categories of Statements	III-7-23
7.7	Common Options and Rules for Statement Formats	III-7-26
7.7.1	Intermediate Data Item	III-7-26
7.7.2	The ROUNDED Option	III-7-26
7.7.3	The SIZE ERROR Option	III-7-26

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

		Page
7.7.4	The CORRESPONDING Option	III-7-27
7.7.5	The Arithmetic Statements	III-7-28
7.7.6	Overlapping Operands	III-7-28
7.7.7	Multiple Results in Arithmetic Statements	III-7-28
7.7.8	Incompatible Data	III-7-29
7.7.9	I-O Status	III-7-29
7.7.10	The INVALID KEY Condition	III-7-32
7.7.11	The AT END Condition	III-7-33
7.7.12	The FROM Option	III-7-33
7.7.13	The INTO Option	III-7-33
7.8	The ACCEPT Statement	III-7-35
7.9	The ADD Statement	III-7-38
7.10	The CALL Statement	III-7-40
7.11	The CANCEL Statement	III-7-42
7.12	The CLOSE Statement	III-7-43
7.13	The COMPUTE Statement	III-7-48
7.14	The DELETE Statement	III-7-50
7.15	The DISABLE Statement	III-7-52
7.16	The DISPLAY Statement	III-7-54
7.17	The DIVIDE Statement	III-7-56
7.18	The ENABLE Statement	III-7-59
7.19	The ENTER Statement	III-7-61
7.20	The EXIT Statement	III-7-62
7.21	The GENERATE Statement	III-7-63
7.22	The GO TO Statement	III-7-66
7.23	The IF Statement	III-7-67
7.24	The INITIALIZE Statement	III-7-69
7.25	The INITIATE Statement	III-7-71
7.26	The INSPECT Statement	III-7-72
7.27	The MERGE Statement	III-7-80
7.28	The MOVE Statement	III-7-85
7.29	The MULTIPLY Statement	III-7-89
7.30	The OPEN Statement	III-7-91
7.31	The PERFORM Statement	III-7-97
7.32	The PURGE Statement	III-7-105
7.33	The READ Statement	III-7-106
7.34	The RECEIVE Statement	III-7-110
7.35	The RELEASE Statement	III-7-113
7.36	The RETURN Statement	III-7-114
7.37	The REWRITE Statement	III-7-116
7.38	The SEARCH Statement	III-7-119
7.39	The SEND Statement	III-7-124
7.40	The SET Statement	III-7-129

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

		Page
7.41	The SORT Statement	III–7–134
7.42	The START Statement	III–7–140
7.43	The STOP Statement	III–7–143
7.44	The STRING Statement	III–7–144
7.45	The SUBTRACT Statement	III–7–147
7.46	The SUPPRESS Statement	III–7–149
7.47	The SUSPEND Statement	III–7–150
7.48	The TERMINATE Statement	III–7–152
7.49	The UNSTRING Statement	III–7–153
7.50	The USE Statement	III–7–157
7.51	The WRITE Statement	III–7–162

CHAPTER 8. SEGMENTATION

8.1	General Description	III–8–1
8.1.1	Scope	III–8–1
8.1.2	Organization	III–8–1
8.1.3	Segment Classification	III–8–2
8.1.4	Segmentation Control	III–8–2
8.2	Structure of Program Segments	III–8–3
8.2.1	Segment–Numbers	III–8–3
8.2.2	Segment Limit	III–8–3

CHAPTER 9. THE COBOL LIBRARY FACILITY

9.1	Introduction	III–9–1
9.2	The COPY Statement	III–9–2

CHAPTER 10. REFERENCE FORMAT

10.1	General Description	III–10–1
10.2	Reference Format Representation	III–10–1
10.3	Division, Section, Paragraph Formats	III–10–4
10.4	Data Division Entries	III–10–5
10.5	Declaratives	III–10–5
10.6	Comment Lines	III–10–6

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

Page

CHAPTER 11. THE DEBUGGING FACILITY

11.1	General Description	III-11-1
11.2	Concepts	III-11-1
11.3	A Compile-Time Switch	III-11-1
11.4	An Object-Time Switch	III-11-2
11.5	The USE FOR DEBUGGING Statement	III-11-3
11.6	Debugging Lines	III-11-12

CHAPTER 12. THE COBOL DATA BASE FACILITY — DATA MANIPULATION

12.1	General Description	III-12-1
12.2	Organization	III-12-1
12.3	Data Base Special Registers	III-12-2
12.4	Data Base Status Indicators	III-12-4
12.5	Data Base Conditions	III-12-10
12.6	Record Selection Expressions	III-12-12
12.7	Monitored Mode & Extended Monitored Mode	III-12-18
12.8	Currency Indicators	III-12-19
12.9	Privacy Locks and Privacy Keys	III-12-21
12.10	The ACCEPT Statement	III-12-22
12.11	The CONNECT Statement	III-12-24
12.12	The DISCONNECT Statement	III-12-26
12.13	The ERASE Statement	III-12-27
12.14	The FIND Statement	III-12-29
12.15	The FINISH Statement	III-12-31
12.16	The FREE Statement	III-12-32
12.17	The GET Statement	III-12-33
12.18	The KEEP Statement	III-12-34
12.19	The MODIFY Statement	III-12-35
12.20	The ORDER Statement	III-12-37
12.21	The READY Statement	III-12-40
12.22	The REMONITOR Statement	III-12-42
12.23	The STORE Statement	III-12-44
12.24	The USE Statement	III-12-47

CHAPTER 13. COBOL RESERVED WORDS III-13-1

Withdrawn/Retirée

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVES OF SECTION III

The specifications of a programming language are written for two principal groups: (1) the implementor of the language for a language processor (compiler) in a given hardware environment and, (2) the user of the language who will write source programs that can be translated for operation in a given hardware environment by the language processor. These specifications for a programming language are often two different sets but must reflect the same interpretive result, which is compatible understanding of the programming language definition.

Section III of this manual attempts to meet the needs of both the compiler-writing programmers and the source language programmers in that the rules of the language are described in a source program environment that is very nearly independent of hardware considerations. Where necessary, however, Section III describes the additional rules which must be followed by the user to interface with specific hardware units, or when the user should refer to other software manuals to determine dependent source program and other related entries.

1.2 HISTORY OF COBOL SPECIFICATION DOCUMENTS

Nine official COBOL specification documents have been printed and released by the CODASYL Executive Committee. These are known as:

COBOL — 60, published 1960
COBOL — 61, published 1961
COBOL — 61, Extended, published 1963
COBOL, Edition 1965, published 1965
COBOL, Journal of Development—1968, published 1968
COBOL, Journal of Development—1969, published 1970
COBOL, Journal of Development—1970, published 1971
COBOL, Journal of Development—1973, published 1973
COBOL, Journal of Development—1976, published 1976

The initial three documents principally addressed themselves to specifications for total definition of the language with interspersed paragraphs reflecting historical development, philosophy of the language, examples of the language use, etc. Section III of the later documents is a definition of the complete specifications with by-product paragraphs of philosophy, examples, etc., removed. The effect of Section III is to gather into one place all of the pertinent definitive rules separate from comments surrounding the specifications.

Whereas the COBOL — 60 specification is incompatible in language design with the COBOL — 61 specification, all of the subsequent specifications, including the 1976 COBOL edition, are based on the COBOL

Withdrawn/Retired

— 61 design. These include only extensions, resolutions of ambiguities, deletion of redundancies, or removal of unused or poor language specifications.

1.3 ORGANIZATION OF SECTION III

Section III is constructed to be a stand-alone document within the total presentation of COBOL. This recognizes that many readers of this document are mainly concerned with the COBOL language specifications and need not retain or maintain the other Sections.

Section III contains a specific chapter entitled Glossary. For a complete understanding of terms used within Section III, specific definitions for COBOL usages are considered a basic part of the language definition. Experience has shown the developers, implementors, and users of the language that many incorrect interpretations, ambiguous understandings, and clarification problems resulted from the lack of precise definitions for the COBOL usage of data processing or conventional English terms.

It is suggested that every reader become acquainted with the COBOL terminology expressed in the 'Glossary' to ensure common understanding before attempting to interpret the COBOL specifications in later chapters.

The flowcharts in Section III illustrate the logic of the statement under which they are contained and are not meant to dictate implementation.

1.4 NOTATION USED IN FORMATS AND RULES

1.4.1 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 manual 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, clauses may appear in sequences other than that shown. Applications, requirements or restrictions are shown as rules.

1.4.1.1 Syntax Rule

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.

Withdrawn/Retiree

1.4.1.2 General Rule

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.

1.4.1.3 Elements

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

1.4.1.4 Words

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

Lower-case 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 appendage to the term serves to identify that term for explanation or discussion.

1.4.1.5 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.

1.4.1.6 Brackets, Braces, and Choice Indicators

When a portion of a General Format is enclosed in brackets, [ ], that portion may be included or omitted at the user's choice. Braces, { }, enclosing a portion of a General Format mean selection of one of the options contained within the braces must be made. Choice indicators, ‖ and ‖, enclosing a portion of a General Format mean that a selection of one or more of the options contained within the choice indicators must be made, but the same sequence of words must not be chosen more than once in that entry or statement. In all cases, a choice is indicated by vertically stacking the possibilities. When brackets or braces enclose a portion of a format, but only one possibility is shown, the function of the brackets or braces is to delimit that portion of the format to which a following ellipsis applies. (See 1.4.1.7, The Ellipsis.) If an option within braces contains only reserved words that are not key words, then that option is a default option (implicitly selected unless one of the other options is explicitly indicated).

Withdrawn/Retirée

1.4.1.7 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 '. . .'; continue scanning right to left and determine the logically matching '[' (left bracket) or '{' (left brace) delimiter; the '. . .' applies to the portion of the format between the determined pair of delimiters.

1.4.1.8 Format Punctuation

The separators comma and semicolon are used to improve the readability of the program. Suggested uses are shown in formats, however, their use is optional. In the source program, separators comma, semicolon and space are interchangeable.

If desired, a semicolon or comma may be used between statements in the Procedure Division.

Paragraphs within the Identification and Procedure Divisions and entries within the Environment and Data Division must be terminated by the separator period.

1.4.1.9 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.

Withdrawn/Retired

CHAPTER 2

GLOSSARY

2.1 INTRODUCTION

The terms in this chapter are defined in accordance with their meaning in COBOL, and may not have the same meaning for other languages.

These definitions are also intended as either reference or introductory material to be reviewed prior to reading the detailed language specifications that follow. For this reason, these definitions are, in most instances, brief and do not include detailed syntactical rules. Complete specifications for elements defined in this chapter can be located in other chapters of Section III.

2.2 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.

ABNORMAL TERMINATION
The premature end of execution of a program due to the detection, by the operating system, of a situation that prevents further successful execution of that program.

ACCESS MODE
The manner in which records are to be operated upon within a file.

ACTUAL DECIMAL POINT
The physical representation, using the decimal point characters period (.) or comma (,), of the decimal point position in a data item.

ALIAS DESCRIPTION ENTRY
An entry in the Alias Section of the Sub-schema that defines an alias, to be used in the Sub-schema and COBOL program, for a name defined in Schema.

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 5.3.3, The SPECIAL-NAMES Paragraph.)

Withdrawn/Retirée

ALPHABETIC CHARACTER
A character that belongs to the following set of 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, and the space.

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 of the character(s), 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.

Withdrawn/Retirée

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 for the file or when an optional file is not present.

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.

AUTOMATIC MEMBER
A record that implicitly becomes a member of a given set when the record is initially stored in the data base.

AVAILABLE MODE
The state of a realm or a record in which it is accessible to the DBCS and is not under the exclusive control of a different run unit.

BIT
The smallest unit in a computer's storage structure capable of expressing two distinct alternatives.

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. (See 3.3.1.2, Conceptual Characteristics of a File.) The term is synonymous with Physical Record.

BODY GROUP
Generic name for a report group of TYPE DETAIL, CONTROL HEADING or CONTROL FOOTING.

BOOLEAN CHARACTER
A unit of information that consists of either of the boolean values, zero or one. It may be stored as either a bit or a Standard Data Format character.

BOOLEAN DATA ITEM
A data item consisting entirely of boolean characters.

Withdrawn/Retiree

BOOLEAN EXPRESSION
An identifier of a boolean data item, a boolean literal, such identifiers and/or literals separated by a boolean operator, two boolean expressions separated by a boolean operator, or a boolean expression enclosed in parentheses.

BOOLEAN LITERAL
A literal composed of one or more boolean characters delimited on the left by the separator 'B''' and on the right by the quotation mark separator.

BOOLEAN OPERATOR
A reserved word used in the construction of a boolean expression. The permissible operators and their meanings are:

Boolean Operator	Meaning
AND	Boolean Conjunction
OR	Boolean Inclusive Disjunction
EXOR	Boolean Exclusive Disjunction
NOT	Boolean Negation

BOOLEAN POSITION
The amount of physical storage required to store a single boolean character. This amount depends on the usage specified in the Data Description entry that defines the data item. If the Data Description entry specifies that the usage is DISPLAY, then a boolean position is synonymous with a character position. Further characteristics of the physical storage are defined by the implementor.

BOTTOM MARGIN
An empty area which follows the page body.

CALCULATION KEY
The location mode data items defined in a record whose contents are used by the DBCS in its algorithm to calculate a data base key value when that record is stored.

CALCULATION LOCATION MODE
The location mode in which the user specifies to the DBCS the data items whose contents are to be used in the location mode algorithm.

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.

Withdrawn/Retiree

CALLING PROGRAM
A program which executes a CALL to another program.

CD-NAME
A user-defined word that names an MCS interface area described in a Communication Description entry within the Communication Section of the Data Division.

CHARACTER
The basic indivisible unit of the language.

CHARACTER DATA ITEM
A data item consisting entirely of Standard Data Format characters.

CHARACTER POSITION
A character position is the amount of physical storage required to store a single Standard Data Format character whose usage is DISPLAY. Further characteristics of the physical storage are defined by the implementor.

CHARACTER-STRING
A sequence of contiguous characters which form a COBOL word, a literal, a PICTURE character-string, or a comment-entry. (See 3.2.2, Character-Strings.)

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.

CLAUSE
A clause is an ordered set of consecutive COBOL character-strings whose purpose is to specify an attribute of an entry.

Withdrawn/Retirée

COBOL CHARACTER SET

The complete COBOL character set, exclusive of the contents of nonnumeric literals, comment-entries, and comment lines, consists of the 59 characters listed below:

Character	Meaning
0 , 1 , . . . , 9	digit
A , B , . . . , Z	letter
	space
+	plus sign
–	minus sign (hyphen)
*	asterisk
/	stroke (virgule, slash)
=	equal sign
$	currency sign
,	comma (decimal point)
;	semicolon
.	period (decimal point, full stop)
"	quotation mark
(	left parenthesis
)	right parenthesis
>	greater than symbol
<	less than symbol
!	exclamation point
#	number sign
%	percent
&	ampersand
'	apostrophe
:	colon
?	question mark
@	commercial at

NOTE: Certain of the characters comprising the COBOL character set may not be graphically represented in definitions of National and International Standard character sets. In these instances, a substitute graphic may be specified to replace the character(s) not represented.

COBOL WORD

(See WORD)

Withdrawn/Retirée

COLLATING SEQUENCE
The sequence in which the characters that are acceptable to a computer are ordered for purposes of sorting, merging and comparing.

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 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 stroke (/) 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.

COMMUNICATION DESCRIPTION ENTRY
An entry in the Communication Section of the Data Division that is composed of the level indicator CD, followed by a cd-name, and then followed by a set of clauses as required. It describes the interface between the Message Control System (MCS) and the COBOL program.

COMMUNICATIONS DEVICE
A mechanism (hardware or hardware/software) capable of sending data to a queue and/or receiving data from a queue. This mechanism may be a computer or a peripheral device. One or more programs containing Communication Description entries and residing within the same computer define one or more of these mechanisms.

COMMUNICATION SECTION
The section of the Data Division that describes the interface areas between the MCS and the program, composed of one or more CD description areas.

COMPILE TIME
The time at which a COBOL source program is translated, by a COBOL compiler, to a COBOL object program.

Withdrawn/Retired

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; NEGATED COMBINED CONDITION.)

COMPUTER-NAME
A system-name that identifies the computer upon which the program is to be compiled or run.

CONCURRENT RUN UNIT
A run unit, other than this run unit, that has been initiated but not terminated during the time in which this run unit has been initiated but not terminated.

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 combination 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 an implementor-defined 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 and qualifiers, subscripts, and indices, 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 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.

Withdrawn/Retirée

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 computers.

CONNECTIVE
A reserved word that is used to:

1. Associate a data-name, paragraph-name, condition-name or text-name with its qualifier.

2. Link two or more operands written in a series.

3. Form conditions (logical connectives). (See LOGICAL OPERATOR.)

CONTROL BREAK
A change in the value of a data item that is referenced in the CONTROLS 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 contents 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.

Withdrawn/Retirée

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 INDICATOR
A conceptual entity that is used by the DBCS as a place marker in accessing the data base.

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
The record which is available in the record area associated with the file.

DATA BASE CONTROL SYSTEM (DBCS)
An input-output control system that directs the processing of the data base.

DATA BASE DESCRIPTION ENTRY
An entry in the Sub-schema that is composed of a level indicator followed by a data-name and then followed by a set of data clauses, as required; or an entry in the Schema that describes a realm, set type or record type.

DATA BASE EXCEPTION CONDITION
The state that exists for a run unit when the DBCS detects a situation for that run unit that requires special handling.

DATA BASE EXCEPTION SECTION
A section that contains a USE FOR DB-EXCEPTION statement.

DATA BASE KEY
A key whose value is assigned by the DBCS and which uniquely identifies a record within the data base.

Withdrawn/Retired

DATA BASE KEY ITEM
A data item whose contents represent the value of a data base key.

DATA BASE STATUS INDICATOR
A seven character value, consisting of a Statement Code and a Status Code, that indicates the state of the run unit with respect to the data base.

DATA CLAUSE
A clause, appearing in a Data Description entry in the Data Division of a COBOL program or in the Record Section of a Sub-schema, that provides information describing a particular attribute of a data item.

DATA DESCRIPTION ENTRY
An entry, in the Data Division of a COBOL program or the Record Section of a Sub-schema, 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 or by the Sub-schema.

DATA ITEM VALIDATION
The process of verifying the proper contents of data items as they are accessed from, or stored in, the data base.

DATA-NAME
A user-defined word that names a data item described in a Data Description entry. When used in the General Formats, 'data-name' represents a word which must not be reference-modified, subscripted, indexed, or qualified unless specifically permitted by the rules of the 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.

Withdrawn/Retirée

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.

DE-EDIT
The logical removal of all editing characters from a numeric edited data item in order to determine that item's unedited numeric value.

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.

DERIVED DATA ITEM
A data item whose contents are generated by the DBCS.

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.

DESTINATION
The symbolic identification of the receiver of a transmission from a queue.

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. Further characteristics of the physical storage are defined by the implementor.

DIRECT LOCATION MODE
The location mode in which the user may specify to the DBCS a data base key value for use in the location mode algorithm.

Withdrawn/Retirée

DIVISION
A collection 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 (4) divisions in a COBOL program:

IDENTIFICATION
ENVIRONMENT
DATA
PROCEDURE

There are three (3) divisions in a COBOL Sub-schema:

TITLE
MAPPING
STRUCTURE

DIVISION HEADER
A combination of words, followed by a separator period, that indicates the beginning of a division. The division headers in a COBOL program are:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION      [USING data-name-1 . . . ] .
```

The division headers in a Sub-schema are:

```
TITLE DIVISION.
MAPPING DIVISION.
STRUCTURE DIVISION.
```

DYNAMIC ACCESS
An access mode in which specific logical records can be obtained from or placed into a mass storage file in a non-sequential manner and obtained from a file in a sequential manner during the scope of the same OPEN statement. (See RANDOM ACCESS; SEQUENTIAL ACCESS; and 6.10, The File Description — Complete Entry Skeleton.)

Withdrawn/Retirée

EDITING CHARACTER
A single character or a fixed two-character combination belonging to the following set:

Character	Meaning
B	space
0	zero
+	plus
–	minus
CR	credit
DB	debit
Z	zero suppress
*	check protect
$	currency sign
,	comma (decimal point)
.	period (decimal point)
/	stroke (virgule, slash)

ELEMENTARY ITEM
A data item that is described as not being further logically subdivided.

EMPTY SET
A set containing no member records.

END OF PROCEDURE DIVISION
The physical position of a COBOL source program after which no further procedures appear.

ENTRY
Any descriptive set of consecutive clauses terminated by a separator period and written in the Identification Division, Environment Division, or Data Division of a COBOL program, or in any division of a Sub-schema.

ENVIRONMENT CLAUSE
A clause that appears as part of an Environment Division entry.

EXCLUSIVE USAGE MODE
The state of a realm during which no other run unit can place it in ready mode.

EXECUTION TIME
(See OBJECT TIME.)

Withdrawn/Retirée

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 without the REEL or UNIT phrase for that file.

EXTENDED MONITORED MODE
The state of any record during which the DBCS notes if a concurrent run unit modifies that record or its record relationships.

EXTERNAL SWITCH
A hardware or software device, defined and named by the implementor, which is used to indicate that one of two alternate states exist.

FIGURATIVE CONSTANT
A compiler-generated value referenced through the use of certain reserved words. (See 3.2.2.2.4, Figurative Constant Values.)

FILE
A collection of records.

FILE CLAUSE
A clause that appears as part of any of the following Data Division entries:

File Description	(FD)
Sort–Merge File Description	(SD)
Saved Area Description	(SA)

FILE-CONTROL
The name of an Environment Division paragraph in which the data files for a given source program are declared.

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. (See 5.4.1, The FILE-CONTROL Paragraph.)

Withdrawn/Retirée

FILE SECTION
The section of the Data Division that contains File Description entries, Sort-Merge File Description entries and Saved Area Description entries together with their associated Record Descriptions.

FIXED-LENGTH RECORD
A record associated with a file whose File Description or Sort-Merge Description entry requires that all records contain the same number of character positions.

FIXED SET ORDER
A characteristic of a set type whose Set Description entry specifies that the logical order of its member records is fixed permanently as defined in the Set Description entry. A set with fixed set order, however, may be temporarily reordered during execution of a run unit, but the set's predefined fixed order is restored upon termination of that run unit.

FOOTING AREA
The position of the page body adjacent to the bottom margin.

FORMAT
A specific arrangement of a set of data.

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 specific input-output techniques, rerun points, sharing of same areas by several data files, and multiple file storage on a single input-output device, are specified.

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 without the REEL or UNIT phrase for that file.

IDENTIFIER
A syntactically correct combination of a data-name, reference modifier and qualifiers, subscripts and indices, as required for uniqueness of reference, that names a data item. The rules for 'identifier' associated with the General Formats may, however, specifically prohibit reference modification, qualification, subscripting or indexing.

Withdrawn/Retirée

IMPERATIVE STATEMENT
A statement that begins with an imperative verb and specifies an unconditional action to be taken. An imperative statement may consist of a sequence of imperative statements. (See 7.2.3, Imperative Statements and Imperative Sentences.)

IMPLEMENTOR-NAME
A system-name that refers to a particular feature available on that implementor's computing system.

IN-LINE PROCEDURE
The set of statements that constitutes the main or controlling flow of the run unit.

INDEX
A computer storage area or register, the contents of which represent the identification of a particular element in a table.

INDEX DATA ITEM
A data item in which the values associated with an index-name can be stored in a form specified by the implementor.

INDEX-NAME
A user-defined word that names an index associated with a specific table.

INDEXED DATA-NAME
An identifier that is composed of a data-name, followed by one or more index-names enclosed in parentheses.

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. (See 5.4.1, The FILE-CONTROL Paragraph.)

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 without the REEL or UNIT phrase for that file.

Withdrawn/Retiree

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-OUTPUT TECHNIQUE
A system-name that specifies an input-output technique.

INPUT PROCEDURE
A set of statements that is executed each time a record is released to the sort file.

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.

INTEGRITY
The ability to ensure that record relationships and the contents of data items are not adversely affected by concurrent processes.

INTERMEDIATE DATA ITEM
A signed numeric data item provided by the implementor that contains the results developed in the course of an arithmetic operation prior to the final result being moved to the resultant-identifier, if any.

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.

INVOCATION PRIVACY LOCK
A privacy lock that must be satisfied in order to execute a COBOL program.

KEY
A data item which identifies the location of a record, or a set of data items which serve 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.

Withdrawn/Retirée

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

LANGUAGE-NAME
A system-name that specifies a particular programming language.

LEVEL INDICATOR
Two alphabetic characters that identify a specific type of file or a position in a hierarchy.

LEVEL-NUMBER
A user-defined word, expressed as a one or two digit number, which indicates the hierarchical position of a data item or the 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 and 88 identify special properties of a Data Description entry.

LIBRARY-NAME
A user-defined word that names a COBOL library that is to be used by the compiler for a given source program compilation.

LIBRARY TEXT
A sequence of character-strings and/or separators in a COBOL library.

LINAGE-COUNTER
A special register whose value points to the current position within the page body.

LINE
(See REPORT LINE.)

LINE NUMBER
An integer that denotes the vertical position of a report line on a page.

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 the called program.

LITERAL
A character-string whose value is implied by the ordered set of characters comprising the string.

Withdrawn/Retirée

LOCATION MODE
The method by which the DBCS establishes the algorithm used to determine the data base key values associated with records.

LOCATION MODE DATA ITEM
A data item whose contents are used by the DBCS in evaluating a record's location mode or set selection criteria.

LOCKED RESOURCE
A resource for which a privacy lock has been defined.

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. (See REPORT WRITER LOGICAL RECORD.)

LOW ORDER END
The rightmost character of a string of characters.

MANUAL MEMBER
A record that explicitly becomes a member of a given set after the record is initially stored in the data base.

MASS STORAGE
A storage medium in which data may be organized and maintained in both a sequential and nonsequential manner.

MASS STORAGE CONTROL SYSTEM (MSCS)
An input-output control system that directs, or controls, the processing of mass storage files.

MASS STORAGE FILE
A collection of records that is assigned to a mass storage medium.

MCS
(See MESSAGE CONTROL SYSTEM.)

Withdrawn/Retirée

MEMBER CONDITION
The proposition, for which a truth value can be determined, that a set has members.

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.

MESSAGE
Data associated with an End of Message Indicator or an End of Group Indicator. (See MESSAGE INDICATORS.)

MESSAGE CONTROL SYSTEM (MCS)
A communications control system that supports the processing of messages.

MESSAGE COUNT
The count of the number of complete messages that exist in the designated queue of messages.

MESSAGE INDICATORS
EGI (End of Group Indicator), EMI (End of Message Indicator), and ESI (End of Segment Indicator) are conceptual indications that serve to notify the MCS that a specific condition exists (end of group, end of message, end of segment). Within the hierarchy of EGI, EMI, and ESI, an EGI is conceptually equivalent to an ESI, EMI and EGI. An EMI is conceptually equivalent to an ESI and EMI. Thus, a segment may be terminated by an ESI, EMI, or EGI. A message may be terminated by an EMI or EGI.

MESSAGE SEGMENT
Data that forms a logical subdivision of a message, normally associated with an End of Segment Indicator. (See MESSAGE INDICATORS.)

MNEMONIC-NAME
A user-defined word that is associated in the Environment Division with a specific implementor-name.

MODE-NAME
A system-name that refers to a particular method of data representation on a physical storage medium.

MONITORED MODE
The state of the current record of the run unit during which the DBCS notes if a concurrent run unit modifies that record or its record relationships.

MSCS
(See MASS STORAGE CONTROL SYSTEM.)

Withdrawn/Retirée

NATIVE CHARACTER SET
The implementor-defined character set associated with the computer specified in the OBJECT-COMPUTER paragraph.

NATIVE COLLATING SEQUENCE
The implementor-defined collating sequence associated with the computer specified in the OBJECT-COMPUTER paragraph.

NEGATED COMBINED CONDITION
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. (See 3.4.2, Explicit and Implicit Transfers of Control.)

NEXT EXECUTABLE STATEMENT
The next statement to which control will be transferred after execution of the current statement is complete. (See 3.4.2, Explicit and Implicit Transfers of Control.)

NEXT RECORD
The record which logically follows the current record of a file.

NEXT RECORD POINTER
A conceptual entity that points to the next logical record, indicates the at end condition, or is set to indicate that no valid next record has been established.

NONNUMERIC ITEM
A data item whose description permits its contents 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.

Withdrawn/Retirée

NULL
A data attribute associated with currency indicators and data items defined in a Sub-schema. This attribute is independent of the value of the contents of data items.

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 contents 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 6.42, The SIGN Clause.)

NUMERIC LITERAL
A literal composed of one or more numeric characters that 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.

OBJECT-COMPUTER
The name of an Environment Division paragraph in which the computer environment, within which the object program is executed, is described.

OBJECT OF ENTRY
A set of operands and reserved words, within a Data Division entry of a COBOL program or within a Sub-schema, 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 SUB-SCHEMA
A set or group of machine language material that describes that portion of a data base known to a COBOL program.

OBJECT TIME
The time at which an object program is executed.

Withdrawn/Retirée

OPEN MODE
The state of a file after execution of an OPEN statement for that file and before the execution of a CLOSE statement without the REEL or UNIT phrase 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 publication, any lower-case 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 data indicated by the operand.

OPERATIONAL SIGN
An algebraic sign, associated with a numeric data item or a numeric literal, to indicate whether its value is positive or negative.

OPTIONAL FILE
An input file which is declared as being not necessarily present each time the object program is executed. The object program causes an interrogation for the presence or absence of the file.

OPTIONAL WORD
A reserved word that is included in a specific format only to improve the readability of the language and whose presence is optional to the user when the format in which the word appears is used in a source program.

OUT-OF-LINE PROCEDURE
A set of statements not included in the main or controlling flow of the program.

OUTPUT FILE
A file that is opened in either the output mode or extend mode.

OUTPUT MODE
The state of a file after execution of an OPEN statement, with the OUTPUT or EXTEND phrase specified, for that file and before the execution of a CLOSE statement without the REEL or UNIT phrase for that file.

OUTPUT PROCEDURE
A set of statements to which control is given during execution of a SORT statement after the sort function is completed, or during execution of a MERGE statement after the merge function has selected the next record in merged order.

PADDING CHARACTER
An alphanumeric character used to fill the unused character positions in a physical record.

Withdrawn/Retiree

PAGE
A vertical division of a report representing a physical separation of report data, the separation being based on internal reporting requirements and/or external characteristics of the reporting medium.

PAGE BODY
That part of the logical page in which lines can be written and/or spaced. (See 6.30, The LINAGE Clause.)

PAGE FOOTING
A report group that is presented at the end of a report page as determined by the Report Writer Control System.

PAGE HEADING
A report group that is presented at the beginning of a report page as determined by the Report Writer Control System.

PARAGRAPH
In the Procedure Division, a paragraph-name followed by a separator period and by zero, one or more entries. In the Identification and Environment Divisions, a paragraph header followed by zero, one or more entries.

PARAGRAPH HEADER
A reserved word, followed by the separator period, that indicates the beginning of a paragraph in the Identification and Environment Divisions. The permissible paragraph headers are:

In the Identification Division:

PROGRAM-ID.
AUTHOR.
INSTALLATION.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.

In the Environment Division:

SOURCE-COMPUTER.
OBJECT-COMPUTER.
SPECIAL-NAMES.
FILE-CONTROL.
I-O-CONTROL.

Withdrawn/Retirée

PARAGRAPH-NAME
A user-defined word that identifies and begins a paragraph in the Procedure Division.

PERMANENT MEMBER
A record that, once it is a member of a given set, cannot be removed from membership in a set of that set type unless that record is removed from the data base.

PHRASE
A phrase is an ordered set of one or more consecutive COBOL character-strings that form a portion of a COBOL procedural statement or of a COBOL clause.

PHYSICAL RECORD
(See BLOCK.)

PRIME RECORD KEY
A key whose contents uniquely identify a record within an indexed file.

PRINTABLE GROUP
A report group that contains at least one print line.

PRINTABLE ITEM
A data item, the extent and contents of which are specified by an elementary report entry. This elementary report entry contains a COLUMN NUMBER clause, a PICTURE clause, and a SOURCE, SUM or VALUE clause.

PRIVACY KEY
A key whose contents are used by the DBCS to determine if a locked resource is to be made available to the run unit that specified that privacy key value.

PRIVACY LOCK
A value, known to the DBCS, that is associated with a locked resource.

PRIVACY PROCEDURE
A set of statements used to generate a privacy key. A privacy procedure is a part of a privacy section.

PRIVACY SECTION
A section that contains a USE FOR PRIVACY statement.

PROCEDURE
A paragraph or group of logically successive paragraphs, or a section or group of logically successive sections, within the Procedure Division.

Withdrawn/Retirée

PROCEDURE-NAME
A user-defined word which is used to name a paragraph or section in the Procedure Division. It consists of a paragraph-name (which may be qualified), or a section-name.

PROCESSING CYCLE
A single execution of a defined out-of-line procedure.

PROGRAM-NAME
A user-defined word that identifies a COBOL source program.

PROTECTED USAGE MODE
The state of a realm in which its records cannot be modified by another run unit.

PSEUDO-FILE-NAME
A user-defined word that names a file residing on a multiple file tape for which no File Description entry is specified. (See 5.4.2, The I-O-CONTROL Paragraph.)

PSEUDO-TEXT
A sequence of character-strings and/or separators bounded by, but not including, pseudo-text delimiters.

PSEUDO-TEXT DELIMITER
Two contiguous equal sign (=) characters used to delimit pseudo-text.

PUNCTUATION CHARACTER
A character that belongs to the following set:

Character	Meaning
,	comma
;	semicolon
:	colon
.	period (full stop)
"	quotation mark
(	left parenthesis
)	right parenthesis
	space
=	equal sign

QUALIFIED DATA-NAME
An identifier that is composed of a data-name followed by one or more sets of either of the connectives OF and IN followed by a data-name qualifier.

Withdrawn/Retirée

QUALIFIER

a. A data-name which is used in a reference together with another data-name at a lower level in the same hierarchy.

b. A section-name which is used in a reference together with a paragraph-name specified in that section.

c. A library-name which is used in a reference together with a text-name associated with that library.

(See 3.3.8.1, Qualification.)

QUEUE
A logical collection of messages awaiting transmission or processing.

QUEUE NAME
A symbolic name that indicates to the MCS the logical path by which a message or a portion of a completed message may be accessible in a queue.

RANDOM ACCESS
An access mode in which the program-specified value of a key data item identifies the logical record that is obtained from, deleted from or placed into a relative or indexed file. (See 6.10, The File Description — Complete Entry Skeleton.)

READY MODE
The state of a realm after execution of a READY statement for that realm and before the execution of a FINISH statement for that realm.

REALM
A logical subdivision of the data base which contains the records having specific data base key values.

REALM CURRENCY INDICATOR
The conceptual entity that is used by the DBCS as a place marker to indicate a specific record in a specific realm.

REALM DATA ITEM
A data item associated with a given record type whose contents identify the realm in which a given record of that record type is to be stored.

Withdrawn/Retirée

REALM DESCRIPTION ENTRY
An entry in the Schema or in the Realm Section of the Sub-schema that defines a realm and its characteristics.

REALM-NAME
A data-name that names a realm.

RECORD
(See LOGICAL RECORD.)

RECORD AREA
A storage area allocated for the purpose of processing the record described in a Record Description entry in the File Section of the Data Division or in the Record Section of a Sub-schema.

RECORD DESCRIPTION
(See RECORD DESCRIPTION ENTRY.)

RECORD DESCRIPTION ENTRY
The total set of Data Description entries associated with a particular record.

RECORD KEY
A key, either the Prime Record Key or an Alternate Record Key, whose contents identify a record within an indexed file.

RECORD-NAME
A user-defined word that names a record described in a Record Description entry in the Data Division of a COBOL program or in the Record Section of a Sub-schema.

RECORD SELECTION EXPRESSION
A word or group of contiguous words in a COBOL source program that specifies the algorithm to be used by the DBCS to identify a specific data base record.

RECORD TYPE
The collection of records described by a Record Description entry.

RECORD TYPE CURRENCY INDICATOR
The conceptual entity that is used by the DBCS as a place marker to indicate a specific record of a specific record type.

Withdrawn/Retirée

REFERENCE FORMAT
A format that provides a standard method for describing COBOL source programs or COBOL Sub-schema source programs.

REFERENCE MODIFIER
The leftmost-character-position and length used to establish and reference a data item. (See 3.3.8.4, Reference Modification.)

RELATION
(See RELATIONAL OPERATOR.)

RELATION CHARACTER
A character that belongs to the following set:

Character	Meaning
>	greater than
<	less than
=	equal to

RELATION CONDITION
The proposition, for which a truth value can be determined, that the value of an arithmetic expression or data item has a specific relationship to the value of another arithmetic expression or data item. (See RELATIONAL OPERATOR.)

RELATIONAL OPERATOR
A reserved word, a relation character, a group of consecutive reserved words, or a group of consecutive reserved words and relation characters used in the construction of a relation condition. The permissible operators and their meanings are:

Withdrawn/Retirée

Relational Operator	Meaning
IS [NOT] GREATER THAN IS [NOT] >	Greater than or not greater than
IS [NOT] LESS THAN IS [NOT] <	Less than or not less than
IS [NOT] EQUAL TO IS [NOT] =	Equal to or not equal to
IS UNEQUAL TO	Not equal to
EQUALS	Equal to
EXCEEDS	Greater than

RELATIONAL RECORD
A record, defined in a Sub-schema, containing no data items.

RELATIVE FILE
A file with relative organization.

RELATIVE KEY
A key whose contents identify a logical record in a relative file.

RELATIVE ORGANIZATION
The 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. (See 5.4.1, The FILE-CONTROL Paragraph.)

REPEATING GROUP
A group data item whose description contains an OCCURS clause, or a group data item subordinate to a data item whose description contains an OCCURS clause.

REPORT CLAUSE
A clause, in the Report Section of the Data Division, that appears in a Report Description entry or a Report Group Description entry.

Withdrawn/Retirée

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 REPORTS 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, provided by the implementor, 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.

Withdrawn/Retirée

RESERVED WORD
A COBOL word specified in the list of words which may be used in a COBOL source program or a COBOL Sub-schema source program, 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.

RETRIEVAL USAGE MODE
The state of a realm during which its records may be accessed but cannot be modified.

ROUTINE-NAME
A user-defined word that identifies a data base procedure or a procedure written in a language other than COBOL.

RUN UNIT
One or more object programs which interact with one another and which function, at object time, as an entity to provide problem solutions.

RWCS
(See REPORT WRITER CONTROL SYSTEM.)

SCHEMA
The description of the entire data base. The Schema is not specified in the COBOL environment.

SCHEMA-NAME
A user-defined word that identifies a Schema.

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.

Withdrawn/Retiree

In the Environment and Data Division, a section header is composed of reserved words followed by a separator period. The permissible section headers are:

In the Environment Division:

```
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
```

In the Data Division:

```
FILE SECTION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
COMMUNICATION 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.

SECURITY
The ability to prohibit access to a data base record by unauthorized means.

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.

SEPARATOR
A character or two contiguous characters used to delimit character-strings. (See 3.2.1, Separators.)

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. (See 6.10, The File Description — Complete Entry Skeleton.)

Withdrawn/Retirée

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. (See 5.4.1, The FILE-CONTROL Paragraph.)

SEQUENTIAL PROCESSING
(See SYNCHRONOUS PROCESSING.)

SET
A logical collection of records in the data base. Each set consists of a set owner and zero, one, or more set members.

SET DESCRIPTION ENTRY
An entry, in the Schema or in the Set Section of the Sub-schema, that defines a set type and its characteristics.

SET LOCATION MODE
The location mode in which the user may specify to the DBCS a set type for use in the location mode algorithm.

SET MEMBER
A record stored in the data base as a non-owner participant in a specific set.

SET-NAME
A data-name that identifies all the sets of a given set type.

SET ORDERING CRITERIA
The arguments used by the DBCS to determine the relative logical position of a record within a set.

SET OWNER
A record in the data base whose existence establishes the existence of a specific set.

SET SELECTION CRITERIA
The arguments used by the DBCS to select one specific set from a collection of sets constituting a given set type.

SET TYPE
The collection of sets described by a Set Description entry.

Withdrawn/Retirée

SET TYPE CURRENCY INDICATOR
The conceptual entity that is used by the DBCS as a place marker to indicate a specific record in a specific set of a specific set type.

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
tenancy condition
member condition
nullity condition
(simple–condition)

SINGULAR SET
A set for which the DBCS is the set owner.

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
The symbolic identification of the originator of a transmission to a queue.

SOURCE-COMPUTER
The name of an Environment Division paragraph in which the computer environment, within which the source program is compiled, is described.

SOURCE ITEM
An identifier designated by a SOURCE clause that provides the value of a printable item.

Withdrawn/Retirée

SOURCE PROGRAM
Although it is recognized that a source program may be represented by other forms and symbols, in this report it always refers to a syntactically correct set of COBOL statements or COBOL Sub-schema clauses. A COBOL source program or Sub-schema commences with an Identification Division or Title Division, respectively, and terminates with the end of the Procedure Division or the end of the Record Section, respectively. In contexts where there is no danger of ambiguity, the word 'program' alone may be used in place of the phrase 'source program'.

SOURCE SUB-SCHEMA
A syntactically correct set of COBOL Sub-schema statements beginning with the Title Division and ending with the end of the Record Section of the Structure Division.

SPECIAL CHARACTER
A character that belongs to the following set:

Character	Meaning
+	plus sign
–	minus sign
*	asterisk
/	stroke (virgule, slash)
=	equal sign
$	currency sign
,	comma (decimal point)
;	semicolon
.	period (decimal point, full stop)
"	quotation mark
(	left parenthesis
)	right parenthesis
>	greater than symbol
<	less than symbol
!	exclamation point
#	number sign
%	percent
&	ampersand
'	apostrophe
:	colon
?	question mark
@	commercial at

SPECIAL-CHARACTER WORD
A reserved word which is an arithmetic operator or a relation character.

Withdrawn/Retirée

SPECIAL-NAMES
The name of an Environment Division paragraph in which implementor-names are related to user-specified mnemonic-names.

SPECIAL REGISTERS
Certain compiler generated storage areas whose primary use is to store information produced in conjunction with the use of specific COBOL features.

STANDARD DATA FORMAT
The concept used in describing data in a COBOL Data Division or a Sub-schema Record Section 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.

STATEMENT CODE
A value that indicates the type of statement whose execution has caused the DBCS to modify the Data Base Status Indicator. The statement code is a part of the Data Base Status Indicator.

STATUS CODE
A value that indicates the specific Data Base Exception Condition that most recently occurred in a run unit, or indicates that no Data Base Exception Condition occurred.

SUB-QUEUE
A logical hierarchical division of a queue.

SUB-SCHEMA
The named collection of entries in the COBOL Sub-schema Library that describes that portion of the data base accessible to a given COBOL program.

SUB-SCHEMA DESCRIPTION ENTRY
An entry in the Title Division of the Sub-schema that names the Sub-schema and describes some of its characteristics.

SUB-SCHEMA ENTRY
An entry in the Sub-schema Section of the Data Division that specifies the Sub-schema to be accessed by the COBOL program

Withdrawn/Retirée

SUB-SCHEMA-NAME
A user-defined word that identifies a COBOL Sub-schema.

SUB-SCHEMA SECTION
The section of the Data Division that references the Sub-schema to be used by the program.

SUBJECT OF ENTRY
An operand or reserved word that appears immediately following the level indicator or the level-number in a Data Division entry or a Sub-schema entry.

SUBPROGRAM
(See CALLED PROGRAM.)

SUBSCRIPT
An integer whose value 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 an implementor-defined switch, capable of being set to an 'on' or 'off' status, has been set to a specific status.

SYNCHRONOUS PROCESSING
The manner of processing logical records within in-line procedures in the order in which the records are made available.

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 of a COBOL program or in the Record Section of a Sub-schema by means of the OCCURS clause.

Withdrawn/Retirée

TABLE ELEMENT
A data item that belongs to the set of repeated items comprising a table.

TEMPORARY REALM
A realm that exists only during the execution of the run unit that creates it. When that run unit terminates, the temporary realms created by that run unit are logically removed from the data base.

TENANCY CONDITION
The proposition, for which a truth value can be determined, that a record is a member, owner, or either, of one or more sets.

TENANT
A record that is either the owner or a member of a specific set.

TERMINAL
The originator of a transmission to a queue, or the receiver of a transmission from a queue.

TEXT-NAME
A user-defined word which identifies library text.

TEXT-WORD
Any character-string or separator, except space, in a COBOL library or in pseudo-text.

TOP MARGIN
An empty area which precedes the page body.

TRANSIENT MEMBER
A record that can be removed from a given set.

TRUTH VALUE
The representation of the result of the evaluation of a condition in terms of one of two values:

true
false

UNARY BOOLEAN OPERATOR
The reserved word NOT, which precedes a boolean variable or a left parenthesis in a boolean expression, and which has the effect of negating the value of the boolean expression.

Withdrawn/Retirée

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 by +1 or —1 respectively.

UNIT
A module of mass storage the dimensions of which are determined by each implementor.

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.

UPDATE USAGE MODE
The state of a realm during which its records may be both accessed and modified.

USAGE MODE
The state of a realm that is in ready mode which designates the types of access that are permissible for that realm. The particular usage mode is specified in the READY statement.

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-LENGTH DATA ITEM
A variable-length data item is a data item which, although physically fixed in size, contains a logically variable number of characters. Such an item must contain the PICTURE symbol 'L' in its Data Description entry.

VARIABLE-LENGTH RECORD
A record associated with a file whose File Description or Sort-Merge Description entry permits records to contain a varying number of character positions.

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 a Format 2 OCCURS 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.

Withdrawn/Retirée

WORD
A character-string of not more than 30 characters which forms a user-defined word, a system-name, or a reserved word. (See 3.2.2.1, COBOL Words.)

WORKING-STORAGE SECTION
The section of the Data Division that describes Working-Storage data items and constants, composed either of noncontiguous items or of Working-Storage records or of both.

Withdrawn/Retirée

CHAPTER 3

LANGUAGE CONCEPTS

3.1 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. (See Chapter 2, 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.

Withdrawn/Retired

3.2 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.

3.2.1 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, more than one space may be used.

2. The punctuation characters comma, semicolon and period, when immediately followed by a space, are separators. These separators may appear in a COBOL source program only where explicitly permitted by the general formats, by format punctuation rules. (See 1.4.1.8, Format Punctuation; 7.2 Statements and Sentences; and Chapter 10, Reference Format.)

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

4. 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.

5. 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.

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

6. The character 'B' immediately followed by the punctuation character quotation mark is a separator. This separator must be immediately preceded by a space or left parenthesis.

7. The punctuation character colon is a separator and is required when shown in the General Formats.

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

   a. As specified by reference format rules. (See Chapter 10, 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.

Withdrawn/Retirée

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.

3.2.2 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.

3.2.2.1 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, user-defined words and reserved words form disjoint sets; user-defined words and system-names form intersecting sets. The same COBOL word may be used as a user-defined word and as a system-name 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.

3.2.2.1.1 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.

There are twenty-three (23) types of user-defined words:

1. alphabet–name
2. area–name
3. cd–name
4. condition–name
5. data–name
6. file–name
7. index–name
8. level–number
9. library–name
10. mnemonic–name
11. paragraph–name

Withdrawn/Retiree

LANGUAGE STRUCTURE

12. program-name
13. pseudo-file-name
14. realm-name
15. record-name
16. report-name
17. routine-name
18. schema-name
19. section-name
20. segment-number
21. set-name
22. sub-schema-name
23. text-name

Within a given source program, twenty-one (21) of these twenty-three (23) types of user-defined words are grouped into nineteen (19) disjoint sets. The disjoint sets are:

1. alphabet-names
2. area-names
3. cd-names
4. condition-names, data-names, and record-names
5. file-names
6. index-names
7. library-names
8. mnemonic-names
9. paragraph-names
10. program-names
11. pseudo-file-names
12. realm-names
13. report-names
14. routine-names
15. schema-names
16. section-names
17. set-names
18. sub-schema-names
19. 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 given disjoint set must be unique, except as specified in the rules for uniqueness of reference. (See 3.3.8, Uniqueness of Reference.)

With the exception of level-number and segment-number, 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.

Withdrawn/Retirée

3.2.2.1.1.1 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 only in the RERUN clause or 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.

3.2.2.1.1.2 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 5.3.3, The SPECIAL-NAMES Paragraph.)

3.2.2.1.1.3 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.

3.2.2.1.1.4 Pseudo-File-Name

A pseudo-file-name is a word which names a file residing on a multiple file tape for which no File Description entry is specified. A pseudo-file-name may only be specified in the MULTIPLE FILE TAPE clause of the I-O CONTROL paragraph. (See 5.4.2, The I-O-CONTROL Paragraph.)

3.2.2.1.1.5 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.

3.2.2.1.1.6 Other User-Defined Names

All other types of user-defined words are defined in the Glossary. (See Chapter 2, Glossary.)

3.2.2.1.2 System-Names

A system-name is a COBOL word which is used to communicate with the operating environment. Rules for the formation of a system-name are defined by the implementor, except that each character used in the

Withdrawn/Retired

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 four (4) types of system-names:

1. computer–name
2. implementor–name
3. input–output–technique
4. language–name

Within a given implementation these four types of system-names form disjoint sets; a given system-name may belong to one and only one of them. The system-names listed above are individually in the Glossary. (See Chapter 2, Glossary.)

3.2.2.1.3 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 Chapter 13, COBOL Reserved Words.)

There are three types of reserved words:

1. Required Words
2. Optional Words
3. Special Purpose Words

3.2.2.1.3.1 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 upper case and underlined.

2. Special Character Words. These are the arithmetic operators and relation characters.

3.2.2.1.3.2 Optional Words

Within each format, upper-case words that are not underlined are called optional words and may be specified at the user's option with no effect on the semantics of the format.

Withdrawn/Retirée

3.2.2.1.3.3 Special Purpose Words

There are two types of special purpose words:

1. Special Registers
2. Figurative Constants

3.2.2.1.3.3.1 Special Registers

Certain reserved words are used to name and reference special registers. Special registers are certain compiler-generated storage areas whose primary use is to store information produced in conjunction with the use of specific COBOL features.

1. LINE-COUNTER

   The reserved word LINE-COUNTER is a name for a line counter 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 defined by the implementor. The value in LINE-COUNTER is maintained by the Report Writer Control System, and is used to determine the vertical positioning of a report. The value in LINE-COUNTER may be accessed by Procedure Division statements; however, only the RWCS may change the value of LINE-COUNTER.

2. PAGE-COUNTER

   The reserved word PAGE-COUNTER is a name for a page counter 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 defined by the implementor. The value in PAGE-COUNTER is maintained by the Report Writer Control System and is used by the program to number the pages of a report. The value in PAGE-COUNTER may be altered by Procedure Division statements.

3. 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. The implicit description is that of an unsigned integer whose size is equal to the size of integer-1 or the data item referenced by data-name-1 in the LINAGE clause. (See Paragraph 6.30.4, The LINAGE Clause.)

4. DEBUG-ITEM

   The reserved word DEBUG-ITEM is the name for a special register generated automatically by the implementor's 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. (See Chapter 11, The Debugging Facility.)

5. DB-STATUS, DB-REALM-NAME, DB-SET-NAME, DB-RECORD-NAME, DB-CONFLICT, and DB-PRIVACY-KEY.

Withdrawn/Retirée

The reserved words listed above are names for special registers generated automatically by the presence of a Sub-schema Section in the Data Division. (See 12.3, Data Base Special Registers.)

3.2.2.1.3.3.2 Figurative Constants

Certain reserved words are used to name and reference specific constant values. (See 3.2.2.2.4, Figurative Constant Values.)

3.2.2.2 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 three types, boolean, nonnumeric or numeric.

3.2.2.2.1 Boolean Literals

A boolean literal is a character-string delimited on the left by the separator 'B'' and on the right by the quotation mark separator. The character-string consists only of boolean characters. The value of a boolean literal is the string of boolean characters itself, excluding the delimiting separators. All boolean literals are of the category boolean. (See 6.35, The PICTURE Clause.)

3.2.2.2.2 Nonnumeric Literals

A nonnumeric literal is a character-string of any length delimited on both ends by quotation marks and consisting of any allowable 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. The value of a nonnumeric literal in the object program is the string of characters itself, except:

1. The delimiting quotation marks are excluded, and

2. Each embedded pair of contiguous quotation marks represents a single quotation mark character.

All other punctuation characters are part of the value of the nonnumeric literal rather than separators; all nonnumeric literals are category alphanumeric. (See 6.35, The PICTURE Clause.)

3.2.2.2.3 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, and/or the decimal point; the rules for the formation of numeric literals are as follows:

1. A literal must contain at least one digit.

2. A literal must not contain more than one sign character. If a sign is used, it must appear as the leftmost character of the literal. If the literal is unsigned, the literal is positive.

Withdrawn/Retirée

3. 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. If the literal contains no decimal point, the literal is an integer.

   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 6.35, The PICTURE Clause.) The size of a numeric literal in Standard Data Format characters is equal to the number of digits in the string of characters as specified by the user.

3.2.2.2.4 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.

Withdrawn/Retirée

LANGUAGE STRUCTURE

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

ZERO ZEROS ZEROES	Represents the numeric value '0', one or more of the boolean character '0', or one or more of the character '0' from the computer's character set, depending on context.
SPACE SPACES	Represents one or more of the character space from the computer's character set.
HIGH–VALUE HIGH–VALUES	Represents one or more of the character that has the highest ordinal position in the program collating sequence.
LOW–VALUE LOW–VALUES	Represents one or more of the character that has the lowest ordinal position in the program collating sequence.
QUOTE 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''.
ALL literal	Represents one or more of the string of characters comprising the literal. The literal must be either a boolean literal, a nonnumeric literal or a figurative constant other than ALL literal. When a figurative constant is used, the word ALL is redundant and is used for readability only.

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 equal to the size in characters of 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 is not associated with another data item, as when the figurative constant appears in a DISPLAY, STRING, STOP, SUSPEND or UNSTRING statement, the length of the string is one character.

Withdrawn/Retirée

A figurative constant may be used whenever 'literal' appears in a format, except that whenever the literal is restricted to having only numeric or boolean characters in it, the only figurative constant permitted is ZERO (ZEROS, ZEROES).

When the figurative constants HIGH-VALUE(S) or LOW-VALUE(S) are used in the source program, the actual characters associated with each figurative constant depend upon the program collating sequence specified. (See 5.3.2, The OBJECT-COMPUTER Paragraph and 5.3.3, The SPECIAL-NAMES Paragraph.)

Each reserved word which is used to reference a figurative constant value is a distinct character-string with the exception of the construction 'ALL literal' which is composed of two distinct character-strings.

3.2.2.3 PICTURE Character-Strings

A PICTURE character-string consists of certain combinations of characters in the COBOL character set used as symbols. An explanation of the PICTURE character-string and the rules that govern its use are given under the appropriate paragraph. (See 6.35, 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.

3.2.2.4 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.

Withdrawn/Retired

3.3 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 the remaining characters in the COBOL character set to describe nonnumeric data items.

3.3.1 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. In addition, the logical characteristics of data stored in a data base are described apart from the data stored in a file. (See Section IV, COBOL Sub-schema Language Specifications.) The following paragraphs discuss the characteristics of files.

Whether stored in a data base or in a file, the logical characteristics of data records themselves are the same. Thus, the following paragraphs that discuss record and data concepts apply equally to both.

3.3.1.1 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 mode in which the data file is recorded on the external medium.
2. The grouping of logical records within the physical limitations of the file medium.
3. The means by which the file can be identified.

3.3.1.2 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 is convenient to a particular computer for the storage of data on an 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, in the case of mass storage files, a logical record may require more than one physical unit to contain it. There are several source language methods available for describing the relationship of logical records and physical units. When a permissible relationship has been established, control of the

Withdrawn/Retirée

accessibility of logical records as related to the physical unit must be provided by the interaction of the object program on the implementor's hardware and/or software system. In this manual, 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 may be 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.

3.3.1.3 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.

3.3.2 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.

3.3.2.1 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 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.

Withdrawn/Retirée

Two 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 condition-names.

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

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

3.3.3 CONCEPT OF CLASSES OF DATA

There are six categories of data items. (See 6.35, The PICTURE Clause.) These are grouped into four classes: Alphabetic, Boolean, Numeric and Alphanumeric. For Alphabetic, Boolean 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 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 figure depicts the relationship of the class and categories of data items.

Withdrawn/Retirée

LEVEL OF ITEM	CLASS	CATEGORY
Elementary	Alphabetic	Alphabetic
	Boolean	Boolean
	Numeric	Numeric
	Alphanumeric	Numeric Edited Alphanumeric Edited Alphanumeric
Non-elementary (Group)	Alphanumeric	Alphabetic Boolean Numeric Numeric Edited Alphanumeric Edited Alphanumeric

Figure 3.3-1 Relationship of Class and Category of Data Items

Withdrawn/Retirée

3.3.4 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 actually 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. If more than one arithmetic radix is provided, the selection is dependent upon factors included in such clauses as USAGE and RECORDING MODE. 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.

When a computer provides more than one means of representing data, the Standard Data Format must be 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, PICTURE, etc., clauses. Each implementor provides a complete explanation of the possible forms on the computer for which he is

implementing COBOL. The method used in selecting the proper data form is also provided to allow the programmer 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 that required for internal representation.

3.3.5 ALGEBRAIC SIGNS

Algebraic signs fall into two categories: operational signs, which are associated with signed numeric data items and signed numeric literals to indicate their algebraic properties; and editing signs, which appear (e.g.) 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. This clause is optional; if it is not used, operational signs will be represented as defined by the implementor.

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

3.3.6 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 digit 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 digit 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.

4. If the receiving data item is boolean, the sending data is moved to the receiving boolean positions and aligned at the leftmost boolean position in the data item with boolean character zero fill or truncation to the right, as required.

Withdrawn/Retirée

If the JUSTIFIED clause is specified for the receiving item, these standard rules are modified. (See 6.27, The JUSTIFIED Clause.)

3.3.7 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 boundaries, half-word boundaries, 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 procedure (other than READ or WRITE) which stores data in the item.

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.

Each implementor who provides for special types of alignment will specify the precise interpretations which are to be made.

Withdrawn/Retired

DATA DESCRIPTION

3.3.8 UNIQUENESS OF REFERENCE

3.3.8.1 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.

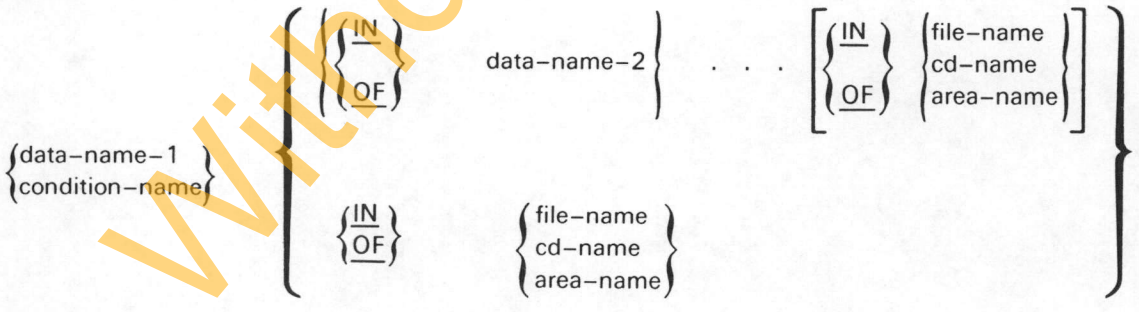
2. It is unique within the context of a REDEFINES CLAUSE. (See 6.39, The REDEFINES Clause.)

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 in the hierarchy. These higher-level names 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 for 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 the 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.

In a data base environment, hierarchical relationships are different from hierarchical relationships in a file. In the hierarchy of qualification for data-names defined in the Data Division, names associated with a level indicator are the most significant, then those names associated with level-number 01 (record-names), then names associated with level-number 02, 03, . . ., 49. In the Sub-schema, the record-names, the set-names and the realm-names are of the same hierarchical level.

The general formats for qualification are:

Format 1



Format 2



Withdrawn/Retirée

Format 3

$$\text{text-name} \quad \left\{\begin{matrix}\underline{\text{OF}}\\ \underline{\text{IN}}\end{matrix}\right\} \quad \text{library-name}$$

Format 4

$$\underline{\text{LINAGE-COUNTER}} \quad \left\{\begin{matrix}\underline{\text{IN}}\\ \underline{\text{OF}}\end{matrix}\right\} \quad \text{file-name}$$

Format 5

$$\left\{\begin{matrix}\underline{\text{PAGE-COUNTER}}\\ \underline{\text{LINE-COUNTER}}\end{matrix}\right\} \quad \left\{\begin{matrix}\underline{\text{IN}}\\ \underline{\text{OF}}\end{matrix}\right\} \quad \text{report-name}$$

Format 6

$$\text{data-name-3} \quad \left\{\begin{matrix}\left\{\begin{matrix}\underline{\text{IN}}\\ \underline{\text{OF}}\end{matrix}\right\} \quad \text{data-name-4} \quad \left[\left\{\begin{matrix}\underline{\text{IN}}\\ \underline{\text{OF}}\end{matrix}\right\} \quad \text{report-name}\right]\\ \left\{\begin{matrix}\underline{\text{IN}}\\ \underline{\text{OF}}\end{matrix}\right\} \quad \text{report-name}\end{matrix}\right\}$$

The rules for qualification are as follows:

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

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

Withdrawn/Retirée

3. 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.

4. 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.

5. If more than one COBOL library is available to the compiler during compilation, text-name must be qualified each time it is referenced.

6. IN and OF are logically equivalent.

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. PAGE-COUNTER and LINE-COUNTER must be qualified each time they are referenced if more than one Report Description entry has been specified in the source program.

9. If record-name is defined in the Data Division, it may be qualified. If record-name is defined in the Sub-schema, it must not be qualified.

3.3.8.2 Subscripting

Subscripts can be used only when reference is made to an individual element within a list or table of like elements that have not been assigned individual data-names. (See 6.33, The OCCURS Clause.)

The subscript can be represented either by a numeric literal that is an integer or by a data-name. The data-name must be a numeric elementary item that represents an integer. When the subscript is represented by a data-name qualification is permissible. However, no subscript can itself be subscripted. In the Report Section, neither a sum counter nor the special registers LINE-COUNTER and PAGE-COUNTER can be used as a subscript.

The subscript may be signed and, if signed, it must be positive. The lowest possible subscript value is 1. This value points to the first element of the table. The next sequential elements of the table are pointed to by subscripts whose values are 2, 3, ... The highest permissible subscript value, in any particular case, is the maximum of occurrences of the item as specified in the OCCURS clause.

The subscript, or set of subscripts, that identifies the table element is delimited by the balanced pair of separators left parenthesis and right parenthesis following the table element data-name. The table element data-name appended with a subscript is called a subscripted data-name or an identifier. When more than one subscript is required, they are written in the order of successively less inclusive dimensions of the data organization.

Withdrawn/Retirée

The general format for subscripting is:

```
{data-name     }
{condition-name}  (subscript [ , subscript ] . . . )
```

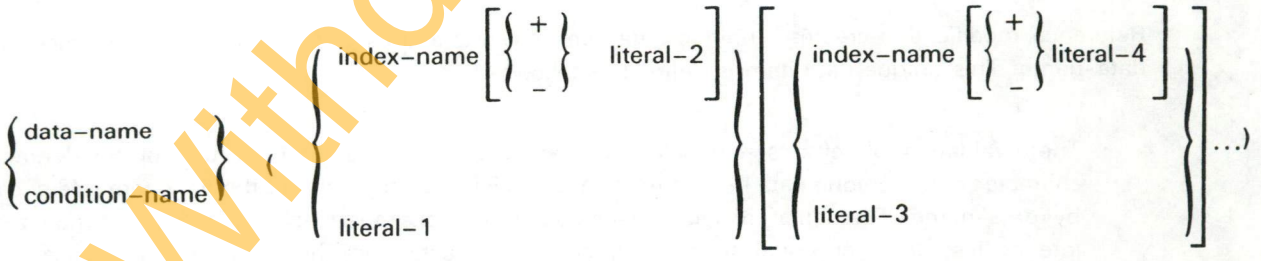
3.3.8.3 Indexing

References can be made to individual elements within a table of like elements by specifying indexing for that reference. An index is assigned to that level of the table by using the INDEXED BY clause in the definition of a table. A name given in the INDEXED BY clause is known as an index-name and is used to refer to the assigned index. The value of an index corresponds to the occurrence number of an element in the associated table. An index must be initialized before it is used as a table reference. An index can be given an initial value by either a SET statement, a SEARCH statement with the ALL phrase or a PERFORM statement with the VARYING phrase.

Direct indexing is specified by using an index-name in the form of a subscript. Relative indexing is specified when an index-name is followed by the operator + or -, followed by an unsigned integer numeric literal all delimited by the balanced pair of separators left parenthesis and right parenthesis following the table element data-name. The occurrence number resulting from relative indexing is determined by incrementing (where the operator + is used) or decrementing (where the operator - is used), by the value of the literal, the occurrence number represented by the value of the index. When more than one index-name is required, they are written in the order of successively less inclusive dimensions of the data organization.

At the time of execution of a statement which refers to an indexed table element, the value contained in the index referenced by the index-name associated with the table element must neither correspond to a value less than one (1) nor to a value greater than the highest permissible occurrence number of an element of the associated table. This restriction also applies to the value resultant from relative indexing.

The general format for indexing is:



3.3.8.4 Reference Modification

3.3.8.4.1 Function

Reference modification defines a data item by specifying a leftmost character and length for the data item.

Withdrawn/Retirée

3.3.8.4.2 General Format

data-name (leftmost-character-position : { length / END })

3.3.8.4.3 Syntax Rules

1. Data-name must reference a data item whose usage is DISPLAY.

2. Leftmost-character-position and length must be arithmetic expressions.

3.3.8.4.4 General Rules

1. Each character of a data item referenced by data-name is assigned an ordinal number incrementing by one (1) from the leftmost position to the rightmost position. The leftmost position is assigned the ordinal number one (1). If the Data Description entry for data-name contains a SIGN IS SEPARATE clause, the sign position is assigned an ordinal number within that data item.

2. If the data item referenced by data-name is described as numeric, numeric-edited, alphanumeric or alphanumeric-edited, it is operated upon for purposes of reference modification as if it were redefined as an alphanumeric data item of the same size as the data item referenced by data-name.

3. Reference modification for an operand is evaluated immediately after evaluation of any subscripts or indices that are specified for that operand.

4. Reference modification creates a unique data item which is a subset of the data item referenced by data-name. This unique data item is defined as follows:

   a. The evaluation of leftmost-character-position specifies the ordinal position of the leftmost character of the unique data item in relation to the leftmost character of the data item referenced by data-name. Evaluation of leftmost-character-position must result in a positive non-zero integer less than or equal to the number of characters in the data item referenced by data-name.

   b. The evaluation of length specifies the size of the data item to be used in the operation. The evaluation of length must result in a positive non-zero integer. The sum of leftmost-character-position and length minus the value one (1) must be less than or equal to the number of characters in the data item referenced by data-name. If the END phrase is specified, the unique data item extends from and includes the character identified by leftmost-

Withdrawn/Retirée

character-position up to and including the rightmost character of the data item referenced by data-name.

5. The unique data item is considered an elementary data item without the JUSTIFIED clause. It has the same class and category as that defined for the data item referenced by data-name except that the categories numeric, numeric-edited and alphanumeric-edited are considered category alphanumeric.

