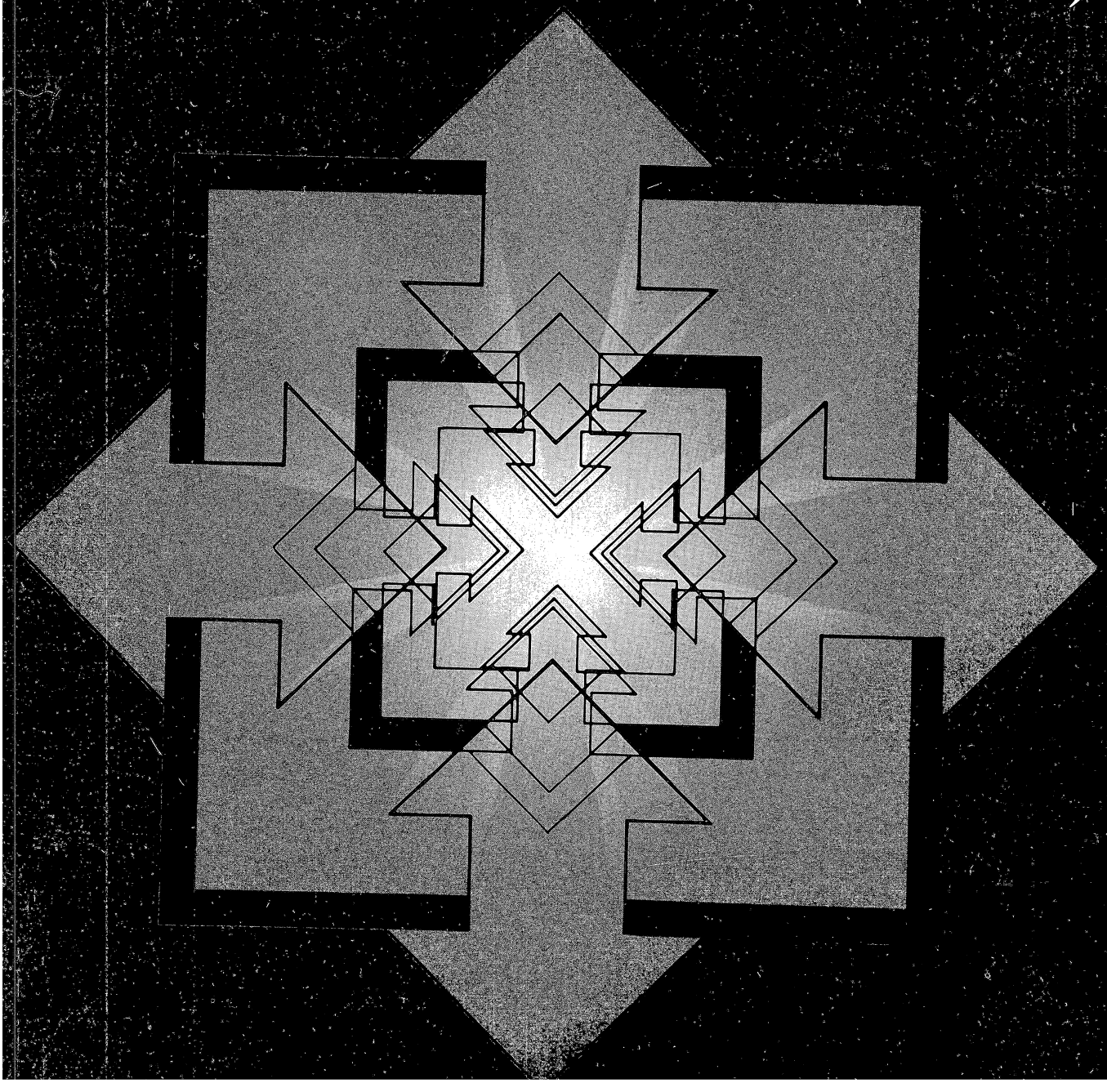
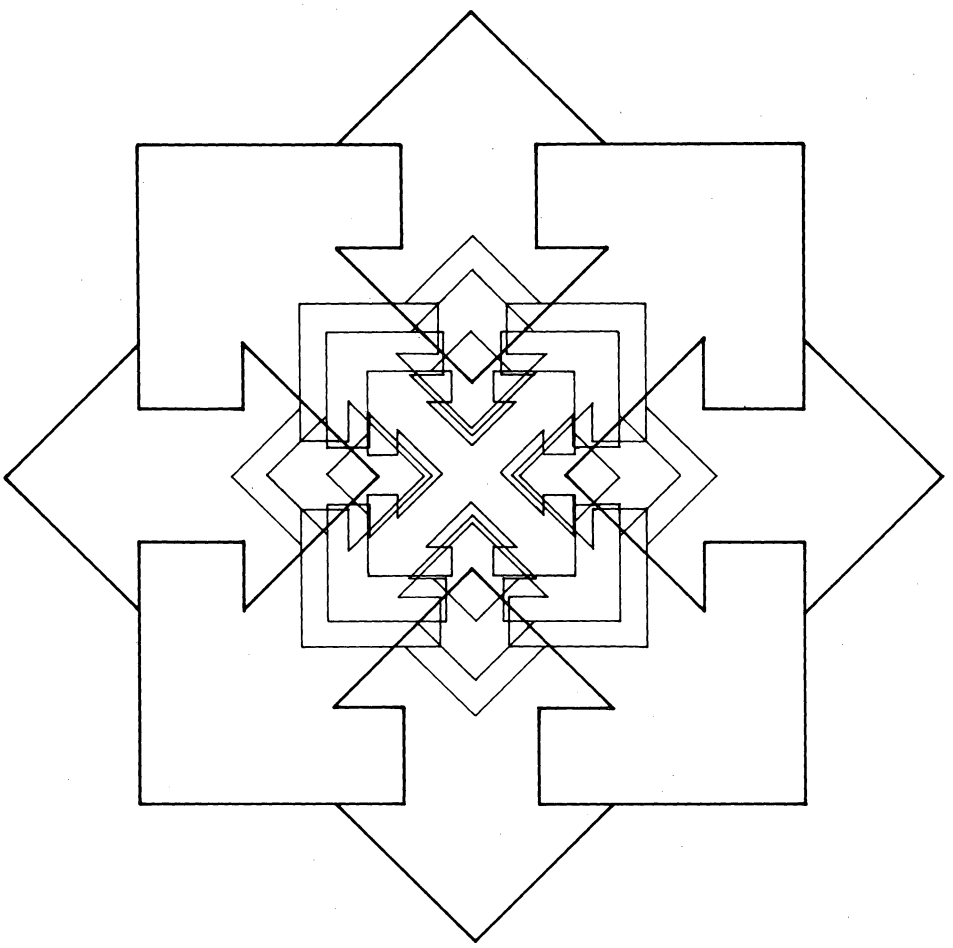
DataGeneral