6. Reference modification is allowed anywhere an alphanumeric identifier is permitted.

3.3.8.5 Identifier

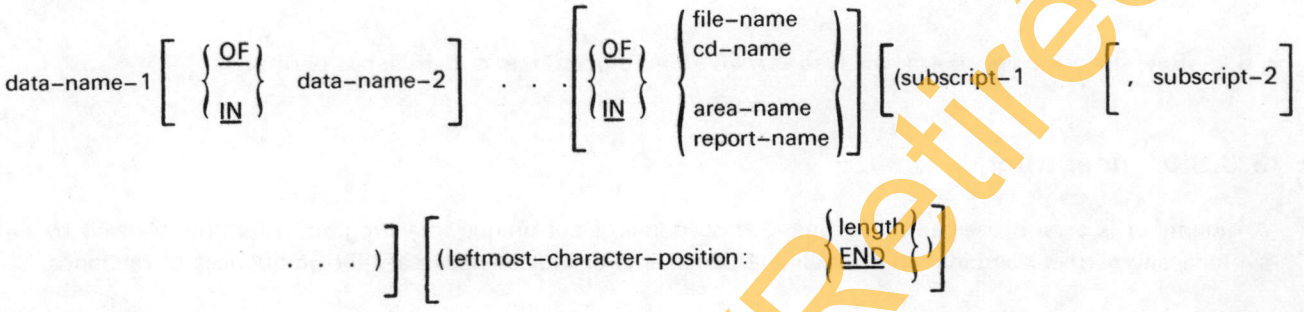
An identifier is a term used to reflect that a data-name, if not unique in a program, must be followed by a syntactically correct combination of qualifiers, subscripts or indices necessary for uniqueness of reference.

Withdrawn/Retirée

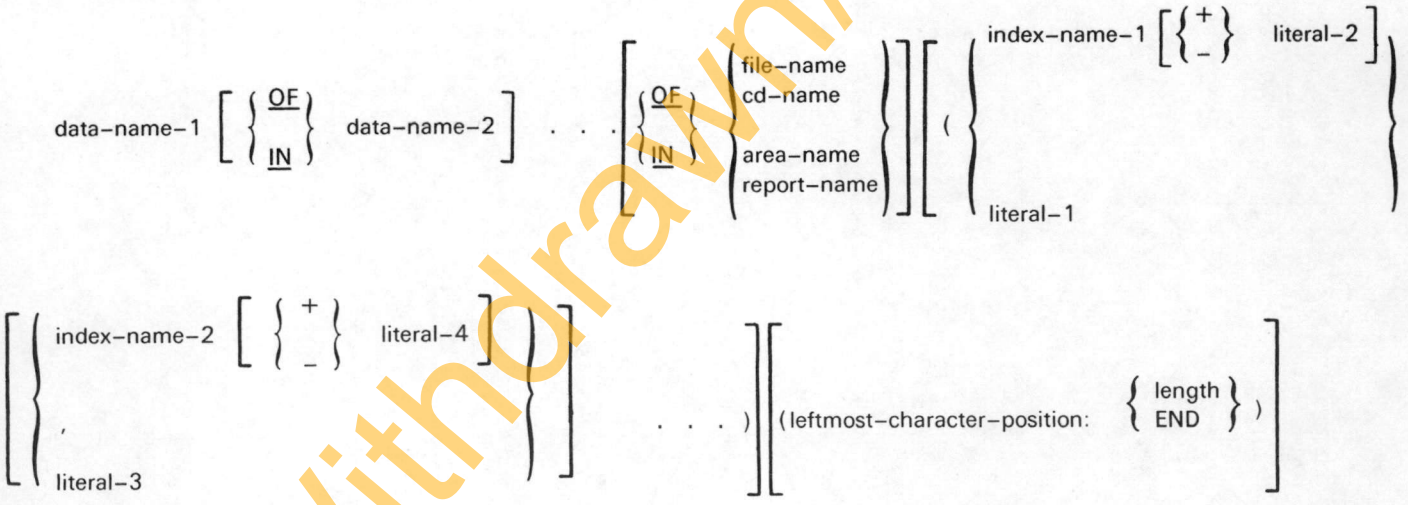
DATA DESCRIPTION

The general formats for identifiers are:

Format 1



Format 2



Restrictions on qualification, subscripting and indexing are:

1. Indexing is not permitted where subscripting is not permitted.

2. An index may be modified only by the SET, SEARCH, and PERFORM statements. Data items described by the USAGE IS INDEX clause permit storage of the values associated with index-names as data in a form specified by the implementor. Such data items are called index data items.

Withdrawn/Retirée

3. Literal-1, literal-3, . . ., in the above format must be positive integers. Literal-2, literal-4, . . ., must be unsigned integers.

4. Each table element reference must be either subscripted or indexed 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. Wherever a data-name is subscripted or indexed, the number of subscripts or indices must equal the number of dimensions in the table whose element is being referenced. That is, there must be a subscript or index for each OCCURS clause in the hierarchy containing the data-name, including the data-name itself.

3.3.8.6 Condition-name

If explicitly referenced, a condition-name must be unique, or be made unique through qualification and/or indexing, or subscripting.

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 indexing or subscripting, then references to any of its condition-names also require the same combination of indexing or subscripting.

The format and restrictions on the combined use of qualification, subscripting, and indexing 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, indexed or subscripted, as necessary.

3.3.9 DATA BASE KEYS

Every record stored in the data base must be uniquely identifiable to the DBCS. Accordingly, the DBCS assigns to every data base record a unique record identifier known as a data base key value. A record's data base key is not a part of the record, but contains a value that is used to identify that specific record.

The DBCS generates and assigns a record's data base key value when that record is placed in the data base for the first time. Once assigned, that record's data base key value remains invariable so long as that record

Withdrawn/Retirée

logically remains in the data base. When a record has been logically removed from the data base, its data base key value may be reused by the DBCS to identify a different record placed into the data base at a later time.

The method used by the DBCS to generate a data base key value is based on the record's description, and possibly on values provided by the COBOL program during execution of the program. The size, format, and characteristics of a data base key are implementor-defined.

There is no inherent relationship between a record's data base key value and its physical location, realm, or set membership. The mapping of data base key values to physical storage locations in the data base is implementor-defined.

The mechanism for specifying which realms contain records with specific data base key values is outside the scope of the COBOL environment.

Withdrawn/Retirée

3.4 EXPLICIT AND IMPLICIT SPECIFICATIONS

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

1. Explicit and Implicit Procedure Division References
2. Explicit and Implicit Transfers of Control
3. Explicit and Implicit Attributes.

3.4.1 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.

3.4.2 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.

Withdrawn/Retirée

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 (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 7.6, 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 causes an explicit transfer of control only when the statement contains the optional PROGRAM phrase and is executed in a called program.

In this manual, 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. (See Chapter 7, The Procedure Division.)

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.

When there is no next executable statement and control is not transferred outside the COBOL program, the program flow of control is undefined.

3.4.3 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 know 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.

Withdrawn/Retired

3.5 EXTERNAL SWITCH

An external switch is a hardware or software device, defined and named by the implementor, which is used to indicate that one of two alternate states exist. These alternate states are referred to as the 'on' status and the 'off' status of the associated external switch. The status of an external switch may be interrogated by testing condition-names associated with that switch. The association of a condition-name with an external switch and the association of a user-specified mnemonic-name with the implementor-name that names an external switch is established in the SPECIAL-NAMES paragraph of the Environment Division. (See 5.3.3, The SPECIAL-NAMES Paragraph.) The implementor defines the scope (program, run unit, etc.) of each external switch and any facility external to COBOL which may be used to modify the status of an external switch. For example, if the scope of an external switch is the run unit, each implementor-name that names such an external switch refers to one and only one such switch, the status of which is available to each object program functioning within that run unit. The status of certain external switches may be altered by the SET statement. (See 7.40, The SET Statement.)

Withdrawn/Retired

CHAPTER 4

IDENTIFICATION DIVISION

4.1 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.

4.2 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.

4.2.1 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. Paragraph 4.3 and 4.4 define the PROGRAM-ID paragraph and the DATE-COMPILED paragraph. While the other paragraphs are not defined, each General Format is formed in the same manner.

4.2.1.1 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] . . . ]
```

4.2.1.2 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 extend to more than one line.

Withdrawn/Retiree

4.3 THE PROGRAM-ID PARAGRAPH

4.3.1 FUNCTION

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

4.3.2 General Format

PROGRAM-ID. program-name.

4.3.3 Syntax Rules

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

4.3.4 General Rules

1. The PROGRAM-ID paragraph must contain the name of the program and must be present in every program.

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

Withdrawn/Retirée

4.4 THE DATE-COMPILED PARAGRAPH

4.4.1 FUNCTION

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

4.4.2 General Format

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

4.4.3 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 line.

4.4.4 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.

Withdrawn/Retirée

CHAPTER 5

ENVIRONMENT DIVISION

5.1 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.

The Environment Division must be included in every COBOL source program.

5.2 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 relates the implementor-names used by the compiler to the mnemonic-names used in the 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. 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.

5.2.1 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. The definitions of the entries for the contents of the paragraphs shown below are given on the following pages.

Withdrawn/Retired

ORGANIZATION

5.2.1.1 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]]]
```

Withdrawn/Retirée

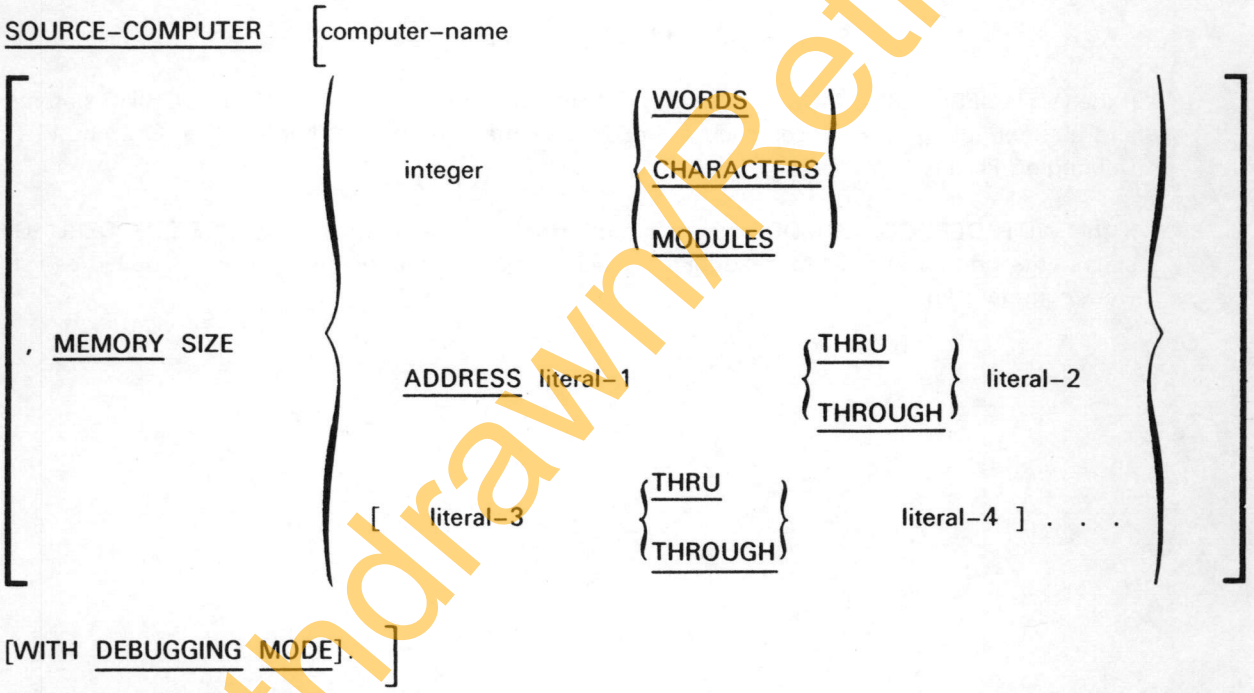
5.3 CONFIGURATION SECTION

5.3.1 THE SOURCE-COMPUTER PARAGRAPH

5.3.1.1 Function

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

5.3.1.2 General Format



5.3.1.3 Syntax Rules

1. Computer-name is a system-name.

2. Each literal may be numeric or nonnumeric; when numeric, it must be an unsigned integer.

Withdrawn/Retiree

3. The words THRU and THROUGH are equivalent.

5.3.1.4 General Rules

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

2. 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 and the effect is as though the SOURCE-COMPUTER paragraph is replaced in the source program with a paragraph of the form:

   SOURCE-COMPUTER . computer-name .

3. If the WITH DEBUGGING MODE clause is specified in a program, all USE FOR DEBUGGING statements and all debugging lines are compiled as specified in the appropriate Chapter. (See Chapter 11, The Debugging Facility.)

   If the WITH DEBUGGING MODE clause is not specified in a program, any USE FOR DEBUGGING statements and all associated debugging sections, and any debugging lines are compiled as if they were comment lines.

Withdrawn/Retirée

5.3.2 THE OBJECT-COMPUTER PARAGRAPH

5.3.2.1 Function

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

5.3.2.2 General Format

```
OBJECT–COMPUTER . [computer–name
[                                                                          ]
[                                  { WORDS      }                          ]
[                    { integer     { CHARACTERS }                          ]
[                                  { MODULES    }                          ]
[      MEMORY SIZE   {                                                     }]
[                    { ADDRESS literal–1 { THRU    } literal–2            }]
[                    {                   { THROUGH }                      }]
[                    { [ , literal–3     { THRU    } literal–4 ] . . .    }]
[                    {                   { THROUGH }                      }]
[                                                                          ]
[ , PROGRAM COLLATING SEQUENCE IS alphabet–name]
[ , SEGMENT–LIMIT IS segment–number ].                                     ]
```

5.3.2.3 Syntax Rules

1. Computer-name is a system-name.
2. Each literal may be numeric or nonnumeric; when numeric, it must be an unsigned integer.
3. Segment-number must be an integer ranging in value from 1 through 49.

Withdrawn/Retirée

4. The words THRU and THROUGH are equivalent.

5.3.2.4 General Rules

1. When the OBJECT-COMPUTER paragraph is not specified, the computer upon which the source program is being compiled is the object computer.

2. When the OBJECT-COMPUTER paragraph is specified, but the object-computer-entry is not specified, the computer upon which the source program is being compiled is the object computer and the effect is as though the OBJECT-COMPUTER paragraph is replaced in the source program with a paragraph of the form:

   OBJECT–COMPUTER . computer–name .

3. If the PROGRAM COLLATING SEQUENCE clause is specified, the collating sequence associated with alphabet-name is used to determine the truth value of any nonnumeric comparisons that are:

   a. Explicitly specified in relation conditions. (See 7.5.1.1, Relation Condition.)

   b. Explicitly specified in condition-name conditions. (See 7.5.1.3, Condition-Name Condition (Conditional Variable).)

   c. Implicitly specified by the presence of a CONTROL clause in a Report Description entry. (See 6.22, The CONTROL Clause.)

4. If the PROGRAM COLLATING SEQUENCE clause is not specified, the native collating sequence is used.

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

6. The PROGRAM COLLATING SEQUENCE clause is also applied to any nonnumeric merge or sort keys unless the COLLATING SEQUENCE phrase of the respective MERGE or SORT statements is specified. (See 7.27, The MERGE Statement and 7.41, The SORT Statement.)

7. The PROGRAM COLLATING SEQUENCE clause applies only to the program in which it is specified.

8. The optional SEGMENT-LIMIT clause gives the limit for segmentation to be varied by the user depending upon the configuration specified in the OBJECT-COMPUTER paragraph.

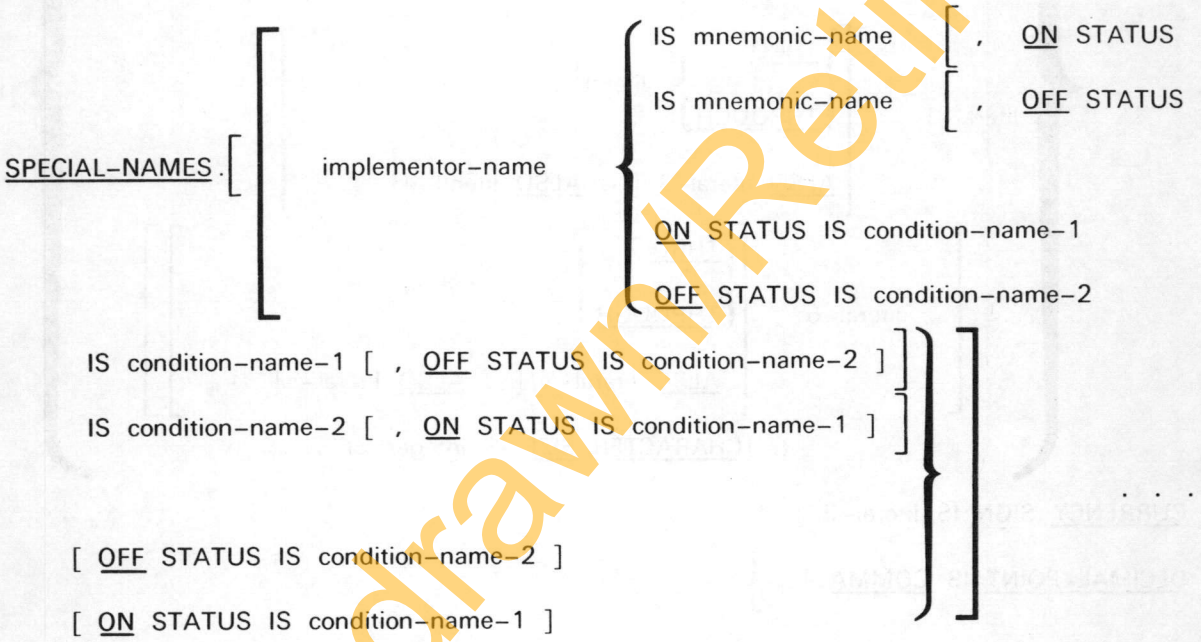
Withdrawn/Retirée

5.3.3 THE SPECIAL-NAMES PARAGRAPH

5.3.3.1 Function

The SPECIAL-NAMES paragraph provides a means of relating implementor-names to user-specified mnemonic-names and of relating alphabet-names to character sets and/or collating sequences.

5.3.3.2 General Format



Withdrawn/Retirée

SPECIAL-NAMES

```
[ , ALPHABET alphabet-name IS
                                                                    ]
    { STANDARD-1                                                    }
    { STANDARD-2                                                    }
    { NATIVE                                                        }
    { implementor-name                                              }
    { identifier [CHARACTER SIZE IS integer BITS]                   }
    { literal-1  [ {THRU    } literal-2                             ] }
    {            [ {THROUGH } literal-2                             ] }
    {            [ ALSO literal-3 [ , ALSO literal-4] . . .         ] }
    {            [ , literal-5 [ {THRU    } literal-6               ] ] }
    {            [ , literal-5 [ {THROUGH } literal-6               ] ] }
    {            [ , literal-5 [ ALSO literal-7 [ , ALSO literal-8] . . . ] ] . . . }
    {            [CHARACTER SIZE IS integer BITS]                   }
[ , CURRENCY SIGN IS literal-9 ]
[ , DECIMAL-POINT IS COMMA ] .
```

5.3.3.3 Syntax Rules

1. The data item referenced by identifier must be defined as one of the following:

   a. An alphanumeric elementary data item.

   b. An unsigned integer elementary data item.

   c. A group data item whose subordinate elementary data items are either alphanumeric or unsigned integer numeric data items.

2. 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 one (1) through the maximum number of characters in the native character set.

Withdrawn/Retirée

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

3. If the literal phrase of the ALPHABET clause is specified:

   a. A given character must not be specified more than once in an ALPHABET clause that is referenced in the PROGRAM COLLATING SEQUENCE clause or that is associated with a file that is opened in either the output or I-O mode.

   b. A given character may be specified more than once in an ALPHABET clause associated only with an input file.

4. The words THRU and THROUGH are equivalent.

5. If implementor-name does not reference an external switch, the associated mnemonic-name may be specified only in the ACCEPT, DISPLAY, SEND or WRITE statements. A condition-name cannot be associated with such an implementor-name.

6. If the implementor-name references an external switch, the associated mnemonic-name may be specified only in the SET statement.

5.3.3.4 General Rules

1. If implementor-name references an external switch, the 'on' status and/or 'off' status of that switch may be associated with condition-names. The status of that switch may be interrogated by testing these condition-names. (See 7.5.1.4, Switch-Status Condition.)

2. If implementor-name references an external switch, the status of that switch may be altered by execution of a Format 4 SET statement which specifies as its operand the mnemonic-name associated with that switch. (See 7.40, The SET Statement.)

3. The ALPHABET clause provides a means for relating a name to a specified character code set and/or collating sequence. When alphabet-name is referenced in the PROGRAM COLLATING SEQUENCE clause (See 5.3.2, The OBJECT-COMPUTER Paragraph.) or the COLLATING SEQUENCE phrase of a SORT or MERGE statement (See 7.27, The MERGE Statement and 7.41, The SORT Statement.), or a SET statement with a SORT, SORT-MERGE, MERGE or PROGRAM phrase (See 7.40, The SET Statement), the ALPHABET clause specifies a collating sequence. When alphabet-name is referenced in a CODE-SET clause in a FILE-CONTROL entry (See 5.4.1, The FILE-CONTROL Paragraph.), or a SET statement with a CODE-SET phrase (See 7.40, The SET Statement.), 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 X3.4-1968, Code for Information Interchange. If the STANDARD-2 phrase is specified, the character code set identified is the International Reference Version of the ISO 7-bit code defined in International Standard 646, '7-Bit Coded

Withdrawn/Retired

Character Set for Information Processing Interchange'. Each character of the Standard character set is associated with its corresponding character of the native character set. The implementor defines the correspondence betweeen the characters of the Standard character set and the characters of the native character set for which there is no correspondence otherwise specified.

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

c. If the implementor-name phrase is specified, the character code set or collating sequence identified is that defined by the implementor. The implementor also defines the correspondence between characters of the character code set specified by implementor-name and the characters of the native character code set.

d. If the identifier phrase is specified, the character code set or collating sequence identified is that defined according to the following rules:

(1) The CHARACTER SIZE phrase, if specified, defines the number of bits used to represent a character. If the CHARACTER SIZE phrase is not specified, the number of bits used to represent a native character is used.

(2) The CHARACTER SIZE phrase applies only when alphabet-name is referenced in a SET statement with a CODE-SET phrase (See 7.40, The SET Statement.) Although the CHARACTER SIZE phrase may be specified in an ALPHABET clause referenced in a PROGRAM COLLATING SEQUENCE clause (See 5.3.2, The OBJECT-COMPUTER Paragraph.), a SET statement with a SORT, SORT-MERGE, MERGE or PROGRAM phrase (See 7.40, The SET Statement.), or a COLLATING SEQUENCE clause (See 7.27, The MERGE Statement and 7.41, The SORT Statement.), it has no effect on the collating sequences used within the program.

(3) The value of each elementary data item subordinate to, or referenced by, identifier specifies:

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

(b) The actual character within the native character set, if the elementary data item is alphanumeric. If the elementary data item is more than one character in length, each character, starting with the leftmost character, is assigned a successive ascending position in the collating sequence being specified or in the character code set used to represent the data.

(4) The ascending order of the elementary data items subordinate to, or referenced by, identifier specifies the ordinal number of the character:

Withdrawn/Retree

(a) Within the collating sequence being specified.

(b) Within the character code set that is used to represent the data. In this case, the ordinal number identifies the relative position of the character in the character code set being specified.

(5) The implementor defines any character code conversion that occurs during an input or output operation for those characters, of either the native character code set or the character code set used to represent data, that are not specified by the contents of identifier.

(6) Any characters within the native collating sequence, which are not explicitly specified by the contents of identifier, 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.

(7) The contents of identifier must not specify a given native character more than once if the ALPHABET clause is referenced in the PROGRAM COLLATING SEQUENCE clause or is associated with a file that is opened in either the output or I-O mode.

(8) The contents of identifier may specify a given native character more than once if the ALPHABET clause is associated only with an input file.

(9) If the contents of the data item referenced by identifier are changed, an appropriate SET statement must be executed prior to the changed contents of the data item referenced by identifier having any effect on the collating sequence or character code conversion. If the contents of the data item referenced by identifier are changed, and an appropriate SET statement is not executed prior to such use of the contents of the data item referenced by identifier, the results are defined by the last executed SET statement which referenced the alphabet-name associated with that identifier.

e. If the literal phrase is specified, the character code set or collating sequence identified is that defined according to the following rules:

(1) 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.

(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 or in the character code set used to represent the data.

Withdrawn/Retiree

(2) The order in which the literals appear in the ALPHABET clause specifies, in ascending sequence, the ordinal number of the character:

(a) Within the collating sequence being specified.

(b) Within the character code set that is used to represent both label and data records. In this case, the ordinal number identifies the relative position of the character in the character code set being specified.

(3) The implementor defines any character code conversion that occurs during an input or output operation for those characters, of either the native character code set or the character code set used to represent data, that are not specified in the literal phrase.

(4) 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.

(5) 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 or in the character code set used to represent the data. 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.

(6) If the ALSO phrase is specified, the characters of the native character set specified by the value of literal-1, literal-3, literal-4, . . ., are assigned to the same position in the collating sequence being specified or in the character code set that is used to represent the data.

(7) The CHARACTER SIZE phrase, if specified, defines the number of bits used to represent a character. If the CHARACTER SIZE phrase is not specified, the number of bits used to represent a native character is used.

(8) The CHARACTER SIZE phrase applies only when alphabet-name is referenced in a SET statement with a CODE-SET phrase (See 7.40, The SET Statement.). Although the CHARACTER SIZE phrase may be specified in an ALPHABET clause referenced in a PROGRAM COLLATING SEQUENCE clause (See 5.3.2, The OBJECT-COMPUTER Paragraph.), a SET statement with a SORT, SORT-MERGE, MERGE or PROGRAM phrase (See 7.40, The SET Statement.) or a COLLATING SEQUENCE clause (See 7.27, The MERGE Statement and 7.41, The SORT Statement.), it has no effect on the collating sequences used within the program.

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

Withdrawn/Retired

position in the program collating sequence, the last character specified is associated with the figurative constant HIGH-VALUE.

5. The character that has the lowest ordinal position in the program collating sequence in effect 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.

6. The literal which appears in the CURRENCY SIGN IS literal 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 any character from the computer's character set except one of the following:

   a. digits '0' through '9';

   b. alphabetic characters 'A', 'B', 'C', 'D', 'L', 'P', 'R', 'S', 'V', 'X', 'Z' or the space;

   c. special characters '*', '+', '—', ',', '.', ';', '(', ')', '"', '=', '/'.

   If this clause is not present, only the currency sign is used in the PICTURE clause.

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

Withdrawn/Retirée

5.4 INPUT-OUTPUT SECTION

5.4.1 THE FILE-CONTROL PARAGRAPH

5.4.1.1 Function

The FILE-CONTROL paragraph names each file and allows specification of other file-related information.

5.4.1.2 General Format

```
FILE-CONTROL.

{SELECT      [OPTIONAL]    file-name
        ASSIGN TO {, implementor-name-1} . . .
      [ ; RESERVE      integer-1 [AREA ]]
                                 [AREAS]
      [ ; [ORGANIZATION IS]          {SEQUENTIAL}]
                                     {RELATIVE  }
                                     {INDEXED   }
      [ ; BLOCK CONTAINS [integer-2 TO]      integer-3      {RECORDS   }]
                                                            {CHARACTERS}
      [; CODE-SET IS alphabet-name]
      [ ; PADDING CHARACTER IS          {literal-1  }]
                                        {data-name-1}
      [ ; RECORD DELIMITER IS implementor-name-2] } . . . .
```

5.4.1.3 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 described in the Data Division must be named once and only once as file-name in the FILE-CONTROL paragraph. Each file specified in the SELECT clause must have a File Description entry or a Sort-Merge File Description entry in the Data Division.

3. If file-name represents a sort or merge file, only the ASSIGN clause is permitted to follow file-name in the FILE-CONTROL paragraph.

Withdrawn/Retirée

4. The OPTIONAL clause may only be specified for input files. Its specification is required for input files that are not necessarily present each time the object program is executed.

5. 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.

6. Literal-1 must be a one character nonnumeric literal.

7. Data-name-1 may be qualified.

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

9. The RECORD DELIMITER clause may only be specified for variable-length records.

5.4.1.4 General Rules

1. The ASSIGN clause specifies the association of the file referenced by file-name to a storage medium.

2. The RESERVE clause allows the user to specify the number of input-output areas allocated. If the RESERVE clause is specified, the number of input-output areas allocated is equal to the value of integer-1. If the RESERVE clause is not specified the number of input-output areas allocated is specified by the implementor.

3. The ORGANIZATION clause specifies the logical structure of a file. The file organization is established at the time a file is created and cannot subsequently be changed.

4. If the ORGANIZATION clause is not specified, sequential organization is implied.

5. The BLOCK CONTAINS clause specifies the size of a physical record. This clause is required except when one or more of the following conditions exist:

   a. A physical record contains one and only one complete logical record.

   b. The hardware device assigned to the file has one and only one physical record size.

   c. The hardware device assigned to the file has more than one physical record size but the implementor has designated one as standard. In this case, the absence of this clause denotes the standard physical record size.

6. The size of a physical record may be stated in terms of records unless one or more of the following situations exists, in which case the RECORDS phrase must not be used:

Withdrawn/Retirée

a. In mass storage files, where logical records may extend across physical records.

b. The physical record contains padding (area not contained in a logical record).

c. Logical records are grouped in such a manner that an inaccurate physical record size would be implied.

7. 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.

8. If integer-2 is not specified, integer-3 represents the exact size of the physical record. If integer-2 and integer-3 are both specified, they refer to the minimum and maximum size of the physical record, respectively.

9. If the CODE-SET clause is specified, alphabet-name specifies the character code convention used to represent data on the external media. It also specifies the algorithm for converting the character codes on the external media from/to the native character codes. This code conversion occurs during the execution of an input or output operation. (See 5.3.3, The SPECIAL-NAMES Paragraph.)

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

11. The code convention and conversion algorithm may be changed during the execution of the object program. (See 7.40, The SET Statement.)

12. If the CODE SET clause is specified for the file, conversion of the character specified by literal-1 or the contents of data-name-1 takes place during the input-output operation.

13. The PADDING CHARACTER clause specifies the character which is to be used for block padding on sequential files. During input operations, any portion of a block which exists beyond the last logical record and consists entirely of padding characters will be bypassed. During input operations, a logical record which consists solely of padding characters will be ignored. During output operations, any portion of a block which exists beyond the last logical record will be filled entirely with padding characters.

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

15. The value of the data item referenced by data-name-1, if specified at the time an OPEN statement is executed for the file, will be used as the value of the padding character.

16. If the PADDING CHARACTER clause is not specified, the value used for the padding character will be defined by the implementor.

Withdrawn/Retirée

17. The RECORD DELIMITER clause is used to indicate the implementor-defined method of determining the length of a variable-length record on the external medium. Any delimiting method used by the implementor will not be reflected in the record area or the record size used within the program.

If the RECORD DELIMITER clause is not specified, the method of determining the length of a variable-length record is undefined.

Withdrawn/Retirée

I-O-CONTROL

5.4.2 THE I-O-CONTROL PARAGRAPH

5.4.2.1 Function

The I-O-CONTROL paragraph specifies the input-output techniques, the points at which rerun is to be established, the memory area which is to be shared by different files, and the location of files on a multiple-file reel.

5.4.2.2 General Format

```
I-O-CONTROL.
[[; APPLY input-output-technique ON file-name-1 [, file-name-2] . . .] . . .

[; RERUN     [ ON { file-name-3      } ]
                  { implementor-name }

                                  { REEL   }
                  [ END OF ]      { UNIT   }          OF file-name-4
                                  {        }
                  integer-1       RECORDS
          EVERY
                  integer-2       CLOCK-UNITS
                  condition-name                                        ] . . .

[; SAME [ RECORD ]
        [ SORT   ]  AREA FOR file-name-5 [, file-name-6] . . . ] . . .
        [ SORT-MERGE ]

[; MULTIPLE FILE TAPE CONTAINS { , { file-name-8      } }
                               {   { pseudo-file-name-1 }

        [ POSITION IS { integer-3    } ] } . . . ] . . . .]
                      { identifier-1 }
                      { LAST         }
```

Withdrawn/Retirée

5.4.2.3 Syntax Rules

1. The I-O-CONTROL paragraph is optional.

2. Contradictory input-output-techniques, as defined by the implementor, must not be specified for the same file.

3. The order of appearance of the clauses is immaterial.

4. A file-name that represents a sort or merge file cannot appear in a RERUN clause or a MULTIPLE FILE clause.

5. The END OF UNIT clause may only be used if file-name-4 is a sequentially-accessed mass storage file. The definition of UNIT is determined by each implementor.

6. When either the integer-1 RECORDS clause or the integer-2 CLOCK-UNITS clause is specified, implementor-name must be given in the RERUN clause.

7. More than one RERUN clause may be specified for a given file-name-4, subject to the following restrictions:

   a. When multiple integer-1 RECORDS clauses are specified, no two of them may specify the same file-name-4.

   b. When multiple END OF REEL or END OF UNIT clauses are specified, no two of them may specify the same file-name-4.

8. Only one RERUN clause containing the CLOCK-UNITS clause may be specified.

9. In the SAME AREA clause, SORT and SORT-MERGE are equivalent.

10. A file-name that represents a sort or merge file must not appear in the SAME clause unless the SORT, SORT-MERGE or RECORD clause is used.

11. The four forms of the SAME clause (SAME AREA, SAME RECORD AREA, SAME SORT AREA, SAME SORT-MERGE AREA) are considered separately in the following:

    More than one SAME clause may be included in a program, however:

    a. A file-name must not appear in more than one SAME AREA clause.

    b. A file-name must not appear in more than one SAME RECORD AREA clause.

    c. A file-name that represents a sort or merge file must not appear in more than one SAME SORT AREA or SAME SORT-MERGE AREA clause.

Withdrawn/Retirée

d. If one or more file-names of a SAME AREA clause appear in a SAME RECORD AREA clause, all of the file-names in that SAME AREA clause must appear in the SAME RECORD AREA clause. However, additional file-names not appearing in that SAME AREA clause may also appear in that SAME RECORD AREA clause. The rule that only one of the files mentioned in a SAME AREA clause can be open at any given time takes precedence over the rule that all files mentioned in a SAME RECORD AREA clause can be open at any given time.

e. If a file-name that does not represent a sort or merge file appears in a SAME AREA clause and one or more SAME SORT AREA or SAME SORT-MERGE AREA clauses, all of the files named in that SAME AREA clause must be named in that SAME SORT AREA or SAME SORT-MERGE AREA clause(s).

12. Each file-name specified in a MULTIPLE FILE TAPE clause must have an associated File Description entry. A pseudo-file-name may specify the position of a file within a series of files on a multiple file tape if that file is not explicitly referenced in any other clause or statement within the program.

13. The same file-name must not be specified more than once in a MULTIPLE FILE TAPE clause. The same file-name must not be specified in more than one MULTIPLE FILE TAPE clause.

14. Identifier-1 must reference an unsigned integer numeric data item.

15. If LAST is specified in the POSITION phrase, then omitted labels may not be specified as the labelling convention.

5.4.2.4 General Rules

1. When the implementor furnishes more than one input-output system technique, the APPLY input-output technique permits the user to select the appropriate technique for his object program. Specific names to designate the individual input-output technique are specified by the implementor.

2. The RERUN clause specifies when and where the rerun information is recorded. Rerun information is recorded in the following ways:

   a. If file-name-3 is specified, the rerun information is written on each reel or unit of an output file and the implementor specifies where, on the reel or file, the rerun information is to be recorded.

   b. If implementor-name is specified, the rerun information is written as a separate file on a device specified by the implementor.

3. There are nine forms of the RERUN clause, based on the several conditions under which rerun points can be established.

Withdrawn/Retirée

a. When either the END OF REEL or END OF UNIT clause is used without the ON clause. In this case, the rerun information is written on the file referenced by file-name-4, which must be an output file.

b. When either the END OF REEL or END OF UNIT clause is used and file-name-3 is specified in the ON clause. In this case, the rerun information is written on the file referenced by file-name-3, which must be an output file. In addition, normal reel-closing or unit-closing functions are performed for the file referenced by file-name-4. File-name-4 may reference either an input or an output file.

c. When either the END OF REEL or END OF UNIT clause is used and implementor-name is specified in the ON clause. In this case, the rerun information is written on a separate rerun unit defined by the implementor. File-name-4 may reference either an input or output file.

d. When the integer-1 RECORDS clause is used and implementor-name is specified in the ON clause. In this case, the rerun information is written on the device specified by implementor-name whenever integer-1 records of the file referenced by file-name-4 have been processed. File-name-4 may reference either an input or output file.

e. When the integer-1 RECORDS clause is used and file-name-3 is specified in the ON clause. In this case, the rerun information is written on the file referenced by file-name-3 whenever integer-1 records of the file referenced by file-name-4 have been processed. File-name-3 must reference an output file and file-name-4 may reference either an input or an output file.

f. When the integer-2 CLOCK-UNITS clause is used and implementor-name is specified in the ON clause. In this case, the rerun information is written on the device specified by implementor-name whenever an interval of time, calculated by an internal clock, has elasped.

g. When the integer-2 CLOCK-UNITS clause is used and file-name-3 is specified in the ON clause. In this case, the rerun information is written on the file referenced by file-name-3 whenever an interval of time, calculated by an internal clock, has elapsed. File-name-3 must reference an output file.

h. When the condition-name clause is used and implementor-name is specified in the ON clause. In this case, the rerun information is written on the device specified by implementor-name whenever a switch assumes a particular status as specified by condition-name. In this case, the associated switch must be defined in the SPECIAL-NAMES paragraph of the Configuration Section of the Environment Division. The implementor specifies when the switch status is interrogated.

i. When the condition-name clause is used and file-name-3 is specified in the ON clause. In this case, the rerun information is written on the file referenced by file-name-3 whenever a switch assumes a particular status as specified by condition-name. File-name-3 must reference an output file. As in the case described in paragraph h above, the associated switch must be defined in the SPECIAL-NAMES paragraph of the Configuration Section of the Environment Division. The implementor specifies when the switch status is interrogated.

Withdrawn/Retired

4. If a rerun condition is caused as the result of the recording of rerun information, that rerun condition is ignored.

5. The SAME AREA clause specifies that two or more files that do not represent sort or merge files are to use the same memory area during processing. The area being shared includes all storage areas (including alternate areas) assigned to the files specified; therefore, it is not valid to have more than one of the files open at the same time. (See 5.4.2.3, Syntax Rule 11.d.)

6. 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 open at the same time. A logical record in the SAME RECORD AREA is considered as a logical record of each opened output file whose file-name appears in this SAME RECORD AREA clause and of the most recently read input file 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.

7. If the SAME SORT AREA or SAME SORT-MERGE AREA clause is used, at least one of the file-names must represent a sort or merge file. Files that do not represent sort or merge files may also be named in the clause. This clause specifies that storage is shared as follows:

   a. The SAME SORT AREA or SAME SORT-MERGE AREA clause specifies a memory area which will be made available for use in sorting or merging each sort or merge file named. Thus any memory area allocated for the sorting or merging of a sort or merge file is available for reuse in sorting or merging any of the other sort or merge files.

   b. In addition, storage areas assigned to files that do not represent sort or merge files may be allocated as needed for sorting or merging the sort or merge files named in the SAME SORT AREA or SAME SORT-MERGE AREA clause. The extent of such allocation will be specified by the implementor.

   c. Files other than sort or merge files do not share the same storage area with each other. If the user wishes these files to share the same storage area with each other, he must also include in the program a SAME AREA or SAME RECORD AREA clause naming these files.

   d. During the execution of a SORT or MERGE statement that refers to a sort or merge file named in this clause, any non sort or merge files named in this clause must not be open.

8. The MULTIPLE FILE TAPE clause is required when the program accesses a file that shares the same physical reel(s) of tape with other files, unless the relative positions of files on a multiple file reel(s) are specified by some means external to the program. It is not required that all of the files named in the clause be contained on the same physical reel of tape; one or more of the files may be contained on the second or subsequent reel of a series of physical reels. Further, any file may be continued from one reel to the next reel in a series of physical reels.

   a. Regardless of the number of files on a reel(s) only file-names for those files that are explicitly referenced in an OPEN statement within the program need to be specified.

Withdrawn/Retirée

b. Each file in a series of files sharing the same physical reel(s) of tape must be created with a uniform labeling convention.

c. Each occurrence of a file-name or pseudo-file-name in a MULTIPLE FILE TAPE clause specifies the ordinal position of a file within the series of files sharing the same reel(s). The position of the file is specified either by the position of the file-name or pseudo-file-name within the sequence of file-names and/or pseudo-file-names or by an associated POSITION phrase.

d. The POSITION phrase specifies the position of a file within a series of files. The first file on the first physical reel is assigned position 1, the second file in the series of files is assigned position 2, etc. Integer or the contents of the data item referenced by identifier-1 specifies the ordinal position of the file relative to the first file on the first, or only, physical reel.

e. The POSITION phrase must be specified for each occurrence of file-name or pseudo-file-name if:

(1) A file-name or pseudo-file-name is not specified for all files in the series of files sharing the reel(s), or

(2) The sequence of file-names and/or pseudo-file-names is not specified in consecutive ordinal order corresponding to the physical placement of the files on the reel(s).

f. The POSITION phrase need not be specified if the sequence of file-names and/or pseudo-file-names includes reference to all files in the series of files in consecutive ordinal order corresponding to the physical placement of the files on the reel(s).

g. LAST specifies the ordinal position of the file as being one greater than the number of files existing on the reel(s) when the file is in the output mode. If the file is in the input or extend modes, LAST specifies the ordinal position of the file as being equal to the number of files existing on the reel(s).

9. Whenever a set of files resides on a multiple file reel(s), the following rules apply:

a. No more than one of the files may be in the open mode at one time.

b. There is no constraint on the order in which files may be opened in the input mode.

c. When one of the files referenced by a file-name-8 is the subject of a DELETE FILE statement or an OPEN statement with the OUTPUT or EXTEND phrase, then all files on the associated multiple file reel(s) whose position numbers are less than the position number of that file must already exist on the reel(s) at the time the DELETE FILE or OPEN statement is executed. Further, no file on that multiple file reel(s) whose position number is greater than the position number of that file can exist at that time on the reel(s).

d. Each of the files must be a sequential file.

Withdrawn/Retired

CHAPTER 6

THE DATA DIVISION

6.1 GENERAL DESCRIPTION

6.1.1 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. Data to be processed falls into four categories:

a. That which is contained in a data base and enters or leaves the internal memory of the computer from a specified realm or realms.

b. That which is contained in files and enters or leaves the internal memory of the computer from a specified input or output device.

c. That which is developed internally and placed into intermediate or working storage, or placed into specific format for output reporting purposes.

d. Constants which are defined by the user.

6.1.2 PHYSICAL AND LOGICAL ASPECTS OF DATA DESCRIPTION

6.1.2.1 Data Division Organization

The Data Division, which is one of the required divisions in a program, is subdivided into sections. These are the Sub-schema, File, Working-Storage, Linkage, Communication, and Report Sections.

The Sub-schema Section names the Data Base Schema and Sub-schema descriptions to be accessed by the 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, or by a File Description entry and one or more Report Description entries. Record Description entries are written immediately following the File Description entry. When the File Description entry specifies a file to be used as a Report Writer output file, only Record Description entries which define label records associated with the report file 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 subordinate 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.

Withdrawn/Retirée

GENERAL DESCRIPTION

The Communication Section describes the data item in the source program that will serve as the interface between the MCS and the program.

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

6.1.2.2 Data Division 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. The structure is illustrated in Figure 6.1-1, Data Division Structure.

6.1.2.2.1 General Format

```
DATA DIVISION.
[SUB-SCHEMA SECTION.
    [sub-schema-entry]]

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

[WORKING-STORAGE SECTION.
    [record-description-entry]    . . . ]

[LINKAGE SECTION.
    [record-description-entry]    . . . ]

[COMMUNICATION SECTION.
    [communication-description-entry    [record-description-entry] . . . ] . . . ]

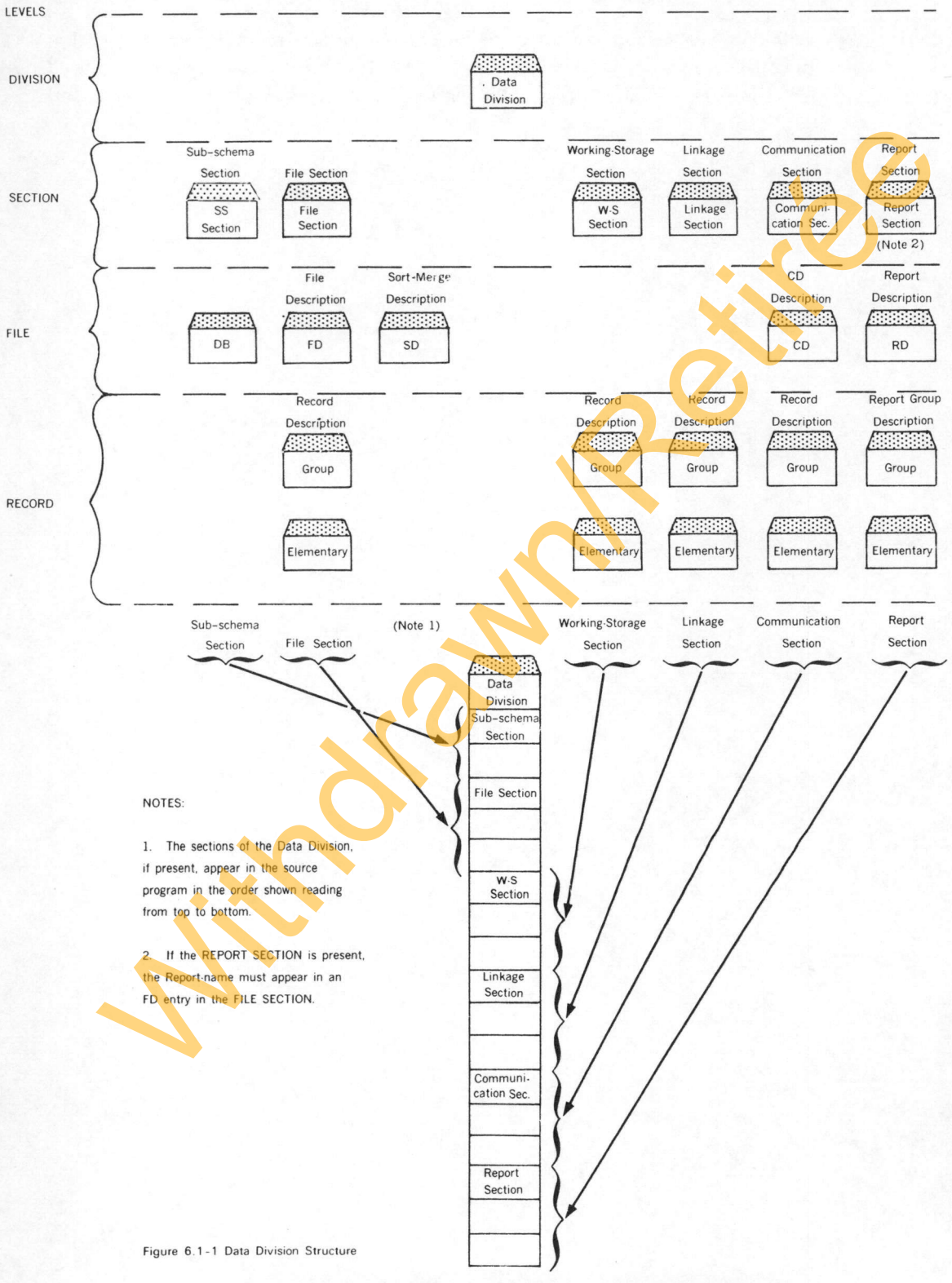
[REPORT SECTION.
    [report-description-entry    {report-group-description-entry} . . . ] . . . ]
```

Withdrawn/Retirée

NOTE: Within a file-description-entry the record-description-entry is required if the DATA RECORDS clause is specified, or if neither the DATA RECORDS clause nor the REPORT clause is specified. (See 6.10, The File Description — Complete Entry Skeleton.)

Withdrawn/Retirée

GENERAL DESCRIPTION



NOTES:

1. The sections of the Data Division, if present, appear in the source program in the order shown reading from top to bottom.

2. If the REPORT SECTION is present, the Report-name must appear in an FD entry in the FILE SECTION.

Figure 6.1-1 Data Division Structure

Withdrawn/Retired

6.2 SUB-SCHEMA SECTION

In the Sub-schema Section there is only one entry, the Sub-schema (DB) entry. It specifies the Data Base Schema and Sub-schema descriptions to be accessed by the program. Only one DB entry can appear in a given program.

Withdrawn/Retirée

6.3 FILE SECTION

In a COBOL program the File Description entry (FD) and the Sort-Merge File Description entry (SD) represent the highest level of their respective organizations in the File Section. The Sort-Merge File Description (SD) is a special type of file description. The File Section header is followed by a File Description entry consisting of a level indicator (FD, or SD), a data-name and a series of independent clauses. The FD clauses specify the manner in which data is recorded on the file, the size of the logical and physical records, the names of the label records contained in the file and values of label items, the names of the data records or reports 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.

An SD File Description gives information about the size and the names of the data records associated with the file to be sorted or merged. 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.

Withdrawn/Retired

6.4 WORKING-STORAGE SECTION

The Working-Storage Section is composed of the section header, followed by Record Description entries.

6.4.1 WORKING-STORAGE RECORDS

Data elements in Working-Storage which bear a definite hierarchic 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.

6.4.2 INITIAL VALUES

The initial value of any item in the Working-Storage Section except an index data item or a data base key item is specified by using the VALUE clause with the data item. The initial value of any index data item or of any data base key item is unpredictable.

Withdrawn/Retirée

6.5 LINKAGE SECTION

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. No space is allocated in the program for data items referenced by data-names in the Linkage Section of that program. Procedure Division references to these data items are resolved at object time by equating the reference in the called program to the location used in the calling program. In the case of index-names, no such correspondence is established. Index-names in the called and calling program always refer to separate indices.

Data items defined in the Linkage Section of the called program may be referenced within the Procedure Division of the called program if and only if they are:

1. Operands of the USING phrase of the Procedure Division header.
2. Subordinate to operands of the USING phrase of the Procedure Division header.
3. Defined with a REDEFINES or RENAMES clause, the object of which is an operand of the USING phrase of the Procedure Division header.
4. Items subordinate to any of the items defined in paragraph 3 above.
5. Condition-names and index-names associated with data items that meet any of the above conditions.

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 Record Description entries.

Data items defined in the Linkage Section of the called program must not be associated with data items defined in the Report Section of the calling program.

6.5.1 LINKAGE RECORDS

Data elements in the Linkage Section which bear a definite hierarchic 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.

6.5.2 INITIAL VALUES

The VALUE clause must not be specified in the Linkage Section except in condition-name entries (level 88).

Withdrawn/Retiree

6.6 COMMUNICATION SECTION

In a COBOL program the Communication Description entries (CD) represent the highest level of organization in the Communication Section. The Communication Section header is followed by a Communication Description entry consisting of a level indicator (CD), a data-name and a series of independent clauses. These clauses indicate the queues and sub-queues, the message date and time, the source, the text length, the status and end keys, and message count. These clauses specify the destination count, the text length, the status and error keys, and destinations for output. The entry itself is terminated by a period. These record areas may be implicitly redefined by user-specified Record Description entries following the various Communication Description clauses.

Withdrawn/Retiree

6.7 REPORT SECTION

In the Report Section the description of each report must begin with a Report Description entry (RD entry) and be followed by the entries that describe the report groups within the report.

6.7.1 REPORT DESCRIPTION ENTRY

In addition to naming the report, the 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 RD 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 FD entry in the File Section must be the subject of an RD entry in the Report Section. Furthermore each report in the Report Section must be named in one and only one FD entry.

6.7.2 REPORT GROUP DESCRIPTION ENTRY

The report groups that will comprise the report are described following the RD 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.

Withdrawn/Retirée

6.8 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 under the appropriate paragraphs. (See 3.3.2, Concept of Levels; 6.14, The Data Description — Complete Entry Skeleton; and Section IV, 4.6, The Record Description — Complete Entry Skeleton.)

Withdrawn/Retired

SS SKELETON

6.9 THE SUB-SCHEMA ENTRY — COMPLETE ENTRY SKELETON

6.9.1 FUNCTION

The Sub-schema entry specifies the Sub-schema in the Sub-schema Library that is to be accessed by the program.

6.9.2 GENERAL FORMAT

DB sub-schema-name WITHIN schema-name

[; PRIVACY KEY IS literal].

6.9.3 SYNTAX RULES

1. The PRIVACY KEY clause must be specified if the Sub-schema referenced by sub-schema-name is defined to have an invocation privacy lock associated with it.

2. Literal must be alphanumeric.

6.9.4 GENERAL RULES

1. The Sub-schema referenced by sub-schema-name, which describes data in the data base described by the Schema referenced by Schema-name, is made available to the program, subject to General Rule 2.

2. The value of literal is matched to the value of the invocation privacy lock associated with the Sub-schema referenced by sub-schema-name, according to the rules for comparison of alphanumeric operands in a relation condition. If the value of literal is equal to the value of the invocation privacy lock, the Sub-schema referenced by sub-schema-name is made available to the program. If the values are not equal, the Sub-schema is not made available to the program.

3. During the execution of a run unit that contains a program that specifies a DB entry, no other program in that run unit may specify a DB entry.

Withdrawn/Retirée

SEAPORT90

Begin New File

SEAPORT IMAGING

Ensign™

Scanner Type: ALL

Function: Begin New File

This is used only with multi-image file formats.

Withdrawn/Retirée

6.10 THE FILE DESCRIPTION — COMPLETE ENTRY SKELETON

6.10.1 FUNCTION

The File Description furnishes information concerning the physical structure, identification, and record names or report names pertaining to a given file.

6.10.2 GENERAL FORMAT

Format 1

```
FD  file-name

[ ; RECORD  { CONTAINS integer-1 CHARACTERS
            { IS VARYING IN SIZE [ [FROM integer-2] [TO integer-3]
            {     CHARACTERS ]   [DEPENDING ON data-name-1]                 } ]

[ ; LABEL   [ RECORDS ARE ]   { STANDARD                                    } ]
            [ RECORD IS   ]   { OMITTED                                     }
                              { , data-name-2 } . . .                       }

[ ; VALUE OF { , data-name-3 IS { data-name-4 } } . . . ]
                                { literal-1   }

[ ; DATA    [ RECORD IS   ]   { , data-name-5 } . . . ]
            [ RECORDS ARE ]

[; FILE STATUS IS data-name-6]

[;[ACCESS MODE IS]  SEQUENTIAL ]

[ ; LINAGE IS { data-name-10 } LINES [ , WITH FOOTING AT { data-name-11 } ]
              { integer-4    }                           { integer-5    }

      [ , LINES AT TOP { data-name-12 } ] [ , LINES AT BOTTOM { data-name-13 } ] ].
                       { integer-6    }                      { integer-7    }
```

Withdrawn/Retirée

FD SKELETON

Format 2

```
FD  file-name

[ ; LABEL      [RECORDS ARE]      { STANDARD }
               [RECORD IS ]       { OMITTED  }
                                  {, data-name-2} . . . ]

[ ; VALUE OF   { , data-name-3  IS  { data-name-4 } } . . . ]
               {                    { literal-1   } }

[; FILE STATUS IS data-name-6]

[;[ACCESS MODE IS] SEQUENTIAL]

    { REPORT  IS }     {, report-name-1} . . .
    { REPORTS ARE}
```

Format 3

```
FD  file-name

[; RECORD CONTAINS integer-1 CHARACTERS]

[ ; LABEL      [RECORDS ARE]      { STANDARD }
               [RECORD IS ]       { OMITTED  }
                                  {, data-name-2} . . . ]

[ ; VALUE OF   { , data-name-3  IS  { data-name-4 } } . . . ]
               {                    { literal-1   } }

[ ; DATA       [RECORD IS ]       {, data-name-5} . . . ]
               [RECORDS ARE]

[; FILE STATUS IS data-name-6]

[ ; [ACCESS MODE IS]  { SEQUENTIAL  [, RELATIVE KEY IS data-name-7] }
                      { RANDOM      , RELATIVE KEY IS data-name-7   }
                      { DYNAMIC                                     } ]
```

Withdrawn/Retiree

FD SKELETON

Format 4

```
FD  file-name

[ ; RECORD  { CONTAINS  integer-2  CHARACTERS                                          } ]
            { IS VARYING IN SIZE  [[FROM integer-2]  [TO integer-3]                    }
              [ CHARACTERS ]      [DEPENDING ON  data-name-1]                          }

[ ; LABEL   [RECORDS ARE]   { STANDARD                                                 } ]
            [RECORD IS  ]   { OMITTED                                                  }
                            { {, data-name-2} . . .                                    }

[ ; VALUE OF  { , data-name-3  IS  { data-name-4 } } . . . ]
              {                    { literal-1   } }

[ ; DATA  [RECORDS IS]  {, data-name-5} . . . ]
          [RECORDS ARE]

[; FILE STATUS IS data-name-6]

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

  ; RECORD KEY IS  data-name-8

[; ALTERNATE RECORD KEY IS  data-name-9  [WITH DUPLICATES]]  . . .
```

Withdrawn/Retiree

FD SKELETON

6.10.3 SYNTAX RULES

ALL FORMATS

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

2. The clauses which follow the name of the file may appear in any order.

FORMATS 1 AND 2

3. File-name may only reference a sequential file.

FORMATS 1, 3 AND 4

4. One or more Record Description entries must follow the File Description entry.

FORMAT 2

5. Each report named in the REPORT clause must be the subject of a Report Description entry in the Report Section.

6. If the REPORT clause is specified, no Record Description entries which define data records may follow the File Description entry. Record Description entries defining label records associated with a report file may follow the File Description entry.

FORMAT 3

7. File-name may only reference a relative file.

8. If a relative file is referenced by a START statement, the RELATIVE KEY clause must be specified for that file.

FORMAT 4

9. File-name may only reference an indexed file.

10. Data-name-9 must not reference an item whose leftmost character position corresponds to the leftmost character position of an item referenced by data-name-8 or by any other data-name-9 associated with this file.

Withdrawn/Retirée

6.11 THE SORT-MERGE FILE DESCRIPTION — COMPLETE ENTRY SKELETON

6.11.1 FUNCTION

The Sort-Merge File Description furnishes information concerning the physical structure, identification, and record names of the file to be sorted or merged.

6.11.2 GENERAL FORMAT

```
SD  file-name

[ ; RECORD { CONTAINS integer-1 CHARACTERS                                              }
           { IS VARYING IN SIZE [ [FROM integer-2] [TO integer-3]                       }
                CHARACTERS ]    [DEPENDING ON data-name-1]                              } ]

[ ; DATA { RECORD IS  } data-name-2 [, data-name-3] . . . ] .
         { RECORDS ARE }
```

6.11.3 SYNTAX RULES

1. The level indicator SD identifies the beginning of the Sort-Merge File Description and must precede the file-name.

2. The clauses which follow the name of the file 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 file.

Withdrawn/Retiree

CD SKELETON

6.12 THE COMMUNICATION DESCRIPTION — COMPLETE ENTRY SKELETON

6.12.1 FUNCTION

The Communication Description specifies the interface area between the MCS and a COBOL program.

6.12.2 GENERAL FORMAT

Format 1

```
CD  cd-name;        FOR    [INITIAL]    INPUT
    [
      [ ; SYMBOLIC QUEUE IS data-name-1 ]
          [ ; SYMBOLIC SUB-QUEUE-1     IS  data-name-2 ]
          [ ; SYMBOLIC SUB-QUEUE-2     IS  data-name-3 ]
          [ ; SYMBOLIC SUB-QUEUE-3     IS  data-name-4 ]
          [ ; MESSAGE DATE             IS  data-name-5 ]
          [ ; MESSAGE TIME             IS  data-name-6 ]
          [ ; SYMBOLIC SOURCE          IS  data-name-7 ]
          [ ; TEXT LENGTH              IS  data-name-8 ]
          [ ; END KEY                  IS  data-name-9 ]
          [ ; STATUS KEY               IS  data-name-10 ]
          [ ; MESSAGE COUNT            IS  data-name-11 ]
      [ data-name-1 , data-name-2 , . . . , data-name-11 ]
    ]
```

Withdrawn/Retirée

Format 2

```
CD   cd-name;    FOR    OUTPUT
  [ ; DESTINATION COUNT          IS        data-name-1 ]
  [ ; TEXT LENGTH                IS        data-name-2 ]
  [ ; STATUS KEY                 IS        data-name-3 ]
    [ ; DESTINATION TABLE OCCURS     integer-2        TIMES
      [ ; INDEXED BY index-name-1    [, index-name-2]  . . .]]
  [ ; ERROR KEY                  IS        data-name-4 ]
  [ ; SYMBOLIC DESTINATION       IS        data-name-5 ]
```

6.12.3 SYNTAX RULES

FORMAT 1

1. A CD must appear only in the Communication Section.

2. Within a single program, the INITIAL clause may be specified in only one CD. The INITIAL clause must not be used in a program that specifies the USING phrase of the Procedure Division header. (See 7.1.4.1, Procedure Division Header.)

3. Except for the INITIAL clause, the optional clauses may be written in any order.

4. If neither option in the format is specified, a level 01 Data Description entry must follow the CD Description entry. Either option may be followed by a level 01 Data Description entry.

5. Record Description entries following an input CD implicitly redefine the record area established by the input CD and must describe a record of exactly 87 Standard Data Format characters. Multiple redefinitions of this record are permitted; however, only the first redefinition may contain VALUE clauses. The MCS will always reference the record according to the data description defined in General Rule 2. (See 6.48, The VALUE Clause.)

6. Data-name-1, data-name-2, . . ., data-name-11 must be unique within the CD. Within this series, any data-name may be replaced by the reserved word FILLER.

Withdrawn/Retirée

CD SKELETON

FORMAT 2

7. A CD must appear only in the Communication Section.

8. If none of the optional clauses of the CD is specified, a level 01 Data Description entry must follow the CD description entry.

9. Record Description entries subordinate to an output CD implicitly redefine the record area established by the output CD. Multiple redefinitions of this record are permitted; however, only the first redefinition may contain VALUE clauses. However, the MCS will always reference the record according to the data descriptions defined in General Rule 16. (See 6.48, The VALUE Clause.)

10. Data-name-1, data-name-2, . . ., data-name-5 must be unique within a CD.

11. If the DESTINATION TABLE OCCURS clause is not specified, one (1) ERROR KEY and one (1) SYMBOLIC DESTINATION area is assumed. In this case, neither subscripting nor indexing is permitted when referencing these data items.

12. If the DESTINATION TABLE OCCURS clause is specified, data-name-4 and data-name-5 may only be referred to by subscripting or indexing.

6.12.4 GENERAL RULES

FORMAT 1

1. The input CD information constitutes the communication between the MCS and the program as information about the message being handled. This information does not come from the terminal as part of the message.

2. For each input CD, a record area of 87 contiguous character positions is allocated. This record area is defined to the MCS as follows:

   a. The SYMBOLIC QUEUE clause defines data-name-1 as the name of an elementary alphanumeric data item of 12 characters occupying positions 1-12 in the record.

   b. The SYMBOLIC SUB-QUEUE-1 clause defines data-name-2 as the name of an elementary alphanumeric data item of 12 characters occupying positions 13-24 in the record.

   c. The SYMBOLIC SUB-QUEUE-2 clause defines data-name-3 as the name of an elementary alphanumeric data item of 12 characters occupying positions 25-36 in the record.

   d. The SYMBOLIC SUB-QUEUE-3 clause defines data-name-4 as the name of an elementary alphanumeric data item of 12 characters occupying positions 37-48 in the record.

Withdrawn/Retirée

e. The MESSAGE DATE clause defines data-name-5 as the name of a data item whose implicit description is that of an integer of 6 digits without an operational sign occupying character positions 49-54 in the record.

f. The MESSAGE TIME clause defines data-name-6 as the name of a data item whose implicit description is that of an integer of 8 digits without an operational sign occupying character positions 55-62 in the record.

g. The SYMBOLIC SOURCE clause defines data-name-7 as the name of an elementary alphanumeric data item of 12 characters occupying positions 63-74 in the record.

h. The TEXT LENGTH clause defines data-name-8 as the name of an elementary data item whose implicit description is that of an integer of 4 digits without an operational sign occupying character positions 75-78 in the record.

i. The END KEY clause defines data-name-9 as the name of an elementary alphanumeric data item of 1 character occupying position 79 in the record.

j. The STATUS KEY clause defines data-name-10 as the name of an elementary alphanumeric data item of 2 characters occupying positions 80-81 in the record.

k. The MESSAGE COUNT clause defines data-name-11 as the name of an elementary data item whose implicit description is that of an integer of 6 digits without an operational sign occupying character positions 82-87 in the record.

The second option may be used to replace the above clauses by a series of data-names which, taken in order, correspond to the data-names defined by these clauses.

Withdrawn/Retirée

CD SKELETON

Use of either option results in a record whose implicit description is equivalent to the following:

IMPLICIT DESCRIPTION		COMMENT
01 data-name-0.		
02 data-name-1	PICTURE X (12).	SYMBOLIC QUEUE
02 data-name-2	PICTURE X (12).	SYMBOLIC SUB-QUEUE-1
02 data-name-3	PICTURE X (12).	SYMBOLIC SUB-QUEUE-2
02 data-name-4	PICTURE X (12).	SYMBOLIC SUB-QUEUE-3
02 data-name-5	PICTURE 9 (06).	MESSAGE DATE
02 data-name-6	PICTURE 9 (08).	MESSAGE TIME
02 data-name-7	PICTURE X (12).	SYMBOLIC SOURCE
02 data-name-8	PICTURE 9 (04).	TEXT LENGTH
02 data-name-9	PICTURE X.	END KEY
02 data-name-10	PICTURE XX.	STATUS KEY
02 data-name-11	PICTURE 9 (06).	MESSAGE COUNT

In the above, the information under 'COMMENT' is for clarification and is not part of the description.

3. The contents of the data items referenced by data-name-2, data-name-3, and data-name-4, when not being used must contain spaces.

4. The data items referenced by data-name-1, data-name-2, data-name-3, and data-name-4 contain symbolic names designating queues, sub-queues, . . . respectively. All symbolic names must follow the rules for the formation of system-names, and must have been previously defined to the MCS.

5. A RECEIVE statement causes the serial return of the next message or a portion of a message from the queue as specified by the entries in the CD.

   At the time of execution of a RECEIVE statement, the input CD area must contain, in the contents of data-name-1, the name of a symbolic queue. The data items specified by data-name-2, data-name-3, and data-name-4 may contain symbolic sub-queue names or spaces. When a given level of the queue structure is specified, all higher levels must also be specified. If less than all the levels of the queue hierarchy are specified, the MCS determines the next message or portion of a message to be accessed within the queue and/or sub-queue specified in the input CD.

   After the execution of a RECEIVE statement, the contents of the data items referenced by data-name-1 through data-name-4 will contain the symbolic names of all the levels of the queue structure.

6. Whenever a program is scheduled by the MCS to process a message, the symbolic names of the queue structure that demanded this activity will be placed in the data items referenced by data-name-1 through data-name-4 of the CD associated with the INITIAL clause as applicable. In all other cases, the contents of the data items referenced by data-name-1 through data-name-4 of the CD associated with the INITIAL clause are initialized to spaces.

Withdrawn/Retired

The symbolic names are inserted or the initialization to spaces is completed prior to the execution of the first Procedure Division statement.

The execution of a subsequent RECEIVE statement naming the same contents of the data items referenced by data-name-1 through data-name-4 will return the actual message that caused the program to be scheduled. Only at that time will the remainder of the CD be updated.

7. If the MCS attempts to schedule a program lacking an INITIAL clause, the results are undefined.

8. Data-name-5 has the format 'YYMMDD' (year, month, day). Its contents represent the date on which the MCS recognizes that the message is complete.

   The contents of the data item referenced by data-name-5 are only updated by the MCS as part of the execution of a RECEIVE statement.

9. The contents of data-name-6 has the format 'HHMMSSTT' (hours, minutes, seconds, hundredths of a second) and its contents represent the time at which the MCS recognizes that the message is complete.

   The contents of the data item referenced by data-name-6 are only updated by the MCS as part of the execution of the RECEIVE statement.

10. During the execution of a RECEIVE statement, the MCS provides, in the data item referenced by data-name-7, the symbolic name of the communications terminal that is the source of the message being transferred. However, if the symbolic name of the communications terminal is not known to the MCS, the contents of the data item referenced by data-name-7 will contain spaces.

11. The MCS indicates via the contents of the data item referenced by data-name-8 the number of character positions filled as a result of the execution of the RECEIVE statement. (See 7.34, The RECEIVE Statement.)

12. The contents of the data item referenced by data-name-9 are set only by the MCS as part of the execution of a RECEIVE statement according to the following rules:

    a. When the RECEIVE MESSAGE phrase is specified, then:

       1. If an end of group has been detected, the contents of the data item referenced by data-name-9 are set to 3;

       2. If an end of message has been detected, the contents of the data item referenced by data-name-9 are set to 2;

       3. If less than a message is transferred, the contents of the data item referenced by data-name-9 are set to 0.

Withdrawn/Retirée

b. When the RECEIVE SEGMENT phrase is specified, then:

1. If an end of group has been detected, the contents of the data item referenced by data-name-9 are set to 3;

2. If an end of message has been detected, the contents of the data item referenced by data-name-9 are set to 2;

3. If an end of segment has been detected, the contents of the data item referenced by data-name-9 are set to 1;

4. If less than a message segment is transferred, the contents of the data item referenced by data-name-9 are set to 0.

c. When more than one of the above conditions is satisfied simultaneously, the rule first satisfied in the order listed determines the contents of the data item referenced by data-name-9.

13. The contents of the data item referenced by data-name-10 indicate the status condition of the previously executed RECEIVE, ACCEPT COUNT, ENABLE INPUT, or DISABLE INPUT statement.

The actual association between the contents of the data item referenced by data-name-10 and the status condition itself is defined in Figure 6.12-1.

14. The contents of the data item referenced by data-name-11 indicate the number of messages that exist in a queue, sub-queue-1, ... The MCS updates the contents of the data item referenced by data-name-11 only as part of the execution of an ACCEPT statement with the COUNT phrase.

FORMAT 2

15. The nature of the output CD information is such that it is not sent to the terminal, but constitutes the communication between the program and the MCS as information about the message being handled.

16. For each output CD, a record area of contiguous character positions is allocated according to the following formula: (10 plus 13 times integer-2). The implicit description of this record area is:

a. The DESTINATION COUNT clause defines data-name-1 as the name of a data item whose implicit description is that of an integer without an operational sign occupying character positions 1-4 in the record.

b. The TEXT LENGTH clause defines data-name-2 as the name of an elementary data item whose implicit description is that of an integer of 4 digits without an operational sign occupying character positions 5-8 in the record.

c. The STATUS KEY clause defines data-name-3 to be an elementary alphanumeric data item of 2 characters occupying positions 9-10 in the record.

Withdrawn/Retirée

d. Character positions 11-23 and every set of 13 characters thereafter will form table items of the following description:

1. The ERROR KEY clause defines data-name-4 as the name of an elementary alphanumeric data item of 1 character.

2. The SYMBOLIC DESTINATION clause defines data-name-5 as the name of an elementary alphanumeric data item of 12 characters.

Use of the above clauses results in a record whose implicit description is equivalent to the following:

IMPLICIT DESCRIPTION		COMMENT
01 data-name-0.		
02 data-name-1	PICTURE 9(04).	DESTINATION COUNT
02 data-name-2	PICTURE 9(04).	TEXT LENGTH
02 data-name-3	PICTURE XX.	STATUS KEY
02 data-name	OCCURS integer-2 TIMES.	DESTINATION TABLE
03 data-name-4	PICTURE X.	ERROR KEY
03 data-name-5	PICTURE X(12).	SYMBOLIC DESTINATION

In the above, the information under 'COMMENT' is for clarification and is not part of the description.

17. During the execution of a SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement, the contents of the data item referenced by data-name-1 will indicate to the MCS the number of symbolic destinations that are to be used from the area referenced by data-name-5.

The MCS finds the first symbolic destination in the first occurrence of the area referenced by data-name-5, the second symbolic destination in the second occurrence of the area referenced by data-name-5 . . . up to and including the occurrence of the area referenced by data-name-5 indicated by the contents of data-name-1.

If during the execution of a SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement the value of the data item referenced by data-name-1 is outside the range of 1 through integer-2, an error condition is indicated and the execution of the SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement is terminated.

18. It is the responsibility of the user to insure that the value of the data item referenced by data-name-1 is valid at the time of execution of the SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement.

19. As part of the execution of a SEND statement, the MCS will interpret the contents of the data item referenced by data-name-2 to be the user's indication of the number of leftmost character positions

Withdrawn/Retirée

of the data item referenced by the associated SEND identifier from which data is to be transferred. (See 7.39, The SEND Statement.)

20. Each occurrence of the data item referenced by data-name-5 contains a symbolic destination previously known to the MCS. These symbolic destination names must follow the rules for the formation of system-names.

21. The contents of the data item referenced by data-name-3 indicate the status condition of the previously executed SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement.

    The actual association between the contents of the data item referenced by data-name-3 and the status condition itself is defined in Figure 6.12-1.

22. If, during the execution of a SEND, PURGE, ENABLE OUTPUT, or DISABLE OUTPUT statement, the MCS determines that any specified destination is unknown, the contents of the data item referenced by data-name-3 and all occurrences of the data items referenced by data-name-4 are updated.

    The contents of the data item referenced by data-name-4 when equal to 1 indicate that the associated value in the area referenced by data-name-5 has not been previously defined to the MCS. Otherwise, the contents of the data item referenced by data-name-4 are set to zero (0).

Figure 6.12-1 indicates the possible contents of the data items referenced by data-name-10 for Format 1 and by data-name-3 for Format 2 at the completion of each statement shown. An 'X' on a line in a statement column indicates that the associated code shown for that line is possible for that statement.

Withdrawn/Retirée

RECEIVE	SEND	PURGE	ACCEPT (with COUNT)	ENABLE INPUT (without TERMINAL)	ENABLE INPUT (with TERMINAL)	ENABLE OUTPUT	DISABLE INPUT (without TERMINAL)	DISABLE INPUT (with TERMINAL)	DISABLE OUTPUT	STATUS KEY CODE	
X	X	X	X	X	X	X	X	X	X	00	No error detected. Action completed.
	X	X								10	One or more destinations are disabled. Action completed.
	X	X				X			X	20	One or more destinations unknown. Action completed for known destinations. No action taken for unknown destinations. Data-name-4 (ERROR KEY) indicates known or unknown.
X			X	X			X			20	One or more queues or subqueues unknown. No action taken.
					X			X		20	The source is unknown. No action taken.
	X	X				X			X	30	Content of DESTINATION COUNT invalid. No action taken.
				X	X	X	X	X	X	40	Password invalid. No enabling/disabling action taken.
	X									50	Character count greater than length of sending field. No action taken.
	X									60	Partial segment with either zero character count or no sending area specified. No action taken.
		X								70	One or more destinations do not have partial messages associated with them. Action completed for other destinations.

Figure 6.12-1
Communication Status Key Condition

Withdrawn/Retiree

RD SKELETON

6.13 THE REPORT DESCRIPTION — COMPLETE ENTRY SKELETON

6.13.1 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.

6.13.2 GENERAL FORMAT

```
RD report-name

    [ ; CODE literal-1 ]

[ ;  { CONTROL IS  }  { data-name-1                [ , data-name-2 ]    . . . }  ]
     { CONTROLS ARE}  { FINAL   [ , data-name-1    [ , data-name-2 ] . . . ] }

[ ;  PAGE  [ LIMIT IS  ]  integer-1  [ LINE  ]  [ , HEADING integer-2 ]
           [ LIMITS ARE]             [ LINES ]

     [ , FIRST DETAIL  integer-3 ]   [ , LAST DETAIL  integer-4 ]
     [ , FOOTING integer-5 ] ]
```

6.13.3 SYNTAX RULES

1. The report-name must appear in one and only one REPORT clause.

2. The order of appearance of the clauses following the report-name is immaterial.

3. Report-name is the highest permissible qualifier that may be specified for LINE-COUNTER, PAGE-COUNTER and all data-names defined within the Report Section.

6.13.4 PAGE-COUNTER RULES

1. PAGE-COUNTER is the reserved word used to reference a special register that is automatically created for each report specified in the Report Section. (See 3.2.2.1.3.3.1, Special Registers.)

2. In the Report Section, a reference to PAGE-COUNTER can only appear in a SOURCE clause. Outside of the Report Section, PAGE-COUNTER may be used in any context in which a data-name of integral 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.

Withdrawn/Retirée

In the Report Section an unqualified reference to PAGE-COUNTER is implicitly qualified by the name of the report in which the reference is made. Whenever the PAGE-COUNTER of a different report is referenced, PAGE-COUNTER must be explicitly qualified by that report-name.

4. Execution of the INITIATE statement causes the Report Writer Control System to set the PAGE-COUNTER of the referenced report to one (1).

5. PAGE-COUNTER is automatically incremented by one (1) each time the Report Writer Control System executes a page advance.

6. PAGE-COUNTER may be altered by Procedure Division statements.

6.13.5 LINE-COUNTER RULES

1. LINE-COUNTER is the reserved word used to reference a special register that is automatically created for each report specified in the Report Section. (See 3.2.2.1.3.3.1, Special Registers.)

2. In the Report Section a reference to LINE-COUNTER can only appear in a SOURCE clause. Outside of the Report Section, LINE-COUNTER may be used in any context in which a data-name of integral value may appear. However, only the Report Writer Control System can change the contents 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 (0). The Report Writer Control System also automatically 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 non-printable 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 6.15.5, Presentation Rules Tables.)

Withdrawn/Retirée

DATA DESCRIPTION SKELETON

6.14 THE DATA DESCRIPTION — COMPLETE ENTRY SKELETON

6.14.1 FUNCTION

A Data Description entry specifies the characteristics of a particular item of data.

6.14.2 GENERAL FORMAT

Format 1

```
level-number  [ data-name-1 ]
              [ FILLER      ]
[ ;  REDEFINES data-name-2 ]
[ ; { PICTURE } IS character-string                [ DEPENDING ON identifier ] ]
  { PIC     }
[                                                                              ]
[ ; [ USAGE IS ] { BIT                                                         } ]
[                { COMPUTATIONAL                                               } ]
[                { COMP                                                        } ]
[                { COMPUTATIONAL-n                                             } ]
[                { COMP-n                                                      } ]
[                { DB-KEY                                                      } ]
[                { DISPLAY                                                     } ]
[                { DISPLAY-n                                                   } ]
[                { INDEX                                                       } ]
[                { INDEX-n                                                     } ]
[                                                                              ]
[ ; [ SIGN IS ] { LEADING  } [ SEPARATE CHARACTER ] ]
                { TRAILING }
```

Withdrawn/Retirée

DATA DESCRIPTION SKELETON

```
[ ;   OCCURS integer-1 TIMES
          [ { ASCENDING  } KEY IS data-name-4 [ , data-name-5 ] . . . ] . . .
          [ { DESCENDING }
          [ INDEXED BY index-name-1 [ , index-name-2 ] . . . ]
  ;   OCCURS integer-2 TO integer-3 TIMES DEPENDING ON data-name-6
          [ { ASCENDING  } KEY IS data-name-4 [ , data-name-5 ] . . . ] . . .
          [ { DESCENDING }
          [ INDEXED BY index-name-1 [ , index-name-2 ] . . . ] ]

[ ;   { SYNCHRONIZED } [ LEFT  ] ]
      { SYNC         } [ RIGHT ]

[ ;   { JUSTIFIED } RIGHT ]
      { JUST      }

[ ;   BLANK WHEN ZERO ]

[ ;   CHECK IS { { literal-1    } [ { THRU    } literal-2 ] } . . . }   ]
                { { NULL       }   { THROUGH }             }
                { { routine-name }                         }

[ ;   PRIVACY LOCK [ FOR { { GET    } } IS { literal-5      } ]
                     { { MODIFY } }        { implementor-name }
                     { { STORE  } }        { routine-name-2   }

[ ;   VALUE IS literal-6 ] .
```

Withdrawn/Retiree

DATA DESCRIPTION SKELETON

Format 2

```
66 data-name-1; RENAMES data-name-2 [ { THRU      } data-name-3 ] .
                                     { THROUGH   }
```

Format 3

```
88 condition-name ; { VALUE IS  } literal-1 [ { THRU    } literal-2 ]
                    { VALUES ARE }           { THROUGH }

    [ , literal-3 [ { THRU    } literal-4 ] ] . . .
                    { THROUGH }
```

6.14.3 SYNTAX RULES

1. Level-number in Format 1 may be any number from 01 through 49.

2. The clauses may be written in any order with two exceptions: the data-name-1 or FILLER clause, if specified, must immediately follow the level-number; the REDEFINES clause, when used, must immediately follow the data-name-1 or FILLER clause, if specified.

3. The PICTURE clause must be specified for every elementary item except an index data item, data base key 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.

6.14.4 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, SYNCHRONIZED or USAGE (other than USAGE IS DISPLAY).

Withdrawn/Retirée

d. An index data item.
e. A data base key item.

3. Multiple level 01 entries subordinate to any given level indicator represent implicit redefinitions of the same area.

Withdrawn/Retirée

REPORT GROUP SKELETON

6.15 THE REPORT GROUP DESCRIPTION — COMPLETE ENTRY SKELETON

6.15.1 FUNCTION

The Report Group Description entry specifies the characteristics of a report group and of the individual items within a report group.

6.15.2 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          }

                  { REPORT HEADING }
                  { RH             }
                  { PAGE HEADING   }
                  { PH             }
                  { CONTROL HEADING }          { data-name-2 }
                  { CH              }          { FINAL       }
; TYPE IS         { DETAIL }
                  { DE     }
                  { CONTROL FOOTING }          { data-name-3 }
                  { CF              }          { FINAL       }
                  { PAGE FOOTING   }
                  { PF             }
                  { REPORT FOOTING }
                  { RF             }

[ ; [ USAGE IS ] { DISPLAY   } ]  .
                 { DISPLAY-n }
```

Withdrawn/Retiree

REPORT GROUP SKELETON

Format 2

```
level-number          [ data-name-1 ]

[ ; LINE NUMBER IS { integer-1          [ ON NEXT PAGE ] } ]
                   { PLUS integer-2                    }
[ ; [ USAGE IS ] { DISPLAY   } ] .
                 { DISPLAY-n }
```

Format 3

```
level-number          [ data-name-1 ]

[ ; BLANK WHEN ZERO ]

[ ; COLUMN NUMBER IS integer-3 ]

[ ; GROUP INDICATE ]

[ ; { JUSTIFIED } RIGHT ]
   { JUST      }

[ ; LINE NUMBER IS { integer-1          [ ON NEXT PAGE ] } ]
                   { PLUS integer-2                    }

   ; { PICTURE } IS character-string
     { PIC     }

{ ; SOURCE IS identifier-1                                                                    }
{ ; VALUE IS literal-1                                                                        }
{ ; SUM identifier-2 [ , identifier-3 ] . . . [ UPON data-name-2                              }
{                                                                                             }
{                    [ , data-name-3 ] . . . ] } . . . [ RESET ON { data-name-4 } ]           }
{                                                                 { FINAL       }           }
```

Withdrawn/Retirée

REPORT GROUP SKELETON

```
[ ; [ USAGE IS ]  { DISPLAY   } ]
                  { DISPLAY-n }
```

6.15.3 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 data 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.

Withdrawn/Retirée

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. Data-name-1 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 selected 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.

Withdrawn/Retirée

REPORT GROUP SKELETON

A blank indicates that the clause is not permitted.

CLAUSES

PERMISSIBLE COMBINATIONS

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

Figure 6.15-1
Permissible Clause Combinations
in Format 3 Entries

6.15.4 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.

6.15.5 PRESENTATION RULES TABLES

6.15.5.1 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 RWCS gives to these clauses.

Withdrawn/Retirée

6.15.5.2 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 6.15.5.8, 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 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 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 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 6.31, The LINE NUMBER Clause.)

Withdrawn/Retirée

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 RWCS places in LINE-COUNTER after presenting report groups are specified by the Final LINE-COUNTER Setting Rules.

6.15.5.3 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 consist of the two specified consecutive sequences. For example 'AR' 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).

6.15.5.4 LINE NUMBER Clause Sequence Substitutions

Where 'AR' is shown to be a permissible sequence in the Presentation Rules Table, 'A' is also permissible and the same presentation rules are applicable.

Where '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.

6.15.5.5 Saved Next Group Integer Description

Saved Next Group Integer is a data item that is addressable only by the 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.

Withdrawn/Retiree

6.15.5.6 REPORT HEADING Group Presentation Rules

The following figure 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 the PAGE clause is omitted.	
Sequence of LINE NUMBER clauses*	NEXT GROUP clause	Upper Limit	Lower Limit	First Print Line Position	Next Group	Final LINE-COUNTER Setting	First Print Line Position	Final LINE-COUNTER Setting
A R	Absolute	1	2a	3a	4a	5a	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R	Relative	1	2a	3a	4b	5b	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R	NEXT PAGE	1	2b	3a	4c	5c	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R		1	2a	3a		5d	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
R	Absolute	1	2a	3b	4a	5a	Illegal Combination. (See 6.32.3.3, The NEXT GROUP Clause.)	
R	Relative	1	2a	3b	4b	5b	3d	5b
R	NEXT PAGE	1	2b	3b	4c	5c	Illegal Combination. (See 6.32.3.3, The NEXT GROUP Clause.)	
R		1	2a	3b		5d	3d	5e
				3c		5e	3c	5e

Figure 6.15-2 REPORT HEADING Group Presentation Rules

Withdrawn/Retirée

REPORT GROUP SKELETON

* The meaning of the abbreviations used in column 1 has been previously stated. (See 6.15.5.3, LINE NUMBER Clause Notation.)

** 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.

6.15.5.6.1 Figure 6.15-2 Presentation Rules

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 contents of its LINE-COUNTER (in this case, zero) to the integer of the first LINE NUMBER clause.

Withdrawn/Retiree

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 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 non-printable report group.

Withdrawn/Retirée

REPORT GROUP SKELETON

6.15.5.7 PAGE HEADING Group Presentation Rules

The following figure 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

Figure 6.15-3 PAGE HEADING Group Presentation Rules

* The meaning of the abbreviations used in column 1 has been previously stated. (See 6.15.5.3, LINE NUMBER Clause Notation.)

** 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 6.46.3.3, The TYPE Clause.)

6.15.5.7.1 Figure 6.15-3 Presentation Rules

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.

Withdrawn/Retirée

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 (1) 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 (1) 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 non-printable report group.

6.15.5.8 Body Group Presentation Rules

The following figure 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.

Withdrawn/Retired

REPORT GROUP SKELETON

**		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	Fit Test	First Print Line Position	Next Group	Final LINE-COUNTER Setting	First Print Line Position	Final LINE-COUNTER Setting
A R	Absolute	1	2	3a	4a	5	6a	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R	Relative	1	2	3a	4a		6b	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R	NEXT PAGE	1	2	3a	4a		6c	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
A R		1	2	3a	4a		6d	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
R	Absolute	1	2	3b	4b	5	6a	Illegal Combination. (See 6.32.3.3, The NEXT GROUP Clause.)	
R	Relative	1	2	3b	4b		6b	4d	6f
R	NEXT PAGE	1	2	3b	4b		6c	Illegal Combination. (See 6.32.3.3, The NEXT GROUP Clause.)	
R		1	2	3b	4b		6d	4d	6d
NP R	Absolute	1	2	3c	4a	5	6a	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
NP R	Relative	1	2	3c	4a		6b	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
NP R	NEXT PAGE	1	2	3c	4a		6c	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
NP R		1	2	3c	4a		6d	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
					4c		6e	4c	6e

Figure 6.15-4 Body Group Presentation Rules

* The meaning of the abbreviations used in column 1 has been previously stated. (See 6.15.5.3, LINE NUMBER Clause Notation.)

** 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.

Withdrawn/Retirée

6.15.5.8.1 Figure 6.15-4 Presentation Rules

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 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 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 Saved Next Group Integer was so set, the RWCS moves the Saved Next Group Integer into LINE-COUNTER, resets 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 contents 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.)

Withdrawn/Retiree

If 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 Saved Next Group Integer was so set, the RWCS moves the Saved Next Group Integer into LINE-COUNTER, resets Saved Next Group Integer to zero, and computes a trial sum in a work location.

The trial sum is computed by adding the contents of LINE-COUNTER to the integer one (1) 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 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 Saved Next Group Integer was so set, the RWCS moves the Saved Next Group Integer into LINE-COUNTER and resets 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 contents of LINE-COUNTER and the integer of the first LINE NUMBER clause of the current body group.

Withdrawn/Retirée

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 contents 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 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.

Withdrawn/Retirée

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 non-printable 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.

6.15.5.9 PAGE FOOTING Presentation Rules

The following figure 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 Position	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

Figure 6.15-5 PAGE FOOTING Presentation Rules

Withdrawn/Retiree

* The meaning of the abbreviations used in column 1 have been previously stated. (See 6.15.5.3, LINE NUMBER Clause Notation.)

** 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 FOOTING report group may not be defined. (See 6.46.3.3, The TYPE Clause.)

6.15.5.9.1 Figure 6.15-5 Presentation Rules

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.

Withdrawn/Retirée

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 non-printable report group.

6.15.5.10 REPORT FOOTING Presentation Rules

The following figure 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 Position	Next Group	Final LINE-COUNTER Setting	First Print Line Position	Final LINE-COUNTER Setting
A R		1a	2	3a		4a	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
R		1a	2	3b		4a	3d	4a
NP R		1b	2	3c		4a	Illegal Combination. (See 6.31.3.5, The LINE NUMBER Clause.)	
				3e		4b	3e	4b

Figure 6.15-6 REPORT FOOTING Presentation Rules

Withdrawn/Retiree

* The meaning of the abbreviations used in column 1 has been previously stated. (See 6.15.5.3, LINE NUMBER Clause Notation.)

** 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.

6.15.5.10.1 Figure 6.15-6 Presentation Rules

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.

Withdrawn/Retiree

REPORT GROUP SKELETON

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 contents 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 non-printable report group.

Withdrawn/Retirée

6.16 THE ACCESS MODE CLAUSE

6.16.1 FUNCTION

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

6.16.2 GENERAL FORMAT

Format 1

```
[ACCESS MODE IS]   SEQUENTIAL
```

Format 2

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

Format 3

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

6.16.3 SYNTAX RULES

1. Data-name-1 may be qualified.
2. Data-name-1 must be defined as an unsigned integer.
3. Data-name-1 must not be defined in a Record Description entry associated with that file-name.
4. If file-name is specified in the USING or GIVING phrases of a SORT or MERGE statement, the ACCESS MODE IS RANDOM clause must not be specified.

6.16.4 GENERAL RULES

ALL FORMATS

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:

Withdrawn/Retirée

ACCESS MODE

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 the order of ascending record key values within a given Key of Reference.

FORMATS 2 AND 3

3. If the access mode is random, the value of the RELATIVE KEY data item for relative files or 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.

FORMAT 2

5. All records stored in a relative file are uniquely identified by relative record numbers. The 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 one (1), and subsequent logical records have relative record numbers of 2, 3, 4, ...

6. The data item specified by data-name-1 is used to communicate a relative record number between the user and the MSCS.

Withdrawn/Retiree

6.17 THE ALTERNATE RECORD KEY CLAUSE

6.17.1 FUNCTION

The ALTERNATE RECORD KEY clause specifies an Alternate Record Key access path to the records in an indexed file.

6.17.2 GENERAL FORMAT

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

6.17.3 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 variable-length data item or a group item that contains a variable-occurrence data item.

6.17.4 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.

Withdrawn/Retirée

6.18 THE BLANK WHEN ZERO CLAUSE

6.18.1 FUNCTION

The BLANK WHEN ZERO clause permits the blanking of an item when its value is zero.

6.18.2 GENERAL FORMAT

BLANK WHEN ZERO

6.18.3 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 6.35, The PICTURE clause.)

2. This clause cannot be used for variable length items.

6.18.4 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.

Withdrawn/Retirée

6.19 THE CHECK CLAUSE

6.19.1 FUNCTION

The CHECK clause specifies data item validation that occurs during access to a data base.

6.19.2 GENERAL FORMAT

```
CHECK IS { literal-1                                          }
         { NULL                                               }
         { routine-name  [ { THRU    } literal-2 ]  . . .     }
                         [ { THROUGH } ]                      }
```

6.19.3 SYNTAX RULES

1. The CHECK clause may be specified only in a Sub-schema.

2. The words THRU and THROUGH are equivalent.

3. The CHECK clause may be used only to describe an elementary item.

4. The value of literal-2 must be greater than the value of literal-1.

5. Numeric literals must be used to describe data items of the category numeric. Nonnumeric literals must be used to describe data items of the categories alphabetic, alphanumeric, alphanumeric edited, and numeric edited. Boolean literals must be used to describe data items of the category boolean.

6. If the data item with which the CHECK clause is associated is a boolean data item, the THROUGH phrase must not be specified.

7. The word NULL may be specified only once within any one CHECK clause.

6.19.4 GENERAL RULES

1. During the execution of any statement that accesses, in the data base, the data items associated with this CHECK clause, the DBCS checks the validity of the contents of the data item. If the DBCS determines that the contents of the data items are not valid, execution of the statement is not successful, no contents of any data items are changed, and a Data Base Exception Condition occurs. (See 12.4, Data Base Status Indicators.)

Withdrawn/Retirée

CHECK

2. If a range of values is specified by the THROUGH phrase, the contents of the data item are valid if its value is equal to or greater than literal-1, and equal to or less than literal-2, according to the rules for comparison of operands in a relation condition.

3. If a range of values is not specified by the THROUGH phrase, the contents of the data item are valid if its value is equal to literal-1, according to the rules for comparison of operands in a relation condition.

4. If NULL is specified, the contents of the data item are valid if the null data attribute is associated with them.

5. If literals and NULL are both specified, the contents of the data item are valid if they satisfy the criteria in General Rule 2, 3, or 4.

6. If routine-name is specified, the routine referenced by routine-name is executed, and the algorithm for determining valid data item content must be contained in the routine.

7. If the Schema description for this data item specifies the valid values for the contents of this data item, the values specified in this CHECK clause must conform to the range specified in the Schema description.

Withdrawn/Retirée

6.20 THE CODE CLAUSE

6.20.1 FUNCTION

The CODE clause specifies a two character literal that identifies each print line as belonging to a specific report.

6.20.2 GENERAL FORMAT

CODE literal-1

6.20.3 SYNTAX RULES

1. Literal-1 is a two character nonnumeric literal.

2. If the CODE clause is specified for any report in a file, then it must be specified for all reports in the same file.

6.20.4 GENERAL RULES

1. When the CODE clause is specified, literal-1 is automatically 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.

Withdrawn/Retirée

6.21 THE COLUMN NUMBER CLAUSE

6.21.1 FUNCTION

The COLUMN NUMBER clause identifies a printable item and specifies the column number position of the item on a print line.

6.21.2 GENERAL FORMAT

COLUMN NUMBER IS integer-1

6.21.3 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.

6.21.4 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 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.

Withdrawn/Retirée

6.22 THE CONTROL CLAUSE

6.22.1 FUNCTION

The CONTROL clause establishes the levels of the control hierarchy for the report.

6.22.2 GENERAL FORMAT

```
{CONTROL IS  }  {data-name-1      [ , data-name-2 ] . . .                    }
{CONTROLS ARE}  {FINAL      [ , data-name-1      [ , data-name-2 ] . . . ]  }
```

6.22.3 SYNTAX RULES

1. Data-name-1 and data-name-2 must not be defined in the Report Section. Data-name-1 and data-name-2 may be qualified but must not be subscripted or indexed.

2. Each data-name must identify a different data item.

3. Data-name-1, data-name-2, . . ., must not have subordinate to it a variable-occurrence data item.

6.22.4 GENERAL RULES

1. The data-names 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, data-name-2 is an intermediate control, etc. The last data-name specified is the minor control.

2. The execution of the chronologically first GENERATE statement for a given report causes the 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. The control break is associated with the highest level for which a change of value is noted. (See 7.21, The GENERATE Statement.)

3. The Report Writer Control System tests for a control break by comparing the contents of each control data item with the prior contents saved from the execution of the previous GENERATE statement for the same report. 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.

Withdrawn/Retirée

c. If the control data item is a boolean data item, the relation test is for the comparison of two boolean data items.

d. If the control data item is a data item other than as described in paragraph 3a, 3b and 3c, the relation test is for the comparison of two nonnumeric operands.

The inequality relation test is further explained in the appropriate paragraph. (See 7.5.1.1, Relation Condition.)

4. FINAL is used when the most inclusive control group in the report is not associated with a control data-name.

Withdrawn/Retiree

6.23 THE DATA-NAME OR FILLER CLAUSE

6.23.1 FUNCTION

A data-name specifies the name of the data item being described. The key word FILLER may be used to specify a data item which is not referenced explicitly.

6.23.2 GENERAL FORMAT

```
[data–name]
[FILLER  ]
```

6.23.3 SYNTAX RULES

1. In the File, Working-Storage, Communication and Linkage Sections, a data-name or the key word FILLER, if either is specified, must be the first word following the level-number in each Data Description entry.

2. In the Report Section a data-name need not appear in a Data Description entry and the key word FILLER must not be used.

6.23.3a SYNTAX RULES FOR THE SUB-SCHEMA ONLY

1. If data-name describes an elementary data item, the data-name of that data item must match the data-name of a data item defined in the Schema; unless that data item is redefined and one of its other data-names matches a data-name defined in the Schema.

6.23.4. GENERAL RULES

1. If this clause is omitted, the data item being described is treated as though FILLER had been specified.

2. The key word FILLER may be used to name a data item. 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 the value contained therein.

3. In the Report Section, data-name must be given in the following cases:

   a. When the data-name 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.

Withdrawn/Retirée

DATA–NAME OR FILLER

c. When a DETAIL report group is referenced in the UPON phrase of the SUM clause.

d. When the data-name is required to provide sum counter qualification.

Withdrawn/Retirée

6.24 THE DATA RECORDS CLAUSE

6.24.1 FUNCTION

The DATA RECORDS clause serves only as documentation for the names of data records with their associated file.

6.24.2 GENERAL FORMAT

```
DATA  [ RECORD IS  ]  data-name-1  [ , data-name-2 ] . . .
      [ RECORDS ARE ]
```

6.24.3 SYNTAX RULES

1. Data-name-1, data-name-2, . . ., are the names of data records and must be defined by 01 level-number Record Description entries subordinate to the File Description entry containing the DATA RECORDS clause.

2. Data-name-1, data-name-2, . . ., must not be named in the LABEL RECORDS clause within the same File Description entry.

6.24.4 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.

Withdrawn/Retirée

6.25 THE FILE STATUS CLAUSE

6.25.1 FUNCTION

The FILE STATUS clause specifies a data item which contains the status of an I/O operation.

6.25.2 GENERAL FORMAT

FILE STATUS IS data-name-1

6.25.3 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, Report Section or the Communication Section.

6.25.4 GENERAL RULES

1. If the FILE STATUS clause is specified, a value will be moved by the operating system into the data item specified by data-name-1 after the execution of every statement that references that file. This value indicates the status of execution of the statement. (See 7.7.9, I-O Status.)

Withdrawn/Retirée

6.26 THE GROUP INDICATE CLAUSE

6.26.1 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.

6.26.2 GENERAL FORMAT

GROUP INDICATE

6.26.3 SYNTAX RULES

1. The GROUP INDICATE clause may only appear in a DETAIL report group entry that defines a printable item.

6.26.4 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.

Withdrawn/Retirée

6.27 THE JUSTIFIED CLAUSE

6.27.1 FUNCTION

The JUSTIFIED clause specifies non-standard positioning of data within a receiving data item.

6.27.2 GENERAL FORMAT

$$\left\{\begin{matrix}\underline{\text{JUSTIFIED}}\\ \underline{\text{JUST}}\end{matrix}\right\} \quad \text{RIGHT}$$

6.27.3 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 a variable-length data item.

6.27.4 GENERAL RULES

1. When the 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, or boolean character zero fill in the case of boolean data items, for the leftmost character positions.
2. When the JUSTIFIED clause is omitted, the standard rules for aligning data within an elementary item apply. (See 3.3.6, Standard Alignment Rules.)

Withdrawn/Retirée

6.28 THE LABEL RECORDS CLAUSE

6.28.1 FUNCTION

The LABEL RECORDS clause specifies whether labels are present and, if present, identifies the label.

6.28.2 GENERAL FORMAT

```
LABEL  [ RECORDS ARE ]  { OMITTED                                          }
       [ RECORD  IS  ]  { STANDARD                                         }
                        { data-name-1  [ , data-name-2 ]  . . .            }
```

6.28.3 SYNTAX RULES

1. Data-name-1, data-name-2, . . ., are names of label records and must not appear in the DATA RECORDS clause and must be the subject of a Record Description associated with the file.

2. The LABEL RECORDS clauses specified in the File Description entries associated with file-names specified in a MULTIPLE FILE TAPE clause in the I-O-CONTROL paragraph must reflect a uniform labeling convention.

6.28.4 GENERAL RULES

1. If the LABEL RECORDS clause is not specified for a file, label records for that file must conform to the implementor's label specifications.

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

3. STANDARD specifies that labels exist for the file or the device to which the file is assigned and the labels conform to the implementor's label specifications.

4. All Procedure Division references to the data-names specified in this clause, or to any items subordinate to these data-names, must appear within USE Procedures.

Withdrawn/Retirée

6.29 LEVEL-NUMBER

6.29.1 FUNCTION

The level-number indicates the position of a data item within the hierarchical structure of a logical record or report group. In addition, it is used to identify entries for condition-names, Working-Storage items, Linkage items, and the RENAMES clause.

6.29.2 GENERAL FORMAT

level-number

6.29.3 SYNTAX RULES FOR THE DATA DIVISION

1. A level-number is required as the first element in each Data Description entry.

2. Data Description entries subordinate to a CD, FD or SD entry must have level-numbers with the values 01 through 49, 66 or 88.

3. Data Description entries subordinate to an RD entry must have level-numbers 01 thru 49 only.

4. Data Description entries in the Working-Storage Section and Linkage Section must have level-numbers 01, 02, ..., 49, 66 or 88.

6.29.3a SYNTAX RULES FOR THE SUB-SCHEMA

1. A level-number is required as the first element in each Record Description entry and each Data Description entry.

2. Data Description entries must have level-numbers with the values 02 through 49, 66, or 88.

6.29.4 GENERAL RULES

1. The level-number 01 identifies the first entry in each Record Description or a report group.

2. Special level-numbers have been assigned to certain entries where there is no real concept of hierarchy.

   a. Level-number 66 is assigned to identify RENAMES entries and can be used only as described by Format 2 of the Data Description Skeleton. (See 6.14, The Data Description — Complete Entry Skeleton.)

   b. Level-number 88 is assigned to entries which define condition-names associated with a conditional variable and can be used only as described by Format 3 of the Data Description Skeleton. (See 6.14, The Data Description — Complete Entry Skeleton.)

Withdrawn/Retirée

3. Multiple level 01 entries subordinate to any given level indicator, other than RD, represent implicit redefinitions of the same area.

Withdrawn/Retirée

6.30 THE LINAGE CLAUSE

6.30.1 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.

6.30.2 GENERAL FORMAT

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

[, LINES AT TOP    {data-name-3}]   [, LINES AT BOTTOM {data-name-4}]
                   {integer-3}  ]   [                  {integer-4}  ]
```

6.30.3 SYNTAX RULES

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

2. The LINAGE clause and the REPORT clause are not allowed in the same File Description entry.

3. The value of integer-1 must be greater than zero.

4. The value of integer-2 must not be greater than integer-1.

5. The value of integer-3, integer-4 may be zero.

6.30.4 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 for these functions are zero. If the FOOTING phrase is not specified, the assumed value is equal to integer-1, or the contents of the data item referenced by data-name-1, whichever is specified.

   There is not necessarily any relationship between the size of the logical page and the size of a physical page.

2. The value of integer-1 or 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.

Withdrawn/Retirée

3. The value of integer-3 or 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.

4. The value of integer-4 or 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.

5. The value of integer-2 or 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 the value of integer-1 or the data item referenced by data-name-1.

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

6. The value of integer-1, integer-3, and integer-4, if specified, will be 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. The value of integer-2, if specified, will be used at that time to define the footing area. These values are used for all logical pages written for that 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, will be 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, will be 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 page overflow condition occurs will be used to specify the number of lines that are to comprise each of the indicated sections for the next logical page. (See 7.51, 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, will be 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 will be 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.

Withdrawn/Retiree

b. LINAGE-COUNTER may be referenced, but may not be modified, by Procedure Division statements. 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 automatically modified, according to the following rules, during the execution of a WRITE statement to an associated file:

(1) When the ADVANCING PAGE phrase of the WRITE statement is specified, the LINAGE-COUNTER is automatically reset to one (1).

(2) When the ADVANCING identifier-2 or integer phrase of the WRITE statement is specified, the LINAGE-COUNTER is incremented by integer or the value of the data item referenced by identifier-2.

(3) When the ADVANCING phrase of the WRITE statement is not specified, the LINAGE-COUNTER is incremented by the value one (1). (See 7.51.4, The WRITE Statement.)

(4) The value of LINAGE-COUNTER is automatically reset to one (1) when the device is repositioned to the first line that can be written on for each of the succeeding logical pages. (See 7.51.4, The WRITE Statement.)

d. The value of LINAGE-COUNTER is automatically set to one (1) 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.

Withdrawn/Retirée

6.31 THE LINE NUMBER CLAUSE

6.31.1 FUNCTION

The LINE NUMBER clause specifies vertical positioning information for its report group.

6.31.2 GENERAL FORMAT

```
LINE NUMBER IS { integer-1                [ ON NEXT PAGE ] }
               { PLUS integer-2                            }
```

6.31.3 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 that report group type, as defined by the PAGE clause. (See 6.34, The PAGE Clause.)

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. The integers need not be consecutive.

5. If the PAGE clause is omitted from a given Report Group 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.

   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.

7. Every entry that defines a printable item (See 6.21, The COLUMN NUMBER Clause.) must either contain a LINE NUMBER clause, or be subordinate to an entry that contains a LINE NUMBER clause.

8. The first LINE NUMBER clause specified within a PAGE FOOTING report group must be an absolute LINE NUMBER clause.

Withdrawn/Retirée

LINE NUMBER

6.31.4 GENERAL RULES

1. A LINE NUMBER clause must be specified to establish each print line of a report group.

2. The 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 6.15.5, 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 6.15.5, Presentation Rules Tables.)

Withdrawn/Retirée

6.32 THE NEXT GROUP CLAUSE

6.32.1 FUNCTION

The NEXT GROUP clause specifies information for vertical positioning of a page following the presentation of the last line of a report group.

6.32.2 GENERAL FORMAT

```
NEXT GROUP IS  { integer-1          }
               { PLUS integer-2     }
               { NEXT PAGE          }
```

6.32.3 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.

6.32.4 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 6.15.5, 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 6.15.5, 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. The NEXT GROUP clause of a REPORT HEADING report group can affect the location at which the PAGE HEADING report group is

Withdrawn/Retirée

presented. The NEXT GROUP clause of a PAGE FOOTING report group can affect the location at which the REPORT FOOTING report group is presented. (See 6.15.5, Presentation Rules Tables.)

Withdrawn/Retirée

6.33 THE OCCURS CLAUSE

6.33.1 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.

6.33.2 GENERAL FORMAT

Format 1

```
OCCURS integer-2 TIMES
    [ { ASCENDING  } KEY IS data-name-2 [ , data-name-3 ] . . . ] . . .
      { DESCENDING }
    [ INDEXED BY index-name-1 [ , index-name-2 ] . . . ]
```

Format 2

```
OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1
    [ { ASCENDING  } KEY IS data-name-2 [ , data-name-3 ] . . . ] . . .
      { DESCENDING }
    [ INDEXED BY index-name-1 [ , index-name-2 ] . . . ]
```

6.33.3 SYNTAX RULES

1. Where both integer-1 and integer-2 are used the value of integer-1 must be less than integer-2. The value of integer-1 may be zero, but integer-2 cannot be zero.

2. Data-name-1 must describe an elementary unsigned integer.

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

4. Data-name-2 must either be the name of the entry containing the OCCURS clause or the name of an entry subordinate to the entry containing the OCCURS clause.

5. Data-name-3, . . ., must be the name of an entry subordinate to the group item which is the subject of this entry.

Withdrawn/Retirée

6. Items identified by the data-names in the KEY IS phrase must not contain an OCCURS clause except where data-name-2 is the subject of the entry.

7. There must not be any entry that contains an OCCURS clause between the items identified by the data-names in the KEY IS phrase and the subject of the entry.

8. Data-name-2, data-name-3, . . ., must be specified without the subscripting or indexing normally required.

9. 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.

10. If Format 2 is specified in a Record Description entry associated with a file which contains variable-length records, the data item referenced by data-name-1 must be in the same record.

11. 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.

12. The OCCURS clause must not be specified in a Data Description entry that:

    a. Has a level-number of 01, 66 or 88, or

    b. Has a variable-occurrence data item subordinate to it.

13. 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.

14. Index-name-1, index-name-2, . . . must be unique words within the program.

15. Data-name-1, data-name-2, data-name-3, . . ., must not reference a boolean data item.

6.33.3a SYNTAX RULES FOR THE SUB-SCHEMA ONLY

1. The OCCURS clause may be specified only in a Data Description entry that:

    a. describes a data item that is described in the Schema as a table element, or

    b. describes a group item consisting only of data items that are described in the Schema as table elements.

Withdrawn/Retirée

2. The value of integer-2 must not be greater than the number of occurrences defined for this data item in the Schema.

3. If the OCCURS clause describes a group item that is not defined in the Schema, the value of integer-2 must not be greater than the least number of occurrences defined in the Schema for any of the data items subordinate to the group item.

4. An elementary table element must be described in the Sub-schema with the same number of dimensions that is described for it in the Schema.

6.33.4 GENERAL RULES

1. The OCCURS clause is used in defining tables and other homogeneous sets of repeated data items.

2. 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.

3. 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, unpredictable.

4. When a group item, having subordinate to it an entry that specifies Format 2 of the OCCURS clause, 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 and the value of the data item referenced by data-name-1 will be ignored, and will be unchanged by the operation unless it is within the group.

5. The KEY IS phrase is used to indicate that the repeated data is arranged in ascending or descending order according to the values contained in data-name-2, data-name-3, ... The ascending or descending order is determined according to the rules for the comparison of operands. (See 7.5.1.1.1,

Withdrawn/Retirée

Comparison of Numeric Operands and 7.5.1.1.2, Comparison of Nonnumeric Operands.) The data-names are listed in their descending order of significance.

6. If Format 2 is specified in a Record Description entry and the associated File Description or Sort-Merge Description entry contains the VARYING phrase of the RECORD clause, the records are variable-length. If the DEPENDING ON phrase of the RECORD clause is not specified, the contents of the data item referenced by data-name-1 of the OCCURS clause must be set to the number of occurrences to be written before the execution of any RELEASE, REWRITE or WRITE statement.

Withdrawn/Retired

6.34 THE PAGE CLAUSE

6.34.1 FUNCTION

The PAGE clause defines the length of a page and the vertical subdivisions within which report groups are presented.

6.34.2 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 ]
```

6.34.3 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 (3) significant digits in length.

3. Integer-2 must be greater than or equal to one (1).

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. Integer-1 must be greater than or equal to integer-5.

8. 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 6.34.5, 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.

Withdrawn/Retirée

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.

9. All report groups must be described such that they can be presented on one page. The RWCS never splits a multi-line report group across page boundaries.

6.34.4 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.

Withdrawn/Retirée

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 (1) 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 6.15.5, Presentation Rules Tables.)

Withdrawn/Retiree

6.34.5 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 Number Of The Region	Last Line Number 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

Figure 6.34-1 Page Regions

Withdrawn/Retirée

6.35 THE PICTURE CLAUSE

6.35.1 FUNCTION

The PICTURE clause describes the general characteristics and editing requirements of an elementary item.

6.35.2 GENERAL FORMAT

$$\left\{\begin{matrix}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{matrix}\right\} \text{ IS character-string } [\ \underline{\text{DEPENDING}}\ \text{ON data-name}\ ]$$

6.35.3 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 the character-string is 30.
4. The DEPENDING ON clause must not be specified in the Report Section.
5. Data-name must describe an elementary unsigned integer. It must not be defined in the Report Section. Data-name may be qualified.
6. The PICTURE clause must be specified for every elementary item except an index data item, a data base key item, or the subject of the RENAMES clause. In these cases the use of this clause is prohibited.
7. PIC is an abbreviation for PICTURE.
8. The asterisk when used as the Zero Suppression symbol and the clause BLANK WHEN ZERO may not appear in the same entry.
9. The DEPENDING ON clause must be specified if, and only if, the PICTURE character-string contains the character 'L'.
10. The DEPENDING ON clause must not appear in a Data Description entry which contains a JUSTIFIED clause.

Withdrawn/Retirée

11. Data-name must not reference a boolean data item.

12. Relational records must not be described with any clauses other than the PICTURE clause and the PRIVACY LOCK clause.

6.35.4 GENERAL RULES

1. There are six categories of data that can be described with a PICTURE clause: Boolean, Alphabetic, Numeric, Alphanumeric, Alphanumeric Edited and Numeric Edited. If the PICTURE clause is specified in a Sub-schema, it may also describe a relational record.

2. To define an item as Boolean:

   a. Its PICTURE character-string can only contain the symbol '1'; and

   b. Its contents, when represented in Standard Data Format, must be a combination of the Arabic numerals '0' and '1'.

3. To define an item as Alphabetic:

   a. Its PICTURE character-string can contain only the symbols 'A', 'L'; and

   b. Its contents when represented in Standard Data Format must be one or more alphabetic characters.

4. To define an item as Numeric:

   a. Its PICTURE character-string can contain only 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 contents 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 6.42, The SIGN Clause.)

5. To define an item as Alphanumeric:

   a. Its PICTURE character-string is restricted to certain combinations of the symbols 'A', 'L', 'X', '9', and the item is treated as if the character-string contained all 'X's. A PICTURE character-string which contain all 'A's (or only 'A's and the symbol 'L') or all '9's does not define an Alphanumeric item, and;

   b. Its contents when represented in Standard Data Format must be one or more characters in the computer's character set.

Withdrawn/Retirée

6. 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 '/' (stroke).

   b. Its contents when represented in Standard Data Format must be one or more characters in the computer's character set.

7. 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', 'DB', and the currency symbol. The allowable combinations are determined from the order of precedence of symbols and the editing rules; and

      1. The number of digit positions that can be represented in the PICTURE character-string must range from 1 to 18 inclusive; and

      2. The character-string must contain at least one '0', 'B', '/', 'Z', '*', '+', ',', '.', '-', 'CR', 'DB', or the currency symbol.

   b. The content of each of the character positions of these symbols that represents a digit in Standard Data Format must be a numeric character.

8. To define a relational record, its PICTURE character-string can only contain a single symbol 'R'.

9. 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 non-zero integer which is enclosed in parentheses following the symbols 'A', 'X', '9', '1', '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', 'L', '.', 'CR', and 'DB'.

   The number of valid character positions contained within a variable-length data item may vary. (See General Rule 10, the PICTURE symbol 'L'.)

10. 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.

Withdrawn/Retiree

PICTURE

L The 'L' is used to define a variable-length data item; and, when present, must appear as the first symbol in the PICTURE character-string.

The 'L' can be used to describe a data item of the category alphanumeric or alphabetic. While the size of the data item is fixed, the number of valid character positions contained in the data item varies. The PICTURE character-string describes the fixed size of the data item; the 'L' is not counted in determining the size of the data item.

When referenced in the Procedure Division, the data item is considered to contain a number of valid character positions equal to the value of the data item referenced by data-name. The content of any character positions in excess of the number specified by data-name is undefined. The valid character positions are contiguous and begin at the leftmost character position in the data item.

When such a data item is explicitly referenced, or implicitly referenced by an INITIALIZE statement or a statement with the CORRESPONDING phrase, any character positions in excess of the value indicated by the contents of that data item referenced by data-name do not participate in the operation.

During the execution of any statement which explicitly or implicitly references a variable-length data item, the value of the data item referenced by data-name must be within the range one (1) through the size of the data item as defined by the PICTURE character-string.

If a group item which contains a variable-length data item is referenced, all character positions in the variable-length data item participate in the operation; i.e., the variability of the valid contents of the data item(s) is ignored, as is its USAGE, category, synchronization, etc.

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 to the left or right as a continuous string of 'P's within a PICTURE description; 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 character 'P' and the insertion character '.' (period) cannot both occur in the same PICTURE character-string. In any operation that references such a data item, 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.

R The 'R' is used to define a relational record, and may appear only in a Sub-schema Record Description entry.

S The letter '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)

Withdrawn/Retirée

of the elementary item unless the entry is subject to a SIGN clause which specifies the optional SEPARATE CHARACTER phrase. (See 6.42, 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 contents of that character position is zero. Each 'Z' is counted in the size of the item.

9 Each '9' in the character-string represents a digit position which contains a numeral and is counted in the size of the item.

1 Each '1' in the character-string represents a boolean position which contains a boolean character 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 '/' (stroke) in the character-string represents a character position into which the stroke 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. This character position is counted in the size of the item.

. When the character '.' (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.

Withdrawn/Retirée

* Each '*' (asterisk) in the character-string represents a leading numeric character position into which an asterisk will be placed when the contents 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.

6.35.5 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	None
Boolean	None
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 and editing by Zero Suppression and Replacement are mutualy exclusive in a PICTURE clause. Only one type of Replacement may be used with Zero Suppression in a PICTURE clause.

Withdrawn/Retired

4. Simple Insertion Editing.

The ',' (comma), 'B' (space), '0' (zero) and '/' (stroke) 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. 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. 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:

	RESULT	
EDITING SYMBOL IN PICTURE CHARACTER-STRING	DATA ITEM POSITIVE OR ZERO	DATA ITEM NEGATIVE
+	+	-
-	space	-
CR	2 spaces	CR
DB	2 spaces	DB

Withdrawn/Retired

PICTURE

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. This string of floating insertion characters may have the fixed insertion characters 'CR' and 'DB' immediately to the right of this 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 non-zero digit in the data represented by the insertion symbol string, whichever is farther to the left in the PICTURE character-string. The character positions preceding the insertion character are replaced with spaces.

When the floating insertion character is the editing control symbol '+' or '-', the character inserted depends upon the value of the data item:

	RESULT	
EDITING SYMBOL IN PICTURE CHARACTER-STRING	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.

Withdrawn/Retired

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.

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 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 non-zero 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, including any editing characters, is spaces. If the value is zero and the suppression symbol is '*', the entire data item, including any insertion editing symbols except the actual decimal point, will be '*'. 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.

6.35.6 PRECEDENCE RULES

The following figure 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'.

Withdrawn/Retirée

PICTURE

First Symbol / Second Symbol		Non-Floating Insertion Symbols									Floating Insertion Symbols						Other Symbols							
		B	0	/	,	.	{+ −}	{+ −}	{CR DB}	cs	{Z *}	{Z *}	{+ −}	{+ −}	cs	cs	9	A X	L	S	V	P	P	1
Non-Floating Insertion Symbols	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	x	
	(CR DB)	x	x	x	x	x				x	x	x			x	x	x				x	x	x	
	cs						x																	
Floating Insertion Symbols	(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 Symbols	9	x	x	x	x	x	x			x	x		x		x		x	x	x	x	x		x	
	A X	x	x	x													x	x	x					
	L																							
	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	
	1																							x

Figure 6.35-1 Order of Precedence

Withdrawn/Retirée

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 character-string.

Non-floating insertion symbols '+' and '-', floating insertion symbols 'Z', '*', '+', '-', and 'cs', and other symbol 'P' appear twice in the above 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.

Withdrawn/Retired

6.36 THE PRIVACY LOCK CLAUSE

6.36.1 FUNCTION

The PRIVACY LOCK clause specifies privacy locks to be associated with the use of a data item.

6.36.2 GENERAL FORMAT

```
PRIVACY LOCK [ FOR { GET    } ] IS { literal             }
               [   { MODIFY }   ]   { implementor-name   }
               [   { STORE  }   ]   { routine-name       }
```

6.36.3 SYNTAX RULES

1. The PRIVACY LOCK clause may be specified only in the Sub-schema.
2. The PRIVACY LOCK clause must not be used in a level 01 entry.
3. The PRIVACY LOCK clause may be used only to describe an elementary data item.
4. Literal must be nonnumeric.

6.36.4 GENERAL RULES

1. The PRIVACY LOCK clause causes a privacy lock to be associated with a data item.
2. If a literal is specified, the value of the literal is the privacy lock associated with the data item.
3. If implementor-name is specified, the feature referenced by the implementor-name is the privacy lock associated with the data item.
4. If a routine-name is specified, the privacy lock associated with the data item is dependent upon the nature of the routine referenced by routine-name.
5. If the FOR phrase is not specified, the privacy lock must be satisfied in order to execute a GET, MODIFY, or STORE statement that references the associated data item.
6. If the GET, MODIFY, or STORE phrase is specified, the privacy lock must be satisfied in order to execute a GET, MODIFY, or STORE statement, respectively, that references the associated data item.

Withdrawn/Retirée

6.37 THE RECORD CLAUSE

6.37.1 FUNCTION

The RECORD clause specifies the number of character positions in a fixed-length record, or specifies the range of character positions in a variable-length record. If the number of character positions does vary, the clause specifies the minimum and maximum number of character positions.

6.37.2 GENERAL FORMAT

Format 1

```
RECORD CONTAINS integer-1 CHARACTERS
```

Format 2

```
RECORD IS VARYING IN SIZE [ [FROM integer-2] [TO integer-3] ]
  CHARACTERS ] [DEPENDING ON data-name-1]
```

6.37.3 SYNTAX RULES

FORMAT 1

1. No Record Description entry for the file may specify a number of character positions greater than integer-1.

FORMAT 2

2. Record descriptions for the file must not describe records which contain a lesser number of character positions than that specified by integer-2 nor records which contain a greater number of character positions than that specified by integer-3.
3. Integer-3 must be greater than integer-2.
4. Data-name-1 must describe an elementary unsigned integer in the Working-Storage or Linkage Section.
5. In a Sort-Merge Description entry, all the sort or merge keys specified in any SORT or MERGE statement referencing that sort or merge file must be contained within the first integer-2 number of character positions.
6. For an indexed file, all the record keys for the file must be contained within the first integer-2 number of character positions.

Withdrawn/Retirée

6.37.4 GENERAL RULES

1. If the RECORD clause is not specified, it is as if Format 1 were specified with integer-1 being equal to the greatest number of character positions described for a record in that file.

FORMAT 1

2. Format 1 is used to specify fixed-length records. Integer-1 specifies the number of character positions contained in each record in the file.

FORMAT 2

3. Format 2 is used to specify variable-length records. Integer-2 specifies the minimum number of character positions to be contained in any record of the file. Integer-3 specifies the maximum number of character positions in any record of the file.

4. The number of character positions associated with a record description is determined by the sum of the number of character positions in all elementary data items excluding redefinitions and renamings, plus any implicit FILLER due to synchronization. If a table is specified, the maximum number of table elements described in the record is included in the summation above. The resultant sum may be different from the physical size of the record. (See 3.3.4, Selection of Character Representation and Radix; 6.45, The SYNCHRONIZED Clause; and 6.47, The USAGE Clause.)

5. If integer-2 is not specified, the minimum number of character positions to be contained in any record of the file is equal to the least number of character positions described for a record in that file.

6. If integer-3 is not specified, the maximum number of character positions to be contained in any record of the file is equal to the greatest number of character positions described for a record in that file.

7. For any execution of a RELEASE, REWRITE or WRITE Statement, if data-name-1 is specified, the contents of the data item referenced by data-name-1 determines the number of character positions for that record.

8. If data-name-1 is specified, the number of character positions in the record must be placed into the data item referenced by data-name-1 before any RELEASE, REWRITE or WRITE statement is executed for the file. After the successful execution of a READ or RETURN statement for the file, the contents of the data item referenced by data-name-1 will indicate the number of character positions in the record just read. The execution of a DELETE or a START statement or the unsuccessful execution of a READ or RETURN statement does not alter the contents of the data item referenced by data-name-1.

9. During the execution of a RELEASE, REWRITE or WRITE statement, the number of character positions in the record is determined by the following conditions:

Withdrawn/Retirée

a. If data-name-1 is specified, by the contents of the data item referenced by data-name-1.

b. If data-name-1 is not specified and the record does not contain a variable-occurrence data item, by the number of character positions in the record.

c. If data-name-1 is not specified and the record does contain a variable-occurrence data item, by the sum of the fixed portion and that portion of the table described by the number of occurrences at the time of execution of the output statement.

Withdrawn/Retiree

6.38 THE RECORD KEY CLAUSE

6.38.1 FUNCTION

The RECORD KEY clause specifies the Prime Record Key access path to the records in an indexed file.

6.38.2 GENERAL FORMAT

RECORD KEY IS data-name-1

6.38.3 SYNTAX RULES

1. Data-name-1 may be qualified.

2. Data-name-1 must be defined as a data item of the category alphanumeric with 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 variable-length data item or a group item that contains a variable-occurrence data item.

6.38.4 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 the 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.

Withdrawn/Retirée

6.39 THE REDEFINES CLAUSE

6.39.1 FUNCTION

The REDEFINES clause allows the same computer storage area to be described by different Data Description entries.

6.39.2 GENERAL FORMAT

level-number [data-name-1 / FILLER] ; REDEFINES data-name-2

Level-number, data-name-1, FILLER and the semicolon are shown in the above format to improve clarity. Level-number, data-name-1 and FILLER are not part of the REDEFINES clause.

6.39.3 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. This clause must not be used in level 01 entries in the Communication Section.

5. 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 or indexed. Neither the original definition nor the redefinition can include a variable-occurrence data item.

6. The number of character positions contained in the data item referenced by the subject of the entry must be less than or equal to the number of character positions contained in the data item referenced by data-name-2.

7. 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.

8. 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.

9. The entries giving the new description of the character positions must not contain any VALUE clauses, except in condition-name entries.

Withdrawn/Retirée

10. 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.

11. 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.

6.39.4 GENERAL RULES

1. Redefinition starts at data-name-2 and continues over a storage area sufficient to contain the number 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.

Withdrawn/Retirée

6.40 THE RENAMES CLAUSE

6.40.1 FUNCTION

The RENAMES clause permits alternative, possibly overlapping, groupings of elementary items.

6.40.2 GENERAL FORMAT

66 data-name-1; RENAMES data-name-2 [{ THRU / THROUGH } data-name-3]

Level-number 66, data-name-1 and the semicolon are shown in the above format to improve clarity. Level-number and data-name-1 are not part of the RENAMES clause.

6.40.3 SYNTAX RULES

1. 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.

2. Data-name-1 cannot be used as a qualifier, and can only be qualified by the names of the associated level 01, FD, CD, 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.

3. 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 an 88 or 01 level entry.

4. Data-name-2 and data-name-3 may be qualified.

5. None of the items within the range, including data-name-2 and data-name-3, if specified, can be variable-occurrence data items.

6. The words THRU and THROUGH are equivalent.

7. 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.

Withdrawn/Retirée

RENAMES

6.40.4 GENERAL RULES

1. One or more RENAMES entries can be written for a logical record.

2. When data-name-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).

3. When data-name-3 is not specified, all of the data attributes of data-name-2 become the data attributes for data-name-1.

Withdrawn/Retired

6.41 THE REPORT CLAUSE

6.41.1 FUNCTION

The REPORT clause specifies the names of reports that comprise a report file.

6.41.2 GENERAL FORMAT

```
{REPORT IS  }
{REPORTS ARE}  report-name-1  [ , report-name-2 ] . . .
```

6.41.3 SYNTAX RULES

1. Each report-name specified in a REPORT clause 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. A report-name must appear in only one REPORT clause.

3. The subject of a File Description entry that specifies a REPORT clause may only be referred to by the OPEN statement with the OUTPUT or EXTEND phrase or the CLOSE statement.

6.41.4 GENERAL RULES

1. The presence of more than one report-name in a REPORT clause indicates that that 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, that report file is under the control of the RWCS. While a report file is under the control of the RWCS, no input-output statement may be executed which references that report file or any label record description associated with that report file.

Withdrawn/Retirée

6.42 THE SIGN CLAUSE

6.42.1 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.

6.42.2 GENERAL FORMAT

[SIGN IS] { LEADING / TRAILING } [SEPARATE CHARACTER]

6.42.3 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 have the DISPLAY usage attribute.

3. At most one SIGN clause may apply to any given numeric Data Description entry.

4. If the CODE-SET clause is specified, any signed numeric Data Description entries associated with that File Description entry must be described with the SIGN IS SEPARATE clause.

6.42.4 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. 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 implementor will define the position and representation of the operational sign. General Rules 3 through 5 do not apply to such signed numeric data items.

3. 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.

Withdrawn/Retirée

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 implementor defines what constitutes valid sign(s) for data items.

4. 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.

5. 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 automatically.

Withdrawn/Retirée

6.43 THE SOURCE CLAUSE

6.43.1 FUNCTION

The SOURCE clause identifies the sending data item that is moved to an associated printable item defined within a Report Group Description entry.

6.43.2 GENERAL FORMAT

SOURCE IS identifier-1

6.43.3 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. A sum counter of the report within which the SOURCE clause appears.

2. Identifier-1 specifies the sending data item of the implicit MOVE statement that the RWCS will execute to move 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 7.28, The MOVE Statement.)

6.43.4 GENERAL RULES

1. The RWCS formats the print lines of a report group just prior to presenting the report group. (See 6.46, The TYPE Clause.) It is at this time that the implicit MOVE statements specified by SOURCE clauses are executed by the RWCS.

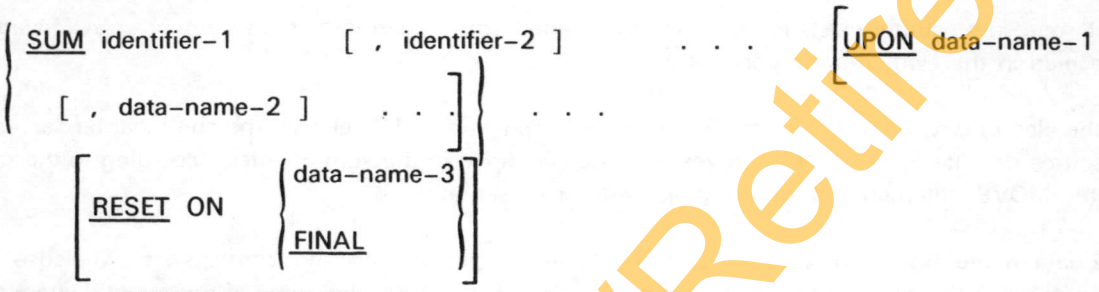
Withdrawn/Retirée

6.44 THE SUM CLAUSE

6.44.1 FUNCTION

The SUM clause establishes a sum counter and names the data items to be summed.

6.44.2 GENERAL FORMAT



6.44.3 SYNTAX RULES

1. The data item that is the subject of the Data Description entry in which the SUM clause appears must be defined as a numeric data item. Identifier-1 and identifier-2 must be the names of sum counters.

   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 and data-name-2 must be the names of DETAIL report groups described in the same report as the CONTROL FOOTING report group in which the SUM clause appears. Data-name-1 and data-name-2 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-3 must be one of the data-names specified in the CONTROL clause for this report. Data-name-3 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.

Withdrawn/Retirée

6.44.4 GENERAL RULES

1. The data item that is the subject of the Data Description entry in which the SUM clause appears is a sum counter. At object time, the RWCS adds into the sum counter each of the values contained in identifier-1 and identifier-2. This addition is consistent with the rules for arithmetic statements. (See 7.7.1, Intermediate Data Item; 7.7.5, The Arithmetic Statements; and 7.7.6, Overlapping Operands.)

2. Only one sum counter exists for an elementary report entry regardless of the number of SUM clauses specified in the elementary report entry.

3. 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.

4. 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.

5. It is permissible for Procedure Division statements to alter the contents of sum counters.

6. Addition of the identifiers into sum counters is performed by the RWCS 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 during execution of GENERATE statements only, after any control break processing but before processing of the DETAIL report group. (See 7.21, The GENERATE Statement.) Crossfooting and rolling forward are accomplished during the processing of CONTROL FOOTING report groups. (See 6.46, The TYPE Clause.)

7. The UPON phrase provides the capability to accomplish selective subtotalling for the DETAIL report groups named in the phrase.

8. The RWCS 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.

Withdrawn/Retirée

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 subtotalled when any GENERATE data-name statement is executed for the report in which the SUM clause appears.

9. 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.

10. For the subtotalling that occurs when a GENERATE report-name statement is executed. (See 7.21, The GENERATE Statement.)

11. In the absence of an explicit RESET phrase, the 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 6.46.4.10, The TYPE Clause.)

Sum counters are initially set to zero by the RWCS during the execution of the INITIATE statement for the report containing the sum counter.

Withdrawn/Retired

6.45 THE SYNCHRONIZED CLAUSE

6.45.1 FUNCTION

The SYNCHRONIZED clause specifies the alignment of an elementary item on the natural boundaries of the computer memory. (See 3.3.7, Item Alignment for Increased Object-Code Efficiency.)

6.45.2 GENERAL FORMAT

$$\left\{\begin{matrix}\underline{\text{SYNCHRONIZED}}\\ \underline{\text{SYNC}}\end{matrix}\right\} \quad \left[\begin{matrix}\underline{\text{LEFT}}\\ \underline{\text{RIGHT}}\end{matrix}\right]$$

6.45.3 SYNTAX RULES

1. This clause may only appear with an elementary item.

2. SYNC is an abbreviation for SYNCHRONIZED.

6.45.4 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 character positions redefined when this data 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. The specific positioning is, however, determined by the implementor.

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, as shown in the PICTURE clause, is used in determining any action that depends on size, such as justification, truncation or overflow.

Withdrawn/Retirée

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. This clause is hardware dependent and in addition to rules 1 through 7, the implementor must 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 automatically generated FILLER positions are included in:

      (1) The size of any group to which the FILLER item belongs; and

      (2) 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.

9. An implementor may, at his option, specify automatic alignment for any internal data formats except, within a record, data items whose usage is DISPLAY. However, the record itself may be synchronized.

10. Any rules for synchronization of the records of a data file, as this effects the synchronization of elementary items, will be specified by the implementor.

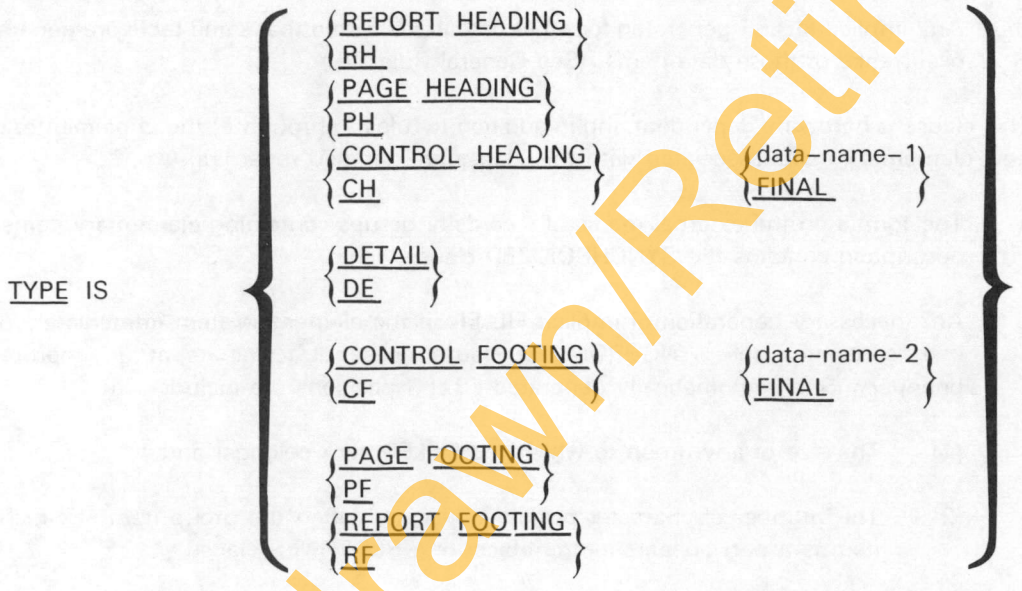
Withdrawn/Retirée

6.46 THE TYPE CLAUSE

6.46.1 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.

6.46.2 GENERAL FORMAT



6.46.3 SYNTAX RULES

1. RH is an abbreviation for REPORT HEADING.
   PH is an abbreviation for PAGE HEADING.
   CH is an abbreviation for CONTROL HEADING.
   DE is an abbreviation for DETAIL.
   CF is an abbreviation for CONTROL FOOTING.
   PF is an abbreviation for PAGE FOOTING.
   RF is an abbreviation for REPORT FOOTING.

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.

Withdrawn/Retirée

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.

6.46.4 GENERAL RULES

1. DETAIL report groups are processed by the 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.

Withdrawn/Retirée

(See 6.15.5, 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 7.48, 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 6.15.5, 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 7.48, 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.

Withdrawn/Retired

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. Otherwise, the RWCS formats the print lines and presents the report group according to the presentation rules for that type of report group. (See 6.15.5, 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 6.44, 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 6.44, 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. Otherwise 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 6.44, 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 paragraph 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

Withdrawn/Retired

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 6.44, 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. Otherwise the RWCS formats the print lines and presents the report group according to the presentation rules for DETAIL report groups. (See 6.15.5, 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.

Withdrawn/Retiree

6.47 THE USAGE CLAUSE

6.47.1 FUNCTION

The USAGE clause specifies the format of a data item in the computer storage.

6.47.2 GENERAL FORMAT