Interactive COBOL

Interactive COBOL User's Guide (MP/OS)



Interactive COBOL User's Guide (MP/OS)



DataGeneral

069-705013-00

A Small Business Systems Publication

DATA GENERAL CORPORATION, Westboro, Massachusetts 01580

NOTICE

Data General Corporation (DGC) has prepared this document for use by DGC personnel, customers, and prospective customers. The information contained herein shall not be reproduced in whole or in part without DGC's prior written approval.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS DOCUMENT OR THE INFORMATION CONTAINED IN IT, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

Ordering Number 069-705013

Revision 00, May 1982

Printed in U.S.A.

Contents

Figures

Tables

Preface

How the Interactive COBOL Documentation Is Organized

We have followed a modular approach in documenting Data General's Interactive COBOL language. The modules, of which there are five, comprise one manual. Two of the modules are system-dependent; that is, they have been written for the particular operating system on which Interactive COBOL will be run: MP/OS, RDOS, or AOS. The remainder are system-independent; the information they contain is not restricted to a particular operating environment. The Interactive COBOL Programmer's Reference, for example, is a system-independent document; it is the nucleus around which the other modules have been written. Thus, no matter what your operating environment, the Interactive COBOL Programmer's Reference will be your primary resource document. The following table shows the modules you will need to run Interactive COBOL on a particular operating system.

MP/OS	RDOS	AOS
1A,2,3A,4,5	1B,2,3B,4,5	1C,2,3B,4,5

Module	Code	Title	Document Number
1	1A	MP/OS Interactive COBOL User's Guide	069-705013
	1B	RDOS Interactive COBOL User's Guide	069-705014
	1C	AOS Interactive COBOL User's Guide	069-705015
2	2	Interactive COBOL Programmer's Reference	093-705013
3	3A	MP/OS Interactive COBOL Utilities	069-705019
	3B	RDOS/AOS Interactive COBOL Utilities	069-705020
4	4	IC/EDIT: Interactive COBOL Editor	055-004-01
5	5	SCREEN: Screen Format Editor	055-006-01

A summary of the contents of Interactive COBOL documents relevant to the MP/OS operating environment is given in the Related Documents section, below.