```
[ USAGE IS ]  { BIT
                COMPUTATIONAL
                COMP
                COMPUTATIONAL-n
                COMP-n
                DB-KEY
                DISPLAY
                DISPLAY-n
                INDEX
                INDEX-n }
```

6.47.3 SYNTAX RULES

1. The PICTURE character-string of a COMPUTATIONAL item can contain only '9's, the operational sign character 'S', the implied decimal point character 'V', one or more 'P's. (See 6.35, The PICTURE Clause.)

2. The USAGE clause for a report group item can specify only USAGE IS DISPLAY or USAGE IS DISPLAY-n.

3. COMP is an abbreviation for COMPUTATIONAL;
   COMP-n is an abbreviation for COMPUTATIONAL-n.

4. 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.

5. The SYNCHRONIZED, JUSTIFIED, PICTURE, VALUE and BLANK WHEN ZERO clauses cannot be used to describe group or elementary items described with the USAGE IS INDEX clause.

6. Boolean data items must not be described with USAGE clauses other than BIT or DISPLAY.

7. Data base key items may be referenced explicitly only in Formats 4 and 5 of the ACCEPT statement, in Formats 5 and 6 of the SET statement, in a relation condition, in a nullity condition, or in the USING phrase of either the CALL statement or the Procedure Division header.

Withdrawn/Retirée

8. The SYNCHRONIZED, JUSTIFIED, PICTURE, VALUE and BLANK WHEN ZERO clauses must not be used to describe data base key items.

6.47.3a SYNTAX RULES FOR THE SUB-SCHEMA ONLY

1. A data item defined with the USAGE IS DB-KEY clause must be defined in the Schema to be a data base key item.

6.47.4 GENERAL RULES

1. The USAGE clause can be written at any level. If the USAGE clause is written at a group level, it applies to each elementary item in the group. The USAGE clause of an elementary item cannot contradict the USAGE clause of a group to which the item belongs.

2. This 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. Suffixing a USAGE clause with a hyphen, followed by a single digit number (1-9), allows for several variations to a general type of USAGE. Any implementor who provides such variations (binary, decimal, and floating-point computations, etc.) assigns the variations in the order of their preference (that is, efficiency or frequency of use). If no suffix is written, or if the integer given in the suffix is unassigned, the recommended (first) variation is used.

4. A COMPUTATIONAL item is capable of representing a value to be used in computations and must be numeric. If a group item is described as COMPUTATIONAL, the elementary items in the group are COMPUTATIONAL. The group item itself is not COMPUTATIONAL (cannot be used in computations). If the USAGE clause specified at the group level is suffixed, the elementary items are considered to be the same, and may not have a contradicting form of computational usage specified.

5. The USAGE IS DISPLAY clause indicates that the format of the data is a Standard Data Format.

6. 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.

7. An elementary item described with the USAGE IS INDEX clause is called an index data item and contains a value which must correspond to an occurrence number of a table element. The elementary item cannot be a conditional variable. The method of representation and the actual value assigned are determined by the implementor. 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 cannot be used in the SEARCH or SET statement or in a relation condition.

8. An index data item can be part of a group which is referred to in a MOVE or input-output statement, in which case no conversion will take place.

Withdrawn/Retirée

9. The external and internal format of an index data item is specified by the implementor.

10. USAGE IS DISPLAY-n permits the implementor to specify variations of different character representations for those hardware units having this capability.

11. The USAGE IS BIT clause specifies that bits are to be used to represent a boolean data item.

12. An elementary item described with the USAGE IS DB-KEY clause is called a data base key item. It must not be a conditional variable. The method of representation and the actual value assigned are determined by the implementor. If a group item is described with the USAGE IS DB-KEY clause, the elementary items in the group are all data base key items. The group itself is not a data base key item and cannot be used in an ACCEPT or SET statement or in a relation condition.

13. A data base key item can be part of a group which is referenced in a MOVE or input-output statement, in which case no conversion of the data base key item takes place.

14. The external and internal format of a data base key item is implementor-defined.

Withdrawn/Retirée

6.48 THE VALUE CLAUSE

6.48.1 FUNCTION

The VALUE clause defines the value of Report Section printable items, the initial values of Communication Section and Working-Storage data items, and the values associated with condition-names.

6.48.2 GENERAL FORMAT

Format 1

```
VALUE IS literal
```

Format 2

```
{VALUE IS  }
{VALUES ARE} literal-1 [{THRU    } literal-2]
                       {THROUGH }
[ , literal-3 [{THRU    } literal-4] ] . . .
              {THROUGH }
```

6.48.3 SYNTAX RULES

1. The words THRU and THROUGH are equivalent.

2. The VALUE clause may be used with a variable-length data item. In this case, the initial value of the data item will be determined as if the data item were not described as variable-length, i.e., as if its PICTURE character-string did not contain the symbol 'L'.

3. A signed numeric literal must have associated with it a signed numeric PICTURE character-string.

4. 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 non-zero digits. Nonnumeric literals in a VALUE clause of an item must not exceed the size indicated by the PICTURE clause.

5. If the data item with which the VALUE clause is associated is a boolean data item, the THROUGH phrase must not be specified.

6.48.4 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:

Withdrawn/Retirée

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. (See 3.3.6, 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. The literal is aligned in the data item as if the data item had been described as alphanumeric. (See 3.3.6, Standard Alignment Rules.) Editing characters in the PICTURE clause are included in determining the size of the data item but have no effect on initialization of the data item. (See 6.35, The PICTURE Clause.) Therefore, the VALUE for an edited item is presented in an edited form.

c. If the category of the item is boolean, all literals in the VALUE clause must be boolean literals. Boolean literals are aligned in the data item according to the standard alignment rules. (See 3.3.6, Standard Alignment Rules.)

d. Initialization is not affected by any BLANK WHEN ZERO or JUSTIFIED clause that may be specified.

6.48.5 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-name. (See 3.2.2.1.1.1, Condition-Name.) Wherever the THRU phrase is used, literal-1 must be less than literal-2, literal-3 less than literal-4, etc.

6.48.6 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 and Linkage Section, the VALUE clause may be used only in condition-name entries.

   b. In the Working-Storage and Communication Sections, the VALUE clause must be used in condition-name entries and may be used to specify the initial value of data items. If the VALUE clause is not used in an item's description, the initial value is undefined.

   c. 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 6.26, The GROUP INDICATE Clause.)

Withdrawn/Retired

VALUE

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.

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 written for a group containing items with descriptions including JUSTIFIED, SYNCHRONIZED or USAGE (other than USAGE IS DISPLAY).

Withdrawn/Retirée

6.49 THE VALUE OF CLAUSE

6.49.1 FUNCTION

The VALUE OF clause particularizes the description of an item in the label records associated with a file.

6.49.2 GENERAL FORMAT

```
VALUE OF { , data-name-1 } IS { { data-name-2 } } . . .
                               { { literal-1   } }
```

6.49.3 SYNTAX RULES

1. Data-name-1, data-name-2, . . ., may be qualified. Data-name-1, data-name-2, . . ., must not be index items or data base key items.

2. Data-name-1, . . ., must reference a data item defined within a label record associated with the file whose File Description entry contains the VALUE OF clause; data-name-2, . . ., must be defined in the Working-Storage Section.

6.49.4 GENERAL RULES

1. For an Input File, the appropriate label routine checks to see if the value of the data item referenced by data-name-1 is equal to the value of literal-1, or of the data item referenced by data-name-2, whichever has been specified.

   For an Output File, at the appropriate time the value of the data item referenced by data-name-1 is made equal to the value of literal-1, or of the data item referenced by data-name-2, whichever has been specified.

Withdrawn/Retirée

CHAPTER 7

THE PROCEDURE DIVISION

7.1 GENERAL DESCRIPTION

The Procedure Division must be included in every COBOL source program. This division may contain Declaratives and nondeclarative procedures.

7.1.1 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. (See 7.50, The USE Statement.)

7.1.2 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.

The end of the Procedure Division and the physical end of the program is that physical position in a COBOL source program after which no further procedures appear.

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 a period and a space 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 a period followed by a space.

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.

7.1.3 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 in this chapter indicate some other order.

Withdrawn/Retirée

GENERAL DESCRIPTION

7.1.4 PROCEDURE DIVISION STRUCTURE

7.1.4.1 Procedure Division Header

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

PROCEDURE DIVISION [USING data-name-1 [, data-name-2] . . .].

The USING phrase is present 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.

Each of the operands in the USING phrase of the Procedure Division header must be defined as a data item in the Linkage Section of the program in which this header occurs, and it must have a 01 level-number.

Within a called program, Linkage Section data items are processed according to their data descriptions given in the called program.

When the USING phrase is present, the object program operates as if data-name-1, of the Procedure Division header in the called program and data-name-1 in the USING phrase of the CALL statement in the calling program refer to a single set of data that is equally available to both the called and calling programs. Except that they must define an equal number of character positions, their descriptions need not be the same. In like manner, there is an equivalent relationship between data-name-2, . . ., in the USING phrase of the called program and data-name-2, . . ., in the USING phrase of the CALL statement in the calling program. A data-name must not appear more than once in the USING phrase in the Procedure Division header of the called program; however, a given data-name may appear more than once in the USING phrase of a CALL statement.

Data items defined in the Linkage Section of the called program may be referenced within the Procedure Division of the called program if and only if they are:

1. Operands of the USING phrase of the Procedure Division header.
2. Subordinate to operands of the USING phrase of the Procedure Division header.
3. Defined with a REDEFINES or RENAMES clause, the object of which is an operand of the USING phrase of the Procedure Division header.
4. Items subordinate to any of the items defined in paragraph 3 above.
5. Condition-names and index-names associated with data items that meet any of the above conditions.

Withdrawn/Retirée

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

  END DECLARATIVES. ]

    {section-name SECTION      [ segment-number ] .
    [paragraph-name.      [ sentence ] . . .  ]  . . .}  . . .
```

Format 2

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

Withdrawn/Retirée

7.2 STATEMENTS AND SENTENCES

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

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

7.2.1 CONDITIONAL STATEMENTS AND CONDITIONAL SENTENCES

7.2.1.1 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:

a. An IF, SEARCH or RETURN statement.

b. A READ statement that specifies the AT END or INVALID KEY phrase.

c. A WRITE statement that specifies the INVALID KEY or END-OF-PAGE phrase.

d. A START, REWRITE or DELETE statement that specifies the INVALID KEY phrase.

e. An arithmetic statement (ADD, COMPUTE, DIVIDE, MULTIPLY, SUBTRACT) that specifies the SIZE ERROR phrase.

f. A RECEIVE statement that specifies a NO DATA phrase.

g. A STRING or UNSTRING statement that specifies the ON OVERFLOW phrase.

h. A CALL statement that specifies the ON EXCEPTION phrase.

7.2.1.2 Definition of Conditional Sentence

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

Withdrawn/Retirée

7.2.2 COMPILER DIRECTING STATEMENTS AND COMPILER DIRECTING SENTENCES

7.2.2.1 Definition of Compiler Directing Statement

A compiler directing statement consists of a compiler directing verb and its operands. The compiler directing verbs are COPY, ENTER and USE. (See 9.2, The COPY Statement; 7.19, The ENTER Statement; 7.50, The USE Statement.) A compiler directing statement causes the compiler to take a specific action during compilation.

7.2.2.2 Definition of Compiler Directing Sentence

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

7.2.3 IMPERATIVE STATEMENTS AND IMPERATIVE SENTENCES

7.2.3.1 Definition of Imperative Statement

An imperative statement indicates a specific unconditional action to be taken by the object program. An imperative statement is any statement that is neither a conditional statement, nor a compiler directing 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	ERASE	MODIFY	SEND
ADD (1)	EXIT	MOVE	SET
CALL (7)	FIND	MULTIPLY (1)	SORT
CANCEL	FINISH	OPEN	START (2)
CLOSE	FREE	ORDER	STOP
COMPUTE (1)	GENERATE	PERFORM	STORE
CONNECT	GET	PURGE	STRING (3)
DELETE (2)	GO TO	READ (5)	SUBTRACT (1)
DISABLE	INITIALIZE	READY	SUPPRESS
DISCONNECT	INITIATE	RECEIVE (4)	SUSPEND
DISPLAY	INSPECT	RELEASE	TERMINATE
DIVIDE (1)	KEEP	REMONITOR	UNSTRING (3)
ENABLE	MERGE	REWRITE (2)	WRITE (6)

(1) Without the optional SIZE ERROR phrase.
(2) Without the optional INVALID KEY phrase.
(3) Without the optional ON OVERFLOW phrase.
(4) Without the optional NO DATA phrase.
(5) Without the optional AT END phrase or INVALID KEY phrase.
(6) Without the optional INVALID KEY phrase or END-OF-PAGE phrase.
(7) Without the optional ON EXCEPTION phrase.

Withdrawn/Retired

STATEMENTS AND SENTENCES

Whenever 'imperative-statement' appears in the General Format of statements described in this Chapter, imperative-statement' refers to that sequence of consecutive imperative statements that must be ended by a period or an ELSE phrase associated with a previous IF statement or a WHEN phrase associated with a previous SEARCH statement.

7.2.3.2 Definition of Imperative Sentence

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

Withdrawn/Retirée

7.3 ARITHMETIC EXPRESSIONS

7.3.1 DEFINITION OF AN ARITHMETIC EXPRESSION

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 operators and parentheses are given in Figure 7.3-1, 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.

7.3.2 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 Arithmetic Operator	Meaning
+	Addition
–	Subtraction
*	Multiplication
/	Division
**	Exponentiation

Unary Arithmetic Operator	Meaning
+	The effect of multiplication by the numeric literal +1
–	The effect of multiplication by the numeric literal –1

7.3.3 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:

Withdrawn/Retirée

ARITHMETIC EXPRESSIONS

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 (Figure 7.3-1), where:

a. The letter 'P' indicates a permissible pair of symbols.
b. The character '–' indicates an invalid pair.
c. 'Variable' indicates an identifier or literal.

FIRST SYMBOL	SECOND SYMBOL				
	Variable	*/** + –	Unary + or –	(	)
Variable	–	P	–	–	P
*/** + –	P	–	P	P	–
Unary + or –	P	–	–	P	–
(	P	–	P	P	–
)	–	P	–	–	P

Figure 7.3-1 Combination of Symbols in Arithmetic Expressions.

4. An arithmetic expression may only begin with the symbol '(', '+', '-', or a variable and may only end with a ')' or a variable. 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.

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 7.7.3, The SIZE ERROR Option.)

Withdrawn/Retiree

b. If the evaluation yields both a positive and a 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. When no reference to a resultant-identifier exists in a statement, an intermediate data item is used to store the results of the arithmetic expression. (See 7.7.1, Intermediate Data Item.)

Withdrawn/Retirée

7.4 BOOLEAN EXPRESSIONS

7.4.1 DEFINITION OF A BOOLEAN EXPRESSION

A boolean expression is an identifier of a boolean data item, a boolean literal, such identifiers and/or literals separated by a boolean operator, two boolean expressions separated by a boolean operator or a boolean expression enclosed in parentheses. Any boolean expression may be preceded by the unary boolean operator. The permissible combinations of variables, boolean literals, boolean operators and parentheses are given in Figure 7.4-1, Combination of Symbols in Boolean Expressions.

7.4.2 BOOLEAN OPERATORS

There are three binary boolean operators and one unary boolean operator that may be used in boolean expressions. They are represented by the following reserved words that must be both preceded and followed by the separator space.

Binary Boolean Operator	Meaning
AND	Boolean Conjunction
OR	Boolean Inclusive Disjunction
EXOR	Boolean Exclusive Disjunction

Unary Boolean Operator	Meaning
NOT	Boolean Negation

7.4.3 BOOLEAN FORMATION AND EVALUATION RULES

1. Whenever two boolean expressions are separated only by 'AND', 'OR' or 'EXOR', or whenever a boolean expression is immediately preceded by 'NOT', the 'AND', 'OR', 'EXOR' or 'NOT' is a boolean operator.

2. Parentheses may be used in boolean 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 – Negation (NOT)
   2nd – Conjunction (AND)
   3rd – Disjunction (OR and EXOR)

Withdrawn/Retiree

3. 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.

4. The ways in which boolean operators, variables and parentheses may be combined in a boolean expression are summarized in Figure 7.4-1, Combination of Symbols in Boolean Expressions.

   In this figure:

   a. The letter 'P' indicates a permissible pair.

   b. The character '-' indicates an invalid pair.

   c. 'Variable' indicates a boolean literal or an identifier that references a boolean data item.

First Symbol	Second Symbol				
	Variable	AND OR EXOR	NOT	(	)
Variable	–	P	–	–	P
AND, OR, EXOR	P	–	P	P	–
NOT	P	–	–	P	–
(	P	–	P	P	–
)	–	P	–	–	P

Figure 7.4-1 Combination of Symbols in Boolean Expressions.

5. A boolean expression may only begin with the symbol '(', an identifier that references a boolean data item, a boolean literal or the operator 'NOT'. A boolean expression may only end with the symbol ')', an identifier that references a boolean data item or a boolean literal. There must be a one-to-one correspondence between left and right parentheses of a boolean expression such that each left parenthesis is to the left of its corresponding right parenthesis.

6. Conjunction, disjunction and exclusive disjunction of boolean operands are made regardless of their usage. If the operands of a conjunction, disjunction or exclusive disjunction are of equal size, then the operation proceeds by conjoining, disjoining or exclusively disjoining boolean characters in corresponding boolean character positions starting from the high order end and continuing to the lower order end. If the operands are of unequal length, then the operation proceeds as though the shorter operand was

Withdrawn/Retiree

extended on the low order end by a sufficient number of boolean zeroes to make the operands of equal size.

Withdrawn/Retirée

7.5 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 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.

7.5.1 SIMPLE CONDITIONS

The simple conditions are the relation, class, condition-name, switch-status, sign, tenancy, member, and nullity 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.

7.5.1.1 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, or the value resulting from an arithmetic expression or a boolean expression. A relation condition has a truth value of 'true' if the relation exists between the operands. Comparison of two numeric operands or two boolean 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; however, if one of the operands is a boolean expression, both operands must be boolean expressions.

The format for a relation condition not involving boolean expressions is as follows:

```
{identifier-1   }                                                {identifier-2   }
{literal-1      }  { IS [ NOT ] GREATER THAN                     {literal-2      }
{arithmetic-    }  { IS [ NOT ]         >                        {arithmetic-    }
{  expression-1 }  { IS [ NOT ] LESS THAN                        {  expression-2 }
                   { IS [ NOT ]         <
                   { IS [ NOT ] EQUAL TO
                   { IS [ NOT ]         =
                   { IS UNEQUAL TO
                   { EQUALS
                   { EXCEEDS
```

The format for a relation condition involving boolean expressions is as follows:

```
boolean-expression-1   { IS [ NOT ] EQUAL TO   }   boolean-expression-2
                       { IS [ NOT ]         =  }
                       { IS UNEQUAL TO         }
                       { EQUALS                }
```

The first operand (identifier-1, literal-1, arithmetic-expression-1 or boolean-expression-1) is called the subject of the condition; the second operand (identifier-2, literal-2, arithmetic-expression-2 or boolean-

Withdrawn/Retiree

expression-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 as follows:

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] =
Not equal to	IS UNEQUAL TO
Equal to	EQUALS
Greater than	EXCEEDS

7.5.1.1.1 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 the number of digits represented, is not significant. Zero is considered a unique value regardless of the 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.

7.5.1.1.2 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. (See 5.3.2, The OBJECT-COMPUTER Paragraph and 7.40, The SET Statement.) If one of the operands is specified as numeric, it must be an integer data item or an integer literal and:

a. 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 contents of this

Withdrawn/Retired

alphanumeric data item were then compared to the nonnumeric operand. (See 7.28.4, The MOVE Statement and 6.35.4.8 The PICTURE Character 'P'.)

b. 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 contents of this group item were then compared to the nonnumeric operand. (See 7.28.4, The MOVE Statement and 6.35.4.10 The PICTURE Character 'P'.)

c. A non-integer 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 characters compare equally through the last pair, when the low order end is reached.

   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. (See 7.5.1.1.2.1, Operands of Equal Size.)

7.5.1.1.3 Comparisons Involving Index-Names and/or Index Data Items

Relation tests may be made 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) or literal. 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.

Withdrawn/Retiree

4. The result of the comparison of an index data item with any data item or literal not specified above is undefined.

7.5.1.1.4 Comparison of Boolean Operands

A comparison of boolean operands is made regardless of their usage. If the operands are of equal size, comparison effectively proceeds by comparing boolean characters in corresponding boolean character positions starting from the high order end and continuing until either a pair of unequal boolean 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 boolean characters compare equally through the last pair, when the low order end is reached. If the operands are of unequal size, comparison proceeds as though the shorter operand were extended on the right by sufficient boolean character zeros to make the operands of equal size.

7.5.1.1.5 Comparisons of Data Base Key Items

Relation tests may be made between two data base key items only to determine whether the two items contain equal values. The result of a comparison of two data base key items that is not a comparison to determine equality or inequality is undefined. The result of a comparison of a data base key item with any data item that is not a data base key item is undefined.

7.5.1.2 Class Condition

The class condition determines whether the operand is numeric, that is, consists entirely of the characters '0', '1', '2', '3', . . ., '9', with or without an operational sign, or alphabetic, that is, consists entirely of the characters 'A', 'B', 'C', . . ., 'Z', space. The general format for the class condition is as follows:

identifier IS [NOT] { NUMERIC / ALPHABETIC }

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 contents 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 contents 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 '-'; the implementor defines what constitutes valid sign(s) for data items not described with the SIGN IS SEPARATE clause.

Withdrawn/Retired

The ALPHABETIC test cannot be used with an item whose data description describes the item as numeric. The item being tested is determined to be alphabetic only if the contents consist of any combination of the alphabetic characters 'A' through 'Z' and the space.

7.5.1.3 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 a condition-name. The general format for the condition-name condition is as follows:

condition–name

If the condition-name 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 the condition-name equals the value of its associated conditional variable.

7.5.1.4 Switch-Status Condition

A switch-status condition determines the 'on' or 'off' status of an implementor-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 as follows:

condition–name

The result of the test is true if the switch is set to the specified position corresponding to the condition-name.

7.5.1.5 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 as follows:

arithmetic–expression 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

Withdrawn/Retirée

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. The arithmetic expression must contain at least one reference to a variable.

7.5.1.6 Tenancy Condition

The tenancy condition is used only in a data base environment. (See 12.5, Data Base Conditions.)

7.5.1.7 Member Condition

The member condition is used only in a data base environment. (See 12.5, Data Base Conditions.)

7.5.1.8 Nullity Condition

The nullity condition is used only in a data base environment. (See 12.5, Data Base Conditions.)

7.5.2 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 all the stated logical operators on the individual truth values of simple conditions, or the intermediate truth values of conditions logically connected or logically negated.

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 by a space and followed by a space.

(See 7.4.3, Boolean Formation and Evaluation Rules.)

Withdrawn/Retirée

7.5.2.1 Negated Simple Conditions

A simple condition is negated through the use of the logical operator 'NOT'. (See 7.5.1, Simple Conditions.) The negated simple condition effects the opposite truth value for a simple condition. Thus the truth value of a negated simple condition is 'true' if and only if the truth value of the simple condition is 'false'; the truth value of a negated simple condition is 'false' if and only if the truth value of the simple condition is 'true'. The inclusion in parentheses of a negated simple condition does not change the truth value.

The general format for a negated simple condition is:

NOT simple-condition

7.5.2.2 Combined and Negated 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 { { AND / OR } condition } . . .

Where 'condition' may be:

1. A simple condition, or
2. A negated simple condition, or
3. A combined condition, or
4. A negated combined condition; i.e., the 'NOT' logical operator followed by a combined condition enclosed within parentheses, or
5. Combinations of the above, specified according to the rules summarized in Figure 7.5-1, Combinations of Conditions, Logical Operators, and Parentheses.

Although parentheses need never be used when either 'AND' or 'OR' (but not both) is used exclusively in a combined condition, parentheses may be used to effect a final truth value when a mixture of 'AND', 'OR' and 'NOT' is used. (See Figure 7.5-1, Combinations of Conditions, Logical Operators, and Parentheses; and 7.5.4, Condition Evaluation Rules.)

Figure 7.5-1 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.

Withdrawn/Retirée

CONDITIONAL EXPRESSIONS

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-condition	Yes	Yes	OR, NOT, AND, (	OR, AND,)
OR or AND	No	No	simple-condition,)	simple-condition, NOT, (
NOT	Yes	No	OR, AND, (	simple-condition, (
(	Yes	No	OR, NOT, AND, (	simple-condition, NOT, (
)	No	Yes	simple-condition,)	OR, AND,)

Figure 7.5-1
Combinations of Conditions, Logical Operators, and Parentheses

Thus, the element pair 'OR NOT' is permissible while the pair 'NOT OR' is not permissible; 'NOT (' is permissible while 'NOT NOT' is not permissible.

7.5.3 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 preceeding 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 } . . .

Withdrawn/Retirée

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 Figure 7.5-1. 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))
a / b UNEQUAL c AND NOT d	((a / b) UNEQUAL c) AND (NOT ((a / b) UNEQUAL d))
NOT (a NOT > b AND c AND NOT d)	NOT ((((a NOT > b) AND (a NOT > c)) AND (NOT (a NOT > d))))

7.5.4 CONDITION EVALUATION RULES

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. (See 7.3.3, Formation and Evaluation Rules.)

Withdrawn/Retirée

CONDITIONAL EXPRESSIONS

2. Truth values for simple conditions are established in the following order:

   relation
   class
   condition–name
   switch–status
   sign

3. Truth values for negated simple conditions are established.

4. Truth values for combined conditions are established:

   'AND' logical operators, followed by
   'OR' logical operators.

5. Truth values for negated combined conditions are established.

6. 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.

Withdrawn/Retirée

7.6 CATEGORIES OF STATEMENTS

Category	Verbs
Arithmetic	ADD COMPUTE DIVIDE MULTIPLY SUBTRACT
Boolean	COMPUTE
Compiler Directing	COPY ENTER USE
Conditional	ADD (SIZE ERROR) CALL (EXCEPTION) COMPUTE (SIZE ERROR) DELETE (INVALID KEY) DIVIDE (SIZE ERROR) IF MULTIPLY (SIZE ERROR) READ (AT END or INVALID KEY) RECEIVE (NO DATA) RETURN (AT END) REWRITE (INVALID KEY) SEARCH (AT END) START (INVALID KEY) STRING (OVERFLOW) SUBTRACT (SIZE ERROR) UNSTRING (OVERFLOW) WRITE (INVALID KEY or END-OF-PAGE)

Withdrawn/Retirée

CATEGORIES OF STATEMENTS

Category	Verbs
Data Base Manipulation	CONNECT DISCONNECT ERASE FIND FINISH FREE GET KEEP MODIFY ORDER READY REMONITOR STORE
Data Movement	INITIALIZE INSPECT (CONVERTING) INSPECT (REPLACING) MOVE STRING UNSTRING
Ending	STOP
Input-Output	ACCEPT CLOSE DELETE DISABLE DISPLAY ENABLE OPEN PURGE READ RECEIVE REWRITE SEND START STOP (literal) SUSPEND WRITE
Inter-Program Communicating	CALL CANCEL

Withdrawn/Retirée

CATEGORIES OF STATEMENTS

Category	Verbs
Ordering	MERGE RELEASE RETURN SORT
Procedure Branching	CALL EXIT GO TO PERFORM
Report Writing	GENERATE INITIATE SUPPRESS TERMINATE
Table Handling	SEARCH SET

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

7.6.1 SPECIFIC STATEMENT FORMATS

The specific statement formats, together with a detailed discussion of the restrictions and limitations associated with each, appear hereafter in alphabetic order. (See 7.8 et seq.)

Withdrawn/Retirée

7.7 COMMON OPTIONS AND RULES FOR STATEMENT FORMATS

In the statement descriptions that follow, several options appear frequently: the ROUNDED phrase, the SIZE ERROR phrase, the CORRESPONDING phrase, the FROM phrase and the INTO phrase.

7.7.1 INTERMEDIATE DATA ITEM

An intermediate data item is a signed numeric data item provided by the implementor that contains the results developed in the course of an arithmetic operation prior to the final result being moved to the resultant-identifier, if any. This data item is 18 digits in length and contains the 18 most significant digits of the result being developed during the execution of an arithmetic operation. During execution of an arithmetic operation, the magnitude of the mathematical result is maintained and all low-order digits which are truncated are considered to be zero for the remainder of this arithmetic operation.

7.7.2 THE ROUNDED OPTION

At the completion of an arithmetic operation for which rounding is specified, the absolute value of the low-order digit of the resultant-identifier is increased by one (1) whenever the absolute value of the next least significant digit of the intermediate data item is greater than or equal to five (5).

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.

7.7.3 THE SIZE ERROR OPTION

If, after decimal point alignment, the absolute value of a result exceeds the largest value that can be contained in the associated resultant-identifier, a size error condition exists. The algebraic value of the final result of the arithmetic operation(s) must be accurate to the precision specified by the resultant-identifier. Division by zero always terminates the arithmetic operation and always causes a size error condition. Violation of the rules for evaluation of exponentiation always terminates the arithmetic operation and always causes a size error condition. (See 7.3.3, Formation and Evaluation Rules.) The size error condition applies only to the final results of an arithmetic operation and does not apply to intermediate results. If the ROUNDED phrase is specified, rounding takes place before checking for size error. When such a size error condition occurs, the subsequent action depends on whether or not the SIZE ERROR phrase is specified.

1. If the SIZE ERROR phrase is not specified and a size error condition occurs, the value of those resultant-identifier(s) affected is undefined. Values of resultant-identifier(s) for which no size error condition occurs are unaffected by size errors that occur for other resultant-identifier(s) during execution of this operation.

2. If the SIZE ERROR phrase is specified and a size error condition occurs, then the values of resultant-identifier(s) affected by the size errors are not altered. Values of resultant-identifier(s) for which no size error condition occurs are unaffected by size errors that occur for other resultant-

Withdrawn/Retirée

identifier(s) during execution of this operation. After completion of the execution of this operation, the imperative statement in the SIZE ERROR phrase is executed.

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, the imperative statement in the SIZE ERROR phrase is not executed until all of the individual additions or subtractions are completed.

7.7.4 THE CORRESPONDING OPTION

For the purposes of this discussion, identifier-1, identifier-2, identifier-3, ..., 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 or with the USAGE IS INDEX clause.

   e. No identifier may be reference modified.

2. Data items subordinate to identifier-1 correspond with data items subordinate to identifier-2, identifier-3, ..., 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, identifier-3, ...

   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.

Withdrawn/Retirée

f. FILLER data items and data items subordinate thereto are ignored.

7.7.5 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 eighteen (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 contents 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. (See 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; and 7.28, The MOVE Statement.)

7.7.6 OVERLAPPING OPERANDS

When a sending and a receiving item in an arithmetic statement or an INSPECT, MOVE, SET, STRING or UNSTRING statement share a part of their storage areas, the result of the execution of such a statement is undefined.

7.7.7 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 which performs all arithmetic necessary to arrive at the result to be stored in the receiving items, and stores that result in a temporary storage location.

2. A sequence of statements transferring or combining the value of this temporary location with a single result. These statements are considered to be written in the same left-to-right sequence that the multiple results are listed.

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
```

where 'temp' is an intermediate result item provided by the implementor.

Withdrawn/Retirée

7.7.8 INCOMPATIBLE DATA

Except for the class condition, when the contents of a data item are referenced in the Procedure Division and the contents of that data item are 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, 7.5.1.2.)

7.7.9 I-O STATUS

If the FILE STATUS clause is specified in a File Description entry, a value is placed into the specified two-character data item during the execution of a CLOSE, DELETE, DELETE FILE, OPEN, READ, REWRITE, START or WRITE statement and before any applicable Format 1 USE procedure is executed, to indicate to the COBOL program the status of that input-output operation.

7.7.9.1 Status Key 1

The leftmost character position of the FILE STATUS data item is known as Status Key 1 and is set to indicate one of the following conditions upon completion of the input-output operation.

'0' indicates Successful Completion

'1' indicates At End

'2' indicates Invalid Key

'3' indicates Permanent Error

'9' indicates Implementor Defined

The meaning of the above indications are as follows:

0 - Successful Completion. The input-output statement was successfully executed.

1 - At End. A sequential READ statement was unsuccessfully executed as a result of one of the following:

a. No next logical record exists in the file.

b. An optional file was not present.

2 - Invalid Key. The input-output statement was unsuccessfully executed as a result of one of the following:

Sequence Error
Duplicate Key

Withdrawn/Retirée

No Record Found
Boundary Violation
Optional File Not Present

3 - Permanent Error. The input-output statement was unsuccessfully executed as the result of a boundary violation for a sequential file or as the result of an input-output error, such as data check, parity error, or transmission error.

9 - Implementor Defined. The input-output statement was unsuccessfully executed as a result of a condition that is specified by the implementor. This value is used only to indicate a condition not indicated by other defined values of Status Key 1, or by specified combinations of the values of Status Key 1 and Status Key 2.

7.7.9.2 Status Key 2

The rightmost character position of the FILE STATUS data item is known as Status Key 2 and is used to further describe the results of the input-output operation. This character will contain a value as follows:

1. If no further information is available concerning the input-output operation, then Status Key 2 contains a value of '0'.

2. When Status Key 1 contains a value of '0' indicating a Successful Completion, Status Key 2 may contain a value of '2' indicating a Duplicate Key, or '5' indicating file not present. These conditions indicate:

   a. For a READ statement, the value '2' indicates that the key value for the current Key of Reference is equal to the value of that same key in the next record within the current Key of Reference.

   b. For a REWRITE or WRITE statement, the value '2' indicates that the record just written created a duplicate key value for at least one Alternate Record Key for which duplicates are allowed.

   c. For a DELETE FILE statement, the value '5' indicates that the referenced file is not available at the time of the execution of the DELETE FILE statement.

3. When Status Key 1 contains a value of '1' indicating an at end condition, Status Key 2 is used to designate the cause of the condition as follows:

   a. A value of '3' in Status Key 2 indicates no next logical record exists in the file.

   b. A value of '5' in Status Key 2 indicates an optional file is not present.

4. When Status Key 1 contains a value of '2' indicating an invalid key condition, Status Key 2 is used to designate the cause of that condition as follows:

Withdrawn/Retired

a. A value of '1' in Status Key 2 indicates a Sequence Error for a sequentially accessed indexed file. The Prime Record Key value has been changed by the COBOL program between the successful execution of a READ statement and the execution of the next REWRITE statement for that file, or the ascending sequence requirements for successive Record Key values have been violated. (See 7.51.4.20, The WRITE Statement.)

b. A value of '2' in Status Key 2 indicates a Duplicate Key value. An attempt has been made to write or rewrite a record that would create a duplicate key in an indexed file, or an attempt has been made to write a record that would create a duplicate key in a relative file.

c. A value of '3' in Status Key 2 indicates No Record Found. An attempt has been made to access a record, identified by a key, and that record does not exist in the file.

d. A value of '4' in Status Key 2 indicates a Boundary Violation. An attempt has been made to write beyond the externally-defined boundaries of a relative or indexed file. The implementor specifies the manner in which these boundaries are defined.

e. A value of '5' in Status Key 2 indicates an optional file is not present.

5. When Status Key 1 contains a value of '3' indicating a permanent error condition, Status Key 2 may contain a value of '4' indicating a boundary violation. This condition indicates that an attempt has been made to write beyond the externally-defined boundaries of a sequential file. The implementor specifies the manner in which these boundaries are defined.

6. When Status Key 1 contains a value of '9' indicating an implementor-defined condition, the value of Status Key 2 is defined by the implementor.

7.7.9.3 Valid Combinations of Status Keys 1 and 2

The valid permissible combinations of the values of Status Key 1 and Status Key 2 are shown in Figure 7.7-1, I-O Status Indicators. An 'X' at an intersection indicates a valid permissible combination.

Withdrawn/Retired

OPTIONS

Status Key 1 \ Status Key 2	No Further Information (0)	Sequence Error (1)	Duplicate Key (2)	No Record Found (3)	Boundary Violation (4)	File Not Present (5)
Successful Completion (0)	X		X			X
At End (1)	X			X		X
Invalid Key (2)		X	X	X	X	X
Permanent Error (3)	X				X	
Implementor Defined (9)						

Figure 7.7-1 I-O Status Indicators

7.7.10 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. 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 paragraphs 7.14, The DELETE Statement; 7.33, The READ Statement; 7.37, The REWRITE Statement; 7.42, The START Statement; and 7.51, The WRITE Statement.)

When the invalid key condition is recognized, the MSCS takes these actions in the following order:

1. A value is placed in the FILE STATUS data item, if specified for this file, to indicate an invalid key condition. (See 7.7.9, I-O STATUS.)

2. If the INVALID KEY phrase is specified in the statement causing the condition, control is transferred to the INVALID KEY imperative-statement. Any USE AFTER EXCEPTION procedure specified for this file is not executed.

Withdrawn/Retiree

3. If the INVALID KEY phrase is not specified, but a USE AFTER EXCEPTION procedure is specified for this file, that procedure is executed.

7.7.11 THE AT END CONDITION

The at end condition can occur as a result of the execution of a READ, RETURN or SEARCH statement. (See 7.33, The READ Statement; 7.36, The RETURN Statement; and 7.38 The SEARCH Statement.)

7.7.12 THE FROM OPTION

The format of the FROM phrase is:

record-name FROM identifier

Record-name and identifier must not refer to the same storage area.

The results of the execution of a RELEASE, REWRITE or WRITE statement with the FROM phrase is equivalent to the execution of the following statements in the order specified:

1. The statement:

   MOVE identifier TO record-name

   according to the rules specified for the MOVE statement.

2. The same RELEASE, REWRITE or WRITE statement without the FROM phrase.

After the execution of the RELEASE, REWRITE or WRITE statement is complete, the information in the area referenced by indentifier is available, even though the information in the area referenced by record-name is not available except as specified by the SAME AREA clause. (See 5.4.2, The I-O-CONTROL Paragraph.)

7.7.13 THE INTO OPTION

The format of the INTO phrase is:

file-name INTO identifier

The storage area associated with identifier and record area associated with the file must not be the same storage area.

The INTO phrase may be specified in READ or RETURN statements:

Withdrawn/Retiree

OPTIONS

1. If only one record description is subordinate to the File Description entry, or

2. If all record-names associated with the file and the data item associated with identifier describe a group item or an elementary alphanumeric item.

The results of the execution of a READ or RETURN statement with the INTO phrase is equivalent to the application of the following rules in the order specified;

1. The execution of the same READ or RETURN statement without the INTO phrase.

2. The current record is moved from the input area to the area specified by identifier 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 or RETURN statement was unsuccessful. Any subscripting or indexing associated with identifier 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 area associated with identifier.

Withdrawn/Retirée

7.8 THE ACCEPT STATEMENT

7.8.1 FUNCTION

The ACCEPT statement causes low volume data to be made available to the specified data item.

7.8.2 GENERAL FORMAT

Format 1

```
ACCEPT identifier-1          [ FROM mnemonic-name ]
```

Format 2

```
ACCEPT identifier-2 FROM { DATE
                         { DAY
                         { DAY-OF-WEEK
                         { TIME
```

Format 3

```
ACCEPT cd-name MESSAGE COUNT
```

Format 4

```
ACCEPT identifier-3 FROM [ record-name ]
                         [ set-name    ]   CURRENCY
                         [ realm-name  ]
```

Format 5

```
ACCEPT identifier-4 FROM [ record-name ]
                         [ set-name    ]   REALM-NAME
                         [ identifier-5 ]
```

7.8.3 SYNTAX RULES

1. The mnemonic-name 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. Cd-name must reference an input CD.

3. Identifier-2 must not reference a boolean data item.

Withdrawn/Retirée

ACCEPT

7.8.4 GENERAL RULES

FORMAT 1

1. The ACCEPT statement causes the transfer of data from the hardware device. This data replaces the contents of the data item named by identifier-1.

2. The implementor will define, for each hardware device, the size of a data transfer.

3. If a hardware device is capable of transferring data of the same size as the receiving data item, the transferred data is stored in the receiving data item.

4. If a hardware device is not capable of transferring data of the same size as the receiving data item, then:

   a. If the size of the receiving data item (or of the portion of the receiving data item not yet currently occupied by transferred data) exceeds the size of the transferred data, the transferred data is stored aligned to the left in the receiving data item (or the portion of the receiving data item not yet occupied), and additional data is requested.

   b. If the size of the transferred data exceeds the size of the receiving data item (or the portion of the receiving data item not yet occupied by transferred data), only the leftmost characters of the transferred data are stored in the receiving data item (or in the portion remaining). The remaining characters of the transferred data which do not fit into the receiving data item are ignored.

5. If the FROM option is not given, the device that the implementor specifies as standard is used.

FORMAT 2

6. The ACCEPT statement causes the information requested to be transferred to the data item specified by identifier-2 according to the rules for the MOVE statement. (See 7.28, The MOVE Statement.) DATE, DAY, DAY-OF-WEEK and TIME are conceptual data items and, therefore, are not described in the COBOL program.

7. 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, July 1, 1968 would be expressed as 680701. DATE, 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 six digits in length.

8. 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, July 1, 1968 would be expressed as 68183. 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.

Withdrawn/Retirée

9. 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. If the hardware does not have the facility to provide fractional parts of TIME, the value is converted to the closest decimal approximation.

10. DAY-OF-WEEK is composed of a single data element whose content represents the day of the week. DAY-OF-WEEK, 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 one (1) digit in length. In DAY-OF-WEEK, the value one (1) represents Monday, 2 represents Tuesday, . . . 7 represents Sunday.

FORMAT 3

11. The ACCEPT statement causes the MESSAGE COUNT field specified for cd-name to be updated to indicate the number of messages that exist in a queue, sub-queue-1, ...

12. Upon execution of the ACCEPT statement, the contents of the area specified by a Communication Description entry must contain at least the name of the symbolic queue to be tested. Testing the condition causes the contents of the data items referenced by data-name-10 (STATUS KEY) and data-name-11 (MESSAGE COUNT) of the area associated with the Communication Description entry to be appropriately updated. (See 6.12.4.14, The Communication Description — Complete Entry Skeleton.)

FORMATS 4 AND 5

13. An explanation of the use of Formats 4 and 5 is given under the appropriate paragraph. (See 12.10, The ACCEPT Statement.)

Withdrawn/Retired

7.9 THE ADD STATEMENT

7.9.1 FUNCTION

The ADD statement causes two or more numeric operands to be summed and the result to be stored.

7.9.2 GENERAL FORMAT

Format 1

```
ADD  { literal-1    }  [ , literal-2    ]
     { identifier-1 }  [ , identifier-2 ]  . . .  TO identifier-m [ ROUNDED ]
  [ , identifier-n [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 2

```
ADD  { literal-1    }  { , literal-2    }  [ , literal-3    ]  . . .
     { identifier-1 }  { , identifier-2 }  [ , identifier-3 ]
     GIVING  identifier-m  [ ROUNDED ]
  [ , identifier-n [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 3

```
ADD  { CORRESPONDING }  identifier-1 TO identifier-2  [ ROUNDED ]
     { CORR          }
  [; ON SIZE ERROR imperative-statement]
```

7.9.3 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. Each literal must be a numeric literal.

3. CORR is an abbreviation for CORRESPONDING.

Withdrawn/Retirée

7.9.4 GENERAL RULES

1. If Format 1 is used, the values of the operands preceding the word TO are added together, then the sum is added to the current value of identifier-m storing the result immediately into identifier-m, and repeating this process respectively for each recurrence of identifier-n.

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 value of each identifier-m, identifier-n, . . ., the resultant-identifiers.

3. If Format 3 is used, data items in identifier-1 are added to and stored in corresponding data items in identifier-2.

4. Addition rules and explanations relative to this statement are given under the appropriate paragraphs. (See 7.7.1, Intermediate Data Item; 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; 7.7.4, The CORRESPONDING Option; 7.7.5, The Arithmetic Statements; 7.7.6, Overlapping Operands; and 7.7.7, Multiple Results in Arithmetic Statements.)

Withdrawn/Retirée

7.10 THE CALL STATEMENT

7.10.1 FUNCTION

The CALL statement causes control to be transferred from one object program to another, within the run unit.

7.10.2 GENERAL FORMAT

```
CALL  { literal-1      }
      { identifier-1   }

        [ USING data-name-1      [ ,      data-name-2 ]    . . . ]

        [ ; ON EXCEPTION imperative-statement ]
```

7.10.3 SYNTAX RULES

1. Literal-1 must be a nonnumeric literal.

2. Identifier-1 must be defined as an alphanumeric data item such that its value can be a program-name.

3. 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 and the number of operands in each USING phrase must be identical.

4. Each of the operands in the USING phrase must have been defined as a data item in the File Section, Working-Storage Section, Communication Section or Linkage Section, and must be a level 01 data item or an elementary data item. Data-name-1, data-name-2, . . ., may be qualified when they reference data items defined in the File Section or the Communication Section.

7.10.4 GENERAL RULES

1. The program whose name is specified by the value of literal-1 or identifier-1 is the called program; the program in which the CALL statement appears is the calling program.

2. The execution of a CALL statement causes control to pass to the called program.

3. A called program 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.

Withdrawn/Retirée

On all other entries into the called program, the state of the program remains unchanged from its state when last exited. This includes all data fields, the status and positioning of all files, and all alterable switch settings.

4. If during the execution of a CALL statement, it is determined that the available object time resources (e.g., memory, hardware device) are incapable of accommodating the program specified in the CALL statement and the ON EXCEPTION phrase is specified, no action is taken and the imperative-statement is executed. Those resources whose availability is checked during execution of the CALL statement are implementor defined.

   If the above condition exists and the ON EXCEPTION phrase is not specified, the effects of the CALL statement are defined by the implementor.

5. Called programs may contain CALL statements. However, a called program must not contain a CALL statement that directly or indirectly calls the calling program.

6. The data-names, specified by the USING phrase of the CALL statement, indicate those data items available to a calling program that may be referred to in the called program. The order of appearance of the data-names in the USING phrase of the CALL statement and the USING phrase in the Procedure Division header is critical. Corresponding data-names refer to a single set of data which is available to the called and calling program. The correspondence is positional, not by name. In the case of index-names, no such correspondence is established. Index-names in the called and calling program always refer to separate indices.

Withdrawn/Retired

7.11 THE CANCEL STATEMENT

7.11.1 FUNCTION

The CANCEL statement releases the memory areas occupied by the referred to program.

7.11.2 GENERAL FORMAT

```
CANCEL { literal-1    }  [ , literal-2    ] . . .
       { identifier-1 }  [ , identifier-2 ]
```

7.11.3 SYNTAX RULES

1. Literal-1, literal-2, . . ., must each be a nonnumeric literal.

2. Identifier-1, identifier-2, . . ., must each be defined as an alphanumeric data item such that its value can be a program-name.

7.11.4 GENERAL RULES

1. Subsequent to the execution of a CANCEL statement, the program referred to therein ceases to have any logical relationship to the run unit in which the CANCEL statement appears. The memory areas associated with the named program are released so as to be made available for disposition by the operating system.

2. A program named in the CANCEL statement must not refer to any program that has been called and has not yet executed an EXIT PROGRAM statement.

3. A logical relationship to a cancelled subprogram is established only by execution of a subsequent CALL statement.

4. A called program is cancelled either by being referred to as the operand of a CANCEL statement or by the termination of the run unit of which the program is a member.

5. No action is taken when a 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 passes to the next statement.

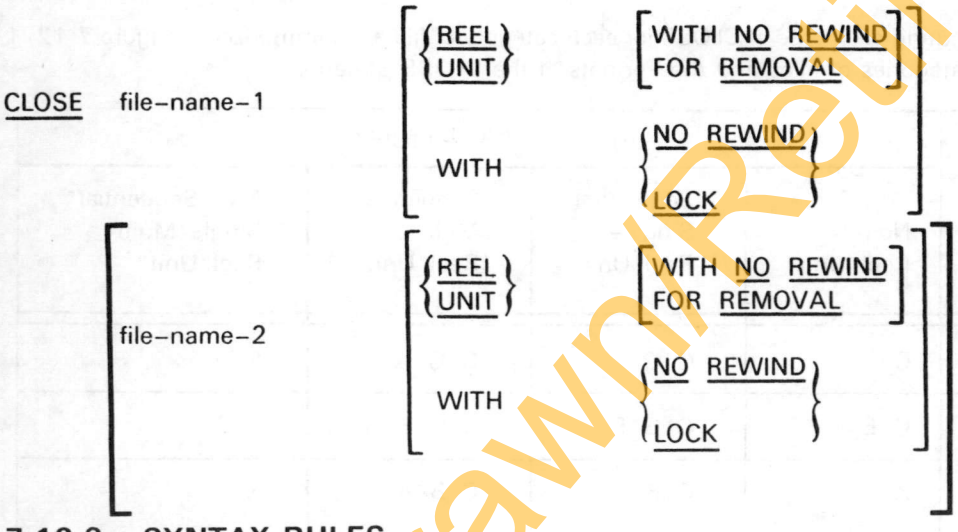
Withdrawn/Retirée

7.12 THE CLOSE STATEMENT

7.12.1 FUNCTION

The CLOSE statement terminates the processing of reels/units and files with optional rewind and/or lock or removal where applicable.

7.12.2 GENERAL FORMAT



7.12.3 SYNTAX RULES

1. File-name-1 or file-name-2 must not be the name of a sort or merge file.

2. The REEL or UNIT phrase must only be used for sequential files.

7.12.4 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. Treatment of a file contained in a multiple file tape environment is logically equivalent to the treatment of a sequential single-reel/unit file if the file is wholly contained on one reel, or to the treatment of a sequential multi-reel/unit file if the file is contained on more than one reel.

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:

Withdrawn/Retirée

a. Non-reel/unit. A file whose input or output medium is such that the concepts of rewind and reels/units have no meaning.

b. Sequential single-reel/unit. A sequential file that is entirely contained on one reel/unit.

c. Sequential multi-reel/unit. A sequential file that is contained on more than one reel/unit.

d. Non-sequential single/multi-reel/unit. A file with organization other than sequential, which resides on a mass storage device.

3. The results of executing each type of CLOSE for each category of file are summarized in Figure 7.12-1, Relationship of Categories of Files and the Formats of the CLOSE Statement.

CLOSE Statement Format	File Category			
	Non-Reel/Unit	Sequential Single-Reel/Unit	Sequential Multi-Reel/Unit	Non-Sequential Single/Multi-Reel/Unit
CLOSE	C	C,G	C,G,A	C
CLOSE WITH LOCK	C,E	C,G,E	C,G,E,A	C,E
CLOSE WITH NO REWIND	X	C,B	C,B,A	X
CLOSE REEL/UNIT	X	X	F,G	X
CLOSE REEL/UNIT FOR REMOVAL	X	X	F,D,G	X
CLOSE REEL/UNIT WITH NO REWIND	X	X	F,B	X

Figure 7.12-1 Relationship of Categories of Files and the Formats of the CLOSE Statement

The definitions of the symbols in the figure are given below. Where the definition depends on whether the file is an input, output or input-output file, alternate definitions are given; otherwise, a definition applies to input, output, and input-output files.

Withdrawn/Retirée

A. Previous Reels/Units Unaffected

Input Files and Input-Output Files:

All reels/units in the file prior to the current reel/unit are processed according to the implementor's standard reel/unit swap procedure, except those reels/units controlled by a prior CLOSE REEL/UNIT statement. If the current reel/unit is not the last in the file, the reels/units in the file following the current one are not processed.

Output Files:

All reels/units in the file prior to the current reel/unit are processed according to the implementor's standard reel/unit swap procedure, except those reels/units controlled by a prior CLOSE REEL/UNIT statement.

B. No Rewind of Current Reel

The current reel/unit is left in its current position.

C. Close File

Input Files and Input-Output Files (Sequential Access Mode):

If the file is positioned at its end and label records are specified for the file, the labels are processed according to the implementor's standard label convention. The behavior of the CLOSE statement when label records are specified but not present, or when label records are not specified but are present, is undefined. If specified by the USE statement, a user's label procedure is executed. The order of execution of these two processes is specified by the USE statement. In addition, other closing operations specified by the implementor are executed. If the file is positioned at its end and label records are not specified for the file, label processing does not take place but other closing operations specified by the implementor are executed. If the file is positioned other than at its end, the closing operations specified by the implementor are executed, but there is no ending label processing.

Input Files and Input-Output Files (Random or Dynamic Access Mode); Output Files (Random, Dynamic, or Sequential Access Mode):

If label records are specified for the file, the labels are processed according to the implementor's standard label convention. The behavior of the CLOSE statement when label records are specified but not present, or when label records are not specified but are present, is undefined. If specified by the USE statement, a user's label procedure is executed. The order of execution of these two processes is specified by the USE statement. In addition, other closing operations specified by the implementor are executed. If label records are not specified for the file, label processing does not take place but other closing operations specified by the implementor are executed.

Withdrawn/Retired

D. Reel/Unit Removal

An implementor-defined technique is supplied to ensure that the current reel or unit is rewound when applicable, and that the operating system is notified that the reel or unit is logically removed from this run unit; however, the reel or unit may be accessed again, in its proper order of reels or units within the file, if a CLOSE statement without the REEL or UNIT phrase is subsequently executed for this file followed by the execution of an OPEN statement for the file.

E. File Lock

An implementor-defined technique is supplied to ensure that this file cannot be opened again during this execution of this run unit.

F. Close Reel/Unit

Input Files:

The following operations take place:

1. A reel/unit swap.

2. The standard beginning reel/unit label procedure and the user's beginning reel/unit label procedure (if specified by the USE statement) are executed. The order of execution of these two label procedures is specified by the USE statement.

The next executed READ statement for that file makes available the next data record on the new reel/unit.

Output Files and Input-Output Files:

The following operations take place:

1. (For output files only.) The standard ending reel/unit label procedure and the user's ending reel/unit label procedure (if specified by the USE statement) are executed. The order of execution of these two procedures is specified by the USE statement.

2. A reel/unit swap.

3. The standard beginning reel/unit label procedure and the user's beginning reel/unit label procedure (if specified by the USE statement) are executed. The order of execution of these two procedures is specified by the USE statement.

For input-output files, the next executed READ statement that references that file makes the next logical data record on the next mass storage unit available. For output files, the next executed

Withdrawn/Retirée

WRITE statement that references that file directs the next logical data record to the next reel/unit of the file.

G. Rewind

The current reel or analogous device is positioned at its physical beginning.

X. Illegal

This is an illegal combination of a CLOSE option and a file category. The results at object time are undefined.

4. The action taken if a file is in the open mode when a STOP RUN statement is executed is specified by the implementor. The action taken for a file that has been opened in a called program and not closed in that program prior to the execution of a CANCEL statement for that program is also specified by the implementor.

5. If an optional file is not present, the standard end-of-file processing is not performed for that file.

6. 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.

7. The WITH NO REWIND and FOR REMOVAL phrases will have no effect at object time if they do not apply to the storage media on which the file resides.

8. Following the successful execution of a CLOSE statement, without the REEL or UNIT phrase, the record area associated with file-name is no longer available. The unsuccessful execution of such a CLOSE statement leaves the availability of the record area undefined.

Withdrawn/Retirée

7.13 THE COMPUTE STATEMENT

7.13.1 FUNCTION

The COMPUTE statement assigns to one or more data items the value of an arithmetic expression or boolean expression.

7.13.2 GENERAL FORMAT

Format 1

```
COMPUTE { , identifier-1 [ROUNDED] }. . . { FROM } arithmetic-expression-1
                                          { =    }
                                          { EQUALS }

        [; ON SIZE ERROR imperative-statement-1]
```

Format 2

```
COMPUTE { , identifier-2 }. . . { FROM } boolean-expression-1
                                { =    }
                                { EQUALS }
```

7.13.3 SYNTAX RULES

1. Identifier-1 must reference either an elementary numeric item or an elementary numeric edited item.

2. Identifier-2 must reference a boolean data item.

3. The words FROM and EQUALS are equivalent to each other and to the symbol ' = '. They may be used interchangeably and the choice is generally made for readability.

7.13.4 GENERAL RULES

1. An arithmetic or boolean expression consisting of a single identifier or literal provides a method of setting the value of the data item referenced by identifier-1 or identifier-2 equal to the value of the literal or data item referenced by the single identifier.

2. If more than one identifier is specified for the result of the operation, that is preceding FROM, ' = ', or EQUALS, the value of the arithmetic or boolean expression is developed, and then this value is stored as the new value of each of the data items referenced by identifier-1, ..., or identifier-2, ..., in turn.

Withdrawn/Retirée

3. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See 7.7.1, Intermediate Data Item; 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; 7.7.5, The Arithmetic Statements; 7.7.6, Overlapping Operands; and 7.7.7, Multiple Results in Arithmetic Statements.)

Withdrawn/Retirée

7.14 THE DELETE STATEMENT

7.14.1 FUNCTION

The DELETE statement logically removes a record from a mass storage file. The DELETE FILE statement makes available the storage media for disposition by the operating system and makes the file unavailable to the I-O Control System.

7.14.2 GENERAL FORMAT

Format 1

```
DELETE      file-name-1      RECORD      [ ; INVALID KEY imperative-statement ]
```

Format 2

```
DELETE FILE { , file-name-2 } . . .
```

7.14.3 SYNTAX RULES

FORMAT 1

1. File-name-1 must not be the name of a sequential file or the name of a sort or merge file.

2. The INVALID KEY phrase must not be specified for a DELETE statement which references a file which is in sequential access mode.

3. 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 Format 1 USE procedure is not specified.

FORMAT 2

4. File-name-2 must not be the name of a sort or merge file.

7.14.4 GENERAL RULES

FORMAT 1

1. The associated file must be open in the I-O mode at the time of the execution of this statement. (See 7.30, The OPEN Statement.)

2. For mass storage 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

Withdrawn/Retired

READ statement. The MSCS logically removes from the file the record that was accessed by that READ statement.

3. For a relative file in random or dynamic access mode, the MSCS logically removes from the file that record identified by the contents of the RELATIVE KEY data item associated with file-name-1. If the file does not contain the record specified by the key, an invalid key condition exists. (See 7.7.10, Invalid Key Condition.)

4. For an indexed file in random or dynamic access mode, the MSCS logically removes from the file the record identified by the contents of the Prime Record Key data item associated with file-name-1. If the file does not contain the record specified by the key, an invalid key condition exists. (See 7.7.10, Invalid Key Condition.)

5. After the successful execution of a DELETE statement, the identified record has been logically removed from the file and can no longer be accessed.

6. The execution of a DELETE statement does not affect the contents of the record area or the contents of the data item referenced by the data-name specified in the DEPENDING ON phrase of the RECORD clause associated with file-name-1.

7. For files in the sequential access mode, the Next Record Pointer is not affected by the execution of a DELETE statement.

8. For files in the dynamic access mode, when the record that is deleted is also pointed to by the Next Record Pointer, then:

   a. For relative files, the Next Record Pointer is updated to the next existing record in the file.

   b. For indexed files, the Next Record Pointer is updated to point to the next existing record established by the Key of Reference for the file.

   c. If no next record exists in the file, the Next Record Pointer is updated to indicate the at end condition.

9. The execution of the DELETE statement causes the value of the specified FILE STATUS data item, if any, associated with file-name-1 to be updated. (See 7.7.9, I-O Status.)

FORMAT 2

10. The file referenced by file-name-2 must not be in the open mode.

11. The execution of a DELETE FILE statement causes the storage media associated with the file referenced by file-name-2 to be made available for disposition by the operating system. The file referenced by file-name-2 is made unavailable to the I-O Control System.

Withdrawn/Retired

7.15 THE DISABLE STATEMENT

7.15.1 FUNCTION

The DISABLE statement notifies the MCS to inhibit data transfer between specified output queues and destinations for output or between specified sources and input queues for input.

7.15.2 GENERAL FORMAT

```
DISABLE { INPUT [ TERMINAL ] } cd-name WITH KEY { identifier-1 }
        { OUTPUT            }                   { literal-1    }
```

7.15.3 SYNTAX RULES

1. Cd-name must reference an input CD when the INPUT phrase is specified.

2. Cd-name must reference an output CD when the OUTPUT phrase is specified.

3. Literal-1 or the contents of the data item referenced by identifier-1 must be defined as alphanumeric or boolean.

7.15.4 GENERAL RULES

1. The DISABLE statement provides a logical disconnection between the MCS and the specified sources or destinations. When this logical disconnection is already in existence, or is to be handled by some other means external to this program, the DISABLE statement is not required in this program. No action is taken when a DISABLE statement is executed which specifies a source or destination which is already disconnected. The logical path for the transfer of data between the COBOL programs and the MCS is not affected by the DISABLE statement.

2. The MCS will insure that the execution of a DISABLE statement will cause the logical disconnection at the earliest time the source or destination is inactive. The execution of the DISABLE statement will never cause the remaining portion of the message to be terminated during transmission to or from a terminal.

3. When the INPUT phrase with the optional word TERMINAL is specified, the logical path between the source and all queues and sub-queues is deactivated. Only the contents of the data item referenced by data-name-7 (SYMBOLIC SOURCE) of the area referenced by cd-name are meaningful.

4. When the INPUT phrase without the optional word TERMINAL is specified, the logical paths for all of the sources associated with the queues and sub-queues specified by the contents of data-name-1 (SYMBOLIC QUEUE) through data-name-4 (SYMBOLIC SUB-QUEUE-3) of the area referenced by cd-name are deactivated.

Withdrawn/Retirée

5. When the OUTPUT phrase is specified, the logical paths for all destinations specified by the contents of the data item referenced by data-name-5 (SYMBOLIC DESTINATION) of the area referenced by cd-name are deactivated.

6. Literal-1 or the contents of the data item referenced by identifier-1 will be matched with a password built into the system. The DISABLE statement will be honored only if literal-1 or the contents of the data item referenced by identifier-1 matches the system password. When literal-1 or the contents of the data item referenced by identifier-1 do not match the system password, the value of the STATUS KEY item in the area referenced by cd-name is updated.

   The MCS must be capable of handling a password of from one to ten characters inclusive.

Withdrawn/Retired

7.16 THE DISPLAY STATEMENT

7.16.1 FUNCTION

The DISPLAY statement causes low volume data to be transferred to an appropriate hardware device.

7.16.2 GENERAL FORMAT

```
DISPLAY { , literal-1    } . . . [UPON mnemonic-name]
        { , identifier-1 }

    [WITH NO ADVANCING]
```

7.16.3 SYNTAX RULES

1. The mnemonic-name is associated with a hardware device in the SPECIAL-NAMES paragraph in the Environment Division.

2. Each literal may be any figurative constant, except ALL.

7.16.4 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 implementor will define, for each hardware device, the size of a data transfer.

3. If a figurative constant is specified as one of the operands, only a single occurrence of the figurative constant is displayed.

4. 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.

5. If a 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.

Withdrawn/Retirée

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.

6. 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 without modifying the positioning of the hardware device between the successive operands.

7. If the UPON phrase is not specified, the implementor's standard display device is used.

8. 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.

9. If the WITH NO ADVANCING phrase is not specified, then after the last operand has been transferred to the hardware device, the positioning of the hardware device will be reset to the leftmost position of the next line of the device.

10. If vertical positioning is not applicable on the hardware device, the operating system will ignore the vertical positioning specified or implied.

Withdrawn/Retired

7.17 THE DIVIDE STATEMENT

7.17.1 FUNCTION

The DIVIDE statement divides one numeric data item into others and sets the values of data items equal to the quotient and remainder.

7.17.2 GENERAL FORMAT

Format 1

```
DIVIDE { identifier-1 } INTO identifier-2 [ROUNDED] [, identifier-3 [ROUNDED]] . . .
       { literal-1    }

    [ ; ON SIZE ERROR imperative-statement ]
```

Format 2

```
DIVIDE { identifier-1 } INTO { identifier-2 } GIVING identifier-3 [ROUNDED]
       { literal-1    }     { literal-2    }

    [, identifier-4 [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 3

```
DIVIDE { identifier-1 } BY { identifier-2 } GIVING identifier-3 [ROUNDED]
       { literal-1    }   { literal-2    }

    [, identifier-4 [ROUNDED]] . . . [; ON SIZE ERROR imperative-statement]
```

Format 4

```
DIVIDE { identifier-1 } INTO { identifier-2 } GIVING identifier-3 [ROUNDED]
       { literal-1    }     { literal-2    }

    REMAINDER identifier-4 [; ON SIZE ERROR imperative-statement]
```

Withdrawn/Retirée

Format 5

```
DIVIDE { identifier-1 }      BY { identifier-2 }      GIVING identifier-3 [ROUNDED]
       { literal-1    }         { literal-2    }

       REMAINDER identifier-4                    [; ON SIZE ERROR imperative-statement]
```

7.17.3 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. Each literal must be a numeric literal.

7.17.4 GENERAL RULES

1. When Format 1 is used, the value of identifier-1 or literal-1 is divided into the value of identifier-2. The value of the dividend (identifier-2) is replaced by this quotient; similarly for identifier-1 or literal-1 and identifier-3, etc.

2. When Format 2 is used, the value of identifer-1 or literal-1 is divided into identifier-2 or literal-2 and the result is stored in identifier-3, identifier-4, etc.

3. When Format 3 is used, the value of identifier-1 or literal-1 is divided by the value of identifier-2 or literal-2 and the result is stored in identifier-3, identifier-4, etc.

4. 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 used, the quotient used to calculate the remainder is an intermediate field which contains the quotient of the DIVIDE statement, truncated rather than rounded.

5. 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 content of the data item referenced by identifier-4, as needed.

6. 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.

Withdrawn/Retiree

b. If the size error occurs in the remainder, the contents 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.

7. Additional rules and explanations relative to this statement are given under the appropriate paragraphs. (See 7.7.1, Intermediate Data Item; 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; 7.7.5, The Arithmetic Statements; 7.7.6, Overlapping Operands; and 7.7.7, Multiple Results in Arithmetic Statements. See also General Rules 4 through 6 above for a discussion of the ROUNDED phrase and the SIZE ERROR phrase as they pertain to Formats 4 and 5.)

Withdrawn/Retired

7.18 THE ENABLE STATEMENT

7.18.1 FUNCTION

The ENABLE statement notifies the MCS to allow data transfer between specified output queues and destinations for output or between specified sources and input queues for input.

7.18.2 GENERAL FORMAT

```
ENABLE { INPUT  [TERMINAL] }  cd-name  WITH KEY { identifier-1 }
       { OUTPUT            }                   { literal-1    }
```

7.18.3 SYNTAX RULES

1. Cd-name must reference an input CD when the INPUT phrase is specified.

2. Cd-name must reference an output CD when the OUTPUT phrase is specified.

3. Literal-1 or the contents of the data item referenced by identifier-1 must be defined as alphanumeric or boolean.

7.18.4 GENERAL RULES

1. The ENABLE statement provides a logical connection between the MCS and the specified sources or destinations. When this logical connection is already in existence, or is to be handled by some other means external to this program, the ENABLE statement is not required in this program. No action is taken when an ENABLE statement is executed which specifies a source or destination which is already connected. The logical path for the transfer of data between the COBOL programs and the MCS is not affected by the ENABLE statement.

2. When the INPUT phrase with the optional word TERMINAL is specified, the logical path between the source and all associated already enabled queues and sub-queues is activated. Only the contents of the data item referenced by data-name-7 (SYMBOLIC SOURCE) of the area referenced by cd-name are meaningful to the MCS.

3. When the INPUT phrase without the optional word TERMINAL is specified, the logical paths for all of the sources associated with the queue and sub-queues specified by the contents of data-name-1 (SYMBOLIC QUEUE) through data-name-4 (SYMBOLIC SUB-QUEUE-3) of the area referenced by cd-name are activated.

4. When the OUTPUT phrase is specified, the logical paths for all destinations specified by the contents of the data item referenced by data-name-5 (SYMBOLIC DESTINATION) of the area referenced by cd-name are activated.

Withdrawn/Retirée

ENABLE

5. Literal-1 or the contents of the data item referenced by identifier-1 will be matched with a password built into the system. The ENABLE statement will be honored only if literal-1 or the contents of the data item referenced by identifier-1 match the system password. When literal-1 or the contents of the data item referenced by identifier-1 do not match the system password, the value of the STATUS KEY item in the area referenced by cd-name is updated.

   The MCS must be capable of handling a password of from one to ten characters inclusive.

Withdrawn/Retirée

7.19 THE ENTER STATEMENT

7.19.1 FUNCTION

The ENTER statement provides a means of allowing the use of more than one language in the same program.

7.19.2 GENERAL FORMAT

ENTER language-name [routine-name]

7.19.3 SYNTAX RULES

1. The language-name may refer to any programming language which the implementor specifies may be entered through COBOL. Language-name is specified by the implementor.

2. The sentence ENTER COBOL must follow the last other-language statement in order to indicate to the compiler where a return to COBOL source language takes place.

7.19.4 GENERAL RULES

1. The other-language statements are executed in the object program as if they had been compiled into the object program following the ENTER statement.

2. Implementors will specify, for their compilers, all details on how the other language(s) are to be written.

3. If the statements in the entered language cannot be written in-line, a routine-name is given to identify the portion of the other-language coding to be executed at this point in the procedure sequence. If the other-language statements can be written in-line, routine-name is not used.

Withdrawn/Retirée

7.20 THE EXIT STATEMENT

7.20.1 FUNCTION

The EXIT statement provides a common end point for a series of procedures, or marks the logical end of a called program.

7.20.2 GENERAL FORMAT

EXIT [PROGRAM]

7.20.3 SYNTAX RULES

1. The EXIT statement without the PROGRAM phrase must appear only in a sentence by itself and comprise the only sentence in the paragraph.

2. 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.

7.20.4 GENERAL RULES

1. An EXIT statement without the optional word PROGRAM 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.

2. An execution of an EXIT PROGRAM statement in a called program causes control to be passed to the calling program. 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.

Withdrawn/Retirée

7.21 THE GENERATE STATEMENT

7.21.1 FUNCTION

The GENERATE statement directs the RWCS to produce a report in accordance with the Report Description that was specified in the Report Section of the Data Division.

7.21.2 GENERAL FORMAT

GENERATE { data-name / report-name }

7.21.3 SYNTAX RULES

1. Data-name must name a TYPE DETAIL report group and may be qualified by a report name.

2. Report-name 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.

7.21.4 GENERAL RULES

1. In response to a GENERATE report-name 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, 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 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 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.

Withdrawn/Retirée

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 6.15.5, 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 6.46, 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 6.46.4.11, 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 6.46, 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:

      1. Enable the CONTROL FOOTING USE procedures and CONTROL FOOTING SOURCE clauses to access the control data item values that the RWCS used to detect a given control break. (See 6.46.4.13, The TYPE Clause.)

      2. 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.

      3. 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 6.46.4.11, The TYPE Clause.)

Withdrawn/Retired

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.

Withdrawn/Retirée

7.22 THE GO TO STATEMENT

7.22.1 FUNCTION

The GO TO statement causes control to be transferred from one part of the Procedure Division to another.

7.22.2 GENERAL FORMAT

Format 1

GO TO procedure-name-1

Format 2

GO TO procedure-name-1 [, procedure-name-2] . . ., procedure-name-n

DEPENDING ON identifier

7.22.3 SYNTAX RULES

1. Identifier is the name of a numeric elementary item described without any positions to the right of the assumed decimal point.

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.

7.22.4 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, procedure-name-2, etc., depending on the value of the identifier being 1, 2, . . ., n. If the value of the identifier 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.

Withdrawn/Retirée

7.23 THE IF STATEMENT

7.23.1 FUNCTION

The IF statement causes a condition (see 7.5, Conditional Expressions) to be evaluated. The subsequent action of the object program depends on whether the value of the condition is true or false.

7.23.2 GENERAL FORMAT

```
IF condition  THEN  { {, statement-1} . . . }
                    {   NEXT SENTENCE       }

    { ; ELSE   {, statement-2} . . . }
    { ; ELSE NEXT SENTENCE           }
```

7.23.3 SYNTAX RULES

1. Statement-1 and statement-2 represent either an imperative statement or a conditional statement optionally preceded by an imperative statement.

2. The ELSE NEXT SENTENCE phrase may be omitted if it immediately precedes the terminal period of the sentence.

7.23.4 GENERAL RULES

1. When an IF statement is executed, the following transfers of control occur:

   a. If the condition is true, statement-1 is executed if specified. If statement-1 contains a Procedure Branching or Conditional statement, control is explicitly transferred in accordance with the rules of that statement. (See 7.6, Categories of Statements.) If statement-1 does not contain a Procedure Branching or Conditional statement, the ELSE phrase, if specified, is ignored and control passes to the next executable sentence.

   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, statement-1 or its surrogate NEXT SENTENCE is ignored, and statement-2, if specified, is executed. If statement-2 contains a Procedure Branching or Conditional statement, control is explicitly transferred in accordance with the rules of that statement. (See 7.6, Categories of Statements.) If statement-2 does not contain a Procedure Branching or Conditional statement, control passes to the next executable sentence.

   d. If the condition is false and the ELSE phrase is not specified, statement-1 is ignored and control passes to the next executable sentence.

Withdrawn/Retirée

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.

2. Statement-1 and/or statement-2 may contain an IF statement. In this case, the IF statement is said to be nested.

IF statements within IF statements may be considered as paired IF and ELSE combinations, proceeding from left to right. Thus, any ELSE encountered is considered to apply to the immediately preceding IF that has not been already paired with an ELSE.

Withdrawn/Retiree

7.24 THE INITIALIZE STATEMENT

7.24.1 FUNCTION

The INITIALIZE statement provides the ability to set selected types of data fields to predetermined values, e.g., numeric data to zeros or alphanumeric data to spaces.

7.24.2 GENERAL FORMAT

```
INITIALIZE  identifier-1  [ , identifier-2 ] . . .  [ REPLACING

{ ALPHABETIC            }
{ ALPHANUMERIC          }
{ BOOLEAN               }
{ NUMERIC               }  DATA BY { identifier-3 } ] ]
{ ALPHANUMERIC-EDITED   }          { literal      }
{ NUMERIC-EDITED        }
```

7.24.3 SYNTAX RULES

1. Identifier-3 and literal represent the sending area; identifier-1, identifier-2, . . . represent the receiving area.

2. The category of the data item referred to by identifier-3 or literal must be consistent with the category indicated by the phrase following the word REPLACING. (See 7.28.4.6, The MOVE Statement.)

3. If any of identifier-1, identifier-2, . . . is described as an elementary item, its category must be identical to the category indicated by the phrase following the word REPLACING.

4. The description of identifier-1, identifier-2, . . . or any items subordinate to any of them may not contain the DEPENDING phrase of the OCCURS clause.

5. An index data item may not appear as an operand of an INITIALIZE statement.

7.24.4 GENERAL RULES

1. Whether identifier-1, identifier-2, . . . is described as an elementary item or a group item, all operations are performed as if a series of MOVE statements had been written, each of which has an elementary item as its receiving field, subject to the following rules:

   If the receiving area is described as a group item, an elementary item within it is affected only if it belongs to the specified category.

   All such elementary receiving fields, including all occurrences of table items within the group, are affected.

Withdrawn/Retirée

INITIALIZE

2. Index data items and elementary FILLER data items are not affected by the execution of an INITIALIZE statement.

3. Any item that is subordinate to a receiving area identifier and which contains the REDEFINES clause or any item that is subordinate to such an item is excluded from this operation. However, a receiving area identifier may itself have a REDEFINES clause or be subordinate to a data item with a REDEFINES clause.

4. When the statement is written with the REPLACING phrase, literal or the area referenced by identifier-3 acts as the sending field of each of the implicit MOVE statements.

5. When the statement is written without the REPLACING phrase, data items of the categories alphabetic, alphanumeric, and alphanumeric edited are set to SPACES; data items of the categories numeric, numeric edited and boolean are set to ZEROS. In this case, the operation is as if each affected data item is the receiving area in an elementary MOVE statement with the indicated source literal (i.e., SPACES or ZEROS).

6. In all cases, the area referenced by identifier-1, identifier-2, . . . is set to the indicated value in the order (left to right) of the appearance of identifier-1, identifier-2, . . . in the INITIALIZE statement. Within this sequence, where identifier-1, identifier-2, . . . is described as a group item, affected elementary items are initialized in the sequence of their definition within the group.

Withdrawn/Retirée

7.25 THE INITIATE STATEMENT

7.25.1 FUNCTION

The INITIATE statement causes the Report Writer Control System to begin the processing of a report.

7.25.2 GENERAL FORMAT

INITIATE report-name-1 [, report-name-2] . . .

7.25.3 SYNTAX RULES

1. Each report-name must be defined by a Report Description entry in the Report Section of the Data Division.

7.25.4 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 (1).

2. The INITIATE statement does not open the file with which the report is associated; 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 a particular report-name must not be executed unless an intervening TERMINATE statement has been executed for that report-name.

Withdrawn/Retirée

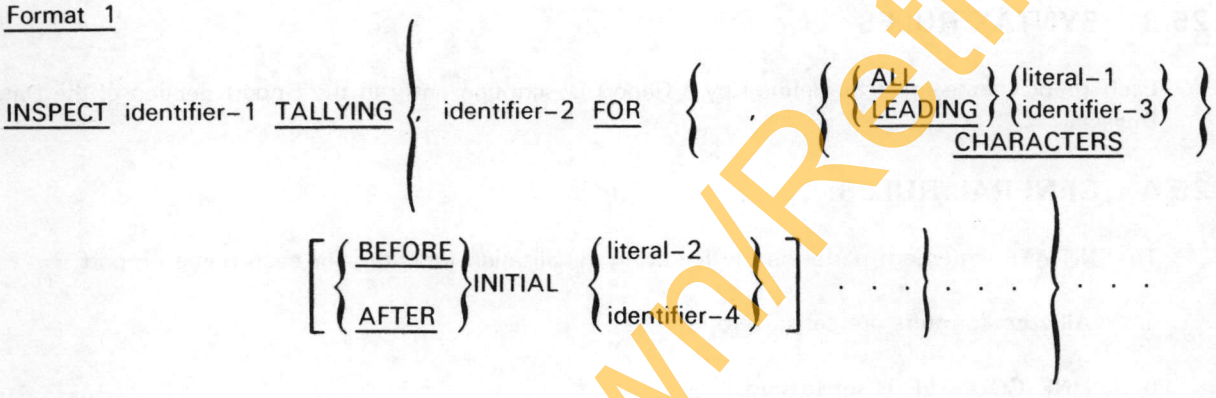
7.26 THE INSPECT STATEMENT

7.26.1 FUNCTION

The INSPECT statement provides the ability to tally or replace occurrences of single characters or groups of characters in a data item.

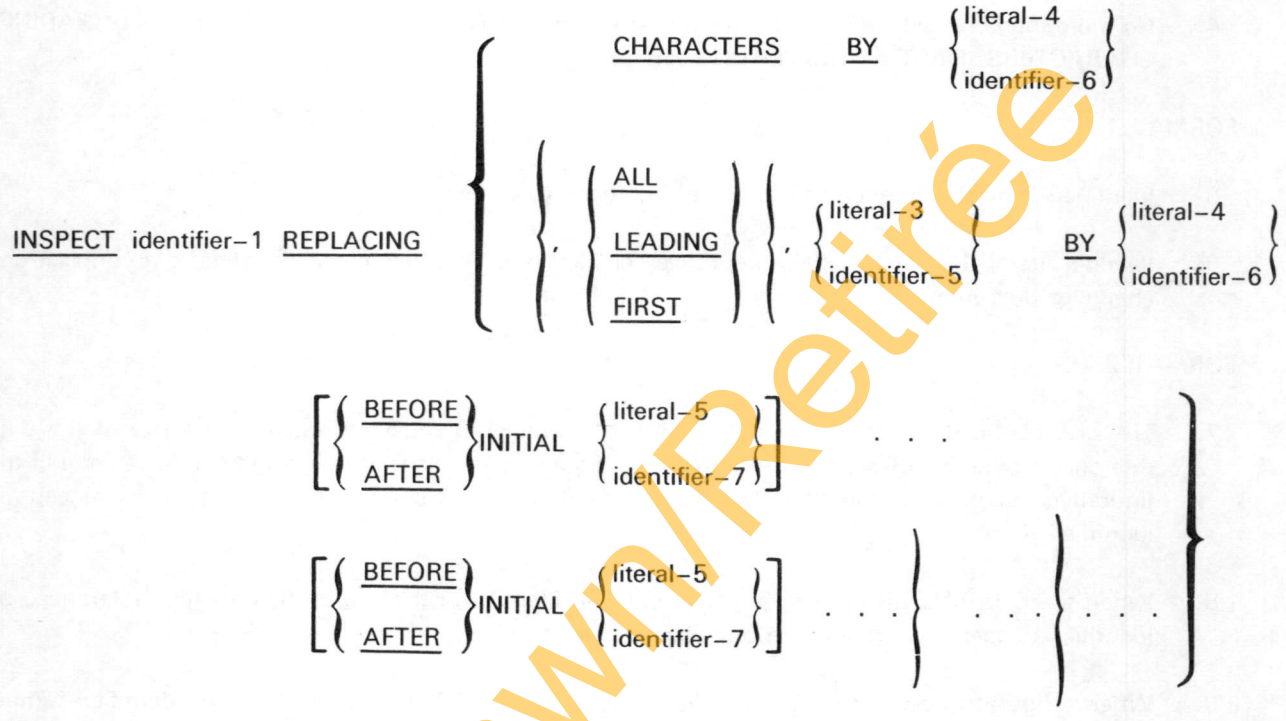
7.26.2 GENERAL FORMAT

Format 1

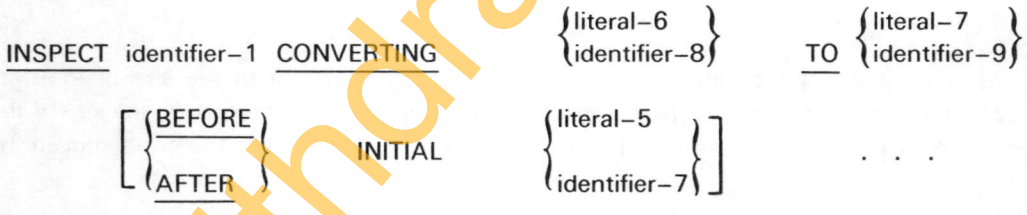


Withdrawn/Retirée

Format 2



Format 3



7.26.3 SYNTAX RULES

ALL FORMATS

1. Identifier-1 must reference either a group item or any category of elementary item, described (either implicitly or explicitly) as USAGE IS DISPLAY.

2. Identifier-3 . . . identifier-n must reference either an elementary alphabetic, alphanumeric or numeric item described (either implicitly or explicitly) as USAGE IS DISPLAY.

Withdrawn/Retirée

3. Each literal must be nonnumeric and may be any figurative constant, except ALL.

4. No more than one BEFORE phrase and one AFTER phrase can be specified for any one ALL, LEADING, CHARACTERS, FIRST or CONVERTING phrase.

FORMAT 1

5. Identifier-2 must reference an elementary numeric data item.

6. If either literal-1 or literal-2 is a figurative constant, the figurative constant refers to an implicit one character data item.

FORMAT 2

7. The size of the data referenced by literal-4 or identifier-6 must be equal to the size of the data referenced by literal-3 or identifier-5. When a figurative constant is used as literal-4, the size of the figurative constant is equal to the size of literal-3 or the size of the data item referenced by identifier-5.

8. When the CHARACTERS phrase is used, literal-4, literal-5 or the size of the data item referenced by identifier-6, identifier-7 must be one character in length.

9. When a figurative constant is used as literal-3, the data referenced by literal-4 or identifier-6 must be one character in length.

FORMAT 3

10. The size of the data referenced by literal-7 or identifier-9 must be equal to the size of the data referenced by literal-6 or identifier-8. When a figurative constant is used as literal-7, the size of the figurative constant is equal to the size of literal-6 or the size of the data item referenced by identifier-8.

11. When a figurative constant is used as literal-6, the data item referenced by literal-7 or identifier-9 must be one character in length.

12. The same character must not appear more than once either in literal-6 or in the data item referenced by identifier-8.

7.26.4 GENERAL RULES

ALL FORMATS

1. Inspection (which includes the comparision cycle, the establishment of boundaries for the BEFORE or AFTER phrase, and the mechanism for tallying and/or replacing) begins at the leftmost character

Withdrawn/Retirée

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 4 through 6.

2. For use in the INSPECT statement, the contents of the data item referenced by identifier-1, identifier-3, identifier-4, identifier-5, identifier-6, identifier-7, identifier-8 or identifier-9 will be treated as follows:

   a. If any of identifier-1, identifier-3, identifier-4, identifier-5, identifier-6, identifier-7, identifier-8 or identifier-9 are described as alphanumeric, 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, identifier-7, identifier-8 or identifier-9 are described as alphanumeric edited, numeric edited, unsigned numeric or boolean, 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, identifier-7, identifier-8 or identifier-9 are described as signed numeric, the data item is inspected as though it had been moved to an unsigned numeric data item of the same length and then the rules in General Rule 2b had been applied. (See 7.28.4, The MOVE Statement.)

3. In General Rules 4 through 11 all references to literal-1, literal-2, literal-3, literal-4, literal-5, literal-6 and literal-7 apply equally to the contents of the data item referenced by identifier-3, identifier-4, identifier-5, identifier-6, identifier-7, identifier-8 and identifier-9, respectively.

4. During inspection of the contents of the data item referenced by identifier-1, each properly matched occurrence of literal-1 is tallied (Format 1) or each properly matched occurrence of literal-3 is replaced by literal-4 (Format 2).

5. The comparison operation to determine the occurrences of literal-1 to be tallied and/or occurrences of literal-3 to be replaced, occurs as follows:

   a. The operands of the TALLYING and REPLACING phrases are considered in the order they are specified in the INSPECT statement from left to right. The first literal-1, literal-3 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, literal-3 and that portion of the contents of the data item referenced by identifier-1 match if, and only if, they are equal, character for character.

   b. If no match occurs in the comparison of the first literal-1, literal-3, the comparison is repeated with each successive literal-1, literal-3, if any, until either a match is found or there is no next successive literal-1, literal-3. When there is no next successive literal-1, literal-3, 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, literal-3.

Withdrawn/Retirée

c. Whenever a match occurs, tallying and/or replacing takes place as described in General Rule 8 through 10. 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, literal-3.

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 5a through 5d above, except that no comparison to the contents of the data item referenced by identifier-1 takes place. This implied character is considered always to match the leftmost character of the contents of the data item referenced by identifier-1 participating in the current comparison cycle.

6. The comparison operation defined in General Rule 5 is affected by the BEFORE and AFTER phrases as follows:

a. If the BEFORE and AFTER phrase is not specified, literal-1, literal-3 or the implied operand of the CHARACTERS phrase participates in the comparison operation as described in General Rule 5.

b. If the BEFORE phrase is specified, the associated literal-1, literal-3 or the implied operand of the CHARACTERS phrase participates only in those comparison cycles which involve that portion of the contents of the data item referenced by identifier-1 from its leftmost character position up to, but not including, the first occurrence of literal-2, literal-5 within the contents 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 5 is begun. If, on any comparison cycle, literal-1, literal-3 or the implied operand of the CHARACTERS phrase is not eligible to participate, it is considered not to match the contents of the data item referenced by identifier-1. If there is no occurrence of literal-2, literal-5 within the contents of the data item referenced by identifier-1, its associated literal-1, literal-3, 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, literal-3 or the implied operand of the CHARACTERS phrase may participate only in those comparison cycles which involve that portion of the contents 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, literal-5 within the contents of the data item referenced by identifier-1 and the rightmost character position 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 5 is begun. If, on any comparison cycle, literal-1, literal-3 or the implied operand of the CHARACTERS phrase is not eligible to participate, it is considered not to match the contents of the data item referenced by identifier-1. If there is no occurrence of literal-2, literal-5 within

Withdrawn/Retired

the contents of the data item referenced by identifier-1, its associated literal-1, literal-3, or the implied operand of the CHARACTERS phrase is never eligible to participate in the comparison operation.

FORMAT 1

7. The contents of the data item referenced by identifier-2 is not initialized by the execution of the INSPECT statement.

8. The rules for tallying are as follows:

   a. If the ALL phrase is specified, the contents of the data item referenced by identifier-2 is incremented by one (1) for each occurrence of literal-1 matched within the contents of the data item referenced by identifier-1.

   b. If the LEADING phrase is specified, the contents of the data item referenced by identifier-2 is incremented by one (1) for each contiguous occurrence of literal-1 matched within the contents 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 contents of the data item referenced by identifier-2 is incremented by one (1) for each character matched, in the sense of General Rule 5e, within the contents of the data item referenced by identifier-1.

FORMAT 2

9. The required words ALL, LEADING, and FIRST are adjectives that apply to each succeeding BY phrase until the next adjective appears.

10. The rules for replacement are as follows:

   a. When the CHARACTERS phrase is specified, each character matched, in the sense of General Rule 5e, in the contents of the data item referenced by identifier-1 is replaced by literal-4.

   b. When the adjective ALL is specified, each occurrance of literal-3 matched in the contents of the data item referenced by identifier-1 is replaced by literal-4.

   c. When the adjective LEADING is specified, each contiguous occurrence of literal-3 matched in the contents of the data item referenced by identifier-1 is replaced by literal-4, provided that the leftmost occurrence is at the point where comparision began in the first comparision cycle in which literal-3 was eligible to participate.

   d. When the adjective FIRST is specified, the leftmost occurrence of literal-3 matched within the contents of the data item referenced by identifier-1 is replaced by literal-4.

Withdrawn/Retiree

INSPECT

FORMAT 3

11. A Format 3 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-6. The effect is as if each of these ALL phrases referenced, as literal-3, a single character of literal-6 and referenced, as literal-4, the corresponding replacement character of literal-7. Correspondence between the characters of literal-6 and the characters of literal-7 is by ordinal position within the data item.

7.26.5 EXAMPLES

1. Following are eight examples of the INSPECT statement. In each case, count is assumed to be zero immediately prior to execution of the INSPECT statement. The first occurrence of 'word' indicates the contents of the data item before execution of the INSPECT statement; all other occurrences indicate the contents after execution.

```
INSPECT word TALLYING count FOR LEADING "L" BEFORE INITIAL "A",
count-1 FOR LEADING "A" BEFORE INITIAL "L"

Where word = LARGE, count = 1, count-1 = 0
Where word = ANALYST, count = 0, count-1 = 1

INSPECT word TALLYING count FOR ALL "L" INSPECT word REPLACING LEADING "A"
BY "E" AFTER INITIAL "L"

Where word = CALLAR, count = 2, word = CALLAR
Where word = SALAMI, count = 1, word = SALEMI
Where word = LATTER, count = 1, word = LETTER

INSPECT word REPLACING ALL "A" BY "G" BEFORE INITIAL "X"

Where word = ARXAX, word = GRXAX
Where word = HANDAX, word = HGNDGX
Where word = HANDAA, word = HGNDGG

INSPECT word TALLYING count FOR CHARACTERS AFTER INITIAL "J" INSPECT word
REPLACING ALL "A" BY "B"

Where word = ADJECTIVE, count = 6, word = BDJECTIVE
Where word = JACK, count = 3, word = JBCK
Where word = JUJMAB, count = 5, word = JUJMBB

INSPECT word REPLACING ALL "X" BY "Y", "B" BY "Z", "W" BY "Q"
AFTER INITIAL "R"
```

Withdrawn/Retired

Where word = RXXBQWY, word = RYYZQQY
Where word = YZACDWBR, word = YZACDWZR
Where word = RAWRXEB, word = RAQRYEZ

INSPECT word REPLACING CHARACTERS BY ''B'' BEFORE INITIAL ''A''

Where word = 12RXZBBCD, word = BBBBBBBBB
Where word = 12RXZABCD, word = BBBBBABCD

INSPECT word REPLACING ALL ''A'' BY ''X'' ALL ''R'' BY ''X'' AFTER INITIAL ''XXL''.

Where word = AALRRRA, word = XXLRRRX
Where word = AXXLRRR, word = XXXLXXX

INSPECT word CONVERTING ''STIR'' TO ''DATA'' AFTER INITIAL QUOTE BEFORE INITIAL ''@''.

Where word = TIRMS''SIR''L@STAR, word = TIRMS''DTA''L@STAR

Withdrawn/Retirée

7.27 THE MERGE STATEMENT

7.27.1 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.

7.27.2 GENERAL FORMAT

```
MERGE file-name-1 { , ON { DESCENDING } } KEY { , data-name-1 } . . . } . . .
                  { ASCENDING }               {               }
      [COLLATING SEQUENCE IS alphabet-name]
      USING file-name-2 { , file-name-3 } . . .
                        {               }
{ OUTPUT PROCEDURE IS section-name-1 [ { THRU      } section-name-2 ] }
{                                    [ { THROUGH   }                ] }
{ GIVING { , file-name-4 } . . .                                      }
         {               }
```

7.27.3 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. If the USING phrase is specified and the file referenced by file-name-1 contains variable-length records, the size of the records contained in the file referenced by file-name-2 must not be less than the smallest record nor larger than the largest record described for file-name-1. If the file referenced by file-name-1 contains fixed-length records, the size of the records contained in the file referenced by file-name-2 must not be larger than the largest record described for file-name-1.

4. 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 variable-length data items, nor may they name group items which contain variable-occurrence data items.

Withdrawn/Retirée

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.

f. The data items identified by KEY data-names must not be described as boolean data items.

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.

6. No two files specified in any one MERGE statement may reside on the same multiple file reel(s).

7. File-names must not be repeated within the MERGE statement.

8. No pair of file-names in a MERGE statement may be specified in the same SAME AREA or SAME SORT-MERGE AREA clause. 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. (See 5.4.2, The I-O-CONTROL Paragraph.)

9. Section-name-1 represents the name of an output procedure.

10. The words THRU and THROUGH are equivalent.

11. If file-name-4 references an indexed file, then the major key must be associated with the ASCENDING phrase and the first data-name must specify the same character positions in its record as are specified for the Prime Record Key for that file.

12. If the GIVING phrase is specified and the file referenced by file-name-4 contains variable-length records, the size of the records contained in the file referenced by file-name-1 must not be less than the smallest record nor larger than the largest record described for file-name-4. If the file referenced by file-name-4 contained fixed-length records, the size of the records contained in the file referenced by file-name-1 must not be larger than the largest record described for file-name-4.

7.27.4 GENERAL RULES

1. The MERGE statement will merge all records contained on the files referenced by file-name-2 and file-name-3.

2. If the file referenced by file-name-1 contains only fixed-length records, any record in the file referenced by file-name-2 or file-name-3 containing fewer character positions than that fixed length is space-filled on the right beginning with the first character position after the last character in the record when that record is released to the file referenced by file-name-1.

Withdrawn/Retired

MERGE

3. 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 most 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.

   These implicit functions are performed such that any associated USE procedures are executed.

4. 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.

5. The collating sequence that applies to the comparison of the nonnumeric key data items specified is determined at the beginning of the execution of the MERGE statement in the following order of precedence:

   a. First, the collating sequence established by the most recently executed SET statement with a MERGE or SORT-MERGE phrase.

   b. Second, the collating sequence established by the COLLATING SEQUENCE phrase, if specified, in that MERGE statement.

   c. Third, the collating sequence established as the program collating sequence.

6. The results of the merge operation 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 clauses associated with the MERGE statement.

7. 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 time 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:

Withdrawn/Retiree

SEAPORT90

Begin New File

Withdrawn/Retirée

SEAPORT IMAGING

Ensign™

Scanner Type: ALL

Function: Begin New File

This is used only with multi-image file formats.

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 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.

8. The output procedure must consist of one or more sections that appear contiguously in a source program and do not form a part of any other procedure. In order to make merged records available for processing, the output procedure must include the execution of at least one RETURN statement. Control must not be passed to the output procedure except when a related MERGE statement is being executed. The output procedure may consist of any procedures needed to select, modify, or copy the records that are being returned one at a time in merged order, from the file referenced by file-name-1. The restrictions on the procedural statements within the output procedure are as follows:

   a. The output procedure must not contain any transfers of control to points outside the output procedure; GO TO and PERFORM statements in the output procedure are not permitted to refer to procedure-names outside the output procedure. COBOL statements are allowed that will cause an implied transfer of control to Declaratives.

   b. The output procedures must not contain any SORT or MERGE statements.

   c. The remainder of the Procedure Division must not contain any transfers of control to points inside the output procedures; GO TO, and PERFORM statements in the remainder of the Procedure Division are not permitted to refer to procedure-names within the output procedure.

9. 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. Before control passes the last statement in the output procedure, the files referenced by file-name-2 and file-name-3 are in the open mode. 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.

10. During the execution of the output procedure or any USE AFTER EXCEPTION procedure, no statement manipulating the files referenced by, or accessing the record areas associated with file-name-2, file-name-3 or file-name-4 may be executed.

11. 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 time of the 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:

Withdrawn/Retirée

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.

For a relative file, the Relative Key data item for the first record returned contains the value '1'; for the second record returned, the value '2', etc. After execution of the MERGE statement, the contents of the Relative Key data item indicates the last record returned to the file.

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 procedures are executed. On the first attempt to write beyond the externally-defined boundaries of the file, any USE AFTER EXCEPTION procedure specified for the file is executed; if control is returned from the USE procedure or if no USE procedure is specified, the processing of the file is terminated as in paragraph 7c above.

12. If the file referenced by file-name-4 contains only fixed-length records, any record in the file referenced by file-name-1 containing fewer character positions than that fixed-length is space-filled on the right beginning with the first character position after the last character in the record when that record is returned from the file referenced by file-name-4.

Withdrawn/Retirée

7.28 THE MOVE STATEMENT

7.28.1 FUNCTION

The MOVE statement transfers data, in accordance with the rules of editing, to one or more data areas.

7.28.2 GENERAL FORMAT

$$\underline{\text{MOVE}} \left\{ \begin{matrix} [\underline{\text{CORRESPONDING}}] \\ [\underline{\text{CORR}}] \end{matrix} \right. \begin{matrix} \text{identifier-1} \\ \text{literal} \end{matrix} \left. \right\} \underline{\text{TO}} \text{ identifier-2 } [\text{, identifier-3}] \ldots$$

7.28.3 SYNTAX RULES

1. Identifier-1 and literal represent the sending area; identifier-2, identifier-3, . . ., represent the receiving area.

2. CORR is an abbreviation for CORRESPONDING.

3. When the CORRESPONDING phrase is used, all identifiers must be group items.

4. Neither an index data item nor a data base key item may appear as an operand of a MOVE statement.

7.28.4 GENERAL RULES

1. If the CORRESPONDING phrase is used, selected items within identifier-1 are moved to selected items within identifier-2, identifier-3, . . ., according to the rules specified under the appropriate paragraph. (See 7.7.4, The CORRESPONDING Option.) The results are the same as if the user had referred to each pair of corresponding identifiers in separate MOVE statements.

2. The data designated by the literal or identifier-1 is moved first to identifier-2, then to identifier-3, ... The rules referring to identifier-2 also apply to the other receiving areas. Any subscripting or indexing associated with identifier-2, . . ., is evaluated immediately before the data is moved to the respective data item.

   Any subscripting or indexing associated with identifier-1 is evaluated only once, immediately before data is moved to the first of the receiving operands.

Withdrawn/Retirée

MOVE

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 implementor.

3. Any move in which the sending and receiving items are both elementary items is an elementary move. Every elementary item belongs to one of the following categories: numeric, alphabetic, alphanumeric, numeric edited, alphanumeric edited, boolean. (See 6.35, The PICTURE Clause.) Numeric literals belong to the category numeric, nonnumeric literals belong to the category alphanumeric and boolean literals belong to the category boolean. The figurative constant ZERO (ZEROS, ZEROES), when moved to a numeric or numeric edited item, belongs to the category numeric. When moved to a boolean item it has the category boolean. In all other cases, it belongs to the category alphanumeric. 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, an alphanumeric edited or alphabetic data item must not be moved to a numeric, numeric edited or boolean data item.

   b. A numeric literal, a boolean literal, the figurative constant ZERO, a numeric data item, a numeric edited data item or a boolean data item must not be moved to an alphabetic data item.

   c. A non-integer numeric literal or a non-integer numeric data item must not be moved to an alphanumeric or alphanumeric edited data item.

   d. A numeric data item, a numeric literal or a figurative constant other than ZERO must not be moved to a boolean data item.

   e. 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, or de-editing implied by, 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 3.3.6, Standard Alignment Rules.) If the size of the sending item is greater than the size of the receiving item, the excess characters are truncated on the right after the receiving item is filled. If the sending item is described as being signed numeric, the operational sign will not be moved; if the operational sign occupied a separate character position, that character will not be moved and the size of the

Withdrawn/Retirée

sending item will be considered to be one less than its actual size (in terms of Standard Data Format characters). (See 6.42, The SIGN Clause.)

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 zeros are replaced because of editing requirements. (See 3.3.6, Standard Alignment Rules.) When the sending item is numeric edited, de-editing is implied to establish the item's unedited numeric value (which may be signed); then the unedited numeric value is moved to the receiving field.

(1) When a signed numeric item is the receiving item, the sign of the sending item is placed in the receiving item. (See 6.42, 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.

(2) 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.

(3) When a data item described as alphanumeric is the sending item, data is moved as if the sending item were described as an unsigned numeric integer.

c. When a receiving field is described as alphabetic, justification and any necessary space-filling takes place as previously defined. (See 3.3.6, Standard Alignment Rules.) If the size of the sending item is greater than the size of the receiving item, the excess characters are truncated on the right after the receiving item is filled.

d. When a boolean item is the receiving item, justification and any necessary boolean character zero filling takes place as previously defined. (See 3.3.6, Standard Alignment Rules.) If the size of the sending item is greater than the size of the receiving item, the excess characters are truncated on the right after the receiving item is filled.

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 General Rule 4 of the OCCURS clause. (See 6.33.4, The OCCURS Clause.)

6. Data in the following figure summarizes the legality of the various types of MOVE statements. The general rule reference indicates the rule that prohibits the move or the behavior of a legal move.

Withdrawn/Retired

MOVE

Category of Sending Data Item		Category of Receiving Data Item			
		Alphabetic	Alphanumeric Alphanumeric Edited	Boolean	Numeric Integer Numeric Non-integer Numeric Edited
ALPHABETIC		Yes/4c	Yes/4a	No/3a	No/3a
ALPHANUMERIC		Yes/4c	Yes/4a	Yes/3d	Yes/4b
ALPHANUMERIC EDITED		Yes/4c	Yes/4a	No/3a	No/3a
BOOLEAN		No/3b	Yes/3a, 3d	Yes/4d	No/3d
NUMERIC	INTEGER	No/3b	Yes/4a	No/3d	Yes/4b
	NON-INTEGER	No/3b	No/3c	No/3d	Yes/4b
NUMERIC EDITED		No/3b	Yes/4a	No/3a	Yes/4b

Figure 7.28-1 Legality of Types of MOVE Statements

7. The null data attribute, or the lack of the null data attribute, associated with any sending data item is not moved to the receiving data items.

Withdrawn/Retiree

7.29 THE MULTIPLY STATEMENT

7.29.1 FUNCTION

The MULTIPLY statement causes numeric data items to be multiplied and sets the values of data items equal to the results.

7.29.2 GENERAL FORMAT

Format 1

```
MULTIPLY { identifier-1 }      BY identifier-2 [ROUNDED] [ , identifier-3 [ROUNDED] ] . . .
         { literal-1    }

    [ ; ON SIZE ERROR imperative-statement ]
```

Format 2

```
MULTIPLY { identifier-1 }      BY { identifier-2 }      GIVING identifier-3 [ROUNDED]
         { literal-1    }         { literal-2    }

    [ , identifier-4 [ROUNDED] ]      . . .

    [ ; ON SIZE ERROR imperative-statement ]
```

7.29.3 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.

7.29.4 GENERAL RULES

1. When Format 1 is used, the value of identifier-1 or literal-1 is multiplied by the value of identifier-2. The value of the multiplier (identifier-2) is replaced by this product; similarly for identifier-1 or literal-1 and identifier-3, etc.

2. When Format 2 is used, the value of identifier-1 or literal-1 is multiplied by identifier-2 or literal-2 and the result is stored in identifier-3, identifier-4, etc.

Withdrawn/Retirée

MULTIPLY

3. Additional rules and explanations relative to this statement are given under the appropriate paragraphs.

   (See 7.7.1, Intermediate Data Item; 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; 7.7.5, The Arithmetic Statements; 7.7.6, Overlapping Operands; and 7.7.7, Multiple Results in Arithmetic Statements.)

Withdrawn/Retirée

7.30 THE OPEN STATEMENT

7.30.1 FUNCTION

The OPEN statement initiates the processing of files. It also performs checking and/or writing of labels and other input-output operations.

7.30.2 GENERAL FORMAT

```
OPEN { INPUT file-name-1 [WITH NO REWIND]
                [, file-name-2 [WITH NO REWIND]] . . .
     { OUTPUT file-name-3 [WITH NO REWIND]
                [, file-name-4 [WITH NO REWIND]] . . .
     { I-O file-name-5 [, file-name-6] . . .
     { EXTEND file-name-7 [, file-name-8] . . .
```

7.30.3 SYNTAX RULES

1. The OPEN statement must not reference a sort or merge file.

2. The OPEN statement for a report file must contain only the OUTPUT phrase or the EXTEND phrase.

3. The NO REWIND phrase must only be used with sequential files. (See 7.12, The CLOSE Statement.)

4. The I-O phrase must only be used for mass storage files.

5. The EXTEND phrase must only be used for sequential files.

7.30.4 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.

   A file is available if it is physically present and is recognized by the I-O Control System. Figure 7.30-1 shows the results of opening available and unavailable files.

Withdrawn/Retirée

OPEN

	File is available	File is unavailable
INPUT	Normal open	Error
INPUT (Optional File)	Normal open	Open is normal. The first read causes an at end condition.
I-O	Normal open	The open causes the file to be created.
OUTPUT	Implementor defined	The open causes the file to be created.
EXTEND	Normal open	The open causes the file to be created.

Figure 7.30-1 Availability of a File

2. The successful execution of an OPEN statement makes the associated record area available to the program.

3. Prior to the successful execution of an OPEN statement for a given file, no statement (except for a DELETE FILE, MERGE or SORT statement with the USING or GIVING phrases) can be executed that references that file, either explicitly or implicitly.

4. An OPEN statement must be successfully executed prior to the execution of any of the permissible input-output statements. In Figure 7.30-2, Permissible Statements, 'X' at an intersection indicates that the specified statement, used in the access mode given for that row, may be used with the file organization and open mode given at the top of the column.

Withdrawn/Retiree

FILE ACCESS MODE		State–ment \ Open Mode	FILE ORGANIZATION											
			SEQUENTIAL				RELATIVE				INDEXED			
			I	O	I–O	E	I	O	I–O	E	I	O	I–O	E
	SEQUENTIAL	READ	X		X		X		X		X		X	
		WRITE		X		X		X				X		
		REWRITE			X				X				X	
		START					X		X		X		X	
		DELETE							X				X	
	RANDOM	READ					X		X		X		X	
		WRITE						X	X			X	X	
		REWRITE							X				X	
		START												
		DELETE							X				X	
	DYNAMIC	READ					X		X		X		X	
		WRITE						X	X			X	X	
		REWRITE							X				X	
		START					X		X		X		X	
		DELETE							X				X	

Figure 7.30–2 Permissible Statements

5. 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.

6. A file may be opened with the INPUT, OUTPUT, EXTEND and I-O phrases in the same program. 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 REEL, UNIT or LOCK phrase, for that file.

Withdrawn/Retiree

7. Execution of the OPEN statement does not obtain or release the first data record.

8. 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 accordance with the implementor's specified conventions for input label checking.

   b. When the OUTPUT phrase is specified, the execution of the OPEN statement causes the labels to be written in accordance with the implementor's specified conventions for output label writing.

   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. If specified by the USE statement, a user's label procedure is also executed. The order of execution of these two processes is specified by the USE statement. If label records are indicated as present by a LABEL RECORDS clause, the user's beginning label procedure, if specified by a USE statement, is executed before or after (as indicated) checking but prior to writing, or subsequent to reading, the first label.

9. The File Description entry for file-name-1, file-name-2, file-name-5, file-name-6, file-name-7, or file-name-8 must be equivalent to that used when this file was created.

10. The NO REWIND phrase must only be used with:

    a. Sequential single-reel/unit files. (See 7.12, The CLOSE Statement.)

    b. Sequential files wholly contained within a single reel of tape within a multiple file tape environment. (See 5.4.2, The I-O-CONTROL Paragraph.)

11. The NO REWIND phrase will be ignored if it does not apply to the storage media on which the file resides.

12. If the storage medium for the file permits rewinding, the following rules apply:

    a. When neither 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.

13. When files are opened with the INPUT or I-O phrase, the OPEN statement sets the Next Record Pointer to the first record currently existing within the file. For indexed files, the Prime Record Key is established as the Key of Reference and is used to determine the first record to be accessed. If no records exist in the file, the Next Record Pointer is set such that the next sequential READ statement executed for the file will result in an at end condition.

Withdrawn/Retirée

14. When the EXTEND phrase is specified, the OPEN statement updates the Next Record Pointer to point to the position immediately following the last logical record of that file. Subsequent WRITE statements referencing that file will add records to the file as though the file had been opened with the OUTPUT phrase.

15. 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.

16. The I-O phrase permits the opening of a mass storage file for both input and output operations.

17. 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 in accordance with the implementor's specified conventions for input-output label checking.

    b. The user's beginning label procedure, if one is specified by the USE statement, is executed as detailed in the first General Rule of the appropriate paragraph. (See 7.50, The USE Statement.)

    c. The new labels are written in accordance with the implementor's specified conventions for input-output label writing.

18. Treatment of a file contained in a multiple file tape environment is logically equivalent to the treatment of a sequential file contained in a single file tape environment.

19. Whenever a set of files resides on a multiple file reel(s), and one of this set of files is referenced in an OPEN statement, the following rules apply:

    a. Not more than one of the files may be in the open mode at one time.

    b. There is no constraint on the order in which files may be opened in the input mode.

    c. When one of the files referenced by a file-name is the subject of an OPEN statement with the OUTPUT or EXTEND phrase, then all files on the associated multiple file reel(s) whose position numbers are less than the position number of that file must already exist on the reel(s) at the

Withdrawn/Retiree

time the OPEN statement is executed. Further, no file on that multiple file reel(s) whose position number is greater than the position number of that file can exist at that time on the reel(s).

d. Each of the files must be a sequential file.

e. If the POSITION phrase of the MULTIPLE FILE TAPE clause is specified with an identifier, then the contents of the referenced data item determines the ordinal position of the file. If the value of the data item is zero, then the referenced file is assumed to be the first.

20. The successful execution of an OPEN statement with an EXTEND or I-O phrase creates the file, if that file is unavailable. The successful execution of an OPEN statement with an OUTPUT phrase creates the file. At that time the associated file contains no data records.

Withdrawn/Retired

7.31 THE PERFORM STATEMENT

7.31.1 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.

7.31.2 GENERAL FORMAT

Format 1

```
PERFORM procedure-name-1 [ { THRU      } procedure-name-2 ]
                         { THROUGH }
```

Format 2

```
PERFORM procedure-name-1 [ { THRU      } procedure-name-2 ]
                         { THROUGH }

        { identifier-10 }
        { integer-1     }  TIMES
```

Format 3

```
PERFORM procedure-name-1 [ { THRU      } procedure-name-2 ]  UNTIL condition-1
                         { THROUGH }
```

Withdrawn/Retirée

PERFORM

Format 4

```
PERFORM procedure-name-1 [ { THRU      } procedure-name-2 ]
                           { THROUGH   }

    VARYING { identifier-1 }  FROM { identifier-2 }
            { index-name-1 }       { index-name-2 }
                                   { literal-1    }

    BY { identifier-3 }  UNTIL condition-1
       { literal-2    }

[   AFTER { identifier-4 }  FROM { identifier-5 }
          { index-name-3 }      { index-name-4 }
                                { literal-3    }

    BY { identifier-6 }  UNTIL condition-2 ] . . .
       { literal-4    }
```

7.31.3 SYNTAX RULES

1. Each identifier represents a numeric elementary item described in the Data Division. In Format 2, identifier-10 must be described as a numeric integer.

2. Each literal represents a numeric literal.

3. The words THRU and THROUGH are equivalent.

4. If an index-name is specified in the VARYING or AFTER phrase, then:

   a. The identifier in the associated FROM and BY phrases must be 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 non-zero integer.

Withdrawn/Retirée

5. If an index-name is specified in the FROM phrase, then:

   a. The identifier in the associated VARYING or AFTER phrase must be an integer data item.

   b. The identifier in the associated BY phrase must be an integer data item.

   c. The literal in the associated BY phrase must be an integer.

6. Literal in the BY phrase must not be zero.

7. Condition-1, condition-2, ..., may be any conditional expression. (See 7.5, Conditional Expressions.)

8. Where procedure-name-1 and procedure-name-2 are both specified and either is the name of a procedure in the Declarative Section of the program then both must be procedure-names in the same Declarative Section.

7.31.4 GENERAL RULES

1. The data items referenced by identifier-3, identifier-6, ..., 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, then the data item referenced by the identifier must have a positive value.

3. When the PERFORM statement is executed, control is transferred to the first statement of the procedure named procedure-name-1 (except as indicated in General Rules 6b, 6c and 6d). This transfer of control occurs only once for each execution of a PERFORM statement. For those cases where a transfer of control to the named procedure does take place, an implicit transfer of control to the next executable statement following the PERFORM statement is established as follows:

   a. If procedure-name-1 is a paragraph-name and procedure-name-2 is not specified, then 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, then 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, then the return is after the last statement of the paragraph.

   d. If procedure-name-2 is specified and it is a section-name, then the return is after the last statement of the last paragraph in the section.

4. 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

Withdrawn/Retirée

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.

5. If control passes to these procedures by means other than a PERFORM statement, control will pass through the last statement of the procedure to the next executable statement as if no PERFORM statement mentioned these procedures.

6. The PERFORM statements operate as follows with Rule 5 above applying to all formats:

   a. Format 1 is the basic PERFORM statement. A procedure referenced by this type of PERFORM statement is executed once and then control passes to the next executable statement following the PERFORM statement.

   b. Format 2 is the PERFORM. . .TIMES. The procedures are performed the number of times specified by integer-1 or by the initial value of the data item referenced by identifier-10 for that execution. If, at the time of the execution of a PERFORM statement, the value of the data item referenced by identifier-10 is equal to zero or is negative, control passes to the next executable statement following the PERFORM statement. Following the execution of the procedures the specified number of times, control is transferred to the next executable statement following the PERFORM statement.

      During execution of the PERFORM statement, reference to identifier-10 cannot alter the number of times the procedures are to be executed from that which was indicated by the initial value of identifier-10.

   c. Format 3 is the PERFORM. . .UNTIL. The specified procedures are performed until the condition specified by the UNTIL phrase is true. When the condition is true, control is transferred to the next executable statement after 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 next executable statement following the PERFORM statement.

   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-1 or identifier-4 at the beginning of the PERFORM statement must be equal to an occurence 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

Withdrawn/Retiree

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.

In Format 4, when one identifier is varied, identifier-1 is set to the value of literal-1 or the current value of identifier-2 at the point of initial execution of the PERFORM statement; then, if the condition of the UNTIL clause is false, the sequence of procedures, procedure-name-1 through procedure-name-2, is executed once. The value of identifier-1 is augmented by the specified increment or decrement value (the value of identifier-3 or literal-2) and condition-1 is evaluated again. The cycle continues until this condition is true, at which point control is transferred to the next executable statement following the PERFORM statement. If condition-1 is true at the beginning of execution of the PERFORM statement, control is transferred to the next executable statement following the PERFORM statement.

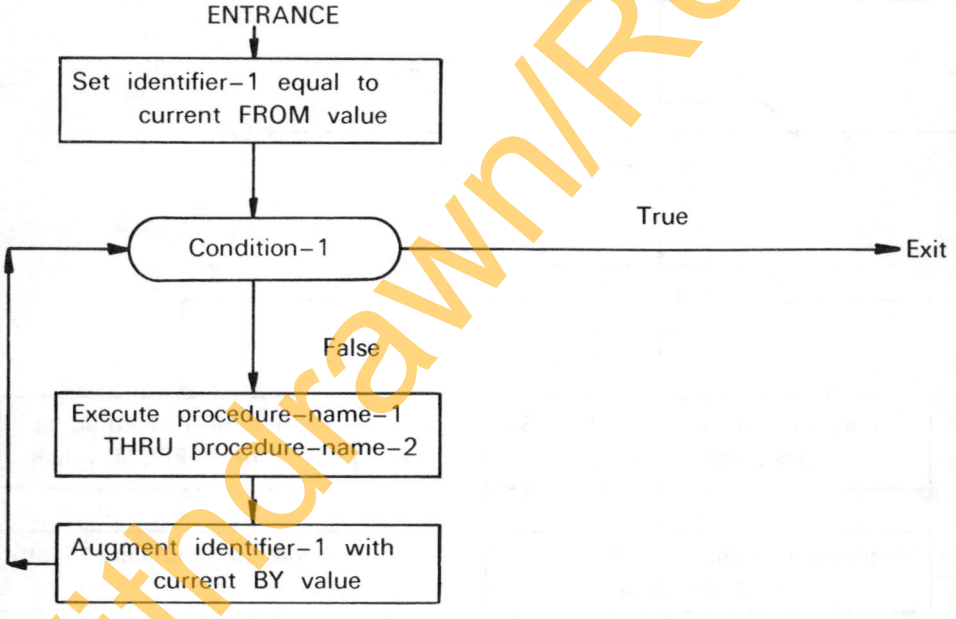


Figure 7.31–1 The VARYING Option of a PERFORM Statement Having One Condition.

In Format 4, when two identifiers are varied, identifier-1 and identifier-4 are set to the current value of identifier-2 and identifier-5, respectively. After the identifiers have been set, condition-1 is evaluated; if true, control is transferred to the next executable statement; if false, condition-2 is evaluated. If condition-2 is false, procedure-name-1 through procedure-name-2 is executed once, then identifier-4 is augmented by identifier-6 or literal-4 and condition-2 is evaluated again. This cycle of evaluation and augmentation continues until this condition is true. When condition-2 is true, identifier-4 is set to the value of literal-3 or the current value of identifier-5, identifier-1 is augmented by identifier-3 and condition-1 is reevaluated. The

Withdrawn/Retired

PERFORM

PERFORM statement is completed if condition-1 is true; if not, the cycles continue until condition-1 is true.

During the execution of the procedures associated with the PERFORM statement, any change to the VARYING variable (identifier-1 and index-name-1), the BY variable (identifier-3), the AFTER variable (identifier-4 and index-name-3), or the FROM variable (identifier-2 and index-name-2) will be taken into consideration and will affect the operation of the PERFORM statement.

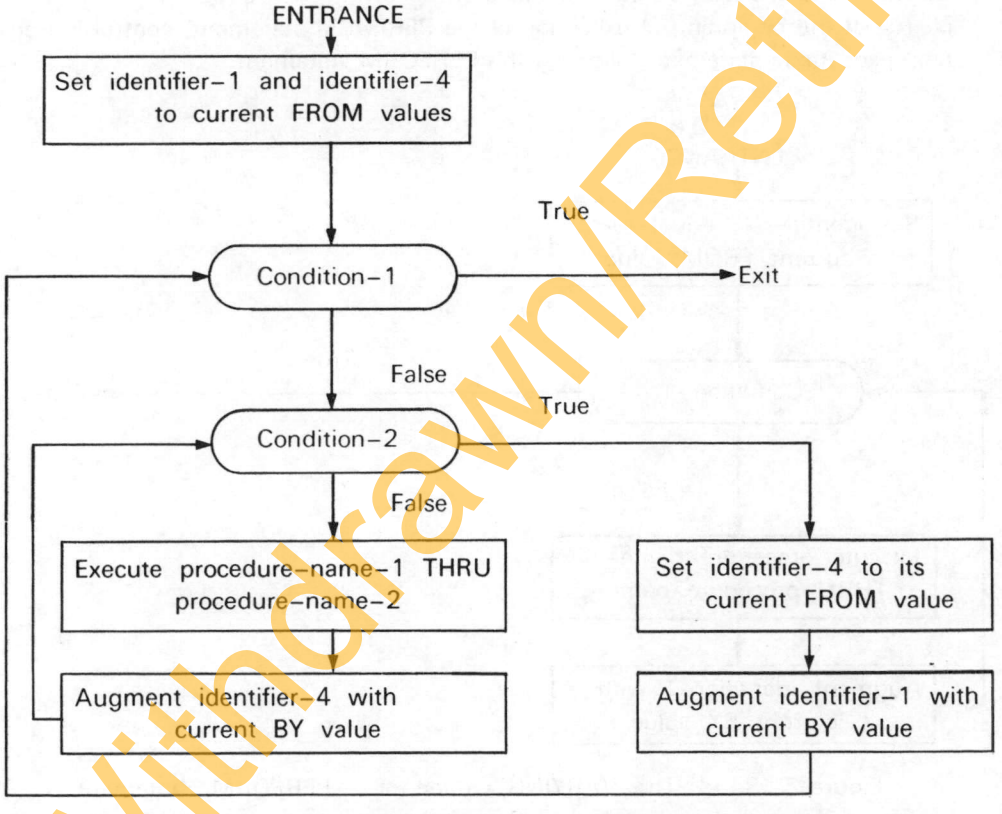


Figure 7.31-2 The VARYING Option of a PERFORM Statement Having Two Conditions.

At the termination of the PERFORM statement, identifier-4 contains the current value of identifier-5. Identifier-1 has a value that exceeds the last used setting by an increment or decrement value, unless condition-1 was true when the PERFORM statement was entered, in which case identifier-1 contains the current value of identifier-2.

When two identifiers are varied, identifier-4 goes through a complete cycle (FROM, BY, UNTIL) each time identifier-1 is varied.

Withdrawn/Retiree

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.

After the completion of a Format 4 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-1 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-1 contains the current value of the data item referenced by identifier-2.

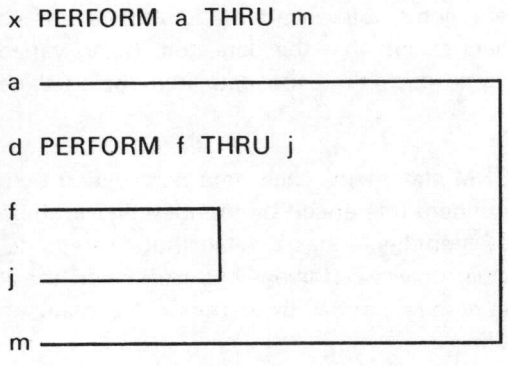
e. During the execution of the procedures associated with a Format 3 or Format 4 PERFORM statement, any change to the program collating sequence is taken into consideration and affects the evaluation of the condition. (See 7.40, The SET Statement.)

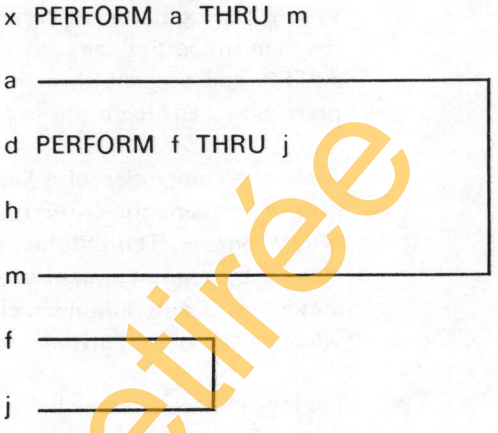
7. The range of a PERFORM statement consists logically of all those statements that are executed as a result of executing the PERFORM statement through execution of the implicit transfer of control to the next executable statement following the PERFORM statement. The range includes all statements that are executed as the result of a transfer of control by GO TO, PERFORM and CALL 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. There is no requirement for the statements in the range of a PERFORM to appear consecutively in the source program.

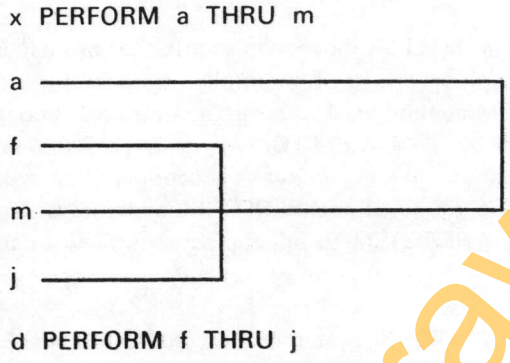
8. If a sequence of statements referred to by a PERFORM statement includes another PERFORM statement, the sequence of procedures associated with the included PERFORM must itself either be totally included in, or totally excluded from, the logical sequence referred to by the first PERFORM. Thus, an active 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 illustrations below:

Withdrawn/Retired

PERFORM







Withdrawn/Retirée

7.32 THE PURGE STATEMENT

7.32.1 FUNCTION

The PURGE statement eliminates from the MCS a partial message which has been released by one or more SEND statements.

7.32.2 GENERAL FORMAT

PURGE cd-name

7.32.3 SYNTAX RULES

1. Cd-name must reference an output CD.

7.32.4 GENERAL RULES

1. Execution of a PURGE statement causes the MCS to eliminate any partial message awaiting transmission to the destinations specified in the CD referenced by cd-name.

2. Any message that has associated with it an EMI or EGI is not affected by the execution of a PURGE statement.

3. The contents of the data item referenced by data-name-3 (STATUS KEY) and the contents of the data item referenced by data-name-4 (ERROR KEY) of the area referenced by cd-name are updated by the MCS. (See 6.12.4, The Communication Description — Complete Entry Skeleton.)

Withdrawn/Retirée

7.33 THE READ STATEMENT

7.33.1 FUNCTION

1. For sequential access, the READ statement makes available the next logical record from a file.

2. For random access, the READ statement makes available a specified record from a mass storage file.

7.33.2 GENERAL FORMAT

Format 1

READ file-name [NEXT] RECORD [INTO identifier]

[; AT END imperative-statement]

Format 2

READ file-name RECORD [INTO identifier]

[; KEY IS data-name]

[; INVALID KEY imperative-statement]

7.33.3 SYNTAX RULES

1. File-name must not be the name of a report file or the name of a sort or merge file.

2. Data-name must be the name of a data item specified as a Record Key associated with file-name.

3. Data-name may be qualified.

4. Format 1 must be used for all 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.

6. Format 2 is used for files in random access mode or for files in dynamic access mode when records are to be retrieved randomly.

7. The KEY phrase may be specified only for indexed files.

8. The INVALID KEY phrase or the AT END phrase must be specified if no applicable Format 1 USE procedure is specified for file-name.

Withdrawn/Retirée

7.33.4 GENERAL RULES

1. The explanation of the INTO phrase is stated under the appropriate paragraph. (See 7.7.13, The INTO Option.)

2. The associated file must be open in the INPUT or I-O mode at the time this statement is executed. (See 7.30, The OPEN Statement.)

3. For sequential files the NEXT phrase is optional and has no effect on the execution of the READ statement.

4. The execution of a Format 1 READ statement may cause the following to occur:

   a. The record, pointed to by the Next Record Pointer, is made available in the record area associated with file-name.

   b. For sequential or relative files, the Next Record Pointer is updated to point to the next existing record in the file.

   c. For indexed files, the Next Record Pointer is updated to point to the next existing record established by the Key of Reference for the file.

   d. If no next record exists in the file, the Next Record Pointer is updated to indicate the at end condition.

   e. If the Next Record Pointer indicates that no valid next record has been established, the execution of the READ statement is unsuccessful.

5. The execution of the READ statement causes the value of the FILE STATUS data item, if any, associated with file-name to be updated. (See 7.7.9, I-O Status.)

6. 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.

7. 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 READ statement.

8. If the end of a reel or unit is recognized during execution of a Format 1 READ statement, and the logical end of the file has not been reached, the following operations are executed:

Withdrawn/Retiree

a. The standard ending reel/unit label procedure and the user's ending reel/unit label procedure, if specified. The order of execution of these two procedures is specified by the USE statement.

b. A reel/unit swap.

c. The standard beginning reel/unit label procedure and the user's beginning reel/unit label procedure, if specified. The order of execution is specified by the USE statement.

d. The first data record of the new reel/unit is made available.

For mass storage files, this is only possible for sequential files. Thus, end-of-unit and beginning-of-unit labels are processed for sequential files only.

9. If an input file is an optional file, the object program causes an interrogation for the presence or absence of the file. If the optional file is not present at the time the file is opened, then at the time of execution of the READ statement for the file one of the following occurs:

a. If a Format 1 READ statement is executed, then the at end condition exists. (See 7.7.11, The At End Condition.)

b. If a Format 2 READ statement is executed, then the invalid key condition exists. (See 7.7.10, The Invalid Key Condition.)

(See 6.10, The File Description — Complete Entry Skeleton; 7.7.9, I-O Status.)

10. If, at the time of the execution of a Format 1 READ statement, the Next Record Pointer indicates the at end condition, the following occurs in the order specified:

a. A value is placed into the FILE STATUS data item, if specified for this file, to indicate an at end condition. (See 7.7.9, I-O Status.)

b. If the AT END phrase is specified in the statement causing the condition, control is transferred to the AT END imperative-statement. Any USE AFTER EXCEPTION procedure specified for this file is not executed.

c. If the AT END phrase is not specified, then a USE AFTER EXCEPTION procedure must be specified for this file, and that procedure is executed.

When the at end condition occurs, execution of the input-output statement which caused the condition is unsuccessful.

11. Following the unsuccessful execution of any READ statement, the contents of the associated record area and the setting of the Next Record Pointer are undefined. For indexed files the Key of Reference is also undefined.

Withdrawn/Retiree

12. For a relative or indexed file for which dynamic access mode is specified, a Format 1 READ statement with the NEXT phrase specified causes the next logical record to be retrieved from that file.

13. For a relative file if the RELATIVE KEY clause is specified, the execution of a Format 1 READ statement updates the contents of the RELATIVE KEY data item such that it contains the relative record number of the record made available.

14. For a relative file, the execution of a Format 2 READ statement sets the Next Record Pointer to, and makes available, the record whose relative record number is contained in the data item named in the RELATIVE KEY clause for the file. If the file does not contain such a record, the invalid key condition exists and execution of the READ statement is unsuccessful. (See 7.7.10, The Invalid Key Condition.)

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 an indexed file if the KEY phrase is specified in a Format 2 READ statement, data-name 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 executions of Format 1 READ statements for the file until a different Key of Reference is established for the file.

17. For an indexed file, if the KEY phrase is not specified in a Format 2 READ statement, the Prime Record Key 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 executions of Format 1 READ statements for the file until a different Key of Reference is established for the file.

18. For an indexed file, execution of a Format 2 READ statement causes the value in the Key of Reference to be 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. The Next Record Pointer is set to this record which is then made available. If no record can be so identified, the invalid key condition exists and execution of the READ statement is unsuccessful. (See 7.7.10, The Invalid Key Condition.)

19. If the number of character positions in the record that is read is less than the minimum size or greater than the maximum size specified for the file, the execution of the READ statement is unsuccessful.

Withdrawn/Retirée

7.34 THE RECEIVE STATEMENT

7.34.1 FUNCTION

The RECEIVE statement makes available to the COBOL program a message, message segment, or a portion of a message or segment, and pertinent information about that data from a queue maintained by the Message Control System.

The RECEIVE statement allows for a specific imperative statement when no data is available.

7.34.2 GENERAL FORMAT

```
RECEIVE cd-name { MESSAGE } INTO identifier-1    [; NO DATA imperative-statement]
                { SEGMENT }
```

7.34.3 SYNTAX RULES

1. Cd-name must reference an input CD.

7.34.4 GENERAL RULES

1. The contents of the data items specified by data-name-1 (SYMBOLIC QUEUE) through data-name-4 (SYMBOLIC SUB-QUEUE-3) of the area referenced by cd-name designate the queue structure containing the message. (See 6.12.4, The Communication Description — Complete Entry Skeleton.)

2. The message, message segment, or portion of a message or segment is transferred to the receiving character positions of the area referenced by identifier-1 aligned to the left without space fill.

3. When during the execution of a RECEIVE statement, the MCS makes data available in the data item referenced by identifier-1, control is transferred to the next executable statement, whether or not the NO DATA phrase is specified.

4. When, during the execution of a RECEIVE statement, the MCS does not make data available in the data item referenced by identifier-1:

   a. If the NO DATA phrase is specified, the RECEIVE operation is terminated with the indication that action is complete (see General Rule 5), and the imperative statement in the NO DATA phrase is executed.

   b. If the NO DATA phrase is not specified, execution of the object program is suspended until data is made available in the data item referenced by identifier-1.

Withdrawn/Retiree

c. If one or more queues or sub-queues is unknown to the MCS, control passes to the next executable statement, whether or not the NO DATA phrase is specified. (See Figure 6.12-1, Communication Status Key Condition.)

5. The data items identified by the input CD are appropriately updated by the Message Control System at each execution of a RECEIVE statement. (See 6.12.4, The Communication Description — Complete Entry Skeleton.)

6. A single execution of a RECEIVE statement never returns to the data item referenced by identifier-1 more than a single message (when the MESSAGE phrase is used) or a single segment (when the SEGMENT phrase is used). However, the MCS does not pass any portion of a message to the object program until the entire message is available in the input queue, even if the SEGMENT phrase of the RECEIVE statement is specified.

7. When the MESSAGE phrase is used, end of segment indicators are ignored, and the following rules apply to the data transfer:

   a. If a message is the same size as the area referenced by identifier-1, the message is stored in the area referenced by identifier-1.

   b. If a message size is less than the area referenced by identifier-1, the message is aligned to the leftmost character position of the area referenced by identifier-1 with no space fill.

   c. If a message size is greater than the area referenced by identifier-1, the message fills the area referenced by identifier-1 left to right starting with the leftmost character of the message. The remainder of the message can be transferred to the area referenced by identifier-1 with subsequent RECEIVE statements referring to the same queue, sub-queue . . . . The remainder of the message, for the purposes of applying rules 7a, 7b, and 7c, is treated as a new message.

8. When the SEGMENT phrase is used, the following rules apply:

   a. If a segment is the same size as the area referenced by identifier-1, the segment is stored in the area referenced by identifier-1.

   b. If the segment size is less than the area referenced by identifier-1, the segment is aligned to the leftmost character position of the area referenced by identifier-1 with no space fill.

   c. If a segment size is greater than the area referenced by identifier-1, the segment fills the area referenced by identifier-1 left to right starting with the leftmost character of the segment. The remainder of the segment can be transferred to the area referenced by identifier-1 with subsequent RECEIVE statements calling out the same queue, sub-queue . . . . The remainder of the segment, for the purposes of applying rules 8a, 8b, and 8c, is treated as a new segment.

   d. If the text to be accessed by the RECEIVE statement has associated with it an End of Message Indicator or End of Group Indicator, the existence of an End of Segment Indicator associated with

Withdrawn/Retirée

the text is implied and the text is treated as a message segment according to General Rule 8.

9. Once the execution of a RECEIVE statement has returned a portion of a message, only subsequent execution of RECEIVE statements in that run unit can cause the remaining portion of the message to be returned.

10. After the execution of a STOP RUN statement, the disposition of a remaining portion of a message partially obtained in that run unit is defined by the implementor. (See 7.43.4, The STOP Statement.)

Withdrawn/Retirée

7.35 THE RELEASE STATEMENT

7.35.1 FUNCTION

The RELEASE statement transfers records to the initial phase of a SORT operation.

7.35.2 GENERAL FORMAT

RELEASE record-name [FROM identifier]

7.35.3 SYNTAX RULES

1. A RELEASE statement may only be used within the range of an input procedure associated with a SORT statement for a file whose Sort-Merge File Description entry contains record-name.

2. Record-name must be the name of a logical record in the associated Sort-Merge File Description entry and may be qualified.

7.35.4 GENERAL RULES

1. The explanation of the FROM phrase is stated under the appropriate paragraph. (See 7.7.12, The FROM Option.)

2. The execution of a RELEASE statement causes the record named by record-name to be released to the initial phase of a sort operation.

3. After the execution of the RELEASE statement, the logical record is no longer available in the record area unless the associated sort-merge file is named in a SAME RECORD AREA clause. The logical record is also available to the program as a record of other files referenced in the same SAME RECORD AREA clause as the associated sort-merge file, as well as to the file associated with record-name.

Withdrawn/Retirée

7.36 THE RETURN STATEMENT

7.36.1 FUNCTION

The RETURN statement obtains either sorted records from the final phase of a SORT operation or merged records during a MERGE operation.

7.36.2 GENERAL FORMAT

RETURN file-name RECORD [INTO identifier]; AT END imperative-statement

7.36.3 SYNTAX RULES

1. File-name must be described by a Sort-Merge File Description entry in the Data Division.

2. A RETURN statement may only be used within the range of an output procedure associated with a SORT or MERGE statement for file-name.

7.36.4 GENERAL RULES

1. The explanation of the INTO phrase is stated under the appropriate paragraph. (See 7.7.13, The INTO Option.)

2. 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.

3. Before control is passed to the associated output procedure, the Next Record Pointer is updated to point to the record whose key values make it the first record in the file referenced by file-name. If no records exist in the file, the Next Record Pointer is updated to indicate the at end condition.

4. The execution of a RETURN statement may cause the following to occur:

   a. The record, pointed to by the Next Record Pointer, is made available in the record area associated with file-name.

   b. The Next Record Pointer is updated to point to the next existing record in the file as determined by the keys listed in the SORT or MERGE statement.

   c. If no next record exists in the file, the Next Record Pointer is updated to indicate the at end condition.

Withdrawn/Retirée

5. If, at the time of the execution of a RETURN statement, the Next Record Pointer indicates the at end condition, control is transferred to the imperative-statement of the AT END phrase. The contents of the record areas associated with the file when the at end condition occurs are undefined.

6. When the at end condition occurs, the execution of the MERGE or SORT statement which caused the condition is unsuccessful.

Withdrawn/Retirée

7.37 THE REWRITE STATEMENT

7.37.1 FUNCTION

The REWRITE statement logically replaces a record existing in a mass storage file.

7.37.2 GENERAL FORMAT

REWRITE record-name [FROM identifier]

[; INVALID KEY imperative-statement]

7.37.3 SYNTAX RULES

1. The file associated with record-name must not be a sort or merge file.

2. Record-name is the name of a logical record in the File Section of the Data Division and may be qualified.

3. The INVALID KEY phrase must not be specified for a REWRITE statement which references a sequential or relative file in sequential access mode.

4. The INVALID KEY phrase must be specified in the REWRITE statement for indexed files and for relative files in the random or dynamic access mode, and for which an appropriate Format 1 USE procedure is not specified.

7.37.4 GENERAL RULES

ALL FILES

1. The explanation of the FROM phrase is stated under the appropriate paragraph. (See 7.7.12, The FROM Option.)

2. The file associated with record-name must be a mass storage file and must be open in the I-O mode at the time of execution of this statement. (See 7.30, The OPEN Statement.)

3. For mass storage 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 MSCS logically replaces the record that was accessed by the READ statement.

4. The logical record released by a successful execution of the REWRITE statement is no longer available in the record area unless the associated file is named in a SAME RECORD AREA clause, in which case the logical record is available to the program as a record of other files appearing in the same SAME

Withdrawn/Retirée

RECORD AREA clause as the associated I-O file, as well as to the file associated with record-name.

5. The Next Record Pointer is not affected by the execution of a REWRITE statement.

SEQUENTIAL FILES

6. The number of character positions in the record referenced by record-name must be equal to the number of character positions in the record being replaced.

7. The execution of the REWRITE statement causes the value of the FILE STATUS data item, if any, associated with the file to be updated. (See 7.7.9, I-O Status.)

RELATIVE FILES

8. For a file accessed in either random or dynamic access mode, the MSCS logically replaces the record specified by the contents of the RELATIVE KEY data item associated with the file. If the file does not contain the record specified by the key, the invalid key condition exists. (See 7.7.10, The Invalid Key Condition.) The updating operation does not take place and the data in the record area is unaffected.

INDEXED FILES

9. For a file in the sequential access mode, the record to be replaced is specified by the value contained in the Prime Record Key. When the REWRITE statement is executed the value contained in the Prime Record Key data item 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.

10. For a file in the random or dynamic access mode, the record to be replaced is specified by the Prime Record Key data item.

11. 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 duplicate key values are permitted and 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. The record is logically positioned last within the group of records containing the same Alternate Record Key value as the Alternate Record Key value that was changed in the record.

Withdrawn/Retirée

REWRITE

12. The invalid key condition exists when:

    a. The access mode is sequential and the value contained in the Prime Record Key data item of the record to be replaced is not equal to the value of the Prime Record Key of the last record read from this file, or

    b. The value contained in the Prime Record Key data item does not equal that of any record stored in the file, or

    c. The value contained in an ALTERNATE RECORD KEY data item for which a DUPLICATES clause has not been specified is equal to that of a record already stored in the file.

    The updating operation does not take place and the data in the record area is unaffected. (See 7.7.10, The Invalid Key Condition.)

Withdrawn/Retiree

7.38 THE SEARCH STATEMENT

7.38.1 FUNCTION

The SEARCH statement is used to search a table for a table element that satisfies the specified condition and to adjust the associated index-name to indicate that table element.

7.38.2 GENERAL FORMAT

Format 1

```
SEARCH identifier-1  [ VARYING { index-name-1 } ]
                               { identifier-2 }
    [; AT END imperative-statement-1]

; WHEN condition-1  { imperative-statement-2 }
                    { NEXT SENTENCE        }

[; WHEN condition-2  { imperative-statement-3 } ] . . .
                     { NEXT SENTENCE          }
```

Format 2

```
SEARCH ALL identifier-1  [; AT END imperative-statement-1]

    ; WHEN { data-name-1      } { EQUALS      } { identifier-3                 }
           { condition-name-1 } { IS EQUAL TO } { literal-1                    }
                               { IS =        } { arithmetic-expression-1      }

[ AND { data-name-2      } { EQUALS      } { identifier-4                 } ] . . .
      { condition-name-2 } { IS EQUAL TO } { literal-2                    }
                          { IS =        } { arithmetic-expression-2      }

{ imperative-statement-2 }
{ NEXT SENTENCE          }
```

Withdrawn/Retirée

SEARCH

7.38.3 SYNTAX RULES

1. In both Formats 1 and 2, identifier-1 must not be subscripted, indexed or reference modified, but its description must contain an OCCURS clause and an INDEXED BY clause. The description of identifier-1 in Format 2 must also contain the KEY IS phrase in its OCCURS clause.

2. Identifier-2, when specified, must be described as USAGE IS INDEX or as a numeric elementary data item without any positions to the right of the assumed decimal point.

3. In Format 1, condition-1, condition-2, etc., may be any conditional expression. (See 7.5, Conditional Expressions.)

4. 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 clause of identifier-1. Each data-name-1, data-name-2 may be qualified. Each data-name-1, data-name-2 must be indexed by the first index-name associated with identifier-1 along with other indices or literals as required, and must be referenced in the KEY clause of identifier-1. Identifier-3, identifier-4, or identifiers specified in arithmetic-expression-1, arithmetic-expression-2 must not be referenced in the KEY clause of identifier-1 or be indexed by the first index-name associated with identifier-1.

   In Format 2, when a data-name in the KEY clause of identifier-1 is referenced, or when a condition-name associated with a data-name in the KEY clause of identifier-1 is referenced, all preceding data-names in the KEY clause of identifier-1 or their associated condition-names must also be referenced.

7.38.4 GENERAL RULES

1. If Format 1 of the SEARCH 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 6.33.4.3, 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 next executable sentence.

   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 are satisfied, the index-name for identifier-1 is

Withdrawn/Retirée

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 1a above. If one of the conditions is satisfied upon its evaluation, the search terminates immediately and the imperative statement associated with that condition is executed; the index-name remains set at the occurrence which caused the condition to be satisfied.

2. In a Format 2 SEARCH, 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 ASCENDING/DESCENDING KEY clause associated with the description of identifier-1, and

   b. The contents of the key(s) referenced in the WHEN clause are sufficient to identify a unique table element.

3. If Format 2 of the SEARCH 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 in a manner specified by the implementor, 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 6.33.4.3, The OCCURS Clause.) If any of the conditions specified in the WHEN clause 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 conditions to be satisfied, and control passes to imperative-statement-2.

4. After execution of imperative-statement-1, imperative-statement-2, or imperative-statement-3, that does not terminate with a GO TO statement, control passes to the next executable sentence.

5. 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 clause of identifier-1. Any other index-names for identifier-1 remain unchanged.

6. 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 clause of identifier-1. Any other index-names for identifier-1 remain unchanged.

7. In Format 1, if the VARYING index-name-1 phrase is specified, and if index-name-1 appears in the INDEXED BY clause of 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 clause of identifier-1 is used for the search. In addition, the following operations will occur:

   a. If the VARYING index-name-1 phrase is used, and if index-name-1 appears in the INDEXED BY clause of another table entry, the occurrence number represented by index-name-1 is

Withdrawn/Retirée

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 (1) at the same time as the index referenced by the index-name associated with identifier-1 is incremented.

8. If identifier-1 is 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.

Withdrawn/Retired

A figure of the Format 1 SEARCH operation containing two WHEN phrases follows.

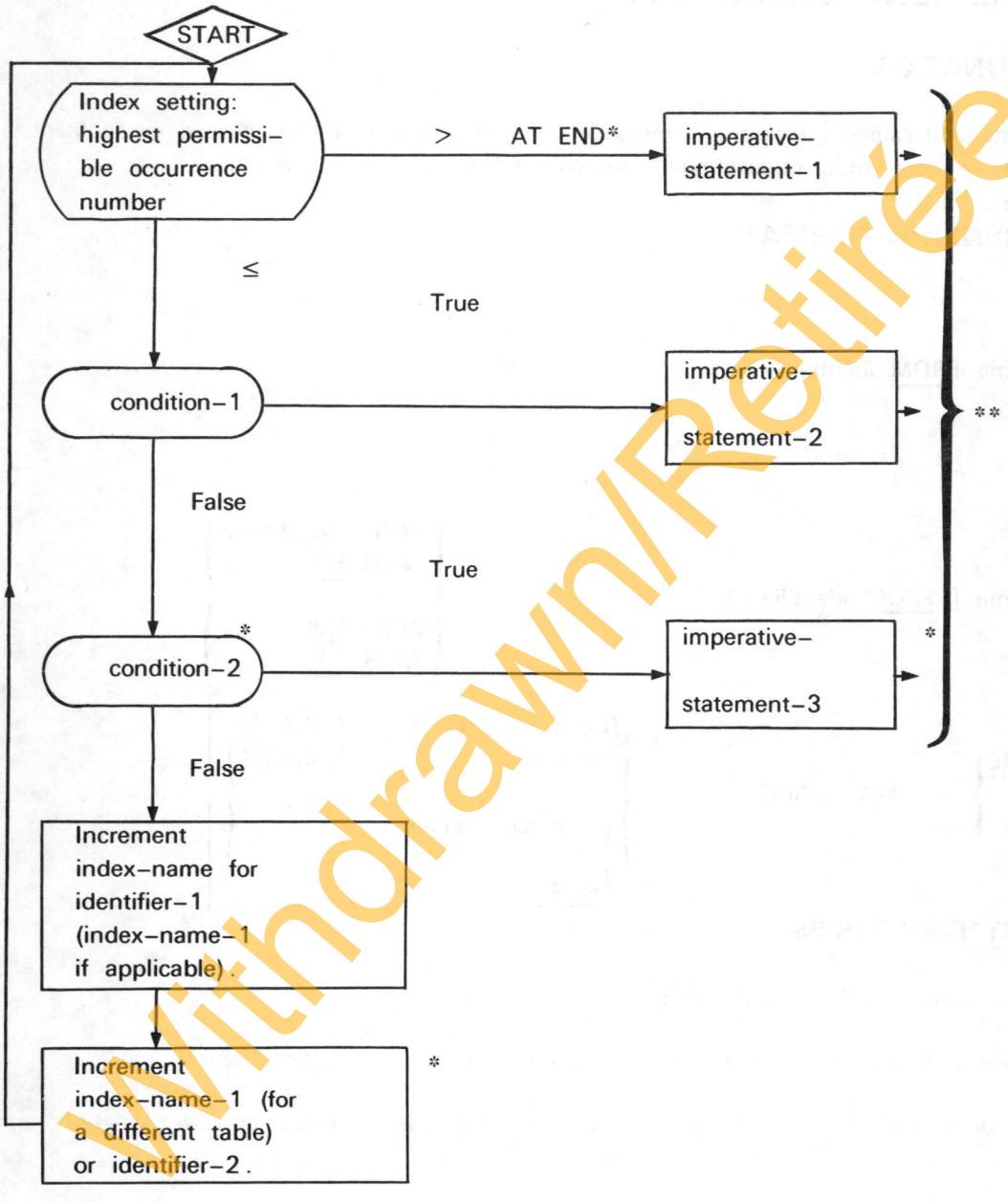


Figure 7.38-1 Format 1 SEARCH Statement Having Two WHEN Phrases

* These operations are options included only when specified in the SEARCH statement.

** Each of these control transfers is to the next executable sentence unless the imperative-statement ends with a GO TO statement.

Withdrawn/Retirée

7.39 THE SEND STATEMENT

7.39.1 FUNCTION

The SEND statement causes a message, a message segment, or a portion of a message or segment to be released to one or more output queues maintained by the Message Control System.

7.39.2 GENERAL FORMAT

Format 1

```
SEND cd-name FROM identifier-1
```

Format 2

```
SEND cd-name [ FROM identifier-1 ]  { WITH identifier-2 }
                                    { WITH ESI        }
                                    { WITH EMI        }
                                    { WITH EGI        }

[ { BEFORE } ADVANCING { { identifier-3 } [ LINE  ] } ]
  { AFTER  }           { { integer      } [ LINES ] }
                       { mnemonic-name              }
                       { PAGE                       }
```

7.39.3 SYNTAX RULES

1. Cd-name must reference an output CD.

2. Identifier-2 must reference a one character integer without an operational sign.

3. When identifier-3 is used in the ADVANCING phrase, it must be the name of an elementary integer item.

4. When the mnemonic-name phrase is used, the name is identified with a particular feature specified by the implementor. The mnemonic-name is defined in the SPECIAL-NAMES paragraph of the Environment Division.

Withdrawn/Retiree

5. Integer or the value of the data item referenced by identifier-3 may be zero.

7.39.4 GENERAL RULES

ALL FORMATS

1. When a receiving communications device (printer, display screen, card punch, etc.) is oriented to a fixed line size:

   a. Each message or message segment will begin at the leftmost character position of the physical line.

   b. A message or message segment that is smaller than the physical line size is released so as to appear space-filled to the right.

   c. Excess characters of a message or message segment will not be truncated. Characters will be packed to a size equal to that of the physical line and then transmitted to the output device. The process continues on the next line with the excess characters.

2. When a receiving communications device (paper tape punch, another computer, etc.) is oriented to handle variable-length messages, each message or message segment will begin on the next available character position of the communications device.

3. As part of the execution of a SEND statement, the MCS will interpret the contents of the data item referenced by data-name-2 (TEXT LENGTH) of the area referenced by cd-name to be the user's indication of the number of leftmost character positions of the data item referenced by identifier-1 from which data is to be transferred.

   If the contents of the data item referenced by data-name-2 (TEXT LENGTH) of the area referenced by cd-name are zero, no characters of the data item referenced by identifier-1 are transferred.

   If the contents of the data item referenced by data-name-2 (TEXT LENGTH) of the area referenced by cd-name are outside the range of zero through the size of the data item referenced by identifier-1 inclusive, an error is indicated by the value of the data item referenced by data-name-3 (STATUS KEY) of the area referenced by cd-name, and no data is transferred. (See Figure 6.12-1, Communication Status Key Condition.)

4. As part of the execution of a SEND statement, the contents of the data item referenced by data-name-3 (STATUS KEY) of the area referenced by cd-name is updated by the MCS. (See 6.12.4, The Communication Description - Complete Entry Skeleton.)

5. The effect of having special control characters within the contents of the data item referenced by identifier-1 is undefined.

Withdrawn/Retirée

SEND

6. A single execution of a SEND statement of Format 1 releases only a single portion of a message segment or a single portion of a message to the MCS.

   A single execution of a SEND statement of Format 2 never releases to the MCS more than a single message or a single message segment as indicated by the contents of the data item referenced by identifier-2 or by the specified indicator ESI, EMI or EGI.

   However, the MCS will not transmit any portion of a message to a communications device until the entire message is placed in the output queue.

7. During the execution of the run unit, the disposition of a portion of a message which is not terminated by an EMI or EGI or which has not been eliminated by the execution of a PURGE statement is undefined. However, the message does not logically exist for the MCS and hence cannot be sent to a destination.

   After the execution of a STOP RUN statement, any portion of a message transferred from the run unit via a SEND statement, but not terminated by an EMI or EGI, is purged from the system. Thus no portion of the message is sent.

8. Once the execution of a SEND statement has released a portion of a message to the MCS, only subsequent execution of SEND statements in the same run unit can cause the remaining portion of the message to be released.

FORMAT 2

9. The contents of the data item referenced by identifier-2 indicate that the contents of the data item referenced by identifier-1, when specified, are to have an associated End of Segment Indicator, End of Message Indicator, End of Group Indicator or no indicator (which implies a portion of a message or a portion of a segment). If identifier-1 is not specified, only the indicator is transmitted to the MCS.

Withdrawn/Retirée

If the contents of the data item referenced by identifier–2 are	then the contents of data item referenced by identifier–1 have an associated	which means
0	no indicator	portion of message or of a segment
1	(ESI) End of Segment Indicator	end of current segment
2	(EMI) End of Message Indicator	end of current message
3	(EGI) End of Group Indicator	end of the current group of messages

If the contents of the data item referenced by identifier-2 is other than '1', '2', or '3', and identifier-1 is not specified, then an error is indicated by the value in the data item referenced by data-name-3 (STATUS KEY) of the area referenced by cd-name, and no data is transferred.

10. The ESI indicates to the MCS that the message segment is complete. The EMI indicates to the MCS that the message is complete.

    The EGI indicates to the MCS that the group of messages is complete. The implementor will specify the interpretation that is given to the EGI by the MCS.

    The MCS will recognize these indications and establish whatever is necessary to maintain group, message, and segment control.

11. The hierarchy of ending indicators is EGI, EMI, and ESI. An EGI need not be preceded by an ESI or EMI. An EMI need not be preceded by an ESI.

12. The ADVANCING phrase allows control of the vertical positioning of each message or message segment on a communications device where vertical positioning is applicable. If vertical positioning is not applicable on the device, the MCS will ignore the vertical positioning specified or implied.

13. If identifier-2 is specified and the content of the data item referenced by identifier-2 is zero, the ADVANCING phrase is ignored by the MCS.

14. On a device where vertical positioning is applicable and the ADVANCING phrase is not specified, automatic advancing will be provided by the implementor to act as if the user had specified AFTER ADVANCING 1 LINE.

Withdrawn/Retiree

15. If the ADVANCING phrase is implicitly or explicitly specified and vertical positioning is applicable, the following rules apply:

a. If identifier-3 or integer is specified, characters transmitted to the communications device will be repositioned vertically downward the number of lines equal to the value associated with the data item referenced by identifier-3 or integer.

b. If mnemonic-name is specified, characters transmitted to the communications device will be positioned according to the rules specified by the implementor for that communications device.

c. If the BEFORE phrase is used, the message or message segment is represented on the communications device before vertical repositioning according to General Rules 15a and 15b above.

d. If the AFTER phrase is used, the message or message segment is represented on the communications device after vertical repositioning according to General Rules 15a and 15b above.

e. If PAGE is specified, characters transmitted to the communications device will be represented on the device before or after (depending upon the phrase used) the device is repositioned to the next page. If PAGE is specified but page has no meaning in conjunction with a specific device, then advancing will be provided by the implementor to act as if the user had specified BEFORE or AFTER (depending upon the phrase used) ADVANCING 1 LINE.

Withdrawn/Retirée

7.40 THE SET STATEMENT

7.40.1 FUNCTION

1. The SET statement establishes reference points for table-handling operations by setting index-names associated with table elements, and is used to alter the status of external switches.

2. The SET statement also names a specific character code set used in character code conversion and/or a specific collating sequence used in nonnumeric comparisons.

3. The SET statement is also used to move the contents of one data base key item to one or more data base key items.

4. The SET statement is also used to control the null data attribute of a data item.

7.40.2 GENERAL FORMAT

Format 1

```
SET { index-name-1 }  [ , index-name-2 ]  . . .  TO { index-name-3 }
    { identifier-1 }  [ , identifier-2 ]           { identifier-3 }
                                                   { integer-1    }
```

Format 2

```
SET index-name-4  [ , index-name-5 ]  . . .  { UP BY   } { identifier-4 }
                                             { DOWN BY } { integer-2    }
```

Format 3

```
SET { { SORT       }                                                    }
    { { MERGE      }  COLLATING SEQUENCE                                }
    { { SORT-MERGE }                                                    }
    { { PROGRAM    }                                                    }
    {                                                                   } TO alphabet-name
    { CODE-SET FOR { file-name  [ , file-name ]  . . . }                }
    {              { ALL FILES                         }                }
```

Format 4

```
SET { mnemonic-name-1  [ , mnemonic-name-2 ]  . . .  TO { ON  } }  . . .
                                                       { OFF }
```

Withdrawn/Retired

SET

Format 5

SET {identifier-5} . . . TO identifier-6

Format 6

SET {identifier-7} . . . TO {NULL / NON-NULL}

7.40.3 SYNTAX RULES

1. All references to index-name-1, identifier-1, and index-name-4 apply equally to index-name-2, identifier-2 and index-name-5, respectively.

2. Identifier-1 and identifier-3 must name either index data items, or elementary items described as an integer.

3. Identifier-4 must be described as an elementary numeric integer.

4. Integer-1 and integer-2 may be signed. Integer-1 must be positive.

5. If the CODE-SET phrase is specified, all data in the specified files must be described as USAGE IS DISPLAY and any signed numeric data must be described with the SIGN IS SEPARATE clause.

6. Each mnemonic-name must be associated with an external switch, the status of which can be altered. The implementor defines which external switches can be referenced by the SET statement.

7. Identifier-5 and identifier-6 must reference data base key items.

8. Identifier-7 must be defined in the Sub-schema as an elementary data item, but it must not be reference modified.

7.40.4 GENERAL RULES

FORMATS 1 and 2

1. Index-names are considered related to a given table and are defined by being specified in the INDEXED BY clause.

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 SEARCH or PERFORM statement may be set to an occurence number that is outside the range of its associated table. (See 7.38.4, The SEARCH Statement and 7.31.4, The PERFORM Statement.)

Withdrawn/Retirée

If index-name-3 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-4 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-4.

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-3, identifier-3, or integer-1. If identifier-3 is an index data item, or if index-name-3 is related to the same table as index-name-1, no conversion takes place.

   b. If identifier-1 is an index data item, it may be set equal to either the contents of index-name-3 or identifier-3 where identifier-3 is also an index data item; no conversion takes place in either case.

   c. If identifier-1 is not an index data item, it may be set only to an occurrence number that corresponds to the value of index-name-3. Neither identifier-3 nor integer-1 can be used in this case.

   d. The process is repeated for index-name-2, identifier-2, ..., if specified. Each time, the value of index-name-3 or the data item referenced by identifier-3 is used as it was at the beginning of the execution of the statement. Any subscripting or indexing associated with identifier-1, ..., is evaluated immediately before the value of the respective data item is changed.

4. In Format 2, the contents of index-name-4 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-4; thereafter, the process is repeated for index-name-5, ... For each repetition the value of the data item referenced by identifier-4 is used as it was at the beginning of the execution of the statement.

5. Data in the following figure represents the validity of various operand combinations in Formats 1 and 2 of the SET statement. The general rule reference indicates the applicable general rule.

Withdrawn/Retirée

Sending Item	Receiving Item		
	Integer Data Item	Index-Name	Index Data Item
Integer Literal	No/3c	Valid/3a	No/3b
Integer Data Item	No/3c	Valid/3a	No/3b
Index-name	Valid/3c	Valid/3a	Valid/3b*
Index Data Item	No/3c	Valid/3a*	Valid/3b*

Figure 7.40-1 Validity of Operand Combinations in Format 1 and 2 SET Statements

*No conversion takes place.

FORMAT 3

6. The execution of Format 3 SET statements changes:

   a. The character code set used to represent data on the external media associated with a file. (See 5.4.1, The FILE-CONTROL Paragraph.)

   b. The program collating sequence. (See 5.3.2, The OBJECT-COMPUTER Paragraph.)

   c. The collating sequence used during sort or merge operations. (See 7.27, The MERGE Statement and 7.41, The SORT Statement.)

7. After the execution of a SET statement with:

   a. A SORT or SORT-MERGE phrase, the collating sequence used for subsequent sort operations is that specified by the ALPHABET clause referenced by alphabet-name. The collating sequence associated with alphabet-name applies only to the ordering of the records made available to the sort operation.

   b. A MERGE or SORT-MERGE phrase, the collating sequence used for subsequent merge operations is that specified by the ALPHABET clause referenced by alphabet-name. The collating sequence associated with alphabet-name applies only to the ordering of the records made available to the merge operation.

   c. A PROGRAM phrase, the program collating sequence used for subsequent nonnumeric comparisons is that specified by the ALPHABET clause referenced by alphabet-name.

Withdrawn/Retirée

d. A CODE-SET phrase, the character code set used for subsequent code conversion for the referenced file is that specified by the ALPHABET clause referenced by alphabet-name. The character code set conversion in effect immediately prior to the execution of an OPEN statement for the associated file is used during the complete execution of that OPEN statement and during the complete execution of the corresponding CLOSE statement for that file. The character code set conversion in effect immediately prior to the execution of the first READ, REWRITE or WRITE statement for the associated file is used for the processing of all data records on that file.

FORMAT 4

8. The status of each external switch associated with the specified mnemonic-name 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. (See 7.5.1.4, Switch-Status Condition.)

FORMAT 5

9. The contents of the data base key item referenced by identifier-6 are moved to the data base key items referenced by identifier-5.

FORMAT 6

10. If NULL is specified, the null data attribute is associated with the data item referenced by identifier-7.

11. If NON-NULL is specified, any null data attribute associated with identifier-7 is removed.

Withdrawn/Retired

7.41 THE SORT STATEMENT

7.41.1 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.

7.41.2 GENERAL FORMAT

```
SORT file-name-1 { , ON { DESCENDING } KEY { , data-name-1 } . . . } . . .
                       { ASCENDING  }
    [WITH DUPLICATES IN SEQUENCE]
    [COLLATING SEQUENCE IS  alphabet-name]

{ INPUT  PROCEDURE IS section-name-1 [ { THRU    } section-name-2 ] }
{                                    { THROUGH }                  }
{ USING { , file-name-2 } . . .                                   }

{ OUTPUT PROCEDURE IS section-name-3 [ { THRU    } section-name-4 ] }
{                                    { THROUGH }                  }
{ GIVING { , file-name-3 } . . .                                  }
```

7.41.3 SYNTAX RULES

1. SORT 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. If the USING phrase is specified and the file referenced by file-name-1 contains variable-length records, the size of the records contained in the file referenced by file-name-2 must not be less than the smallest record nor larger than the largest record described for file-name-1. If the file referenced by file-name-1 contains fixed-length records, the size of the records contained in the file referenced by file-name-2 must not be larger than the largest record described for the file referenced by file-name-1.

Withdrawn/Retiree

4. 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 variable-length data items, nor may they name 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.

   f. The data items identified by KEY data-names must not be described as boolean data items.

5. Section-name-1 represents the name of an input procedure. Section-name-3 represents the name of an output procedure.

6. The words THRU and THROUGH are equivalent.

7. 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.

8. The files referenced by file-name-2 and file-name-3 may reside on the same multiple file reel(s).

9. If file-name-3 references an indexed file, then the major key must be associated with the ASCENDING phrase and the first data-name must specify the same character positions in its record as are specified for the Prime Record Key for that file.

10. No pair of file-names in a SAME SORT statement may be specified in the same SAME SORT-MERGE AREA clause. File-names associated with the GIVING phrase may not be specified in the same SAME AREA clause. (See 5.4.2, The I-O-CONTROL Paragraph.)

11. If the GIVING phrase is specified and the file referenced by file-name-3 contains variable-length records, the size of the records contained in the file referenced by file-name-1 must not be less than the smallest record nor larger than the largest record described for file-name-3. If the file referenced by file-name-3 contains fixed-length records, the size of the records contained in the file referenced by file-name-1 must not be larger than the largest record described for the file referenced by file-name-3.

Withdrawn/Retirée

7.41.4 GENERAL RULES

1. If the file referenced by file-name-1 contains only fixed-length records, any record in the file referenced by file-name-2 containing fewer character positions than that fixed length is space-filled on the right beginning with the first character position after the last character in the record when that record is released to the file referenced by file-name-1.

2. 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.

3. 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.

4. 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.

5. The collating sequence that applies to the comparison of the nonnumeric key data items specified is determined at the beginning of the execution of the SORT statement in the following order of precedence:

   a. First, the collating sequence established by the most recently executed SET statement with a SORT or SORT-MERGE phrase.

   b. Second, the collating sequence established by the COLLATING SEQUENCE phrase, if specified, in the SORT statement.

Withdrawn/Retirée

c. Third, the collating sequence established as the program collating sequence.

6. The input procedure must consist of one or more sections that appear contiguously in a source program and do not form a part of any output procedure. In order to transfer records to the file referenced by file-name-1, the input procedure must include the execution of at least one RELEASE statement. Control must not be passed to the input procedure except when a related SORT statement is being executed. The input procedure can include any procedures needed to select, create, or modify records. The restrictions on the procedural statements within the input procedure are as follows:

   a. The input procedure must not contain any SORT or MERGE statements.

   b. The input procedure must not contain any explicit transfers of control to points outside the input procedure; GO TO, and PERFORM statements in the input procedure are not permitted to refer to procedure-names outside the input procedure. COBOL statements are allowed that will cause an implied transfer of control to Declaratives.

   c. The remainder of the Procedure Division must not contain any transfers of control to points inside the input procedure; GO TO and PERFORM statements in the remainder of the Procedure Division must not refer to procedure-names within the input procedure.

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. Before control passes the last statement in the input procedure, the file referenced by file-name-3 is not in the open mode. The compiler inserts a return mechanism at the end of the last section in the input procedure and when control passes the last statement in the input procedure, the records that have been released to the file referenced by file-name-1 are sorted.

8. During the execution of the input procedure, the output procedure or any USE AFTER EXCEPTION procedure, no statement manipulating the files referenced by, or accessing the record areas associated with file-name-2 or file-name-3 may be executed.

9. 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. At the time of execution of the SORT statement, the file(s) referenced by file-name-2 must not be in the open mode. 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.

Withdrawn/Retired

These implicit functions are performed such that any associated USE procedures are executed.

10. The output procedure must consist of one or more sections that appear contiguously in a source program and do not form part of any input procedure. In order to make sorted records available for processing, the output procedure must include the execution of at least one RETURN statement. Control must not be passed to the output procedure except when a related SORT statement is being executed. The output procedure may consist of any procedures needed to select, modify or copy the records that are being returned, one at a time in sorted order, from the sort file. The restrictions on the procedural statements within the output procedure are as follows:

    a. The output procedure must not contain any SORT or MERGE statements.

    b. The output procedure must not contain any explicit transfers of control to points outside the output procedure; GO TO, and PERFORM statements in the output procedure are not permitted to refer to procedure-names outside the output procedure. COBOL statements are allowed that will cause an implied transfer of control to Declaratives.

    c. The remainder of the Procedure Division must not contain any transfers of control to points inside the output procedure; GO TO and PERFORM statements in the remainder of the Procedure Division are not permitted to refer to procedure-names within the output procedure.

11. 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 file referenced by file-name-2 is not in the open mode. The compiler inserts a return mechanism at the end of the last section 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.

12. 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. At the time of the execution of the SORT statement, the file referenced by file-name-3 must not be in the open mode. 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.

    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.

    For a relative file, the Relative Key data item for the first record returned contains the value '1'; for the second record returned, the value '2', etc. After execution of the SORT statement, the contents of the Relative Key data item indicates the last record returned to the file.

Withdrawn/Retired

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 procedures are executed. On the first attempt to write beyond the externally-defined boundaries of the file, any USE AFTER EXCEPTION procedure specified for the file is executed; if control is returned from the USE procedure or if no USE procedure is specified, the processing of the file is terminated as in paragraph 9c above.

13. If the file referenced by file-name-3 contains only fixed-length records, any record in the file referenced by file-name-1 containing fewer character positions than that fixed length is space-filled on the right beginning with the first character position after the last character in the record when that record is returned from the file referenced by file-name-3.

Withdrawn/Retired

7.42 THE START STATEMENT

7.42.1 FUNCTION

The START statement provides a basis for logical positioning within an indexed or relative file, for subsequent sequential retrieval of records.

7.42.2 GENERAL FORMAT

```
START file-name [ KEY { IS EQUAL TO
                      | EQUALS
                      | IS =
                      | EXCEEDS
                      | IS GREATER THAN
                      | IS >
                      | IS NOT LESS THAN
                      | IS NOT < } data-name ]
    [; INVALID KEY imperative-statement]
```

7.42.3 SYNTAX RULES

1. File-name must be the name of an indexed or relative file.
2. File-name must be the name of a file with sequential or dynamic access.
3. File-name must not be the name of a sort or merge file.
4. Data-name may be qualified.
5. Data-name must not reference a boolean data item.
6. The INVALID KEY phrase must be specified if no applicable Format 1 USE procedure is specified for file-name.
7. If file-name represents a relative file, data-name, if specified, must be the data item specified in the RELATIVE KEY phrase of the associated File-Control entry.
8. If file-name is the name of an indexed file, and if the KEY phrase is specified, data-name may reference a data item specified as a record key associated with file-name, or it may reference any data item of category alphanumeric subordinate to the data-name of a data item specified as a record key associated with file-name whose leftmost character position corresponds to the leftmost character position of that record key data item.

Withdrawn/Retirée

7.42.4 GENERAL RULES

ALL FILES

1. File-name must be open in the INPUT or I-O mode at the time that the START statement is executed. (See 7.30, The OPEN Statement.)

2. If the KEY phrase is not specified, the relational operator 'IS EQUAL TO' is implied.

3. 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 and a data item as specified in General Rules 5, 6, and 7. If file-name references an indexed file and 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 numeric or nonnumeric comparison rules apply. (See 7.5.1.1.1, Comparison of Numeric Operands and 7.5.1.1.2, Comparison of Nonnumeric Operands.)

   a. The Next Record Pointer is set to the first logical record currently existing in the file whose key satisfies the comparison.

   b. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the execution of the START statement is unsuccessful, and the setting of the Next Record Pointer is undefined. (See 7.7.10, The Invalid Key Condition.)

4. The execution of the START statement causes the value of the FILE STATUS data item, if any, associated with file-name to be updated. (See 7.7.9, I-O Status.)

RELATIVE FILES

5. The comparison described in General Rule 3 uses the data item referenced by the RELATIVE KEY clause associated with file-name.

INDEXED FILES

6. If the KEY phrase is specified, the comparison described in General Rule 3 uses the data item referenced by data-name.

7. If the KEY phrase is not specified, the comparison described in General Rule 3 uses the data item referenced in the RECORD KEY clause associated with file-name.

8. Upon completion of the successful execution of the START statement, a Key of Reference is established and used in subsequent Format 1 READ statements as follows: (See 7.33, The READ Statement.)

   a. If the KEY phrase is not specified, the Prime Record Key specified for file-name becomes the Key of Reference.

Withdrawn/Retirée

b. If the KEY phrase is specified, and data-name is specified as a Record Key for file-name, that Record Key becomes the Key of Reference.

c. If the KEY phrase is specified, and data-name is not specified as a Record Key for file-name, the Record Key whose leftmost character position corresponds to the leftmost character position of the data item specified by data-name, becomes the Key of Reference.

9. If the execution of the START statement is not successful, the Key of Reference is undefined.

Withdrawn/Retirée

7.43 THE STOP STATEMENT

7.43.1 FUNCTION

The STOP statement causes a permanent or temporary suspension of the execution of the run unit.

7.43.2 GENERAL FORMAT

STOP { RUN / literal }

7.43.3 SYNTAX RULES

1. Literal may be any figurative constant, except 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.

7.43.4 GENERAL RULES

1. If the RUN phrase is specified, execution of the run unit ceases and control is transferred to the operating system.

2. If STOP literal is specified, the execution of the run unit is suspended and the literal is communicated to the operator. Continuation of the execution of the run unit begins with the next executable statement when the implementor-defined procedure governing run unit reinitiation is instituted.

Withdrawn/Retirée

7.44 THE STRING STATEMENT

7.44.1 FUNCTION

The STRING statement provides juxtaposition of the partial or complete contents of two or more data items into a single data item.

7.44.2 GENERAL FORMAT

```
STRING { identifier-1 } [, identifier-2 ]
       { literal-1    } [, literal-2    ]  . . . DELIMITED BY { identifier-3 }
                                                              { literal-3    }
                                                              { SIZE         }

[, { identifier-4 } [, identifier-5 ]                                        ]
   { literal-4    } [, literal-5    ]  . . . DELIMITED BY { identifier-6 }  ] . . .
                                                          { literal-6    }
                                                          { SIZE         }

    INTO identifier-7                                  [WITH POINTER identifier-8]

    [; ON OVERFLOW imperative-statement]
```

7.44.3 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-8, must be described implicitly or explicitly as USAGE IS DISPLAY.

3. None of the identifiers may reference boolean data items.

4. Identifier-7 must not be reference modified.

5. Identifier-7 must not represent an edited data item and must not be described with the JUSTIFIED clause.

6. Identifier-8 must represent an elementary numeric integer data item of sufficient size to contain a value equal to the size plus 1 of the area referenced by identifier-7.

7.44.4 GENERAL RULES

1. All references to identifier-1, identifier-2, identifier-3, literal-1, literal-2, literal-3 apply equally to identifier-4, identifier-5, identifier-6, literal-4, literal-5 and literal-6, respectively, and all recursions thereof.

Withdrawn/Retirée

2. Identifier-1, literal-1, identifier-2, literal-2, represent the sending items. Identifier-7 represents the receiving item.

3. Literal-3, identifier-3, indicate the character(s) delimiting the move. If the SIZE phrase is used, the complete data item defined by identifier-1, literal-1, identifier-2, literal-2, is moved. If the SIZE phrase is used and identifier-1, identifier-2 describes a variable-length data item, the number of characters currently defined for the data item 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, literal-2, literal-3, 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, literal-2, or from the contents of the data item referenced by identifier-1, identifier-2, are transferred to the contents of identifier-7 in accordance with the rules for alphanumeric to alphanumeric moves, except that no space-filling will be provided. (See 7.28.4.4.a, The MOVE Statement.)

   b. If the DELIMITED phrase is specified without the SIZE phrase, the contents of the data item referenced by identifier-1, identifier-2, or the value of literal-1, literal-2, are 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-3, or by the contents of identifier-3 are encountered. The character(s) specified by literal-3 or by the data item referenced by identifier-3 are not transferred.

   c. If the DELIMITED phrase is specified with the SIZE phrase, the entire contents of literal-1, literal-2, or the contents of the data item referenced by identifier-1, identifier-2, are transferred, in the sequence specified in the STRING statement, to the data item referenced by identifier-7 until all data has been transferred or the end of the data item referenced by identifier-7 has been reached. If identifier-1, identifier-2 describes a variable-length data item, the number of characters currently defined for the data item is moved.

   This behavior is repeated until all literal-1, ..., or data items referenced by identifier-1, ..., have been processed.

6. If the POINTER phrase is specified, identifier-8 is explicitly available to the programmer, and he is responsible for setting its initial value. The initial value must not be less than one.

7. If the POINTER phrase is not specified, the following general rules apply as if the user had specified identifier-8 with an initial value of 1.

8. When characters are transferred to the data item referenced by identifier-7, the moves behave as though the characters were moved one at a time from the source into the character position of the data item referenced by identifier-7 designated by the value associated with identifier-8 (provided the

Withdrawn/Retired

value of identifier-8 does not exceed the length of identifier-7), and then identifier-8 was increased by one prior to the move of the next character. The value associated with identifier-8 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-7 that was referenced during the execution of the STRING statement is changed. All other portions of the data item referenced by identifier-7 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-7, if the value associated with identifier-8 is either less than one or exceeds the number of character positions in the data item referenced by identifier-7, no (further) data is transferred to the data item referenced by identifier-7, and the imperative statement in the ON OVERFLOW phrase is executed, if specified.

11. If the ON OVERFLOW phrase is not specified when the conditions described in General Rule 10 above are encountered, control is transferred to the next executable statement.

Withdrawn/Retiree

7.45 THE SUBTRACT STATEMENT

7.45.1 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.

7.45.2 GENERAL FORMAT

Format 1

```
SUBTRACT { literal-1    } [ , literal-2    ] . . . FROM identifier-m [ROUNDED ]
         { identifier-1 }   [ , identifier-2 ]

    [ , identifier-n [ROUNDED] ]  . . . [; ON SIZE ERROR imperative-statement]
```

Format 2

```
SUBTRACT { literal-1    } [ , literal-2    ] . . . FROM { literal-m    }
         { identifier-1 }   [ , identifier-2 ]        { identifier-m }

    GIVING identifier-n [ROUNDED]  [ , identifier-o [ROUNDED] ] . . .

    [; ON SIZE ERROR imperative-statement]
```

Format 3

```
SUBTRACT { CORRESPONDING } identifier-1 FROM identifier-2 [ROUNDED]
         { CORR          }

    [; ON SIZE ERROR imperative-statement]
```

Withdrawn/Retirée

SUBTRACT

7.45.3 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. CORR is an abbreviation for CORRESPONDING.

7.45.4 GENERAL RULES

1. In Format 1, all literals or identifiers preceding the word FROM are added together and this total is subtracted from the current value of identifier-m storing the result immediately into identifier-m, and repeating this process respectively for each recurrence of identifier-n.

2. In Format 2, all literals or identifiers preceding the word FROM are added together, the sum is subtracted from literal-m or identifier-m and the result of the subtraction is stored as the new value of identifier-n, identifier-o, etc.

3. If Format 3 is used, data items in identifier-1 are subtracted from and stored into corresponding data items in identifier-2.

4. Additional rules and explanations relative to this statement are given under the appropriate paragraph. (See 7.7.1, Intermediate Data Item; 7.7.2, The ROUNDED Option; 7.7.3, The SIZE ERROR Option; 7.7.4, The CORRESPONDING Option; 7.7.5, The Arithmetic Statements; 7.7.6, Overlapping Operands; and 7.7.7, Multiple Results in Arithmetic Statements.)

Withdrawn/Retirée

7.46 THE SUPPRESS STATEMENT

7.46.1 FUNCTION

The SUPPRESS statement causes the Report Writer Control System to inhibit the presentation of a report group.

7.46.2 GENERAL FORMAT

SUPPRESS PRINTING

7.46.3 SYNTAX RULES

1. The SUPPRESS statement may only appear in a USE BEFORE REPORTING procedure.

7.46.4 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.

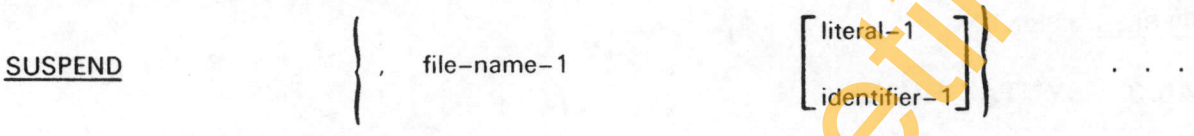
Withdrawn/Retirée

7.47 THE SUSPEND STATEMENT

7.47.1 FUNCTION

The SUSPEND statement stops the release of subsequent logical records to a graphic display device until re-initiated by the operator.

7.47.2 GENERAL FORMAT



7.47.3 SYNTAX RULES

1. The figurative constant ALL must not be used as literal-1 . . . .

7.47.4 GENERAL RULES

1. When the files referenced by file-name-1, . . ., are assigned to a graphic display device, the SUSPEND statement releases a logical record to the file named. When this logical record is transmitted to the graphic display device, the further transmission of logical records to the graphic display device is suspended until re-initiated by the graphic display device operator. When the file referenced by file-name is not assigned to a graphic display device, the effect of the SUSPEND statement is defined by the implementor.

2. The associated file must be open (OUTPUT) at the time of the execution of this statement. (See 7.30, The OPEN Statement.)

3. The object program will stop execution only when the release of logical records to the output device is not possible until re-initiated by the operator.

4. The implementor will specify the means of re-initiating the release of logical records to the graphic display device.

5. The value of literal-1 or identifier-1 is written on the implementor's standard display device. When more than one such device is present, the one most logically associated with the graphic display device will be used.

6. If a figurative constant is specified as one of the operands, only a single occurrence of the figurative constant is displayed.

Withdrawn/Retirée

7. The implementor will define, for each hardware device, the size of a data transfer.

8. 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.

9. If a 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 item being transferred, the transferred data is stored aligned to the left in the receiving hardware device.

Withdrawn/Retirée

7.48 THE TERMINATE STATEMENT

7.48.1 FUNCTION

The TERMINATE statement causes the Report Writer Control System to complete the processing of the specified reports.

7.48.2 GENERAL FORMAT

TERMINATE report-name-1 [, report-name-2] . . .

7.48.3 SYNTAX RULES

1. Each report-name given in a TERMINATE statement must be defined by an RD entry in the Report Section of the Data Division.

7.48.4 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 6.15.5, Presentation Rules Tables.)

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. 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.

Withdrawn/Retirée

7.49 THE UNSTRING STATEMENT

7.49.1 FUNCTION

The UNSTRING statement causes contiguous data in a sending field to be separated and placed into multiple receiving fields.

7.49.2 GENERAL FORMAT

```
UNSTRING identifier-1
[ DELIMITED BY [ALL] { identifier-2 } [ , OR [ALL] { identifier-3 } ] . . . ]
                     { literal-1    }               { literal-2    }
  INTO identifier-4          [ , DELIMITER IN identifier-5] [ , COUNT IN identifier-6]
     [ , identifier-7        [ , DELIMITER IN identifier-8] [ , COUNT IN identifier-9] ] . . .
  [ WITH POINTER identifier-10]                  [TALLYING IN identifier-11]
  [; ON OVERFLOW imperative-statement]
```

7.49.3 SYNTAX RULES

1. Each literal must be a nonnumeric literal. In addition, each literal may be any figurative constant without the optional word ALL.

2. Identifier-1, identifier-2, identifier-3, identifier-5, and identifier-8 must be described, implicitly or explicitly, as an alphanumeric data item.

3. Identifier-4 and identifier-7 may be described as either alphabetic (except that the symbol 'B' may not be used in the PICTURE character-string), alphanumeric, or numeric (except that the symbol 'P' may not be used in the PICTURE character-string), and must be described as USAGE IS DISPLAY.

4. Identifier-6, identifier-9, identifier-10, and identifier-11 must be described as elementary numeric integer data items.

5. No identifier may name a level 88 entry.

6. The DELIMITER IN phrase and the COUNT IN phrase may be specified only if the DELIMITED BY phrase is specified.

Withdrawn/Retirée

7. Identifier-1 must not be reference modified.

7.49.4 GENERAL RULES

1. All references to identifier-2, literal-1, identifier-4, identifier-5, and identifier-6, apply equally to identifier-3, literal-2, identifier-7, identifier-8, and identifier-9, respectively, and all recursions thereof.

2. Identifier-1 represents the sending area.

3. Identifier-4 represents the data receiving area. Identifier-5 represents the receiving area for delimiters.

4. Literal-1 or the data item referenced by identifier-2 specifies a delimiter.

5. 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 identifier-4. This value does not include a count of the delimiter character(s).

6. The data item referenced by identifier-10 contains a value that indicates a relative character position within the area defined by identifier-1.

7. The data item referenced by identifier-11 is a counter which is incremented by 1 for each 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 contents of the data item referenced by identifier-2 are treated as if it were only one occurrence, and this occurrence 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 either space or zero filled according to the description of the receiving area.

10. Literal-1 or the contents 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

Withdrawn/Retirée

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-10. 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 the value of literal-1 or 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 current receiving area is described as a variable-length data item, the current size of the data item is used to determine the number of characters examined.

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 7.28, 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 7.28, 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 characters 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.

Withdrawn/Retiree

g. After data is transferred to the data item referenced by identifier-4, the current receiving area is the data item referenced by identifier-7. 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 contents of the data item referenced by identifier-10 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 contents of the data item referenced by identifier-10 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 contents of the data item referenced by identifier-11 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-10 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. If an ON OVERFLOW phrase has been specified, the imperative statement included in the ON OVERFLOW phrase is executed. If the ON OVERFLOW phrase is not specified, control is transferred to the next executable sentence.

19. The evaluation of subscripting and indexing for the identifiers is as follows:

    a. Any subscripting or indexing associated with identifier-1, identifier-10, identifier-11 is evaluated only once, immediately before any data is transferred as the result of the execution of the UNSTRING statement.

    b. Any subscripting or indexing associated with identifier-2, identifier-3, identifier-4, identifier-5, identifier-6 is evaluated immediately before the transfer of data into the respective data item.

Withdrawn/Retirée

7.50 THE USE STATEMENT

7.50.1 FUNCTION

1. The USE statement specifies procedures for input-output label and error handling that are in addition to the standard procedures provided by the input-output control system. It is also used to specify Procedure Division statements that are executed just before a report group named in the Report Section of the Data Division is presented.

2. The USE statement also serves to identify the user items that are to be monitored by the associated debugging section.

3. The USE statement is also used to specify procedures to be executed when the execution of a statement results in a Data Base Exception Condition and to specify procedures that produce privacy keys for use by the DBCS.

7.50.2 GENERAL FORMAT

Format 1

```
USE AFTER STANDARD {EXCEPTION} {PROCEDURE} ON {file-name-1 [, file-name-2] . . .}
                   {ERROR    }               {INPUT                            }
                                             {OUTPUT                           }
                                             {I-O                              }
                                             {EXTEND                           } .
```

Format 2

```
USE {BEFORE} STANDARD [BEGINNING] [REEL]
    {AFTER }          [ENDING   ] [FILE]
                                  [UNIT]

LABEL PROCEDURE ON {file-name-1 [, file-name-2] . . .}
                   {INPUT                            }
                   {OUTPUT                           }
                   {I-O                              }
                   {EXTEND                           } .
```

Withdrawn/Retirée

USE

Format 3

```
USE BEFORE REPORTING identifier-1 .
```

Format 4

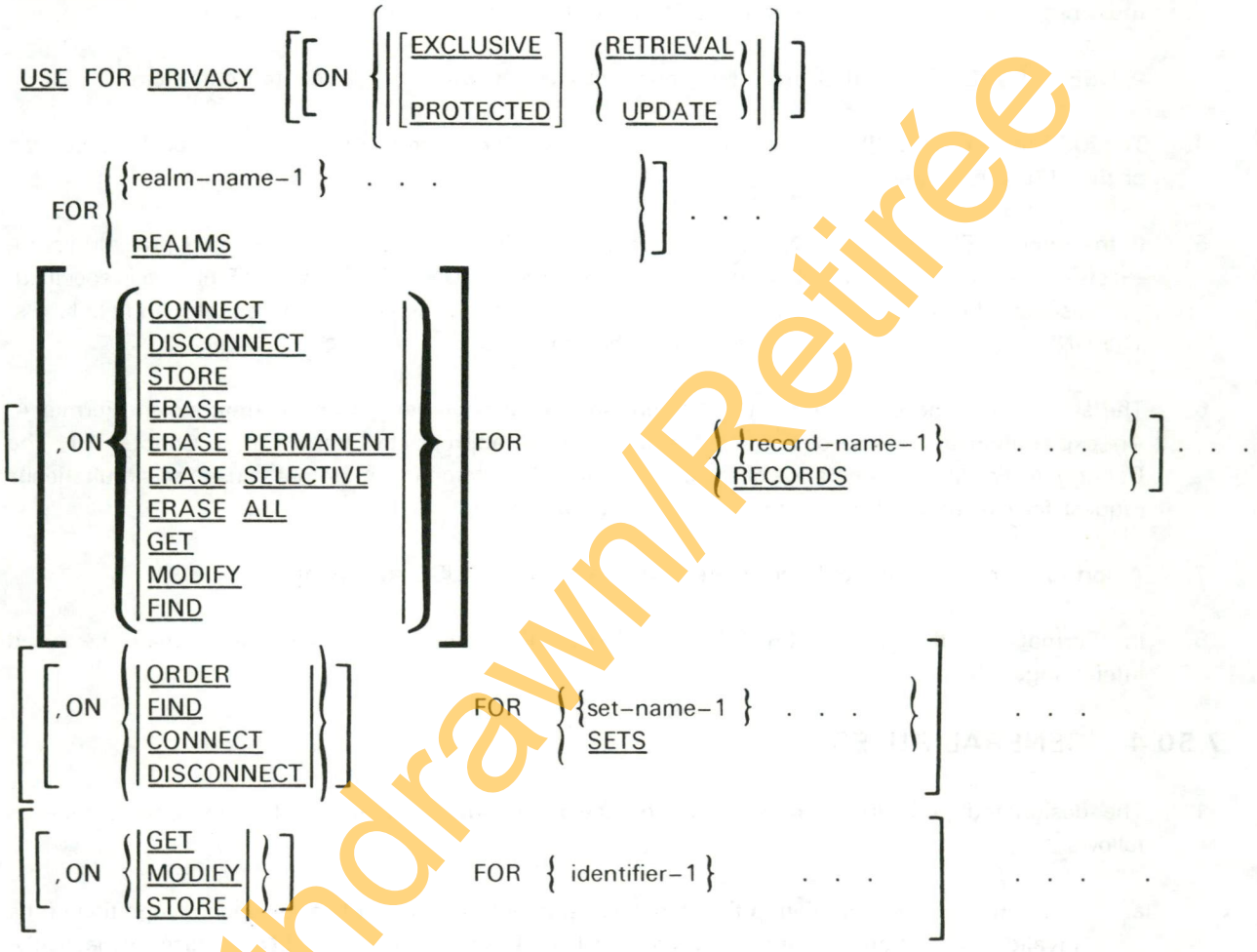
```
USE FOR DEBUGGING ON { cd-name-1                                          }
                     { [ALL REFERENCES OF] identifier-1 [WITH CONVERSION] }
                     { procedure-name-1                                   }
                     { file-name-1                                        }
                     { ALL PROCEDURES                                     }
                     [ cd-name-2                                          ]
                     [ [ALL REFERENCES OF] identifier-2 [WITH CONVERSION] ] . . .
                     [ procedure-name-2                                   ]
                     [ file-name-2                                        ]
                     [ ALL PROCEDURES                                     ]
```

Format 5

```
USE FOR DB-EXCEPTION
[ ON { OTHER                  } ]
     { {literal-1} . . .      }.
```

Withdrawn/Retirée

Format 6



7.50.3 SYNTAX RULES

1. A USE statement, when present, must immediately follow a section header in the Declaratives Section and must be followed by a period followed by a space. The remainder of the section must consist of zero, one or more procedural paragraphs that define the procedures to be used.

2. If the file-name phrase is used as a part of Format 2, the File Description entry for file-name-1, file-name-2, . . ., must not specify a LABEL RECORDS ARE OMITTED clause.

3. In Format 3, identifier-1 represents a report group. Identifier-1 must not appear in more than one Format 3 USE statement.

Withdrawn/Retirée

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.

A USE BEFORE REPORTING procedure must not alter the value of any control data item.

4. The USE statement itself is never executed; it merely defines the conditions calling for the execution of the USE procedures.

5. If the words BEGINNING or ENDING are not specified in Format 2, the designated procedures are executed for both beginning and ending labels. If neither the UNIT, REEL, nor FILE phrase is specified, the designated procedures are executed for both reel or unit, whichever is appropriate, and file labels. The UNIT phrase is not applicable to files in the random access mode.

6. The same file-name can appear in a different specific arrangement of any Format except Format 4. An explanation of the use of Format 4 is given in the appropriate Chapter. (See Chapter 11, The Debugging Facility.) Appearance of a file-name in a USE statement must not cause the simultaneous request for execution of more than one USE procedure.

7. A sort or merge file may only be referenced in a Format 4 USE statement.

8. In Format 1, the words ERROR and EXCEPTION are synonymous and may be used interchangeably.

7.50.4 GENERAL RULES

1. The designated procedures are executed by the input-output system at the appropriate time as follows:

   a. In Format 1, after completing the standard input-output error routine, or upon recognition of the invalid key or at end conditions, when the INVALID KEY phrase or AT END phrase, respectively, has not been specified in the input-output statement.

   b. In Format 2, before or after beginning or ending input label checking is executed.

      Before a beginning or ending output label is created.

      After a beginning or ending output label is created, but before it is written.

      Before or after beginning or ending input-output label checking is executed.

2. In Format 2, if the USE statement specifies a phrase other than the file-name-1 phrase, references in that USE procedure to common label items need not be qualified by a file-name. A common label item is an elementary data item that appears in every label record defined in the program, but at the

Withdrawn/Retirée

same time, does not appear in any data record defined in this program. Furthermore, a common label item must have the same name, description, and relative position in every label record.

The USE procedures do not apply to files that are described with the LABEL RECORDS ARE OMITTED clause.

3. In Format 3, the designated procedures are executed by the Report Writer Control System just before the named report group is produced. (See 6.46, The TYPE Clause.)

4. After execution of a Format 1 USE procedure, control is returned to the invoking routine.

5. Within a USE procedure, there must not be any reference to any nondeclarative procedures. Conversely, in the nondeclarative portion there must be no reference to procedure-names that appear in the declarative portion, except that PERFORM statements may refer to a USE statement having Formats 1, 2, 3 or to the procedures associated with such a USE statement.

6. An explanation of the USE FOR DEBUGGING statement is given in the appropriate Chapter. (See Chapter 11, The Debugging Facility.)

7. An explanation of the USE FOR DB - EXCEPTION statement is given in the appropriate Chapter. (See Chapter 12, The COBOL Data Base Facility - Data Manipulation.)

8. An explanation of the USE FOR PRIVACY statement is given in the appropriate Chapter. (See Chapter 12, The COBOL Data Base Facility - Data Manipulation.)

Withdrawn/Retiree

7.51 THE WRITE STATEMENT

7.51.1 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.

7.51.2 GENERAL FORMAT

Format 1

```
WRITE record-name  [FROM identifier-1]
[ { BEFORE } ADVANCING { { { identifier-2 } [ LINE ] } }
  { AFTER  }           { { { integer    } [ LINES ] } }
                       { { { mnemonic-name } } }
                       { { { PAGE          } } }
]
[ ; AT { END-OF-PAGE } imperative-statement ]
       { EOP        }
```

Format 2

```
WRITE record-name  [FROM identifier-1]  [; INVALID KEY imperative-statement]
```

7.51.3 SYNTAX RULES

1. Format 1 must be used for sequential files.
2. Format 2 must be used for mass storage files with other than sequential organization.
3. A WRITE statement must not reference a Report Group Description entry.
4. The record-name is the name of a logical record in the File Section of the Data Division and may be qualified.

Withdrawn/Retirée

5. Record-name must not be defined within a Sort-Merge File Description entry.

6. The ADVANCING mnemonic-name phrase cannot be specified when writing a record to a file whose File Description entry contains the LINAGE clause.

7. When identifier-2 is used in the ADVANCING phrase, it must be the name of an unsigned integer data item.

8. Integer or the value of the data item referenced by identifier-2 may be zero.

9. When mnemonic-name is specified, the name is associated with a particular feature specified by the implementor. The mnemonic-name is defined in the SPECIAL-NAMES paragraph of the Environment Division.

10. If the END-OF-PAGE phrase is specified, the LINAGE clause must be specified in the File Description entry for the associated file.

11. The words END-OF-PAGE and EOP are equivalent.

12. When Format 2 is used, the INVALID KEY phrase must be specified if an applicable Format 1 USE procedure is not specified for the associated file.

7.51.4 GENERAL RULES

ALL FILES

1. The explanation of the FROM phrase is stated under the appropriate paragraph. (See 7.7.12, The FROM Option.)

2. The associated file must be open in the OUTPUT, I-O, or EXTEND mode at the time of the execution of this statement. (See 7.30, The OPEN Statement.)

3. The logical record released by the execution of the WRITE statement is no longer available in the record area unless the associated file is named in a SAME RECORD AREA clause or the execution of the WRITE statement is unsuccessful due to an invalid key condition or a boundary violation. The logical record is also available to the program as a record of other files referenced in the same SAME RECORD AREA clause as the associated output file, as well as to the file associated with record-name.

Withdrawn/Retirée

4. For mass storage files, the Next Record Pointer is unaffected by the execution of a WRITE statement.

5. The execution of the WRITE statement causes the value of the FILE STATUS data item, if any, associated with the file to be updated. (See 7.7.9, I-O Status.)

6. The maximum record size for a file is established at the time the file is created and must not subsequently be changed.

7. 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.

8. The execution of the WRITE statement releases a logical record to the operating system.

SEQUENTIAL FILES

9. 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 by the implementor to act as if the user had specified AFTER ADVANCING 1 LINE. If the ADVANCING phrase is used, advancing is provided as follows:

   a. If identifier-2 is specified, the representation of the printed page is advanced the number of lines equal to the current value associated with identifier-2.

   b. If integer is specified, the representation of the printed page is advanced the number of lines equal to the value of integer.

   c. If mnemonic-name is specified, the representation of the printed page is advanced according to the rules specified by the implementor for that hardware device.

   d. If the BEFORE phrase is used, the line is presented before the representation of the printed page is advanced according to Rules a, b, and c above.

   e. If the AFTER phrase is used, the line is presented after the representation of the printed page is advanced according to Rules a, b, and c above.

   f. If PAGE is specified, 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. If the record to be written is associated with a file whose File Description entry contains a LINAGE clause, the repositioning is to the first line that can be written on the next logical page as specified in the LINAGE clause. If the record to be written is associated with a file whose File Description entry does not contain a LINAGE clause, the repositioning to the next logical page is accomplished in accordance with an implementor-defined technique. If page has no meaning in conjunction with a specific device, then advancing will be provided by the implementor to act as if the user had specified BEFORE or AFTER (depending on the phrase used) ADVANCING 1 LINE.

Withdrawn/Retired

10. If the logical end of the representation of the printed page is reached during the execution of a Format 1 WRITE statement with the END-OF-PAGE phrase, the imperative-statement specified in the END-OF-PAGE phrase is executed. The logical end is specified in the LINAGE clause associated with record-name.

11. An end-of-page condition is reached whenever 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 the imperative-statement in the END-OF-PAGE phrase is executed.

    An automatic page overflow condition is reached whenever the execution of a given WRITE statement (with or without an END-OF-PAGE phrase) cannot be fully accommodated within the current page body.

    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. The imperative-statement in the END-OF-PAGE clause, if specified, is executed after the record is written and the device has been repositioned.

    If integer-2 or data-name-2 of the LINAGE clause is not specified, no end-of-page condition distinct from the page overflow condition is detected. In this case, the end-of-page condition and page overflow condition occur simultaneously.

    If integer-2 or data-name-2 of the LINAGE CLAUSE is specified, but 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 and integer-1 or the data item referenced by data-name-1, then the operation proceeds as if integer-2 or data-name-2 had not been specified.

12. 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 action takes place:

    a. The value of the FILE STATUS data item, if any, of the associated file is set to a value indicating a boundary violation. (See 7.7.9, I-O Status.)

    b. If a USE AFTER STANDARD EXCEPTION declarative is explicitly or implicitly specified for the file, that declarative procedure will then be executed.

    c. If a USE AFTER STANDARD EXCEPTION declarative is not explicitly or implicitly specified for the file, the result is undefined.

Withdrawn/Retired

WRITE

RELATIVE FILES

13. 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 will cause a record to be released to the MSCS. The first record will have a relative record number of one (1), and subsequent records released will have relative record numbers of 2, 3, 4, ... If the RELATIVE KEY data item has been specified in the SELECT statement for the associated file, the relative record number of the record just released will be placed into the RELATIVE KEY data item by the MSCS during execution of the WRITE 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 MSCS by execution of the WRITE statement.

14. 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 a Format 2 WRITE statement then causes the contents of the record area to be released to the MSCS.

15. The invalid key condition exists under the following circumstances:

    a. When the access mode is random or dynamic, and the RELATIVE KEY data item specifies a record which already exists in the file, or

    b. When an attempt is made to write beyond the externally-defined boundaries of the file.

16. When the invalid key condition is recognized, the execution of the WRITE statement is unsuccessful, the contents of the record area are unaffected, and the FILE STATUS data item, if any, of the associated file 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 7.7.9, I-O Status; and 7.7.10, The Invalid Key Condition.)

INDEXED FILES

17. Execution of a Format 2 WRITE statement causes the contents of the record area to be released. The MSCS utilizes the content 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.

18. The value of the Prime Record Key must be unique within the records in the file.

19. 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.

Withdrawn/Retired

20. If sequential access mode is specified for the file, records must be released to the MSCS in ascending order of Prime Record Key values.

21. If random or dynamic access mode is specified, records may be released to the MSCS in any program-specified order.

22. When the ALTERNATE RECORD KEY clause is specified in the File Description entry for an indexed file, the value of the Alternate Record Key may be non-unique only if the DUPLICATES phrase is specified for that data item. In this case the MSCS 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 MSCS.

23. The invalid key condition exists under the following circumstances:

    a. When sequential access mode is specified for a file opened in the output 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 the file is opened in the output or I-O mode, and the value of the Prime Record Key is equal to the value of a Prime Record Key of a record already existing in the file, or

    c. When the file is opened in the output or I-O mode, and the value of an Alternate Record Key for which duplicates are not allowed equals 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.

24. When the invalid key condition is recognized, the execution of the WRITE statement is unsuccessful, the contents of the record area are unaffected and the FILE STATUS data item, if any, of the associated file is set to a value indicating the cause of the condition. Execution of the program proceeds according to the rules stated in 7.7.10, The Invalid Key Condition. (See 7.7.9, I-O Status; and 7.7.10, The Invalid Key Condition.)

Withdrawn/Retiree

CHAPTER 8

SEGMENTATION

8.1 GENERAL DESCRIPTION

COBOL segmentation is a facility that provides a means by which the user may communicate with the compiler to specify object program overlay requirements.

8.1.1 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.

8.1.2 ORGANIZATION

8.1.2.1 Program Segments

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 non-overlayable or the 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.

8.1.2.2 Non-Overlayable Portion

The non-overlayable portion is defined as that portion of the program which must not be overlaid once it has been referenced explicitly or implicitly.

8.1.2.3 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 non-overlayable or overlayable segment.

8.1.2.4 Segment State

A segment is in its initial state when control is transferred to it either for the first time within a run unit or after execution of a CANCEL statement affecting that segment. The segment is in its last used state for all other transfers of control.

Withdrawn/Retirée

8.1.3 SEGMENT CLASSIFICATION

Sections which are to be segmented are classified, using a system of segment-numbers and the following criteria:

1. 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.

2. Relationship to Other Sections — Sections which frequently communicate with one another should be given the same segment-numbers.

8.1.4 SEGMENTATION CONTROL

The logical sequence of the program is the same as the physical sequence except for specific transfers of control. If any reordering of the object program is required to handle the flow from segment to segment, according to the rules in 8.2.1, the implementor must provide control transfers to maintain the logical flow specified in the source program. The implementor must also provide all controls necessary for a segment to operate whenever the segment is used.

Withdrawn/Retirée

8.2 STRUCTURE OF PROGRAM SEGMENTS

8.2.1 SEGMENT-NUMBERS

Section classification is accomplished by means of a system of segment-numbers. The segment-number is included in the section header.

8.2.1.1 General Format

section-name SECTION [segment-number] .

8.2.1.2 Syntax Rules

1. The segment-number must be an integer ranging in value from 00 through 49.

2. If the segment-number is omitted from the section header, the segment-number is assumed to be 00.

8.2.1.3 General Rules

1. All sections 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 non-overlayable 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.

8.2.2 SEGMENT LIMIT

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 non-overlayable or overlayable segments. The logical properties of these segments remain unchanged.

8.2.2.1 General Format

The SEGMENT-LIMIT clause appears in the OBJECT-COMPUTER paragraph and has the following format:

[, SEGMENT–LIMIT IS segment–number]

8.2.2.2 Syntax Rules

1. Segment-number must be an integer ranging in value from 00 through 49.

Withdrawn/Retirée

8.2.2.3 General Rules

1. When the SEGMENT-LIMIT clause is specified, only those segments having segment-numbers less than the Segment Limit are the non-overlayable portion of the 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 non-overlayable.

Withdrawn/Retirée

CHAPTER 9

THE COBOL LIBRARY FACILITY

9.1 INTRODUCTION

COBOL libraries contain library texts that are available to the compiler for copying at compile time. The effect of the interpretation of the COPY statement is to insert text into the source program, where it will be treated by the compiler as part of the source program.

Withdrawn/Retired

9.2 THE COPY STATEMENT

9.2.1 FUNCTION

The COPY statement incorporates text into a COBOL source program.

9.2.2 GENERAL FORMAT

```
COPY  text-name  [ { OF } library-name ]
                 { IN }

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

9.2.3 SYNTAX RULES

1. If more than one COBOL library is available during compilation, text-name must be qualified by the library-name identifying the COBOL library in which the text associated with text-name resides.

   Within one COBOL library, each text-name must be unique.

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

3. Pseudo-text-1 must not be null, nor may it consist solely of the character space(s), nor may it consist solely of comment lines.

4. Pseudo-text-2 may be null.

5. Character-strings within pseudo-text-1 and pseudo-text-2 may be continued. (See 10.2.2, Continuation of Lines.)

6. Word-1 or word-2 may be any single COBOL word.

7. A COPY statement may occur in the source program anywhere a character-string or a separator may occur except that a COPY statement must not occur within a COPY statement.

9.2.4 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 resultant source program.

Withdrawn/Retirée

2. The effect of processing a COPY statement is that the library text associated with text-name 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.

3. 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, identifier-1, word-1, and literal-1 in the library text is replaced by the corresponding pseudo-text-2, identifier-2, word-2, or literal-2.

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

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

   Any separator comma, semicolon and/or space(s) preceding the leftmost library text-word is copied into the source program. Starting with the leftmost library text-word and the first pseudo-text-1, identifier-1, word-1, or literal-1 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, identifier-1, word-1, or literal-1 match the library text if, and only if, the ordered sequence of text-words that forms pseudo-text-1, identifier-1, word-1, or literal-1 is equal, character for character, to the ordered sequence of library text-words. For purposes of matching, each occurrence of a separator comma or semicolon in pseudo-text-1 or in the library text is considered to be a single space except when pseudo-text-1 consists solely of either a separator comma or semicolon, in which case it participates in the match as a text-word. 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, identifier-1, word-1, or literal-1, 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, identifier-1, word-1, or literal-1 specified in the REPLACING phrase.

   Whenever a match occurs between pseudo-text-1, identifier-1, word-1, or literal-1 and the library text, the corresponding pseudo-text-2, identifier-2, word-2, or literal-2 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, identifier-1, word-1, or literal-1 specified in the REPLACING phrase.

Withdrawn/Retiree

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.

6. A comment line occurring in the library text and pseudo-text-1 is interpreted, for purposes of matching, as a single space. Comment lines appearing in pseudo-text-2 and library text are copied into the source program unchanged.

7. Debugging lines are permitted within library text and pseudo-text-2. Debugging lines are not permitted within pseudo-text-1; text-words within a debugging line participate in the matching rules as if the 'D' did not appear in the Indicator Area. If a COPY statement is specified on a debugging line, then the text that is the result of the processing of the COPY statement will appear as though it were specified on debugging lines with the following exception: comment lines in library text will appear as comment lines in the resultant source program.

8. The text produced as a result of the complete processing of a COPY statement must not contain a COPY statement.

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

10. Each word copied from the library but not replaced is copied into the same area of the resultant program as it is in the library. Each word in pseudo-text-2 that is to be placed in the resultant program is placed in the same area of the resultant program as it appears in pseudo-text-2.

    Library text must conform to the rules for COBOL Reference Format.

11. For purposes of compilation, text-words after replacement are placed in the source program according to the rules for Reference Format. (See Chapter 10, Reference Format.)

Withdrawn/Retirée

CHAPTER 10

REFERENCE FORMAT

10.1 GENERAL DESCRIPTION

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

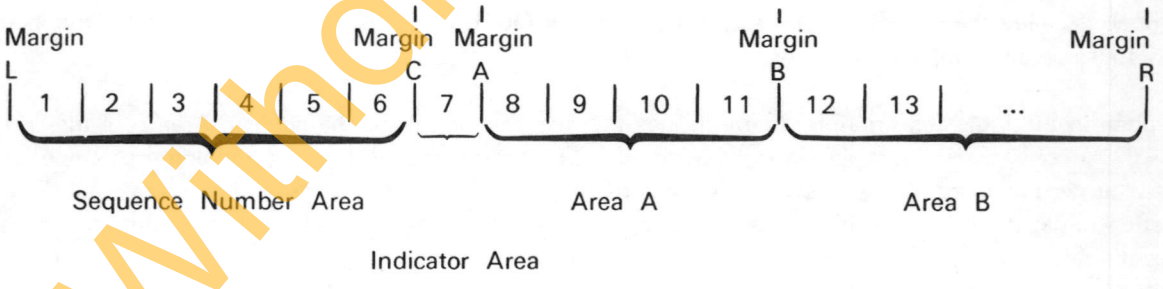
The rules for spacing given in the discussion of the reference format take precedence over all other rules for spacing.

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.

The divisions of a COBOL Sub-schema source program must be ordered as follows: the Title Division, then the Mapping Division, then the Structure Division. Each division must be written according to the rules for the reference format.

10.2 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.

Withdrawn/Retiree

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 specified by the implementor; it begins immediately to the right of Margin B and terminates immediately to the left of Margin R.

10.2.1 SEQUENCE NUMBERS

The sequence number area may be used to label a source program line. The contents of the sequence number area are user-defined and may consist of any character in the computer's character set. There is no requirement that the contents of the sequence number area appear in any particular sequence or be unique.

10.2.2 CONTINUATION OF LINES

Whenever a sentence, entry, phrase, or clause requires more than one line, it 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 nonblank character of the preceding line, excluding intervening comment 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.

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 '==' or 'B''' must be on the same line.

Withdrawn/Retiree

10.2.3 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, except immediately preceding a continuation line. (See 10.2.2, Continuation of Lines.)

Withdrawn/Retirée

10.3 DIVISION, SECTION, PARAGRAPH FORMATS

10.3.1 DIVISION HEADER

The division header must start in Area A.

10.3.2 SECTION HEADER

The section header must start in Area A.

A section consists of paragraphs in the Environment and Procedure Divisions and Data Division entries in the Data Division.

10.3.3 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. Comment entries may be included within a paragraph. (See 10.6, Comment Lines.)

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 10.2.2, Continuation of Lines.)

Withdrawn/Retirée

10.4 DATA DIVISION AND SUB-SCHEMA ENTRIES

Each Data Division entry and each Sub-schema entry begins with a level indicator or a level-number, followed by a space, followed by the name of a data item (except in the Report Section of the Data Division), followed by a sequence of independent clauses describing the item. Each clause, except the last clause of an entry, may be terminated by a separator semicolon. 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: DB, FD, SD, RD, SA, CD.

In the Sub-schema, a level indicator is any of the following: SS, AD, RD, SD.

In those entries that begin with a level indicator, the level indicator begins in Area A, followed by a separator space, then followed, in Area B, with its associated data-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, 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, the level-number begins in Area A followed by a space and followed in Area B with its associated record-name or item-name and appropriate descriptive information.

Successive Data Description entries may have the same format as the first or may be indented according to level-number. The entries in the output listing need be indented only if the input is indented. Indentation does not affect the magnitude of a level-number.

When level-numbers are to be indented, each new level-number may begin in any number of spaces to the right of Margin A. The extent of indentation to the right is determined only by the width of the physical medium.

10.5 DECLARATIVES

The key word DECLARATIVES and the 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.

Withdrawn/Retirée

10.6 COMMENT LINES

A comment line is any line with an asterisk in the Indicator Area of the line. A comment line can appear as any line in a source program after the Identification Division header. 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 and the characters in Area A and Area B will be produced on the listing but serve as documentation only. A special form of comment line represented by a stroke in the Indicator Area of the line causes page ejection prior to printing the comment.

Successive comment lines are allowed.

Withdrawn/Retired

CHAPTER 11

THE DEBUGGING FACILITY

11.1 GENERAL DESCRIPTION

COBOL debugging is a facility that provides a means by which the user can describe his debugging algorithm — the conditions under which data or procedure items 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 debugging facility simply provides a convenient access to pertinent information.

11.2 CONCEPTS

The features of the COBOL language that support the debugging facility are:

a. A compile-time switch — WITH DEBUGGING MODE.

b. An object-time switch.

c. A USE FOR DEBUGGING statement.

d. Special registers — DEBUG-ITEM and DEBUG-SUB-ITEM.

e. Debugging lines.

11.3 A 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 statements written in the program.

When the WITH DEBUGGING MODE clause is specified in a program, all debugging sections and all debugging lines are compiled as specified in this Chapter. When the WITH DEBUGGING MODE clause is not specified, all debugging lines and all debugging sections are compiled as if they were comment lines.

Withdrawn/Retired

11.4 AN OBJECT-TIME SWITCH

An object-time switch dynamically activates the code inserted by the compiler for debugging purposes; namely, for the establishment of DEBUG-ITEM and the execution of the debugging section, but not the execution of debugging lines. 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 defined for the 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.

Withdrawn/Retired

11.5 THE USE FOR DEBUGGING STATEMENT

11.5.1 FUNCTION

The USE FOR DEBUGGING statement identifies the user items that are to be monitored by the associated debugging section.

11.5.2 GENERAL FORMAT

```
section-name    SECTION    [segment-number].

USE FOR DEBUGGING ON  { cd-name-1                                          }
                      { [ALL REFERENCES OF]  identifier-1 [WITH CONVERSION] }
                      { procedure-name-1                                   }
                      { file-name-1                                        }
                      { ALL PROCEDURES                                     }

                    [ cd-name-2                                          ]
                    [ [ALL REFERENCES OF]  identifier-2 [WITH CONVERSION] ]
                    [ procedure-name-2                                   ]
                    [ file-name-2                                        ]
                    [ ALL PROCEDURES                                     ]
```

11.5.3 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, cd-name, file-name, or procedure-name may appear in only one USE FOR DEBUGGING statement and may appear only once in that statement.

Withdrawn/Retirée

7. The ALL PROCEDURES phrase may appear only once in a program.

8. When the ALL PROCEDURES phrase is specified, procedure-name-1, procedure-name-2, . . ., must not be specified in any USE FOR DEBUGGING statement.

9. Identifier-1, identifier-2, . . ., must not reference any data item defined in the Report Section except sum counters.

10. If the Data Description entry of the data item referenced by identifier-1, identifier-2, . . ., contains an OCCURS clause or is subordinate to a Data Description entry that contains an OCCURS clause, identifier-1, identifier-2, . . ., must be specified without the subscripting or indexing normally required.

11. Identifier-1, identifier-2, . . ., must not be reference modified.

12. References to the special registers DEBUG-ITEM or DEBUG-SUB-ITEM are restricted to references from within a debugging section.

13. Identifier-1, identifier-2, . . ., must reference an elementary numeric data item if the WITH CONVERSION phrase is specified.

11.5.4 GENERAL RULES

1. In the following General Rules all references to cd-name-1, identifier-1, procedure-name-1, and file-name-1 apply equally to cd-name-2, identifier-2, procedure-name-2, and file-name-2, respectively.

2. A reference to file-name-1, identifier-1, procedure-name-1 or cd-name-1 as a qualifier does not constitute 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, indexing or reference modifiers constitutes an explicit reference to identifier-1 and causes the associated debugging section to be executed in accordance with General Rules 9 and 10.

3. Where 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 OPEN, CLOSE, 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. When file-name-1 is specified in a USE FOR DEBUGGING statement, that debugging section is executed at each of the following times:

Withdrawn/Retirée

a. Immediately after the execution of any CLOSE, DELETE FILE 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.

7. 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.

8. The ALL PROCEDURES phrase causes the effects described in General Rule 6 to occur for every procedure-name in the program, except those appearing within a debugging section.

9. 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 contents of the data item referenced by identifier-1.

d. In the case of a conditional statement, immediately after the determination of the truth value and before the execution either of any associated imperative statement or of any other statement.

e. In the case of a SEARCH statement:

(1) If the end of the search has occurred, immediately before execution of the imperative statement associated with the AT END phrase, if specified.

(2) Immediately after the SEARCH statement if the AT END phrase is not specified.

Withdrawn/Retirée

(3) Immediately after evaluation of the condition in the last evaluated WHEN phrase.

The debugging sections are executed once for each iteration of the search loop. If an identifier appears in a WHEN phrase that is not evaluated in an iteration, the associated debugging section is not executed for that identifier during that iteration.

f. In the case of any other COBOL statement, immediately after execution of that statement.

If identifier-1 is specified in a phrase that is not executed or evaluated, the associated debugging section is not executed.

10. 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, AFTER or UNTIL phrase references identifier-1, immediately after each initialization, modification or evaluation of the contents of the data item referenced by identifier-1.

c. In the case of a conditional statement, immediately after the determination of the truth value and before the execution of either any associated imperative statement or any other statement.

d. In the case of a FORMAT 1 SEARCH statement in which a VARYING phrase references identifier-1:

(1) If the end of the search has occurred, immediately before the execution of the imperative statement associated with the AT END phrase, if specified.

(2) Immediately after the SEARCH statement if the AT END phrase is not specified.

In the case of a Format 1 SEARCH statement in which a VARYING phrase is not specified, immediately after evaluation of the condition in the last evaluated WHEN phrase.

The debugging section is executed once for each iteration of the search loop.

e. 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 that is not executed or evaluated, or if identifier-1 is specified only in the FROM or BY phrase of a PERFORM statement, the associated debugging section is not executed.

Withdrawn/Retiree

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. 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.

    Within an imperative statement, each individual occurrence of an imperative verb identifies a separate statement for the purpose of debugging.

12. When cd-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 ENABLE, DISABLE, and SEND statement that references cd-name-1.

    b. Immediately after the execution of a RECEIVE statement referencing cd-name-1 that does not result in the execution of the NO DATA imperative-statement.

    c. Immediately after the execution of an ACCEPT statement that references cd-name-1.

13. Associated with each execution of a debugging section are two special registers. DEBUG-ITEM provides information about the conditions that caused the execution of a debugging section. DEBUG-SUB-ITEM provides additional information when the data item that causes the debugging section to be executed is subscripted or indexed; otherwise, its contents are undefined. The implicit descriptions of DEBUG-ITEM and DEBUG-SUB-ITEM are:

Withdrawn/Retiree

```
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-NUM                       PICTURE IS 9(2).
    02  FILLER                              PICTURE IS X VALUE SPACE.
    02  DEBUG-SIZE                          PICTURE IS 9(3).
    02  FILLER                              PICTURE IS X VALUE SPACE.
    02  DEBUG-CONTENTS                      PICTURE IS X(n).
    02  DEBUG-NUMERIC-CONTENTS REDEFINES DEBUG-CONTENTS
                                            PICTURE IS S9(n)V9(n) SIGN IS
                                              LEADING SEPARATE CHARACTER.
    02  REFERENCE-MODIFIER.
        03  DEBUG-START                     PICTURE IS S9999 SIGN IS
                                              LEADING SEPARATE CHARACTER.
        03  FILLER                          PICTURE IS X VALUE SPACE.
        03  DEBUG-LENGTH                    PICTURE IS S9999 SIGN IS
                                              LEADING SEPARATE CHARACTER.

01  DEBUG-SUB-ITEM.
    02  DEBUG-SUB OCCURS 0 TO 48 TIMES DEPENDING ON
          DEBUG-SUB-NUM.
        03  DEBUG-SUB-N                     PICTURE IS S99999
                                              SIGN IS LEADING
                                              SEPARATE CHARACTER.
        03  FILLER                          PICTURE IS X VALUE SPACE.
```

14. Prior to each execution of a debugging section, the contents of the data item referenced by DEBUG-ITEM are space-filled. The contents of data items subordinate to DEBUG-ITEM and DEBUG-SUB-ITEM are then updated, according to the following General Rules, immediately before control is passed to that debugging section. The contents 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 WITH CONVERSION phrase is not specified, when the move is treated exactly as if it were an alphanumeric to alphanumeric elementary move with no conversion of data from one form of internal representation to another.

15. The contents of DEBUG-LINE is the implementor-defined means of identifying a particular source statement.

16. DEBUG-NAME contains the first 30 characters of the name that caused the debugging section to be executed.

Withdrawn/Retiree

All qualifiers of the name are separated in DEBUG-NAME by the words 'IN' or 'OF'. Subscripts/indices, if any, are not entered into DEBUG-NAME.

17. If the reference to a data item that causes the debugging section to be executed is subscripted or indexed, the contents of DEBUG-SUB-NUM are set equal to the total number of subscripts or indices in that reference. The value of the first subscript or index is entered into DEBUG-SUB-N(1), the second into DEBUG-SUB-N(2),..., until all subscripts or indices have been accommodated. If the reference to a data item that causes the debugging section to be executed is not subscripted or indexed, the contents of DEBUG-SUB-NUM are set to zero and the contents of DEBUG-SUB-N are undefined.

18. DEBUG-SIZE contains a count of the number of characters currently in DEBUG-CONTENTS.

19. If a reference to a data-name which is reference modified causes the debugging section to be executed, the results of the arithmetic expressions which determine leftmost-character-position and length are entered into DEBUG-START and DEBUG-LENGTH, respectively.

20. DEBUG-CONTENTS is a data item that is large enough to contain the data required by the following General Rules.

21. 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 procedure-name of the first nondeclarative procedure.

    b. DEBUG-NAME contains the name of that procedure.

    c. DEBUG-SUB-NUM contains zero.

    d. DEBUG-CONTENTS contains 'START PROGRAM'.

22. 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.

    c. DEBUG-SUB-NUM contains zero.

23. 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.

Withdrawn/Retired

b. DEBUG-NAME contains procedure-name-1.

c. DEBUG-SUB-NUM contains zero.

d. DEBUG-CONTENTS contains:

(1) If the reference to procedure-name-1 is in the INPUT phrase of a SORT statement, 'SORT INPUT'.

(2) If the reference to procedure-name-1 is in the OUTPUT phrase of a SORT statement, 'SORT OUTPUT'.

(3) If the reference to procedure-name-1 is in the OUTPUT phrase of a MERGE statement, 'MERGE OUTPUT'.

24. If the transfer of control from the control mechanism associated with a PERFORM statement causes 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-SUB-NUM contains zero.

d. DEBUG-CONTENTS contains 'PERFORM LOOP'.

25. 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-SUB-NUM contains zero.

d. DEBUG-CONTENTS contains 'USE PROCEDURE'.

26. 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.

Withdrawn/Retiree

c. DEBUG-SUB-NUM contains zero.

d. DEBUG-CONTENTS contains 'FALL THROUGH'.

27. If reference to file-name-1, cd-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, cd-name-1.

b. DEBUG-NAME contains the name of file-name-1, cd-name-1.

c. DEBUG-SUB-NUM contains zero.

d. For READ or RETURN, DEBUG-CONTENTS contains the entire record read.

e. For all other references to file-name-1, DEBUG-CONTENTS contains spaces.

f. For any reference to cd-name, DEBUG-CONTENTS contains the contents of the area associated with the cd-name.

28. 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. DEBUG-SUB-NUM contains zero if identifier-1 is not subscripted (or indexed) or the number of subscripts and/or indices if identifier-1 is subscripted.

d. If the Data Description entry of the data item referenced by identifier-1, identifier-2, . . ., contains an OCCURS clause or is subordinate to a Data Description entry that contains an OCCURS clause, DEBUG-CONTENTS contains only the contents of the explicit table element which caused the USE FOR DEBUGGING statement to be executed. If identifier has been reference modified, DEBUG-CONTENTS contains only the portion of identifier which is participating in the operation which caused the USE FOR DEBUGGING statement to be executed.

e. If the WITH CONVERSION phrase is specified, DEBUG-NUMERIC-CONTENTS contains the contents of the data item referenced by identifier-1 at the time that control passes to the debugging section. (See General Rules 8 and 9.)

f. If the WITH CONVERSION phrase is not specified, DEBUG-CONTENTS contains the contents of the data item referenced by identifier-1 at the time that control passes to the debugging section. (See General Rules 8 and 9.)

Withdrawn/Retirée

11.6 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 contents 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.

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 program after the OBJECT-COMPUTER paragraph.

Withdrawn/Retirée

CHAPTER 12

THE COBOL DATA BASE FACILITY — DATA MANIPULATION

12.1 GENERAL DESCRIPTION

COBOL data manipulation of a data base is a facility that provides a means by which the user may access a data base that has been described by a Schema. This facility includes special registers, status indicators, conditions, record selection expressions, record monitoring, currency indicators, and procedural statements.

12.2 ORGANIZATION

The facilities described in this chapter are a part of the Procedure Division of a COBOL program. Specification of these facilities in a COBOL program must conform to the rules governing the Procedure Division. (See Chapter 7, The Procedure Division.) Description in this chapter of a facility which also appears in Chapter 7 is limited to the additional capabilities provided by the presence of a data base.

Withdrawn/Retiree

12.3 DATA BASE SPECIAL REGISTERS

12.3.1 GENERAL DESCRIPTION

Data Base Special Registers are compiler-generated storage areas whose primary use is to store information for use by the program and the Data Base Control System (DBCS) when using the Data Base Facility.

12.3.2 DB-STATUS

The reserved word DB-STATUS is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-STATUS is allocated per run unit. The implicit description is that of an alphanumeric data item capable of containing seven characters. The value in DB-STATUS may be accessed by Procedure Division statements; however, only the DBCS may change the value in DB-STATUS. The DBCS changes the value in DB-STATUS by moving an appropriate status indicator to DB-STATUS during the execution of any statement that references the data base. (See 12.4, Data Base Status Indicators.)

12.3.3 DB-REALM-NAME

The reserved word DB-REALM-NAME is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-REALM-NAME is allocated per run unit. The implicit description of DB-REALM-NAME is that of an alphanumeric data item capable of containing 30 characters.

The value in DB-REALM-NAME may be accessed by Procedure Division statements; however, only the DBCS may change the value in DB-REALM-NAME. The DBCS changes the value in DB-REALM-NAME by moving the name of an appropriate realm to DB-REALM-NAME after the execution of certain statements that reference the data base. (See 12.4, Data Base Status Indicators.)

12.3.4 DB-SET-NAME

The reserved word DB-SET-NAME is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-SET-NAME is allocated per run unit. The implicit description of DB-SET-NAME is that of an alphanumeric data item capable of containing 30 characters.

The value in DB-SET-NAME may be accessed by Procedure Division statements; however, only the DBCS may change the value in DB-SET-NAME. The DBCS changes the value in DB-SET-NAME by moving the name of a set type, if applicable, to DB-SET-NAME after the unsuccessful execution of statements that reference the data base. (See 12.4, Data Base Status Indicators.)

Withdrawn/Retiree

12.3.5 DB-RECORD-NAME

The reserved word DB-RECORD-NAME is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-RECORD-NAME is allocated per run unit. The implicit description of DB-RECORD-NAME is that of an alphanumeric data item capable of containing 30 characters.

The value in DB-RECORD-NAME may be accessed by Procedure Division statements; however, only the DBCS may change the value in DB-RECORD-NAME. The DBCS changes the value in DB-RECORD-NAME by moving the name of the current record type to DB-RECORD-NAME after the execution of certain statements that reference the data base. (See 12.4, Data Base Status Indicators.)

12.3.6 DB-CONFLICT

The reserved word DB-CONFLICT is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-CONFLICT is allocated per run unit. The implicit description is that of an unsigned integer capable of handling a range of values from 01 to 15. The implicit usage for DB-CONFLICT is display.

The value in DB-CONFLICT may be accessed by Procedure Division statements; however, only the DBCS may change the value in DB-CONFLICT. The DBCS changes the value in DB-CONFLICT by moving an appropriate code to DB-CONFLICT when the current run unit attempts to alter a record or a record relationship that has been altered by a concurrent run unit while that record is in a monitored mode or extended monitored mode for the current run unit. (See 12.4, Data Base Status Indicators.)

12.3.7 DB-PRIVACY-KEY

The reserved word DB-PRIVACY-KEY is the name of a special register generated automatically by the implementor's code that supports the Data Base Facility. Only one DB-PRIVACY-KEY will be allocated per run unit. The implicit description of DB-PRIVACY-KEY is that of an alphanumeric data item capable of containing at least 10 characters. The value in DB-PRIVACY-KEY must be set by the COBOL object program to establish run unit authority to execute statements that may reference the data base. The value in DB-PRIVACY-KEY may be changed only in a USE FOR PRIVACY declarative procedure, but may be referenced in any portion of the program.

Withdrawn/Retirée

12.4 DATA BASE STATUS INDICATORS

12.4.1 GENERAL DESCRIPTION

The execution of any data base manipulation statement causes the DBCS to insert a value into the special register DB-STATUS. This value, known as a Data Base Status Indicator, is made up of a Statement Code and a Status Code. If the execution of any data base manipulation statement results in a Data Base Exception Condition, the Statement Code indicates which data base manipulation statement caused the Data Base Exception Condition, and the Status Code indicates which type of Data Base Exception Condition has occurred. If a Data Base Exception Condition does not occur, both the Statement Code and the Status Code are equal to zero, and execution of the statement is successfully completed.

12.4.2 STATEMENT CODE AND STATUS CODE

The leftmost two characters of the Data Base Status Indicator contain the Statement Code. For the value that indicates a specific data base manipulation statement, see Figure 12.4-1, Statement Codes.

The rightmost five characters of the Data Base Status Indicator contain the Status Code. For the value that indicates a specific Data Base Exception Condition, see Figure 12.4-2, Data Base Status Codes.

12.4.3 STATEMENT CODE AND STATUS CODE COMBINATIONS

The valid permissible combinations of Statement Code and Status Code are shown in Figure 12.4-3, Statement Code and Status Code Combinations. An 'X' at an intersection indicates a valid permissible combination.

12.4.4 USE OF SPECIAL REGISTERS FOR DATA BASE EXCEPTION CONDITIONS

When the DBCS recognizes certain Data Base Exception Conditions, it places values into the special registers DB-REALM-NAME, DB-SET-NAME, DB-RECORD-NAME, and DB-CONFLICT.

12.4.4.1 DB-REALM-NAME

During the execution of a data base manipulation statement that is not successfully completed, the DBCS places the realm name of the realm associated with the Data Base Exception Condition in the special register DB-REALM-NAME.

12.4.4.2 DB-SET-NAME

During the execution of a data base manipulation statement that is not successfully completed, the DBCS places the set-name of a set type associated with the Data Base Exception Condition in the special register DB-SET-NAME.

Withdrawn/Retired

12.4.4.3 DB-RECORD-NAME

During the execution of a data base manipulation statement that is not successfully completed, the DBCS places the record-name of a record type associated with the Data Base Exception Condition in the special register DB-RECORD-NAME.

12.4.4.4 DB-CONFLICT

During the execution of a CONNECT, DISCONNECT, ERASE, or MODIFY statement, if the DBCS determines that the object record or its record relationships have been modified by a concurrent run unit, a Data Base Exception Condition occurs and the DBCS places a value in the special register DB-CONFLICT to indicate the type of modification that has occurred. The value placed in the special register is the absolute value resulting from the addition of the values shown below for each type of modification:

VALUE	MODIFICATION
1	The object record has been made a member of one or more sets.
2	The object record has been removed from membership in one or more sets.
4	The contents or the null data attribute of one or more data items in the object record have been changed.
8	The object record has been deleted.

Each value is added to the value in the special register only once during the execution of any data base manipulation statement, regardless of how many times the record or its record relationships has changed since the record was placed in the now current monitored mode or extended monitored mode.

Withdrawn/Retirée

STATUS INDICATORS

Statement	Statement Code
ACCEPT	01
CONNECT	02
DISCONNECT	03
ERASE	04
FIND	05
FINISH	06
FREE	07
GET	08
IF	09
KEEP	10
MODIFY	11
ORDER	12
READY	13
REMONITOR	14
STORE	15

Figure 12.4-1 Statement Codes

Status Code	Data Base Exception Condition
	(Monitored Mode or Extended Monitored Mode Exception Conditions)
01100	Object record or its record relationships have been altered by a concurrent run unit.
01200	The object record is in extended monitored mode.
01300	No record is in extended monitored mode.
	(Data Base Retrieval Exception Conditions)
02100	End of a set or realm has been reached, and an attempt to locate the next record has been made.
02200	An unavailable realm has been requested by the DBCS.
02300	No set can be located to satisfy the set selection criteria.

Withdrawn/Retirée

02400	No record can be located to satisfy the record selection expression.
	(Currency Indicator Exception Conditions)
03100	Current record of realm, set type, or record type is null.
03200	Current record of the run unit is null.
03300	Current record of the run unit is not correct record type.
	(Name Specification Exception Conditions)
04100	Data base key value is inconsistent with realm-name.
04200	Record type or set type requested by the DBCS is not defined in the Sub-schema.
04300	Realm-name is invalid.
	(Data Item Value Exception Conditions)
05100	Contents of data items are duplicated in data base.
05200	Contents of data item are invalid.
05300	Contents of data item are not within range specified in CHECK clause.
	(Deleted Data Base Record Exception Conditions)
07100	Access to a deleted record is specified.
07200	Deletion of a non-empty set is specified.
	(Membership Exception Conditions)
08100	Object record is a member of the set.
08200	Object record is a permanent member of the set.
08300	Object record is not a member of the set.

Withdrawn/Retirée

STATUS INDICATORS

08400	Set members cannot be stored in both permanent and temporary realms.
08500	Set membership change is required.
	(Ready Mode Exception Conditions)
09100	Realm not in ready mode.
09200	Invalid usage mode for realm.
09300	Realm in ready mode.
09400	Usage mode conflict in a concurrent run unit.
7XXXX	Implementor-defined. (X may be any character in the computer character set.)
	(Resource Allocation Exception Conditions)
80100	Space in data base is exhausted.
80200	Space in realm is exhausted.
80300	DBCS space is exhausted.
80400	No data base key values in the data base are available for assignment.
80500	No data base key values in the realm are available for assignment.
	(Privacy Lock Exception Conditions)
90100	Locked resource is specified.
99000	Locked resource is specified resulting in abnormal termination of the run unit.

Figure 12.4-2 Data Base Status Codes

Withdrawn/Retirée

STATUS INDICATORS

STATEMENT CODE / STATUS CODE	01 ACCEPT	02 CONNECT	03 DISCONNECT	04 ERASE	05 FIND	06 FINISH	07 FREE	08 GET	09 IF	10 KEEP	11 MODIFY	12 ORDER	13 READY	14 REMONITOR	15 STORE
01100		X	X	X							X				
01200										X					
01300							X								
02100					X										
02200		X	X	X	X				X		X		X		X
02300					X						X				X
02400					X										
03100	X	X			X						X	X			X
03200	X	X	X	X	X			X	X	X	X			X	
03300		X	X	X	X			X			X				
04100															X
04200				X	X										
04300															X
05100		X	X	X							X	X			X
05200	X							X			X				X
05300								X			X				X
07100					X			X							
07200				X											
08100		X													
08200			X												
08300			X								X				
08400		X									X				X
08500											X				
09100		X	X	X	X	X					X	X			X
09200		X	X	X							X	X			X
09300													X		
09400													X		
7XXXX		X	X	X	X	X	X	X	X	X	X	X	X	X	X
80100		X	X	X							X	X			X
80200		X	X	X							X	X			X
80300		X	X	X	X	X	X	X	X	X	X	X	X	X	X
80400															X
80500															X
90100		X	X	X	X			X			X	X	X		X
99000		X	X	X	X			X			X	X	X		X

Figure 12.4–3 Statement Code and Status Code Combinations

Withdrawn/Retiree

12.5 DATA BASE CONDITIONS

Data base conditions enable the object program to select between alternate paths of control depending upon the truth value of a test involving a current record and its set membership, if any. There are three data base conditions:

tenancy condition
member condition
nullity condition

12.5.1 TENANCY CONDITION

The tenancy condition determines whether or not the current record of the run unit is a member, or owner, or either in one or more specified sets. The General Format of the tenancy condition is:

[NOT] [set-name] { OWNER / MEMBER / TENANT }

If set-name is specified, only the set type referenced by set-name is considered in determining the truth value of the condition. If set-name is not specified, all set types are considered in determining the truth value of the condition.

If the OWNER phrase is specified, the result of the test is true if the current record of the run unit is the owner of at least one of the referenced sets. If the MEMBER phrase is specified, the result of the test is true if the current record of the run unit is a member of at least one of the referenced sets. If the TENANT phrase is specified, the result of the test is true if the current record of the run unit is either a member or an owner in at least one of the referenced sets.

If NOT is specified, the result of the test is reversed.

12.5.2 MEMBER CONDITION

The member condition determines whether or not a specified set has any members. The general format of the member condition is:

set-name IS [NOT] EMPTY

If set-name does not reference a singular set, the object set is the one specified by the current record of the set type referenced by set-name. If set-name references a singular set, the object set is that singular set. If the object set has no members, the result of the test is true. If the object set has members, the result of the test is false.

Withdrawn/Retirée

If NOT is specified, the result of the test is reversed.

12.5.3 NULLITY CONDITION

The nullity condition determines whether or not a specified data item has the null attribute. The general format of the nullity condition is:

identifier IS [NOT] NULL

Identifier must be defined in the Sub-schema, but it must not be reference modified. If the data item referenced by identifier has the null attribute, the result of the test is true. Otherwise, the result of the test is false. If identifier is a group item, the result of the test is true only if every data item subordinate to that group item has the null data attribute.

If NOT is specified, the result of the test is reversed.

Withdrawn/Retirée

12.6 RECORD SELECTION EXPRESSIONS

12.6.1 GENERAL DESCRIPTION

A record selection expression is used to specify to the DBCS the criteria to be used in selecting a record in the data base. The record thus selected becomes the current record of the run unit, upon which subsequent statements may operate when accessing the data base.

A record selection expression can appear only in a FIND statement.

12.6.2 FORMAT OF RECORD SELECTION EXPRESSIONS

There are seven formats of record selection expressions, permitting both absolute and relative selection criteria. Records may be selected based on their contents, their data base key values, or currency indicators. If currency indicators selection is used, either a current record or a record relative to a current record can be selected.

Format 1

```
[record-name-1]; DB-KEY        IS identifier-1
```

Format 2

```
{ANY      }
{DUPLICATE}  record-name-2
```

Format 3

```
DUPLICATE WITHIN set-name-1 USING {identifier-2} . . .
```

Format 4

```
{NEXT      }
{PRIOR     }
{FIRST     }
{LAST      }  [record-name-3]   WITHIN {set-name-2 }
{integer-1 }                           {realm-name-1}
{identifier-3}
```

Format 5

```
CURRENT   [record-name-4]   [WITHIN {set-name-3 }]
                                    {realm-name-2}
```

Withdrawn/Retired

Format 6

OWNER WITHIN set-name-4

Format 7

record-name-5 WITHIN set-name-5 [CURRENT] [USING] {identifier-4 . . .}

12.6.3 SYNTAX RULES

FORMAT 1

1. Record-name-1 is not required. If specified, record-name-1 must reference a record whose data base key value is equal to the value of the data item referenced by identifier-1.

2. Identifier-1 must be defined as a data base key item.

FORMAT 2

3. The record referenced by record-name-2 must have a calculation location mode.

FORMAT 3

4. Identifier-2 must be defined in the Record Description entry of the current record of the set type referenced by set-name-1, but it must not be reference modified.

FORMAT 4

5. Integer-1 may be signed.

6. The data item referenced by identifier-3 must be a signed or unsigned elementary integer.

7. Record-name-3, if specified, must be defined in the Sub-schema.

FORMAT 5

8. Record-name-4, if specified, must be defined in the Sub-schema.

9. If record-name-4 and set-name-3 are specified, a record referenced by record-name-4 must be the current record of the set type referenced by set-name-3.

10. If record-name-4 and realm-name-2 are specified, a record referenced by record-name-4 must be the current record of the realm referenced by realm-name-2.

Withdrawn/Retirée

FORMAT 6

11. The set-type referenced by set-name-4 must not be a singular set.

FORMAT 7

12. Identifier-4 must be defined in the Record Description entry of the record type referenced by record-name-5, but it must not be reference modified.

12.6.4 GENERAL RULES

ALL FORMATS

1. An explanation of Data Base Exception Conditions which may be caused by the evaluation of record selection expressions is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. Evaluation of a record selection expression results in the identification, by the DBCS, of a specific record in the data base.

3. Records identified by a record selection expression must be stored in realms that are in ready mode.

4. All records that participate in the DBCS search for the identified record must be stored in realms that are in available mode.

FORMAT 1

5. The record identified is the record whose data base key value is equal to the value of the data item referenced by identifier-1.

FORMAT 2

6. If the ANY phrase is specified, the calculation key values in the record area associated with the record type referenced by record-name-2 are used to calculate the data base key value of the identified record. If more than one record in the data base contains calculation key values equal to those in the associated record area, the record identified is the one whose data base key value is the lowest.

7. If the DUPLICATE phrase is specified, the calculation key values of the record identified are all equal to the calculation key in the current record of the run unit, according to the rules for comparison of operands in a relation condition. The DBCS searches for the identified record beginning with the current record of the run unit and proceeds in sequence by ascending data base key values. The first record found after the current record of the run unit, with the proper calculation key values, is the record identified.

Withdrawn/Retirée

FORMAT 3

8. The DBCS identifies a member of the set of the set type referenced by set-name-1. That record is the one in which:

   a) Record type is equal to the record type of the current record of the set type referenced by set-name-1, and

   b) The contents of the data items referenced by identifier-2 are equal to all those in the current record of the set type referenced by set-name-1, according to the rules for comparison of operands in a relation condition.

   The DBCS begins its search for the identified record at the current record of the set type and proceeds to search in the order defined by the set ordering criteria of that set type.

FORMAT 4

9. If record-name-3 is not specified, all record types within the set referenced by set-name-2 or the realm referenced by realm-name-1 are considered in evaluating the record selection expression.

10. If record-name-3 is specified, only the record type of the record referenced by record-name-3 is considered in evaluation of the record selection. All other record types are ignored.

11. If the WITHIN set-name-2 phrase is specified, only a record that is a member of a set referenced by set-name-2 is identified.

12. If the WITHIN realm-name-1 phrase is specified, only a record stored in the realm referenced by realm-name-1 is identified.

13. If the NEXT phrase and the WITHIN realm-name-1 phrase are specified, the record identified is the one whose data base key value is the next higher, relative to the data base key value of the current record of the realm referenced by realm-name-1.

14. If the PRIOR phrase and the WITHIN realm-name-1 phrase are specified, the record identified is the one whose data base key value is the next lower, relative to the data base key value of the current record of the realm referenced by realm-name-1.

15. If the FIRST phrase and the WITHIN realm-name-1 phrase are specified, the record identified is the one whose data base key value is the lowest, relative to all records stored in the realm referenced by realm-name-1.

16. If the LAST phrase and the WITHIN realm-name-1 phrase are specified, the record identified is the one whose data base key value is the highest, relative to all records stored in the realm referenced by realm-name-1.

Withdrawn/Retirée

RECORD SELECTION EXPRESSIONS

17. If integer-1 or identifier-3 is specified and the WITHIN realm-name-1 phrase is specified, the record identified is the one whose ordinal position, in the realm, is equal to the value of integer-1 or the contents of the data item referenced by identifier-3. If the value of integer-1 or the contents of the data item referenced by identifier-3 is positive, the ordinal position is relative to the record whose data base key value is the lowest in the realm referenced by realm-name-1. If negative, the ordinal position is relative to the record whose data base key value is the highest in the realm referenced by realm-name-1. The contents of the data item referenced by identifier-3 must not be zero.

18. If the NEXT phrase and the WITHIN set-name-2 phrase are specified, the record identified is the next member record in the set, relative to the current record of the set type referenced by set-name-2 according to the set ordering criteria for that set type.

19. If the PRIOR phrase and the WITHIN set-name-2 phrase are specified, the record identified is the prior member record in the set, relative to the current record of the set type referenced by set-name-2 according to the set ordering criteria for that set type.

20. If the FIRST phrase and the WITHIN set-name-2 phrase are specified, the record identified is the first member of the set in which the current record of the set type referenced by set-name-2 is a tenant, according to the set ordering criteria for that set type.

21. If the LAST phrase and the WITHIN set-name-2 phrase are specified, the record identified is the last member of the set in which the current record of the set type referenced by set-name-2 is a tenant, according to the set ordering criteria for that set type.

22. If integer-1 or identifier-3 is specified and the WITHIN set-name-2 phrase is specified, the record identified is the one whose ordinal position, in the set in which the current record of the set referenced by set-name-2 is a tenant, is equal to the value of integer-1 or the contents of the data item referenced by identifier-3. If the value of integer-1 or the contents of the data item referenced by identifier-3 is positive, the ordinal position is relative to the first member of the set, according to the set ordering criteria of that set type. If negative, the ordinal position is relative to the last member of the set, according to the set ordering criteria of that set type. The contents of the data item referenced by identifier-3 must not be zero.

FORMAT 5

23. If record-name-4 is specified, and the WITHIN phrase is not specified, the record identified is the current record of the record type referenced by record-name-4.

24. If set-name-3 is specified, the record identified is the current record of the set type referenced by set-name-3.

25. If realm-name-2 is specified, the record identified is the current record of the realm referenced by realm-name-2.

26. If neither realm-name-2, set-name-3, nor record-name-4 is specified, the record identified is the current record of the run unit.

Withdrawn/Retirée

FORMAT 6

27. The record identified is the owner of the set referenced by set-name-4 in which the current record of the set type referenced by set-name-4 is a tenant.

FORMAT 7

28. The record identified has a record type equal to that of the record referenced by record-name-5. The record identified is a member of the set type referenced by set-name-5.

29. If the CURRENT phrase is specified, the record identified is a member of the set in which the current record of the set type referenced by set-name-5 is a tenant.

30. If the CURRENT phrase is not specified, the record identified is a member of the set specified by the set selection criteria of the set type referenced by set-name-5.

31. If the USING phrase is specified, the contents of the data items referenced by identifier-4 in the record identified are all equal to the contents of those data items, respectively, in the record area associated with the record referenced by record-name-5, according to the rules for comparison of operands in a relation condition.

32. If the USING phrase is not specified, the record identified is the first member of the referenced set, according to the set ordering criteria for that set type.

33. The DBCS begins its search at the first member record and proceeds in its search for the identified record according to the set ordering criteria for that set type, and the record identified is the first one meeting the requirements specified in General Rules 28 through 32.

Withdrawn/Retirée

12.7 MONITORED MODE AND EXTENDED MONITORED MODE

In an environment in which more than one run unit can be concurrently operating, modification of the data base must be more carefully controlled than in a single run unit environment. Specifically, protection is required to ensure that the integrity of the data base is not affected by the execution of concurrent run units. This protection is achieved by placing a record in monitored mode or extended monitored mode. Records are not locked, that is, made unavailable to concurrent run units, except during the time that the DBCS is altering them or their record relationships.

12.7.1 MONITORED MODE

The current record of the run unit is always in monitored mode or extended monitored mode. No other record can be in monitored mode. Establishment of monitored mode is automatic and occurs coincident with the establishment of the record as the current record of the run unit. Monitored mode is in effect for that record continuously until that record ceases to be the current record of the run unit, or until it is the object of a REMONITOR statement. The execution of a KEEP statement alters the monitored mode of the record to an extended monitored mode, but the resultant extended monitored mode behaves as though the record had been placed in that mode when it became the current record of the run unit.

12.7.2 EXTENDED MONITORED MODE

The current record of the run unit may be placed in extended monitored mode by execution of a KEEP statement. Any number of records can be simultaneously in extended monitored mode. Once established for a record, extended monitored mode continues until a FREE statement removes the record from extended monitored mode, or a REMONITOR statement references the record, or the realm in which the record is stored is removed from ready mode.

12.7.3 EFFECT OF MONITORING

If a record is in monitored mode or extended monitored mode during the execution of a MODIFY statement, CONNECT statement, DISCONNECT statement, or ERASE statement that affects that record, the following action is taken:

1. The DBCS determines whether the affected record has been modified by the execution of a concurrent run unit since the most recent establishment of monitored mode or extended monitored mode for that record.

2. If the affected record has not been modified by the execution of a concurrent run unit, execution of the statement proceeds according to the rules governing that statement.

3. If the execution of a concurrent run unit has modified the affected record, execution of the statement is unsuccessful, the record is not changed, and a Data Base Exception Condition occurs.

Withdrawn/Retired

12.8 CURRENCY INDICATORS

In the environment of a data base, it is necessary that the user be able to access records that are a part of more than one logical structure. Unlike files, in which only one record can logically follow a given record, in a data base, many records can logically follow a given record, depending upon the structure of the data base and the search criteria in use. Thus, there can be several simultaneous current records during the execution of a run unit, and several simultaneous current record pointers. These simultaneous current record pointers are called currency indicators.

To distinguish between the several current records that are possible, they are called the current record of the run unit, the current record of a given set type, the current record of a given realm, and the current record of a given record type. The DBCS maintains conceptual pointers, for each run unit, to each of these records in the data base.

Any of these conceptual pointers may be null, that is, they may specify no record, if there is no current record of that type.

12.8.1 CURRENT RECORD OF THE RUN UNIT

For each run unit, the DBCS maintains a separate conceptual pointer. The conceptual pointer for each run unit is maintained by the DBCS to identify the object record of any data manipulation statement that does not specify an object record. The execution of some types of statements causes the DBCS to update its conceptual pointer to a different record, while the execution of other types of statements does not affect the conceptual pointer to the current record of the run unit.

12.8.2 CURRENT RECORD OF A SET TYPE

For each Set Description entry specified in the Sub-schema, the DBCS maintains a separate conceptual pointer. The conceptual pointer for each set type is maintained by the DBCS to identify the record in that set type that was last referenced by a successfully executed statement. For singular sets, the DBCS is considered to be a tenant. Thus, a null currency indicator for a singular set implies that the DBCS itself is the current record.

12.8.3 CURRENT RECORD OF A REALM

For each Realm Description entry specified in the Sub-schema, the DBCS maintains a separate conceptual pointer. The conceptual pointer for each realm is maintained by the DBCS to identify the record in that realm that was last referenced by a successfully executed statement.

12.8.4 CURRENT RECORD OF A RECORD TYPE

For each Record Description entry specified in the Sub-schema, the DBCS maintains a separate conceptual pointer. The conceptual pointer for each record type is maintained by the DBCS to identify the record of that record type that was last referenced by a successfully executed statement.

Withdrawn/Retired

CURRENCY INDICATORS

12.8.5 SUMMARY OF CURRENCY INDICATORS

Currency indicators are conceptual pointers maintained and used by the DBCS. There are as many currency indicators for a run unit as there are set descriptions, realm descriptions, and record types described in that run unit's Sub-schema, plus one (1) currency indicator for the run unit as a whole. Each currency indicator identifies the last referenced record to which that currency indicator applies, unless the statement referencing that record specifies that one or more currency indicators are to remain unchanged from their previous values. If there is no such record, the currency indicator is null. The DBCS uses the currency indicators as place markers to control its logical sequence of accesses to the data base.

Withdrawn/Retired

12.9 PRIVACY LOCKS AND PRIVACY KEYS

To protect the security of the data base, a system of privacy locks and privacy keys is provided. The description of the data base, as provided by the Schema and Sub-schema, may specify that privacy locks apply to usage modes, realms, sets, records, data item and/or procedural statements. If such privacy locks are specified, the COBOL user must specify the correct privacy key to open each privacy lock before the DBCS will permit the COBOL program to access the locked resource. If a privacy key is specified for a resource that is not locked, the DBCS ignores that privacy key.

12.9.1 PRIVACY LOCKS

A privacy lock may be specified in the Schema or the Sub-schema to be a literal, an implementor-name, or a routine-name. If the privacy lock is a literal or an implementor-name, its value is a character string. If the privacy lock is a routine-name, the nature of the lock is dependent upon the routine. The value of a privacy lock is always available to the DBCS.

12.9.2 PRIVACY KEYS

The value of a privacy key consists of a character string stored in the special register DB-PRIVACY-KEY. These values are placed in DB-PRIVACY-KEY during execution of a USE FOR PRIVACY procedure when a realm is being placed in ready mode by the execution of a READY statement.

12.9.3 MATCHING OF PRIVACY LOCKS AND PRIVACY KEYS

The DBCS controls the matching of privacy locks to privacy keys. If the value of a privacy lock is a character string, the DBCS performs the matching according to the rules for comparison of alphanumeric operands in a relation condition; if the privacy key is thus equal to the corresponding privacy lock, the locked resource is made available to the run unit that supplied the privacy key. If the privacy key does not equal the privacy lock, the locked resource associated with that privacy lock is not made available to the run unit, and the execution of any statement in the run unit that requires access to that locked resource results in the occurrence of a Data Base Exception Condition.

If the privacy lock is the result of the execution of a routine, the DBCS makes the value of the associated privacy key available to the routine. If the privacy key value satisfies the requirements of the routine, the routine unlocks the locked resource associated with that privacy lock. If the privacy key value does not satisfy the requirements of the routine, the routine can cause termination of the execution of the run unit that supplied the privacy key, or it can transfer control back to the run unit. In the later case, execution of any statement in the run unit that requires access to that locked resource results in the occurrence of a Data Base Exception Condition.

Withdrawn/Retired

12.10 THE ACCEPT STATEMENT

12.10.1 FUNCTION

The ACCEPT statement causes the contents of the specified currency indicators to be accessed and provides a means for deriving the realm-name that corresponds to a data base key value.

12.10.2 GENERAL FORMAT

Format 1

```
ACCEPT identifier-1 FROM [ record-name ]          CURRENCY
                         [ realm-name  ]
                         [ set-name    ]
```

Format 2

```
ACCEPT identifier-2 FROM [ record-name ]          REALM-NAME
                         [ set-name    ]
                         [ identifier-3 ]
```

12.10.3 SYNTAX RULES

1. Identifier-1 and identifier-3 must reference data base key items. Identifier-2 must reference an alphanumeric elementary data item.

12.10.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of an ACCEPT statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

FORMAT 1

2. If record-name, realm-name, or set-name is specified, the data base key value for the current record of record-name, realm-name or set-name, respectively, is placed in the data item referenced by identifier-1. That data base key value must not have the null data attribute.

3. If a record-name, realm-name, or set-name is not specified, the data base key value for the current record of the run unit is placed in the data item referenced by identifier-1. That data base key value must not have the null data attribute.

Withdrawn/Retirée

4. The realm-name derived from the data base key value in the data item referenced by identifier-3 or from the specified currency indicator is placed in the data item referenced by identifier-2, according to the rules for an alphanumeric elementary move.

5. If identifier-3 or a currency indicator is not specified, the name of the realm associated with the current record of the run unit is placed in the data item referenced by identifier-2, according to the rules for an alphanumeric elementary move.

6. The contents of the data item referenced by identifier-3 or the currency indicator used to derive the realm-name must not have the null data attribute.

Withdrawn/Retiree

12.11 THE CONNECT STATEMENT

12.11.1 FUNCTION

The CONNECT statement causes a record stored in the data base to become a member of one or more sets in which the record's record type has been defined to be a possible member.

12.11.2 GENERAL FORMAT

CONNECT [record-name] TO { {set-name-1} . . . / ALL }

12.11.3 SYNTAX RULES

1. If record-name is specified, it must be the name of the current record of the run unit.

2. The current record type of the run unit must be defined to be a manual or transient automatic member of all sets specified in the CONNECT statement. If ALL is specified, the record type must be defined as above in at least one set type defined in the Sub-schema.

12.11.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by the execution of a CONNECT statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. The realms in which all affected sets are stored, and in which the current record of the run unit is stored, must be in update usage mode. All records that contain derived data items that are affected by the execution of the CONNECT statement must also be stored in realms in update usage mode.

3. Execution of the CONNECT statement causes the current record of the run unit to become a member of one or more sets provided that:

   a. Its record type is defined as a manual member, or a transient automatic member, of that set type, and

   b. It is not currently a member of any set of that set type.

4. If ALL is specified, the current record of the run unit becomes a member of all sets whose set types are defined in the Sub-schema and in which its record type is defined to be a member. If ALL is not specified, the current record of the run unit becomes a member of each set specified by set-name-1.

Withdrawn/Retirée

5. The current record of the run unit is connected in each affected set in accordance with the set ordering criteria for each such set type.

6. The set or sets to which the current record of the run unit is connected is the set in which the current record of the set type for each affected set is a tenant.

7. The current record of the run unit becomes the current record of each set type to which the record is connected as a result of execution of a CONNECT statement.

8. If the current record of the run unit cannot be connected to every set specified, it is not connected to any set, and execution of the CONNECT statement is unsuccessful.

Withdrawn/Retirée

12.12 THE DISCONNECT STATEMENT

12.12.1 FUNCTION

The DISCONNECT statement logically removes a record from one or more sets.

12.12.2 GENERAL FORMAT

DISCONNECT [record-name] FROM $\left\{ \begin{array}{l} \{\text{set-name-1}\} \ \ldots \\ \text{ALL} \end{array} \right\}$

12.12.3 SYNTAX RULES

1. If record-name is specified, it must be the name of the current record of the run unit.

12.12.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a DISCONNECT statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)
2. The realm in which the current record of the run unit is stored, and the realms in which affected sets are stored, must be in update usage mode. All records that contain derived data items that are affected by execution of the DISCONNECT statement must also be stored in realms that are in update usage mode.
3. The record type of the current record of the run unit must be defined as a transient member of all specified set types.
4. The current record of the run unit must be a current member of all specified sets.
5. If set-name-1 is specified, execution of the DISCONNECT statement causes the current record of the run unit to be removed from membership in each specified set.
6. If ALL is specified, the current record of the run unit is removed from membership in all sets in which its record type is defined as transient.
7. Currency indicators are not affected by the execution of a DISCONNECT statement.

Withdrawn/Retirée

12.13 THE ERASE STATEMENT

12.13.1 FUNCTION

The ERASE statement logically removes one or more records from the data base.

12.13.2 GENERAL FORMAT

ERASE [record-name] [PERMANENT / SELECTIVE / ALL] MEMBERS

12.13.3 SYNTAX RULES

1. Record-name is not required. If specified, record-name must be the name of the current record of the run unit.

12.13.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of the ERASE statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. The realms in which all affected records and sets are stored must be in update usage mode. All records that contain derived data items that are affected by the execution of the ERASE statement must also be stored in realms that are in update usage mode.

3. Execution of the ERASE statement logically removes one or more records from the data base.

4. Execution of the ERASE statement operates on the current record of the run unit. It may additionally operate upon other records as defined in the following rules.

5. The current record of the run unit is logically removed from the data base and is no longer accessible.

6. If neither PERMANENT, SELECTIVE, nor ALL is specified, the current record of the run unit must not be the owner of a set that currently has members.

7. If PERMANENT is specified, and the current record of the run unit is the owner of a set that currently has members, then each permanent member of that set is treated as though it were the current record of the run unit, and is logically removed from the data base. This process is repeated until all such records have been removed from the data base. Any records in the affected sets which are transient members are not removed from the data base but are treated as though they were the object of a DISCONNECT statement specifying that set-name.

Withdrawn/Retirée

ERASE

8. If SELECTIVE is specified, execution of the ERASE statement proceeds as though PERMANENT had been specified, except where a transient member of an affected set is not a member of any other set, in which case that member is logically removed from the data base.

9. If ALL is specified, the current record of the run unit is logically removed from the data base. If that record is the owner of a set, all members of that set are also removed from the data base. Any record so removed which is itself the owner of a set is treated as though it were the current record of the run unit and the process is repeated until all such records have been removed from the data base.

10. At the completion of the execution of an ERASE statement, the value of the run unit currency indicator is null. Other currency indicators are not affected.

11. If execution of an ERASE statement is not able to remove or disconnect all object records, no records are affected, and execution of the ERASE statement is unsuccessful.

Withdrawn/Retired

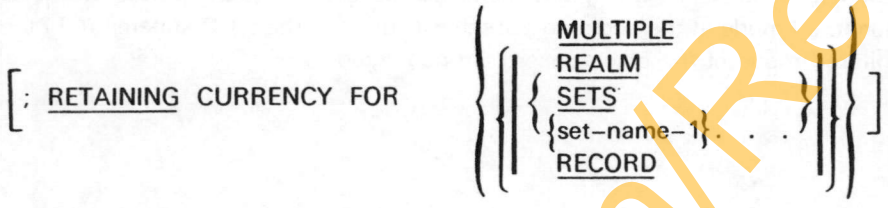
12.14 THE FIND STATEMENT

12.14.1 FUNCTION

The FIND statement is used to establish a specific record occurrence in the data base as the object of certain subsequent statements.

12.14.2 GENERAL FORMAT

FIND record-selection-expression



12.14.3 SYNTAX RULES

1. Record-selection-expression must reference a record stored in a realm that is in ready mode.

12.14.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a FIND statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. An explanation of the record-selection-expression is given under the appropriate paragraph. (See 12.6, Record Selection Expressions.)

3. Execution of the FIND statement causes the record referenced by record-selection-expression to become the current record of the run unit. The realm-name and record-name of that record are placed in the special registers DB-REALM-NAME and DB-RECORD-NAME, respectively.

4. If the RETAINING phrase is not specified, the record referenced by record-selection-expression becomes the current record of its realm-name, the current record of its record-name, and the current record of all sets in which it is a tenant.

5. If the RETAINING phrase with the key word REALM is specified no realm currency indicators are changed.

6. If the RETAINING phrase with the key word SETS is specified, no set type currency indicators are changed.

Withdrawn/Retirée

7. If the retaining phrase with the key word RECORD is specified, no record type currency indicators are changed.

8. If the RETAINING phrase and set-name-1 are specified, the set type currency indicators of set-name-1 are not changed.

9. If the RETAINING phrase with the key word MULTIPLE is specified, no realm, set type, or record type currency indicators are changed.

10. Execution of a FIND statement does not make the selected record available to the program, it merely identifies the record for use in certain subsequent statements.

11. Execution of a FIND statement establishes monitored mode for the object record, unless the object record is in extended monitored mode at the beginning of the execution of the FIND statement. In that case, the extended monitored mode of the object record is unchanged.

Withdrawn/Retirée

12.15 THE FINISH STATEMENT

12.15.1 FUNCTION

The FINISH statement terminates the availability of one or more realms to the program.

12.15.2 GENERAL FORMAT

```
FINISH  { [set-name-1] . . .   }
        { [realm-name-1] . . . }
```

12.15.3 SYNTAX RULES

1. All affected realms must be in ready mode at the beginning of the execution of a FINISH statement for those realms.

12.15.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of the FINISH statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. If set-name-1 is specified, the ready mode of all realms which may contain tenants of sets referenced by set-name-1 is terminated.

3. If realm-name-1 is specified, the ready mode of the realm referenced by realm-name-1 is terminated.

4. If neither set-name-1 nor realm-name-1 is specified, the ready mode of all realms is terminated.

5. If the execution of a FINISH statement is not able to terminate the ready mode of all affected realms, no realms are affected, and execution of the FINISH statement is unsuccessful.

6. At the completion of the execution of a FINISH statement:

   a. All currency indicators which identify records stored in the affected realms are null, and

   b. All records in the affected realms are removed from extended monitored mode.

7. If any realm is in ready mode at the time the run unit is terminated, the action taken is defined by the implementor.

Withdrawn/Retirée

12.16 THE FREE STATEMENT

12.16.1 FUNCTION

The FREE statement removes one or more records from extended monitored mode.

12.16.2 GENERAL FORMAT

```
FREE { ALL                          }
     {{record-name-1} . . .         }
```

12.16.3 SYNTAX RULES

1. Record-name-1 must be defined in the Sub-schema.

12.16.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of the FREE statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. If ALL is specified, the object records are all records that are in extended monitored mode, except that the current record of the run unit is not an object record.

3. If record-name-1 is specified, the object records are the records referenced by record-name-1 that are in extended monitored mode, except that the current record of the run unit is not an object record.

4. There must be at least one object record of each record type specified.

5. Execution of the FREE statement removes the object records from extend monitored mode. (See 12.7, Monitored Mode and Extended Monitored Mode.)

6. If all object records cannot be removed from extended monitored mode, no record is affected by execution of the FREE statement, and execution of the FREE statement is unsuccessful.

Withdrawn/Retirée

12.17 THE GET STATEMENT

12.17.1 FUNCTION

The GET statement is used to make part or all of a data base record available in its associated record area.

12.17.2 GENERAL FORMAT

GET [identifier-1] . . .

12.17.3 SYNTAX RULES

1. Identifier-1, if specified, must be defined in the Record Description entry of the current record of the run unit.

2. If any specified identifier consists solely of a record-name, no other identifiers may be specified.

3. All records that contain data items required by the DBCS to generate values required by execution of the GET statement must be stored in realms that are in available mode.

12.17.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a GET statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. Execution of a GET statement causes the DBCS to place all or part of the current record of the run unit in its associated record area.

3. If identifier-1 is not specified, the entire contents of the current record of the run unit are made available.

4. If identifier-1 consists solely of a record-name, the entire contents of the current record of the run unit are made available.

5. If identifier-1 references data items subordinate to a record-name, only the contents of those data items are made available. The contents of other data items in the associated record area are unchanged.

Withdrawn/Retirée

12.18 THE KEEP STATEMENT

12.18.1 FUNCTION

The KEEP statement causes extended monitored mode to be established for the current record of the run unit.

12.18.2 GENERAL FORMAT

KEEP

12.18.3 SYNTAX RULES

1. The current record of the run unit must be in monitored mode at the beginning of the execution of the KEEP statement.

12.18.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a KEEP statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. Execution of a KEEP statement causes the monitored mode of the current record of the run unit to be changed to extended monitored mode. The effect of the execution of a KEEP statement is as though the current record of the run unit had been in extended monitored mode since its establishment as the current record of the run unit. (See 12.7, Monitored Mode and Extended Monitored Mode.)

Withdrawn/Retirée

12.19 THE MODIFY STATEMENT

12.19.1 FUNCTION

The MODIFY statement is used to alter the contents of one or more data items in a record and/or to change the set membership of the record.

12.19.2 GENERAL FORMAT

Format 1

```
MODIFY [record-name]  [ { INCLUDING } ]
                      [ { ONLY      } ]

    { ALL                          }
    {{set-name-1} . . .            }  MEMBERSHIP ]
```

Format 2

```
MODIFY {identifier-1} . . .

    [ INCLUDING { ALL                          }
                {{set-name-1} . . .            }  MEMBERSHIP ]
```

12.19.3 SYNTAX RULES

1. Record-name is not required. If specified, record-name must be the name of the current record of the run unit.

2. Identifier-1 must not reference derived data items.

3. Identifier-1 must reference data items contained within the current record of the run unit.

4. All realms in which affected records are stored must be in update usage mode. All sets in which membership is affected by execution of the MODIFY statement must be stored in realms that are in update usage mode. All records that contain derived data items that are affected by the execution of the MODIFY statement must be stored in realms that are in update usage mode.

5. All sets referenced by the set selection criteria of the affected set type must be stored in realms that are in available mode.

Withdrawn/Retirée

12.19.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a MODIFY statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. The object of a MODIFY statement is the current record of the run unit.

3. Execution of a MODIFY statement without the ONLY phrase causes the DBCS to replace logically, in the data base, the contents of data items in the current record of the run unit with the contents of data items in the record area associated with that record. All other data items in the data base remain unchanged, unless they are derived data items affected by the MODIFY statement.

4. Execution of a MODIFY statement with the ONLY phrase does not affect the contents, in the data base, of any data items.

5. Execution of a MODIFY statement with either the INCLUDING phrase or the ONLY phrase causes the set membership of the current record of the run unit to change. If ALL is specified, the record's membership in every set in which it is a member is changed. If set-name-1 is specified, the record's membership in the sets referenced by set-name-1 is changed. The sets to which the record is connected, as the result of the execution of a MODIFY statement, are governed by the set selection criteria. The record is connected to each set in accordance with the set ordering criteria of each set type.

6. In Format 1, if the ONLY phrase is not specified the contents of all data items contained in the current record of the run unit are replaced, in the data base, by the contents of the record area associated with that record. Derived data items may be affected indirectly.

7. In Format 2, the contents of the data items referenced by identifier-1 in the current record of the run unit are replaced, in the data base, by the contents of the data items referenced by identifier-1 in the record area associated with that record.

8. After successful execution of a MODIFY statement, the current record of the run unit becomes the current record of the set types specified.

9. If neither the INCLUDING phrase nor the ONLY phrase is specified, execution of the MODIFY statement must not change a data item which affects the record's membership in any set.

Withdrawn/Retirée

12.20 THE ORDER STATEMENT

12.20.1 FUNCTION

The ORDER statement is used to reorder the members of a set logically.

12.20.2 GENERAL FORMAT

```
ORDER set-name [LOCALLY]

{ ON { DESCENDING } KEY { [record-name-1] . . .  RECORD-NAME } } . . .
     { ASCENDING  }     { [record-name-1] . . .  DB-KEY      }
                        { {data-name-1} . . .                }
```

12.20.3 SYNTAX RULES

1. The LOCALLY phrase must be specified if the Set Description entry of set-name specifies that the set order is fixed.

2. Record-name-1 must reference a record type that is defined as a member of the set type referenced by set-name.

3. Data-name-1 must reference data items defined in records that are defined as members of the set type referenced by set-name.

4. Each data-name-1 specified in the same KEY phrase must reference a data item defined in a different Record Description entry.

5. The same record-name must not be referenced more than once in a specific KEY phrase. If data-name-1 is specified, the record types in which data-name-1 is defined must not be referenced in the same KEY phrase.

6. Data-name-1 is a KEY data-name and is subject to the following rules:

   a. KEY data-names may be qualified.

   b. Data items referenced by KEY data-names must not be variable length items.

   c. None of the data items referenced by KEY data-names may be described by an entry which either contains an OCCURS clause or is subordinate to an entry which contains an OCCURS clause.

Withdrawn/Retirée

7. Each data-name-1 including its qualifiers, if any, may be specified only once in the same ORDER statement.

8. Any record-name or data-name referenced in the second or succeeding KEY phrases of an ORDER statement must not reference records that are not referenced in the first KEY phrase of that ORDER statement.

9. If the LOCALLY phrase is specified and all records in the set are in non-temporary realms, those realms must be in exclusive retrieval mode or protected retrieval mode.

10. If the LOCALLY phrase is not specified, all records in the set must be in realms that are in exclusive update usage mode.

11. All records that contain derived data items that are affected by execution of the ORDER statement must be stored in realms that are in update usage mode.

12.20.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may occur during the execution of an ORDER statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)

2. The object records of an ORDER statement are members of the set in which the current record of set-name is a tenant.

   a. If record-name-1 is referenced, all records referenced by record-name-1 are object records.

   b. If data-name-1 is specified, the records in which data-name-1 is defined are the object records.

   c. If neither record-name-1 nor data-name-1 is referenced, all records that are members of the set are object records.

3. Execution of an ORDER statement causes the object records to be logically reordered.

4. Any member records of the set that are not object records are placed logically at the end of the set. Their relative order is not affected.

5. If the set order is fixed or the LOCALLY phrase is specified, the logical reordering of the object records acts only until the run unit is terminated or until the ready mode for all realms in which object records are stored is terminated. At that time, the logical order of the object records at the beginning of the execution of the ORDER statement is restored.

6. If the LOCALLY phrase is not specified, the logical reordering of the object records remains in effect in the data base after the run unit is terminated.

Withdrawn/Retirée

7. The KEY phrases are listed from left to right in the ORDER statement in order of decreasing significance. Within a given KEY phrase, the sequence in which RECORD-NAME, DB-KEY and data-names are specified is not significant.

   a. When the ASCENDING phrase is used, the ordered sequence will be from the lowest value of the keys to the highest value, according to the rules for comparison of operands in a relation condition.

   b. When the DESCENDING phrase is used, the ordered sequence will be from the highest value of the keys to the lowest value, according to the rules for comparison of operands in a relation condition.

8. The values of all keys for a given record are treated as though they were juxtaposed and left-justified. If the size of the entire key for any object record is less than the size of the entire key of any other object record, the smaller is increased in size to equal the size of the larger by the juxtaposition of LOW-VALUES in the rightmost positions of the shorter.

9. If the RECORD-NAME phrase is specified, the values of the record-names of the object records are the key values.

10. If the DB-KEY phrase is specified, the values of the data base keys of the object records are the key values.

Withdrawn/Retirée

12.21 THE READY STATEMENT

12.21.1 FUNCTION

The READY statement prepares one or more realms for processing.

12.21.2 GENERAL FORMAT

```
READY [ [ set-name-1    ]
      [ realm-name-1 ]  . . .

      [ USAGE-MODE IS [ EXCLUSIVE ] { RETRIEVAL } ] . . .
                     [ PROTECTED ] { UPDATE    } ]
```

12.21.3 SYNTAX RULES

1. If realm-name-1 is defined as a temporary realm, or if set-name-1 is defined in a temporary realm, then the EXCLUSIVE UPDATE phrase of the USAGE-MODE phrase must be specified.

12.21.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a READY statement is given under the appropriate paragraph. (See 12.4, Date Base Status Indicators.)

2. A READY statement must not affect a realm already in the ready mode.

3. Execution of any READY statement affects entire realms, regardless of the specific format of the statement. If the execution of a READY statement is unable to place all referenced realms in ready mode, no realms are affected, and execution of the READY statement is unsuccessful.

4. If set-name-1 is specified, all realms which may contain tenants of sets referenced by set-name-1 are prepared for access and placed in ready mode.

5. If realm-name-1 is specified, each realm referenced is prepared for access and placed in ready mode.

6. If neither set-name-1 nor realm-name-1 is specified, all realms in that portion of the data base described by the Sub-schema are prepared for access and placed in ready mode.

7. The USAGE-MODE phrase establishes a usage mode for the affected realms and thereby specifies the type of operations permitted. Usage modes remain in effect for the affected realms until a FINISH statement for the affected realms is executed, or until the termination of the run unit which executed

Withdrawn/Retirée

the READY statement. Each USAGE-MODE phrase establishes the usage mode for the realms specified in the immediately preceding phrase.

8. The UPDATE phrase permits both access to, and modification of, the contents of the affected realms.

9. The RETRIEVAL phrase permits access to records.

10. The EXCLUSIVE phrase prevents any other run unit from executing a READY statement for the affected realms.

11. The PROTECTED phrase prevents any other run unit from updating the affected realms.

12. If the USAGE-MODE phrase is omitted, execution of the READY statement occurs as though USAGE-MODE IS RETRIEVAL had been specified.

13. Execution of a READY statement causes all privacy key procedures, if any, associated with all set types, data items, and operations that are permitted by the specified usage mode on all record types that may be contained in the affected realms, to be executed. In addition, any privacy key procedures associated with the affected realms themselves are executed.

14. If any privacy lock associated with any record type, set type, realm, or data item is modified, that modification has no effect for this run unit until the execution of the next READY statement for the usage mode and realm or realms involved. At that time, the new privacy lock is placed into effect for this run unit immediately prior to the execution of any privacy key procedures caused by the READY statement.

15. Execution of a READY statement makes available the record areas that are associated with each record type that may be contained in the affected realms, unless those record areas are already available as the result of the execution of a previous READY statement.

16. A usage mode must not be specified that conflicts with the usage mode of an affected realm already established by a concurrent run unit.

Withdrawn/Retirée

12.22 THE REMONITOR STATEMENT

12.22.1 FUNCTION

The REMONITOR statement causes a monitored mode or extended monitored mode to be ended, followed immediately by the establishment of a new monitored mode or extended monitored mode for the same record.

12.22.2 GENERAL FORMAT

```
REMONITOR [ ALL                          ]
          [ {record-name-1} . . .        ]
```

12.22.3 SYNTAX RULES

1. Record-name-1 must be defined in the Sub-schema.

12.22.4 GENERAL RULES

1. An explanation of Data Base Exeception Conditions which may be caused by execution of a REMONITOR statement is given under the appropriate paragraph. (See 12.4, Data Base Status Indicators.)
2. If ALL is specified, all records that are in extended monitored mode are object records, except the current record of the run unit.
3. If record-name-1 is specified, all records referenced by record-name-1 that are in extended monitored mode are object records, except the current record of the run unit.
4. If neither ALL nor record-name-1 is specified, the current record of the run unit is the object record.
5. For each record type specified there must be at least one object record.
6. If all object records cannot be affected, no record is affected by execution of the REMONITOR statement, and execution of the REMONITOR statement is unsuccessful.
7. If either ALL or record-name-1 is specified, the extended monitored mode of the object records is ended, followed immediately by the establishment of a new extended monitored mode.
8. If neither ALL nor record-name-1 is specified, the monitored mode or extended monitored mode of the current record of the run unit is ended, followed immediately by the establishment of a new monitored mode for that record.

Withdrawn/Retiree

9. Execution of a REMONITOR statement behaves as if the establishment of monitored mode or extended monitored mode were coincident with the ending of the previous monitored mode or extended monitored mode. Each object record, therefore, is in monitored mode or extended monitored mode continuously during the execution of the REMONITOR statement.

Withdrawn/Retirée

12.23 THE STORE STATEMENT

12.23.1 FUNCTION

The STORE statement causes a record to be stored in the data base.

12.23.2 GENERAL FORMAT

```
STORE  record-name

[ ; RETAINING CURRENCY FOR { { { MULTIPLE } } } ]
                           { { { REALM    } } }
                           { { { SETS     } } }
                           { { { set-name-1 } ... } } }
                           { { { RECORD   } } }
```

12.23.3 SYNTAX RULES

1. The record type referenced by record-name must be defined in the Sub-schema.

2. The record referenced by record-name may only be stored in a realm that is in update usage mode. Any sets, of which this record becomes a member as the result of execution of a STORE statement, must also be stored in realms that are in update usage mode. All records that contain derived data items that are affected by execution of the STORE statement must also be stored in realms that are in update usage mode.

3. All sets referenced by the set selection criteria of the affected set types must be stored in realms that are in available mode.

12.23.4 GENERAL RULES

1. An explanation of Data Base Exception Conditions which may be caused by execution of a STORE statement is given under the appropriate paragraph. (See 12.4 Data Base Status Indicators.)

2. The record referenced by record-name becomes a member of each set type in which it has been declared to be an automatic member.

3. Execution of a STORE statement causes the record referenced by record-name to be established as the owner of an empty set for each set type in which it is defined as an owner.

4. The record referenced by record-name becomes a member of each set in accordance with the set selection criteria and the set ordering criteria defined for each affected set type.

Withdrawn/Retirée

5. Execution of a STORE statement causes the record referenced by record-name to become the current record of the run unit. The realm-name and record-name of the record are placed in the special registers DB-REALM-NAME and DB-RECORD-NAME, respectively.

6. If the RETAINING phrase is not specified, the record referenced by record-name becomes the current record of its realm-name, of its record-name, and of all set types in which it has been declared to be a tenant.

7. If the RETAINING phrase with the key word REALM is specified, no realm currency indicators are changed.

8. If the RETAINING phrase with the key word SETS is specified, no set type currency indicators are changed.

9. If the RETAINING phrase with the key word RECORD is specified, no record type currency indicators are changed.

10. If the RETAINING phrase and set-name-1 are specified, the currency indicators of set-name-1 are not changed.

11. If the RETAINING phrase with the key word MULTIPLE is specified, no realm, set type, or record type currency indicators are changed.

12. The record referenced by record-name remains available to the program in its associated record area.

13. Any realm data item associated with the record is used by the DBCS during execution of a STORE statement.

14. The DBCS assigns a unique value to the data base key associated with the record referenced by record-name. The user, by means of the record's location mode, can specify certain constraints to be placed on the DBCS in its assignment of a data base key value.

    a. If the record has no location mode, the DBCS uses an implementor-defined method of assigning a value.

    b. If the record has a set location mode, the DBCS uses an implementor-defined method of assigning a value. The DBCS assigns a data base key value as though the record was to become a member of a set of the set type associated with the record's location mode. Both the set ordering criteria and set selection criteria of the set type are used by the DBCS in assigning the data base key value.

    c. If the record has a direct location mode, the contents of the location mode data item associated with the record are used to assign a data base key value.

Withdrawn/Retirée

(1) If the location mode data item has the null data attribute the DBCS uses an implementor-defined method of assigning a value.

(2) If the location mode data item does not have the null data attribute and contains a value that does not specify an existing record in the data base, the DBCS assigns that value.

(3) If the location mode data item does not have the null data attribute and contains a value that does specify an existing record in the data base, the DBCS assigns the logically next value that does not specify an existing record in the data base.

d. If the record has a calculation location mode, the contents of the location mode data items associated with the record are used to assign a data base key value.

Withdrawn/Retirée

12.24 THE USE STATEMENT

12.24.1 FUNCTION

The USE statement is used to specify procedures to be executed when the execution of a statement results in a Data Base Exception Condition. It is also used to specify procedures that produce privacy keys for use by the DBCS.

12.24.2 GENERAL FORMAT

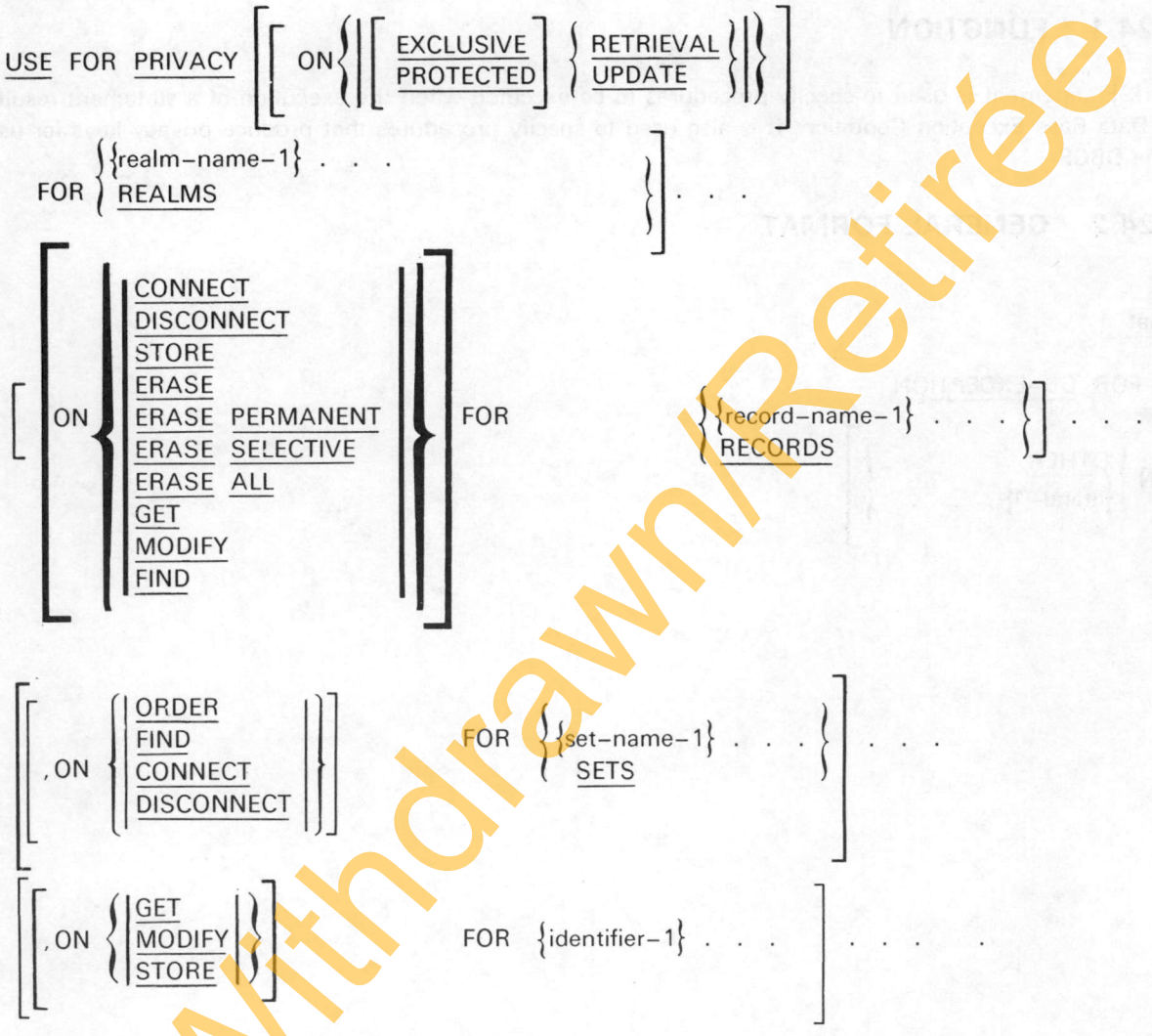
Format 1

```
USE FOR DB-EXCEPTION
[ ON { OTHER       } . . . ] .
     {literal-1}
```

Withdrawn/Retiree

USE

Format 2



12.24.3 SYNTAX RULES

ALL FORMATS

1. A USE statement, when present, must immediately follow a section header in the Declarative portion of the Procedure Division and must be followed by a separator period. The remainder of the section must consist of one or more procedural paragraphs that define the procedures to be used.

2. The USE statement itself is never executed; rather, it defines the conditions calling for the execution of the USE procedures.

Withdrawn/Retirée

3. Within a USE procedure, there must be no reference to any non-declarative procedure.

4. Within a non-declarative procedure there must be no reference to procedure-names that appear in a Declarative Section.

5. Statements appearing within a given Declarative Section may reference procedure-names defined within a different USE procedure only with a PERFORM statement.

FORMAT 1

6. Each literal must be a nonnumeric literal and must represent a Data Base Status Indicator. However, the rightmost five (5) characters of the the literal must not be '00000'.

7. Multiple occurrences of the USE FOR DB-EXCEPTION statement can exist within a program only if the ON phrase is specified in each. The literals in each USE statement must be unique; multiple USE statements must not cause the simultaneous request for execution of more than one USE declarative. The OTHER phrase may be specified only once in a program.

8. If a USE FOR DB-EXCEPTION statement is specified without the optional ON phrase, it must be the only occurrence of Format 1 of the USE statement in the program.

9. If some but not all the Data Base Status Indicators are represented by literals in USE FOR DB-EXCEPTION statements, then one statement must specify the OTHER phrase.

10. At least one Format 1 USE statement must be specified in any program that executes a Data Base Manipulation statement.

FORMAT 2

11. If a USE FOR PRIVACY statement is used, at least one of the optional FOR phrases must be specified.

12.24.4 GENERAL RULES

FORMAT 1

1. Appropriate values are placed in the special registers DB-STATUS, DB-REALM-NAME, DB-SET-NAME, DB-RECORD-NAME, and DB-CONFLICT by the DBCS prior to the execution of a data base exception section.

2. During the execution of any statement, control will be transferred to no more than one data base exception section. If the DBCS detects more than one Data Base Exception Condition during the execution of a statement, the data base exception section to which control is transferred is implementor-defined.

Withdrawn/Retirée

3. If no ON phrase is specified, or if the OTHER phrase is the only phrase specified in the program, the data base exception section is executed whenever the execution of a statement results in a Data Base Exception Condition. At the end of the execution of the data base exception section, if the Status Code associated with Data Base Exception Condition is '99000', control is returned to the system for abnormal termination; otherwise control is returned to the invoking routine.

4. If literals are specified and the execution of a statement results in a Data Base Exception Condition associated with a Status Code other than '99000', execution continues as follows:

   a. If the value of the resultant Data Base Status Indicator is equal to one of the specified literals, the procedures associated with that literal are executed.

   b. If the value of the resultant Data Base Status Indicator is not equal to one of the specified literals, the procedures associated with the OTHER phrase are executed.

5. If literals are specified and the execution of a statement results in a Data Base Exception Condition associated with a Status Code '99000', execution continues as follows:

   a. If the value of the resultant Data Base Status Indicator is equal to one of the specified literals, the procedures associated with that literal are executed and control is transferred to the system for abnormal termination.

   b. If the value of the resultant Data Base Status Indicator is not equal to one of the specified literals, the procedures associated with the OTHER phrase are executed and control is transferred to the system for abnormal termination.

FORMAT 2

6. If the ON phrase is not specified with any given FOR phrase, the privacy key produced by that privacy procedure applies to all phrases that could have been specified in that omitted phrase.

7. The execution of any given privacy procedure is caused by the execution of a READY statement which affects any realms, operations, record types, set types, or data items with which that privacy procedure is associated.

8. The value contained in the special register DB-PRIVACY-KEY at the completion of the execution of the privacy procedure is the privacy key associated with the realms, record types, set types, or data items with which that privacy procedure is associated.

Withdrawn/Retirée

CHAPTER 13

COBOL RESERVED WORDS

With the sole exception of the reserved words listed below and the special-character words, all reserved words will contain at least two characters and consist of only the characters A through Z and hyphen. Following is the list of COBOL reserved words:

ACCEPT
ACCESS
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALPHABET
ALPHABETIC
ALPHANUMERIC
ALPHANUMERIC-EDITED
ALSO
ALTERNATE
AND
ANY
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
BEFORE
BEGINNING
BIT
BITS
BLANK
BLOCK
BOOLEAN
BOTTOM
BY
CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMUNICATION
COMP
COMP-n
COMPUTATIONAL
COMPUTATIONAL-n
COMPUTE
CONFIGURATION
CONNECT
CONTAINS
CONTROL
CONTROLS
CONVERSION
CONVERTING
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
CURRENT
DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DAY-OF-WEEK
DB
DB-CONFLICT
DB-EXCEPTION
DB-KEY
DB-PRIVACY-KEY
DB-REALM-NAME
DB-RECORD-NAME
DB-SET-NAME
DB-STATUS
DE
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LENGTH
DEBUG-LINE
DEBUG-NAME
DEBUG-NUMERIC-CONTENTS
DEBUG-SIZE
DEBUG-START
DEBUG-SUB
DEBUG-SUB-ITEM
DEBUG-SUB-N
DEBUG-SUB-NUM
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISCONNECT
DISPLAY
DISPLAY-n
DIVIDE
DIVISION
DOWN
DUPLICATE
DUPLICATES
DYNAMIC
EGI
ELSE
EMI
EMPTY
ENABLE
END
END-OF-PAGE
ENDING
ENTER
ENVIRONMENT
EOP
EQUAL
EQUALS
ERASE
ERROR
ESI
EVERY
EXCEEDS
EXCEPTION
EXCLUSIVE
EXIT
EXOR
EXTEND
FD
FILE
FILE-CONTROL
FILES
FILLER
FINAL
FIND
FINISH
FIRST
FOOTING
FOR
FREE
FROM
GENERATE
GET
GIVING
GO
GREATER
GROUP
HEADING
HIGH-VALUE

Withdrawn/Retirée

HIGH-VALUES

I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INCLUDING
INDEX
INDEX-n
INDEXED
INDICATE
INITIAL
INITIALIZE
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS
JUST
JUSTIFIED
KEEP
KEY
LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCALLY
LOCK
LOW-VALUE
LOW-VALUES
MEMBER
MEMBERS
MEMBERSHIP
MEMORY
MERGE
MESSAGE
MODE
MODIFY
MODULES
MOVE
MULTIPLE
MULTIPLY
NATIVE
NEGATIVE
NEXT
NO
NON-NULL
NOT
NULL
NUMBER
NUMERIC
NUMERIC-EDITED
OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
ONLY
OPEN
OPTIONAL
OR
ORDER
ORGANIZATION
OTHER
OUTPUT
OVERFLOW
OWNER
PADDING
PAGE
PAGE-COUNTER
PERFORM
PERMANENT
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PRIOR
PRIVACY
PROCEDURE
PROCEDURES

PROGRAM
PROGRAM-ID
PROTECTED
PURGE
QUEUE
QUOTE
QUOTES
RANDOM
RD
READ
READY
REALM
REALMS
REALM-NAME
RECEIVE
RECORD
RECORD-NAME
RECORDS
REDEFINES
REEL
REFERENCE-MODIFIER
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMONITOR
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETAINING
RETRIEVAL
RETURN
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

SAME
SD
SEARCH
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SELECTIVE
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-MERGE
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-1
STANDARD-2
START
STATUS
STOP
STORE
STRING
SUB-QUEUE-n
SUB-SCHEMA
SUBTRACT
SUM
SUPPRESS

Withdrawn/Retiree

SUSPEND	THROUGH	UPDATE	WORKING-STORAGE
SYMBOLIC	THRU	UPON	WRITE
SYNC	TIME	USAGE	ZERO
SYNCHRONIZED	TIMES	USAGE-MODE	ZEROES
TABLE	TO	USE	ZEROS
TALLYING	TOP	USING	+
TAPE	TRAILING	VALUE	-
TENANT	TYPE	VALUES	*
TERMINAL	UNEQUAL	VARYING	/
TERMINATE	UNIT	WHEN	**
TEXT	UNSTRING	WITH	>
THAN	UNTIL	WITHIN	<
THEN	UP	WORDS	=

Withdrawn/Retirée

SECTION IV: COBOL

SUB–SCHEMA LANGUAGE SPECIFICATIONS

Withdrawn/Retirée

TABLE OF CONTENTS

SECTION IV: COBOL SUB-SCHEMA LANGUAGE SPECIFICATIONS

Page

CHAPTER 1. INTRODUCTION

1.1 Objectives of Section IV IV-1-1
1.2 Organization of Section IV IV-1-1
1.3 Notation Used in Formats and Rules IV-1-1

CHAPTER 2. CONCEPTS

2.1 Character Set and Language Structure IV-2-1
2.2 The Schema IV-2-1
2.3 The Sub-schema IV-2-1
2.4 Interaction of the Schema and the Sub-schema IV-2-2
2.4.1 Data Item Validation IV-2-2
2.4.2 Data Transformation IV-2-2
2.4.3 Data Mapping IV-2-2
2.4.4 Renaming Data-names IV-2-4
2.4.5 Schema and Sub-schema Privacy Locks IV-2-4
2.5 Derived Data Items IV-2-5
2.5.1 Actual Derived Data Items IV-2-5
2.5.2 Virtual Derived Data Items IV-2-5
2.5.3 Modification of Actual Derived Data Items IV-2-5

CHAPTER 3. THE COBOL SUB-SCHEMA LIBRARY

3.1 General Description IV-3-1
3.1.1 Sub-schema Organization IV-3-1

CHAPTER 4. THE COBOL SUB-SCHEMA

4.1 Sub-schema Structure IV-4-1
4.1.1 General Format IV-4-1
4.1.2 The Title Division IV-4-1
4.1.3 The Mapping Division IV-4-2
4.1.3.1 The Alias Section IV-4-2
4.1.4 The Structure Division IV-4-2
4.1.4.1 The Realm Section IV-4-2
4.1.4.2 The Set Section IV-4-2
4.1.4.3 The Record Section IV-4-2
4.2 The Sub-schema Description — Complete Entry Skeleton IV-4-4
4.3 The Alias Description — Complete Entry Skeleton IV-4-5
4.4 The Realm Description — Complete Entry Skeleton IV-4-7
4.5 The Set Description — Complete Entry Skeleton IV-4-8

Withdrawn/Retirée

TABLE OF CONTENTS

(Continued)

		Page
4.6	The Record Description — Complete Entry Skeleton	IV-4-10
4.7	The Data Description — Complete Entry Skeleton	IV-4-12
4.8	The PRIVACY KEY Clause	IV-4-15
4.9	The PRIVACY LOCK Clause	IV-4-16
4.10	The SET SELECTION Clause	IV-4-19
4.11	The WITHIN Clause	IV-4-22

CHAPTER 5. COBOL SUB-SCHEMA RESERVED WORDS IV-5-1

Withdrawn/Retired

CHAPTER I

INTRODUCTION

1.1 OBJECTIVES OF SECTION IV

The language specifications contained in this section are intended to serve the purposes of both the language implementor and the language user. While each of these audiences has different needs and sees the language specifications from different viewpoints, the specifications must nevertheless define complete and precise methods of specifying the intent of the user and the results generated by the implementor. The general formats and syntax rules are a precise definition of how to specify intent, while the general rules are a precise definition of the results generated by application of a specific syntax.

1.2 ORGANIZATION OF SECTION IV

Unlike Section III, this section of the manual is not intended to be a stand-alone document, but must be used with Section III in order to define the COBOL Sub-schema Language completely.

The organization of Section IV roughly parallels that of Section III. Chapter 4, The COBOL Sub-schema, of this section is organized similarly to Chapter 6, The Data Division, of Section III.

1.3 NOTATION USED IN FORMATS AND RULES

The notation used within formats and rules in this section is the same as that used in Section III. The rules and definitions applicable to this notation for this section are those stated in Section III. (See Section III, 1.4, Notation Used in Formats and Rules.)

Withdrawn/Retired

CHAPTER 2

CONCEPTS

2.1 CHARACTER SET AND LANGUAGE STRUCTURE

The character set, rules, and definitions applicable to Section IV are those stated in Section III. (See Section III, 3.1, Character Set.)

The language structure rules and definitions applicable to Section IV are those stated in Section III. (See Section III, 3.2, Language Structure.)

2.2 THE SCHEMA

A Schema is the logical description of a data base. It describes the logical data items contained in the data base, as well as the logical relationships that exist in the data base by which data items are grouped into records, sets, and realms. There is one Schema, and only one Schema, per data base.

The language specifications for the Schema are not a part of this document. Because the Schema is a logical description of data, it is independent of the language used to access the data, and thus bears no direct relationship to the specifications of either the COBOL or COBOL Sub-schema Languages. The Schema Language specifications are contained in the CODASYL Data Description Language Journal of Development.

2.3 THE SUB-SCHEMA

A Sub-schema is the logical description of that portion of a data base that a program can access. There can be an arbitrary number of Sub-schemas for a data base, and each Sub-schema can describe as much or as little of the data base as needed. Sub-schema descriptions can overlap each other; that is, a given portion of the data base can be described by an arbitrary number of Sub-schemas. Each run unit that accesses a data base must contain one, and only one, object Sub-schema.

Withdrawn/Retirée

2.4 INTERACTION OF THE SCHEMA AND THE SUB-SCHEMA

The Schema describes an entire data base as it logically exists in external storage. The Sub-schema describes a portion of a data base as it is to be logically made available to a program. Thus, all the data described in a Sub-schema must be described in the Schema for that data base. However, there can be certain differences between the Schema and Sub-schema descriptions of the same data, and therefore differences between the data as stored in the external storage device and data as the program handles it. The DBCS, using the Schema and the Sub-schema, performs the necessary actions to convert the data from one form to another as data is retrieved or stored by the program. This conversion process does not occur in retrieving and storing data in non-data base files. In such files, the data is accessed with no conversion at all, regardless of the record descriptions in the File Section of the Data Division.

2.4.1 DATA ITEM VALIDATION

As the DBCS retrieves and stores data, it may perform certain checks upon the contents of the data items involved. Depending on the specifications in the Schema and/or the Sub-schema, the DBCS can determine whether the value of the contents of a data item falls within a permissible range. Other checks can be performed by user-specified routines to which the DBCS transfers control at specified times during the retrieval or storage of data in the data base.

2.4.2 DATA TRANSFORMATION

Data is logically stored in the data base in accordance with its Schema description. When retrieved, the DBCS converts the data to the format specified in the Sub-schema description for that data item. When the DBCS stores data in the data base, the process is reversed. In both cases, each data item must contain a value consistent with its description.

If the description of a data item in the Schema does not specify a transformation procedure, the conversion of data occurs for that data item in accordance with the rules of the MOVE statement for elementary sending and receiving data items. If such a transformation procedure is specified, the conversion of data is done by that procedure. In either case, no data item may be described in the Schema and the Sub-schema in a manner that would prevent the data from being stored in the associated record area in a format consistent with its Sub-schema description.

2.4.3 DATA MAPPING

Because a Sub-schema describes only a portion of a data base, it is necessary that the DBCS be able to determine which data items are to be retrieved and stored. This determination is made on the basis of matching data-names and record-names in the Schema and the Sub-schema.

2.4.3.1 Matching of Data-names and Record-names

In order for the DBCS to access a given data item, the description of the data item in the Schema and Sub-schema must:

Withdrawn/Retired

a. Specify the same data-name, and

b. Be defined within a record with the same record-name, and

c. Specify an elementary data item.

For purposes of matching, if either the record-name or the data-name of the data item, or both, in the Schema is the subject of the Alias Description entry; the name specified as the object of that Alias Description entry is equivalent to the name that appears in the Schema. If an elementary data item can be explicitly referenced in the Sub-schema by more than one data-name, then one and only one of these data-names may match a data-name in the Schema. The data-names of all elementary data items defined in the Sub-schema must match data-names defined in the Schema.

2.4.3.2 COBOL References to Data-names

A COBOL program that accesses a data base references the data-names defined in the Sub-schema, and access to such data is in accordance with the description of the data items in the Sub-schema.

2.4.3.3 Mapping of Non-Table Elements

If the data-name of a non-table element in the Sub-schema matches the data-name of a non-table element in the Schema, data is moved between the data base and the record area as if the matching data-names had been the operands of a MOVE statement, unless a transformation procedure is specified in the Schema for that data item. In that case, the transformation procedure determines the mapping.

2.4.3.4 Mapping of Table Elements

If the data-name of a table element in the Sub-schema matches the data-name of a table element in the Schema, data is moved between the data base and the record area as if the matching data-names had been the operands of a series of MOVE statements, unless a transformation procedure is specified in the Schema for that data item. In that case, the transformation procedure determines the mapping.

The series of hypothetical MOVE statements behaves as follows:

1. The first occurrence of the table element is moved between the data base and the record area as though the hypothetical MOVE statement had specified the data-name of the table element subscripted by values that indicate the first occurrence of the table element.

2. Further hypothetical MOVE instructions are executed, each one successively increasing the subscript values to indicate the next occurrence of the table element.

Withdrawn/Retirée

3. If the table element is not defined as a variable occurrence data item, and:

   a. The number of occurrences of the table element is defined to be the same in both the Schema and the Sub-schema, the hypothetical MOVE statements continue until all occurrences of the table element have been moved.

   b. The number of occurrences of the table element is defined to be different in the Schema and the Sub-schema, the hypothetical MOVE statements continue until the lesser number of occurrences has been moved.

4. If the table element is defined as a variable occurrence data item, the current numbers of occurrences in the data base and in the record area are treated as the defined numbers of occurrences, and the hypothetical MOVE statements behave as defined in paragraph 3 above.

2.4.4 RENAMING DATA-NAMES

Because the data-names defined in the Schema may not be desirable to use in the Sub-schema, a facility is provided to assign aliases to data-names defined in the Schema. If a data-name defined in the Schema is the subject of an Alias Description entry in the Sub-schema, the DBCS behaves as though the name of the object of that Alias Description entry had been used in the Schema instead.

2.4.5 SCHEMA AND SUB-SCHEMA PRIVACY LOCKS

In both the Schema and the Sub-schema, privacy locks may be declared to be associated with the Schema or Sub-schema itself, or with individual realms, set types, record types, or data items.

If a given realm, set type, record type, or data item has a privacy lock declared for it in either the Schema or the Sub-schema, but not both, that privacy lock must be satisfied by an appropriate privacy key before the DBCS will unlock that resource. If a privacy lock for a given realm, set type, record type, or data item has a privacy lock declared for it in both the Schema and the Sub-schema, the privacy lock specified in the Schema is ignored and only the privacy lock in the Sub-schema is operative.

Withdrawn/Retirée

2.5 DERIVED DATA ITEMS

A derived data item is one whose contents are generated by the DBCS rather than by the COBOL object program. There are two types of derived data items, actual derived data items and virtual derived data items. Derived data items are defined as such in the Schema.

2.5.1 ACTUAL DERIVED DATA ITEMS

An actual derived data item is a data item whose contents are generated by the DBCS and stored as a part of the data base record in which the data item is defined. The data item is a part of the record in the data base as long as the record exists in the data base: The DBCS generates the value of an actual derived data item during execution of a STORE statement for the associated record, and during the execution of any statement that modifies the record or its relationships with other records. In addition, the DBCS modifies the data item whenever the data base is modified in any way that affects the data item's contents.

2.5.2 VIRTUAL DERIVED DATA ITEMS

A virtual derived data item is a data item whose contents are generated by the DBCS during execution of a GET statement for the record in which the data item is stored. A virtual derived data item does not exist in the data base, but exists only in the record area associated with the relevant record type.

2.5.3 MODIFICATION OF ACTUAL DERIVED DATA ITEMS

When a record that contains an actual derived data item is stored in the data base, the DBCS, using the criteria defined in the Schema, generates the contents of the data item and stores it as a part of that data base record. The DBCS then ensures that the contents of the data item are modified whenever necessary to maintain valid contents, based on the criteria defined in the Schema.

There are three conditions under which the DBCS must modify the contents of an existing actual derived data item:

1. Whenever the record in which the data item is stored is modified in the data base, or
2. Whenever the record relationship of the record in which the data item is stored is modified in the data base, or
3. Whenever a different record or a different record's relationships are modified in the data base in such a manner as to affect the criteria for generating the contents of the actual derived data item.

Whenever an actual derived data item must be modified by the DBCS for any reason, the record in which the data item is stored must be in update usage mode.

Withdrawn/Retirée

CHAPTER 3

THE COBOL SUB–SCHEMA LIBRARY

3.1 GENERAL DESCRIPTION

The COBOL Sub-schema Library contains the descriptions of the portions of a data base that are to be accessed by COBOL programs. The source Sub-schemas are placed in the COBOL Sub-schema Library in an implementor-defined format. When a COBOL program requires access to a portion of a data base, that program specifies the name of a specific Sub-schema that resides in the COBOL Sub-schema Library. That Sub-schema is then made available to the COBOL program for use by the DBCS in retrieving and storing data.

3.1.1 SUB-SCHEMA ORGANIZATION

The COBOL Sub-schema Library can contain an arbitrary number of Sub-schemas. Each Sub-schema consists of three divisions: the Title Division, the Mapping Division, and the Structure Division.

Withdrawn/Retired

CHAPTER 4

THE COBOL SUB–SCHEMA

4.1 SUB-SCHEMA STRUCTURE

The following gives the general format of the divisions in the Sub-schema, and defines the order of their presentation in the source Sub-schema.

4.1.1 GENERAL FORMAT

```
TITLE DIVISION

sub–schema–description–entry

MAPPING DIVISION.

[ALIAS SECTION.
   [alias–description–entry]  . . .]

STRUCTURE DIVISION.

[REALM SECTION.
   [realm–description–entry]  . . .]

[SET SECTION.
   [set–description–entry]  . . .]

[RECORD SECTION.
   [record–description–entry]  . . .]
```

4.1.2 THE TITLE DIVISION

The Title Division specifies the sub-schema-name and the schema-name associated with this Sub-schema. It can also specify the values of Privacy locks that must be satisfied before the Sub-schema is made available for use by a program. If the Schema referenced by the Sub-schema has a privacy lock associated with its use, the Title Division specifies the value of the privacy key to be matched against that privacy lock.

Only one Sub-schema Description entry may be specified in a Sub-schema.

Withdrawn/Retirée

SUB–SCHEMA STRUCTURE

4.1.3 THE MAPPING DIVISION

The Mapping Division provides the ability to define the logical relationship between Schema definitions and Sub-schema definitions.

4.1.3.1 The Alias Section

The Alias Section provides the ability to specify aliases by which names defined in the Schema are to be known in the Sub-schema, and thus by the COBOL programs referencing the Sub-schema.

There is one Alias Description entry for each name defined in the Schema for which an alias is required in the Sub-schema.

4.1.4 THE STRUCTURE DIVISION

The Structure Division specifies the content and structure of the data base that are to be made known to a COBOL program.

4.1.4.1 The Realm Section

The Realm Section specifies the realms that can be made available to a COBOL program. It may also define privacy lock values to be associated with the use of the realm by a COBOL program.

One Realm Description entry must be specified for each realm that is to be made available to a COBOL program. However, only one Realm Description entry is required in a Sub-schema if all realms defined in the Schema can be made available to a program using that Sub-schema.

4.1.4.2 The Set Section

The Set Section specifies the set types that are to be made available to a COBOL program. It may also define privacy lock values to be associated with the use of the set type by a COBOL program. In addition, the Set Section can define the set selection criteria to be used to select the specific set of each defined set type.

One Set Description entry must be specified for each set type that is to be made available to a COBOL program. However, only one Set Description entry is required in a Sub-schema if all set types defined in the Schema can be made available to a program using that Sub-schema.

4.1.4.3 The Record Section

The Record Section specifies the record types to be made available to a COBOL program. It may specify that only records stored in specific realms are to be made available, and it may specify the values of privacy locks that are to be associated with use of the record types.

Withdrawn/Retirée

SUB–SCHEMA STRUCTURE

The Record Description entry need not specify an entire record as described in the Schema, nor need the Record Description entry specify data items within the record in the same order in which the Schema describes them.

There is one Record Description entry for each record type to be accessed by a COBOL program.

Withdrawn/Retirée

SS SKELETON

4.2 THE SUB-SCHEMA DESCRIPTION — COMPLETE ENTRY SKELETON

4.2.1 FUNCTION

The Sub-schema Description specifies the name of the Sub-schema, identifies the Schema to which the Sub-schema applies, and may specify privacy lock values associated with the use of the Sub-schema.

4.2.2 GENERAL FORMAT

```
SS  sub–schema–name  WITHIN  schema–name

[ ; PRIVACY LOCK  [ FOR { INVOKING | ALTERING | LOCKS } ]  IS { literal–1 | implementor–name–1 | routine–name–1 } ]  . . .

[ ; PRIVACY KEY IS { literal–2 | implementor–name–2 | routine–name–2 } ]  .
```

4.2.3 SYNTAX RULES

1. There must be one, and only one, Sub-schema Description entry specified in a Sub-schema;
2. Sub-schema-name must conform to the rules for formation of a data-name.
3. The order of appearance of the clauses that follow schema-name is immaterial.
4. Schema-name must reference a Schema known to the DBCS.
5. The combination of sub-schema-name and schema-name must appear only once in a COBOL Sub-schema Library.

4.2.4 GENERAL RULES

1. Sub-schema-name is the name associated with the Sub-schema.
2. Schema-name identifies the Schema with which the Sub-schema is to be associated.

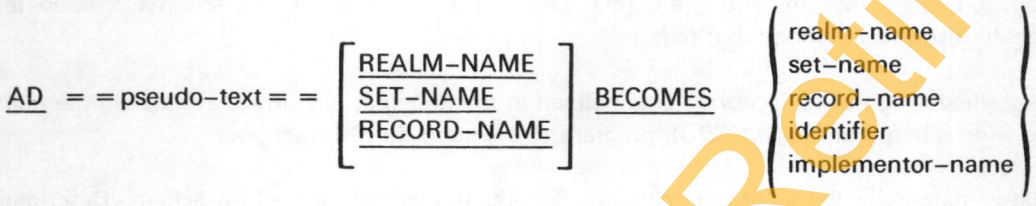
Withdrawn/Retirée

4.3 THE ALIAS DESCRIPTION — COMPLETE ENTRY SKELETON

4.3.1 FUNCTION

The Alias Description assigns aliases to names defined in the Schema.

4.3.2 GENERAL FORMAT



4.3.3 SYNTAX RULES

1. Pseudo-text must reference a name defined in the Schema referenced by the Sub-schema Description entry.

2. If pseudo-text references a realm-name, set-name, or record-name not unique in the Schema referenced by the Sub-schema Description entry, the REALM-NAME, SET-NAME, or RECORD-NAME phrase must be specified.

3. A realm, set type, record type, data item, or implementor-name defined in the Schema referenced by the Sub-schema Description entry may be the subject of only one Alias Description entry.

4. Realm-name, set-name, record-name, identifier, and implementor-name must be used to reference a realm, set type, record type, data item, or implementor-name, respectively, defined in the Schema referenced by the Sub-schema Description entry.

5. Identifier must not be reference modified.

4.3.4 GENERAL RULES

1. If specified, realm-name becomes the alias by which the realm referenced by pseudo-text is known, both to the Sub-schema and to a COBOL program referencing the Sub-schema. Pseudo-text cannot then be used to reference that realm.

2. If specified, set-name becomes the alias by which the set type referenced by pseudo-text is known, both to the Sub-schema and to a COBOL program referencing the Sub-schema, Pseudo-text cannot then be used to reference that set type.

Withdrawn/Retirée

AD SKELETON

3. If specified, record-name becomes the alias by which the record type referenced by pseudo-text is known, both to the Sub-schema and to a COBOL program referencing the Sub-schema. Pseudo-text cannot then be used to reference that record type.

4. If specified, identifier becomes the alias by which the data item referenced by pseudo-text is known, both to the Sub-schema and to a COBOL program referencing the Sub-schema. Pseudo-text cannot then be used to reference that data item.

5. If specified, implementor-name becomes the alias by which the feature referenced by pseudo-text is known, both to the Sub-schema and to a COBOL program referencing the Sub-schema. Pseudo-text cannot then be used to reference that feature.

6. The alias specified does not replace the name defined in the Schema, but serves as a substitute name only in the Sub-schema and in a COBOL program referencing the Sub-schema.

7. Pseudo-text is matched with names defined in the Schema referenced by the Sub-schema Description entry as follows:

   Starting with the leftmost schema-name that is the subject of an entry, pseudo-text is compared to an equivalent number of contiguous Schema names.

   Pseudo-text matches schema-name if, and only if, the ordered sequence of text-words that forms pseudo-text is equal, character for character, to the ordered sequence of Schema-names. For purposes of matching, each occurrence of a separator comma or semicolon in schema-name or in pseudo-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 successive schema-name.

Withdrawn/Retired

4.4 THE REALM DESCRIPTION — COMPLETE ENTRY SKELETON

4.4.1 FUNCTION

The Realm Description specifies the realms that are to be made available to the Sub-schema and a COBOL program. It may also specify privacy lock values to be associated with the use of the realms.

4.4.2 GENERAL FORMAT

```
RD    { ALL          }
      { realm–name   }

[ ; PRIVACY LOCK   [ FOR { [ EXCLUSIVE ]  { RETRIEVAL } } ] ]
                         { [ PROTECTED ]  { UPDATE    } }

    IS { literal–1              } [ . . . ]
       { implementor–name–1     }
       { routine–name–1         }
```

4.4.3 SYNTAX RULES

1. Realm-name must reference a realm defined in the Schema referenced by the Sub-schema Description entry.

2. If the ALL clause is specified, no other RD entry may appear in the Sub-schema.

4.4.4 GENERAL RULES

1. If ALL is specified, every realm defined in the Schema referenced by the Sub-schema Description entry is made available to the Sub-schema and to a COBOL program referencing the Sub-schema.

2. If realm-name is specified, the realm referenced by realm-name that is defined in the Schema referenced by the Sub-schema Description entry is made available to the Sub-schema and to a COBOL program referencing the Sub-schema.

Withdrawn/Retiree

SD SKELETON

4.5 THE SET DESCRIPTION — COMPLETE ENTRY SKELETON

4.5.1 FUNCTION

The Set Description specifies the set types that are to be made available to the Sub-schema and a COBOL program, and may specify the algorithm to be used by the DBCS to distinguish between sets of the set type. It may also specify privacy lock values to be associated with the use of the sets.

4.5.2 GENERAL FORMAT

```
SD { ALL          }
   { set-name-1   }
[ ; SET SELECTION FOR record-name IS

  VIA set-name-1 OWNER { SYSTEM                                          }
                       { CURRENT                                         }
                       { identifier-1                                    }
                       { VALUE OF identifier-2 IS identifier-3           }
                       [ , identifier-4 IS identifier-5 ] . . .          }

[ VIA set-name-2 OWNER VALUE OF identifier-6 IS identifier-7
    [ , identifier-8 IS identifier-9 ] . . . ] . . . ] . . .

[ [ ; PRIVACY LOCK [ FOR { CONNECT    } ] ] IS { literal-1          } ] . . . .
                         { DISCONNECT }       { implementor-name-1 }
                         { FIND       }       { routine-name-1     }
                         { ORDER      }
```

4.5.3 SYNTAX RULES

1. Set-name-1 must reference a set type defined in the Schema referenced by the Sub-schema Description entry.

2. If the ALL clause is specified, no other SD entry may appear in the Sub-schema.

Withdrawn/Retiree

4.5.4 GENERAL RULES

1. If ALL is specified, all set types defined in the Schema referenced by the Sub-schema Description entry are made available to the Sub-schema and to a COBOL program referencing the Sub-schema.

2. If set-name-1 is specified, the set type referenced by set-name-1 that is defined in the Schema referenced by the Sub-schema Description entry is made available to the Sub-schema and to a COBOL program referencing the Sub-schema.

Withdrawn/Retiree

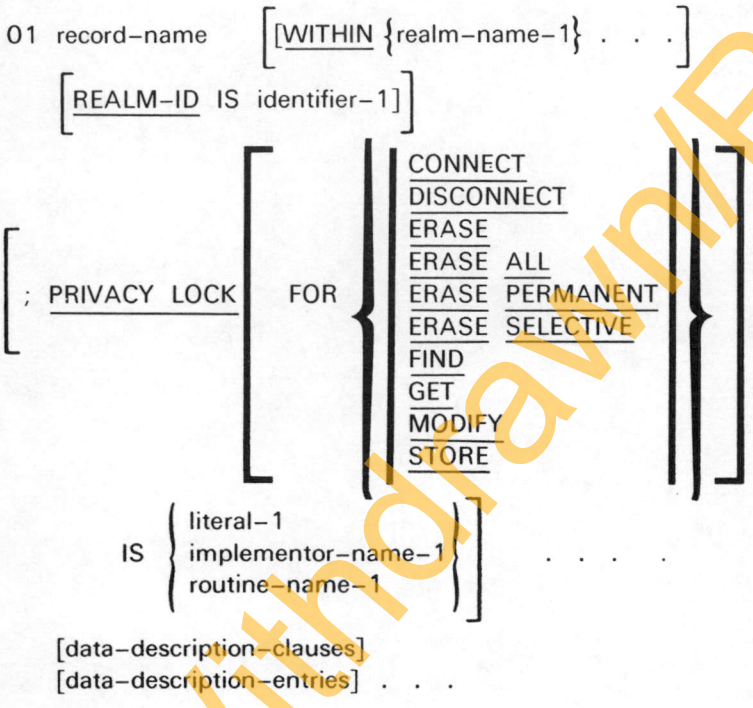
RECORD DESCRIPTION SKELETON

4.6 THE RECORD DESCRIPTION — COMPLETE ENTRY SKELETON

4.6.1 FUNCTION

The Record Description specifies the record types that are to be made available to the Sub-schema and a COBOL program. It specifies the data items within the data base record that are to be made available, specifies their relative order within the record, and specifies the format of the data item within the object Sub-schema and COBOL program. It may also specify privacy lock values to be associated with the use of the records.

4.6.2 GENERAL FORMAT



4.6.3 SYNTAX RULES

1. Record-name must reference a record type defined in the Schema referenced by the Sub-schema Description entry.

2. Record-name must reference a record type that may be contained in at least one of the realms referenced by an RD entry.

Withdrawn/Retired

4.6.4 GENERAL RULES

1. Records of the type referenced by record-name are made available to the Sub-schema and to a COBOL program referencing the Sub-schema.

Withdrawn/Retirée

DATA DESCRIPTION SKELETON

4.7 THE DATA DESCRIPTION — COMPLETE ENTRY SKELETON

4.7.1 FUNCTION

The Data Description specifies the characteristics of a particular item of data.

4.7.2 GENERAL FORMAT

Format 1

```
level-number   [ data-name-1 ]
               [ FILLER      ]

[; REDEFINES data-name-2]

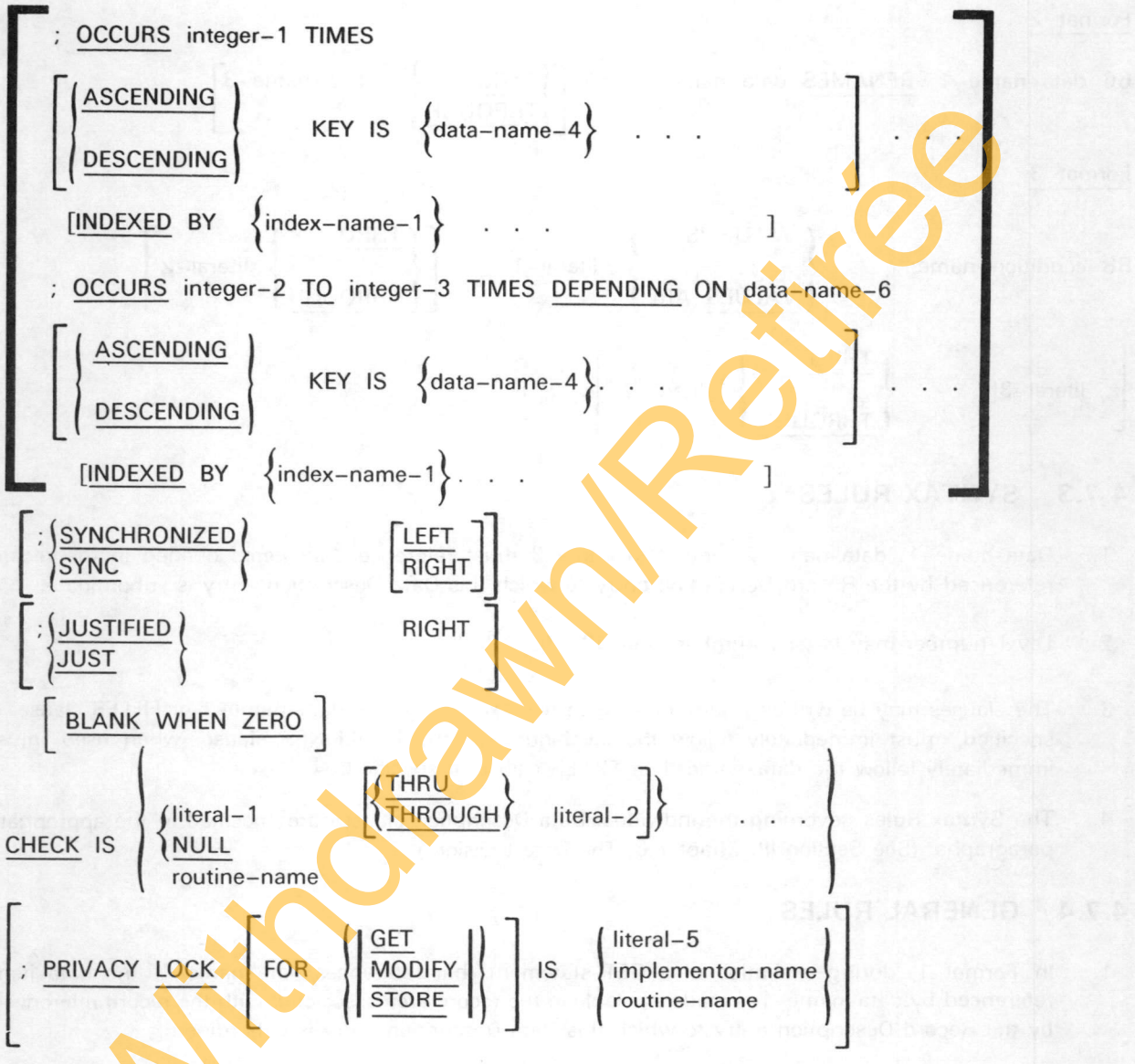
[ ; { PICTURE } IS character-string [DEPENDING ON identifier] ]
    { PIC     }

[ ; [USAGE IS] { BIT                  } ]
               { COMPUTATIONAL        }
               { COMP                 }
               { COMPUTATIONAL-n      }
               { COMP-n               }
               { DB-KEY               }
               { DISPLAY              }
               { DISPLAY-n            }
               { INDEX                }
               { INDEX-n              }

[ ; [SIGN IS] { LEADING  } [SEPARATE CHARACTER] ]
              { TRAILING }
```

Withdrawn/Retirée

DATA DESCRIPTION SKELETON



Withdrawn/Retiree

DATA DESCRIPTION SKELETON

Format 2

```
66 data-name-1; RENAMES data-name-2 [ { THRU     } data-name-3 ] .
                                     { THROUGH }
```

Format 3

```
88 condition-name; { VALUE IS } literal-1 [ { THRU     } literal-2 ]
                   { VALUES ARE }          { THROUGH }

[ , literal-3[ { THRU     } literal-4] ] . . . .
               { THROUGH }
```

4.7.3 SYNTAX RULES

1. Data-name-1, data-name-2, and data-name-3 must reference data items defined in the record referenced by the Record Description entry to which this Data Description entry is subordinate.

2. Level-number may be any number from 02-49.

3. The clauses may be written in any order with two exceptions: the data-name-1 or FILLER clause, if specified, must immediately follow the level-number; the REDEFINES clause, when used, must immediately follow the data-name-1 or FILLER clause, if specified.

4. The Syntax Rules governing the individual Data Description clauses are specified in the appropriate paragraphs. (See Section III, Chapter 6, The Data Division.)

4.7.4 GENERAL RULES

1. In Format 1, during execution of a GET statement that references data-name-1, the data item referenced by data-name-1 is made available in the record area associated with the record referenced by the Record Description entry to which this Data Description entry is subordinate.

2. The relative order of the Data Description entries determines the relative order of the data items in the record area associated with the record referenced by the Record Description entry to which this Data Description entry is subordinate.

3. The General Rules governing the individual Data Description clauses are specified in the appropriate paragraphs. (See Section III, Chapter 6, The Data Division.)

Withdrawn/Retirée

4.8 THE PRIVACY KEY CLAUSE

4.8.1 FUNCTION

The PRIVACY KEY clause specifies the method of producing the privacy key to be associated with the process of placing a Sub-schema in a Sub-schema Library.

4.8.2 GENERAL FORMAT

PRIVACY KEY IS { literal / implementor–name / routine–name }

4.8.3 GENERAL RULES

1. The privacy key specified by the PRIVACY KEY clause is used during the process of placing the Sub-schema in a Sub-schema Library. If, and only if, that privacy key satisfies the privacy lock, if any, associated with that process, the Sub-sechema is placed in the Sub-schema Library.

2. If literal is specified, the value of literal represents the privacy key.

3. If implementor-name is specified, the value associated with the feature referenced by implementor-name represents the privacy key.

4. If routine-name is specified, the privacy key depends upon the nature of the routine referenced by routine-name.

Withdrawn/Retiree

4.9 THE PRIVACY LOCK CLAUSE

4.9.1 FUNCTION

The PRIVACY LOCK clause specifies privacy locks to be associated with the use of a Sub–schema, realm, set type, record type, or data item.

4.9.2 GENERAL FORMAT

Format 1

```
PRIVACY LOCK [ FOR { INVOKING } ] IS { literal–1          }
             [     { ALTERING }     { implementor–name–1 }
             [     { LOCKS    }     { routine–name–1     }
```

Format 2

```
PRIVACY LOCK [ FOR { [EXCLUSIVE] { RETRIEVAL } } ] IS { literal–2          }
             [     { [PROTECTED] { UPDATE    } }     { implementor–name–2 }
                                                     { routine–name–2     }
```

Format 3

```
PRIVACY LOCK [ FOR { CONNECT    } ] IS { literal–3          }
             [     { DISCONNECT }     { implementor–name–3 }
             [     { FIND       }     { routine–name–3     }
             [     { ORDER      }
```

Format 4

```
                  [ { CONNECT                    } ]
                  [ { DISCONNECT                 } ]
                  [ { ERASE                      } ]
                  [ { ERASE PERMANENT            } ]
PRIVACY LOCK [ FOR { ERASE SELECTIVE             } ] IS { literal–4          }
                  [ { ERASE ALL                  } ]     { implementor–name–4 }
                  [ { FIND                       } ]     { routine–name–4     }
                  [ { GET                        } ]
                  [ { MODIFY                     } ]
                  [ { STORE                      } ]
```

Withdrawn/Retirée

Format 5

```
PRIVACY LOCK [ FOR { [ GET ]   } ] IS { literal–5           }
               [ MODIFY ]               { implementor–name–5 }
               [ STORE  ]               { routine–name–5     }
```

4.9.3 SYNTAX RULES

1. Format 1 may be specified only in the Title Division.
2. Format 2 may be specified only in the Realm Section.
3. Format 3 may be specified only in the Set Section.
4. Formats 4 and 5 may be specified only in the Record Section.
5. Format 5 must not be specified in a level 01 entry.
6. Each literal must be nonnumeric.

4.9.4 GENERAL RULES

ALL FORMATS

1. The PRIVACY LOCK clause causes a privacy lock to be associated with a resource.
2. If a literal is specified, the value of the literal is the privacy lock associated with the resource.
3. If an implementor-name is specified, the value of the contents of the feature referenced by the implementor-name is the privacy lock associated with the resource.
4. If a routine-name is specified, the privacy lock associated with the resource is dependent upon the nature of the routine referenced by the routine name.
5. If the FOR phrase is not specified, the privacy locks that must be satisfied are those that would have been specified if the FOR phrase had been specified with all of its possible choices.
6. If the INVOKING phrase is specified, the privacy lock is an invocation privacy lock. The invocation privacy lock must be satisfied in order to execute a run unit containing this Sub-schema.
7. If the ALTERING phrase is specified, the privacy lock must be satisified in order to modify the relevant Sub-schema in the Sub-schema Library.

Withdrawn/Retirée

PRIVACY LOCK

8. If the LOCKS phrase is specified, the privacy lock must be satisfied in order to access any PRIVACY LOCK clause or PRIVACY KEY clause in the relevant Sub-schema in the Sub-schema Library.

FORMAT 2

9. If the EXCLUSIVE phrase is specified, the privacy lock must be satisfied in order to place the associated realm in exclusive usage mode.

10. If the PROTECTED phrase is specified, the privacy lock must be satisfied in order to place the associated realm in protected usage mode.

11. If the RETRIEVAL phrase is specified, the privacy lock must be satisfied in order to place the associated realm in retrieval usage mode.

12. If the UPDATE phrase is specified, the privacy lock must be satisfied in order to place the associated realm in update usage mode.

FORMAT 3

13. If the CONNECT, DISCONNECT, FIND, or ORDER phrase is specified, the privacy lock must be satisfied in order to execute a CONNECT, DISCONNECT, FIND, or ORDER statement, respectively, that references the associated set type.

FORMAT 4

14. If the CONNECT, DISCONNECT, ERASE, ERASE ALL, ERASE PERMANENT, ERASE SELECTIVE, FIND, GET, MODIFY, or STORE phrase is specified, the privacy lock must be satisfied in order to execute a CONNECT, DISCONNECT, ERASE, ERASE ALL, ERASE PERMANENT, ERASE SELECTIVE, FIND, GET, MODIFY or STORE statement, respectively, in which the associated record is an object record.

FORMAT 5

15. The General Rules governing Format 5 are defined in the appropriate paragraphs. (See Section III, 6.36, The PRIVACY LOCK Clause.)

Withdrawn/Retirée

4.10 THE SET SELECTION CLAUSE

4.10.1 FUNCTION

The SET SELECTION clause specifies the arguments to be used by the DBCS to select a specific set of a given set type.

4.10.2 GENERAL FORMAT

```
SET SELECTION FOR record-name IS

VIA set-name-1 OWNER { SYSTEM                                                    }
                     { CURRENT                                                   }
                     { identifier-1                                              }
                     { VALUE OF identifier-2 IS identifier-3[, identifier-4      }
                                                                                 }
                                             IS identifier-5] . . .              }

[VIA set-name-2 OWNER VALUE OF identifier-6 IS identifier-7[, identifier-8
                                             IS identifier-9] . . . ] . . .
```

4.10.3 SYNTAX RULES

1. The same record-name must not be specified more than once in the same SD entry.

2. Record-name must be defined as a member of the set type defined by the SD entry in which the SET SELECTION clause is specified.

3. At least one owner record of the set type referenced by set-name-2 must be a member of a set referenced by set-name-1. This relationship must exist between each successive pair of set-names specified.

4. The set-name specified in the last VIA phrase must reference the set type defined by the SD entry to which this clause is subordinate.

5. Identifier-1 must be defined as a data base key item.

6. The SYSTEM phrase must be specified if, and only if, set-name-1 references a singular set.

7. Identifier-2, identifier-4, . . . , must be defined in the Record Description entry of the record type defined as the owner of the set type referenced by set-name-1. These identifiers must not be reference modified.

Withdrawn/Retirée

8. Identifier-6, identifier-8, . . . , must be defined in the Record Description entry of the record type defined as the owner of the set type referenced by set-name-2. These identifiers must not be reference modified.

9. The contents of the data items referenced by identifier-3, identifier-5, . . . , must be values that are contained in the data items referenced by identifier-2, identifier-4, . . . , respectively, in one and only one owner record of a set referenced by set-name-1.

10. The contents of the data items referenced by identifier-7, identifier-9, . . . , must be values that are contained in the data items referenced by identifier-6, identifier-8, . . . , respectively, in one and only one owner record of a set referenced by set-name-2.

4.10.4 GENERAL RULES

1. The set selection criteria, specified in the Schema for the record type referenced by record-name in the set type defined in the SD entry to which this clause is subordinate, are replaced by the set selection criteria specified in this clause. All other set selection criteria specified in the Schema remain in effect.

2. The set selection criteria, for the record type referenced by record-name in the set type defined by the SD entry to which this clause is subordinate, are established according to the following rules.

3. Each VIA phrase specified selects the owner of a specific set of the set type referenced by the set-name specified in that phrase.

4. If the SYSTEM phrase is specified, the DBCS is the selected owner of the set referenced by set-name-1.

5. If the CURRENT phrase is specified, the owner selected is the owner of the set referenced by set-name-1 in which the current record of the set type referenced by set-name-1 is a tenant.

6. If identifier-1 is specified, the owner selected is the owner of the set referenced by set-name-1 whose data base key value is equal to the contents of the data item referenced by identifier-1.

7. If the VALUE OF phrase is specified in the first VIA phrase, and identifier-2, identifier-4, . . . , reference all the location mode data items associated with the record type referenced by record-name, and no other data items, the owner is selected by that location mode algorithm. In this case, the contents of the data items referenced by identifier-3, identifier-5, . . . , are used in that algorithm as the values of the location mode data items referenced by identifier-2, identifier-4, . . . , respectively.

8. If the VALUE OF phrase is specified in the first VIA phrase and identifier-2, identifier-4, . . . , do not reference all the location mode data items, if any, associated with the record type referenced by record-name, or if they reference any other data items, the owner selected is the record in the set referenced by set-name-1 in which the contents of the data items referenced by identifier-2,

Withdrawn/Retirée

identifier-4, . . . , are equal to the contents of the data items referenced by identifier-3, identifier-5, . . . , respectively, according to the rules for comparison of operands in a relation condition.

9. The only owners in set-name-2, . . . , that can be selected are those owners that are members of the set selected by the criteria defined in the VIA phrase that references set-name-1, . . . , respectively.

10. In the second and subsequent VIA phrases, the owner selected is the record in the set referenced by set-name-2 in which the contents of the data items referenced by identifier-6, identifier-8, . . . , are equal to the contents of the data items referenced by identifier-7, identifier-9, . . . , respectively, according to the rules for comparison of operands in a relation condition.

Withdrawn/Retirée

4.11 THE WITHIN CLAUSE

4.11.1 FUNCTION

The WITHIN clause specifies the realms in which a record type may be stored, and provides a means of differentiating between each such realm.

4.11.2 GENERAL FORMAT

[WITHIN realm–name–1 . . .] [REALM–ID IS identifier]

4.11.3 SYNTAX RULES

1. If records of the record type defined by the Record Description entry in which the WITHIN clause is specified may be stored in more than one realm, the REALM-ID phrase must be specified.

2. The data item referenced by identifier must be defined such that its contents may consist of a realm-name, but it must not be reference modified.

3. If the Schema description of the record type defined by this clause specifies that records of this record type may be stored in more than one realm, then realm-name-1 must reference realms in which records of this record type may be stored.

4. If the REALM-ID phrase is specified, the WITHIN clause must also be specified.

4.11.4 GENERAL RULES

1. If the WITHIN clause is not specified, records of the record type defined by this Record Description entry may be stored only in realms permitted by the Schema description of this record type.

2. If the WITHIN clause is specified, records of the record type defined by this Record Description entry may only be stored in the realms referenced by realm-name-1.

3. If the REALM-ID phrase is specified and, during the execution of a STORE statement that references the record type defined by this Record Description entry, the contents of the data item referenced by identifier are:

   a. null, then the DBCS selects the specific realm in which the object record is stored, according to the location mode associated with that record type.

   b. not null and equal to a realm-name, then the DBCS stores the object record in the realm referenced by that realm-name.

Withdrawn/Retirée

CHAPTER 5

COBOL SUB–SCHEMA RESERVED WORDS

All reserved words listed in Section III are also reserved words in the COBOL Sub-schema language.

In addition, the words in the following list are also reserved in the COBOL Sub-schema:

AD
ALIAS
ALTERING
BECOMES
CHECK
INVOKING
LOCKS
MAPPING
REALM–ID
SELECTION
SET–NAME
SS
STRUCTURE
TITLE
VIA

Withdrawn/Retirée