Who the Manual Is For

The Interactive COBOL documentation set has been written for the experienced COBOL programmer; it is not designed as a tutorial. Knowledgeable COBOL programmers will benefit most from reading the

system-dependent modules (i.e., the User's Guide and Utilities Reference) thoroughly, using the Interactive COBOL Programmer's Reference to resolve questions about language syntax and Data General's extensions to the COBOL programming language.

Organization of This Module

The MP/OS Interactive COBOL User's Guide is divided into five chapters and an appendix.

Chapter 1 presents the file system, including naming and managing files.

Chapter 2 discusses Data General's Interactive COBOL system calls.

Chapter 3 presents the Interactive COBOL runtime system, including its functions and program execution.

Chapter 4 provides instruction on how to operate the Interactive COBOL compiler, including the command line and its switches.

Chapter 5 discusses the Interactive COBOL debugger.

Appendix A presents and explains compiler, data validation, and runtime error messages.

Notational Conventions

The conventions described below are used in this manual to represent the various elements of COBOL language syntax. The following example contains most of the elements used in describing COBOL syntax:

```
DISPLAY  {screen-name}  [  {, screen-name}  ]  ...[WITH NO ADVANCING].
         {id-lit     }  [  {, id-lit     }  ]
```

The brackets, braces, and ellipses are notational symbols; they are not part of the program source.

UPPERCASE Indicates a COBOL reserved word. An underlined uppercase word is required, if the function in which it appears is used. Non-underlined uppercase words are optional and are used to improve readability. In either case, all uppercase words must be spelled as shown; no abbreviations are permitted.

lowercase Indicates a generic term representing words, literals, PICTURE character strings, comment entries, or a complete syntactical entry to be supplied by the programmer. For instance, where "screen-name" appears, the actual screen name that the programmer has chosen should be used.

Note that throughout this manual, the abbreviations id, id-lit, and lit are used in the syntax in place of the common COBOL constructs identifier, identifier-literal, and literal.

yphen (-) A hyphen appearing between uppercase words is required, as in PROGRAM-ID or SOURCE-COMPUTER. A hyphen between lowercase words indicates that the entry chosen by the programmer must not contain any spaces. In the example above, the screen-name could be ACCTS-PAYABLE or ACCTSPAYABLE, but not ACCTS PAYABLE.

If there is no hyphen in the format, spaces between words are permitted, as in "comment entry".

{ } Braces enclosing part of a format mean that the programmer must select one of the options enclosed within the braces. Thus, the example indicates that either a screen-name or an id-lit must appear in the DISPLAY statement.

[ ] Brackets enclose optional portions of a format. In the example, screen-name or id-lit and the phrase WITH NO ADVANCING may both be included.

When both brackets and braces appear, they are read from the outside in. In the example, the outer brackets indicate that the entire argument is optional. If the argument is included, however, only one of the elements enclosed in the braces can be used.

... An ellipsis indicates that the format item immediately preceding it (defined by logically matching brackets or braces) may be repeated one or more times. An ellipsis between statements indicates omitted source text, commands, or other operations. Vertical ellipses are used where convenient.

Format — The period (.) is required when it is present in a Punctuation format. The comma (,) and semicolon (;) are optional, interchangeable punctuation characters used in some formats to improve readability. However, they may be used only in the positions indicated in the format. At least one space must follow a comma or semicolon used to separate statements.

Level Numbers — If a level number is indicated in a format, one is required.

Special Characters — When one of the arithmetic or logical operators (+, -, >, <, or =) appears in a format, it is required. These special characters are not underlined.

The conventions described below are used in program examples in this manual. They should not appear in program source.

< > Angle brackets indicate input or responses by the user at the console, often a specific key or key combination. For example:

```
DIVIDE GROSS-SALES BY TOT-NUM-SALES GIVING<NL>
    AVERAGE SALES; ON SIZE ERROR PERFORM
```

( ) Parentheses enclose an explanation of a specific program action that cannot be easily illustrated.

b This character represents a position where a space occurs.

Throughout this manual, the NEW LINE key is abbreviated as NL.

Related Documents

MP/OS Interactive COBOL Documentation

Interactive COBOL Programmer's Reference 093-705013

Written for the experienced COBOL programmer, this document provides all the information required to write COBOL programs. The Identification, Environment, Data, and Procedure divisions are explained in detail, and a set of COBOL program examples are provided. In addition, a syntax summary section provides a handy quick reference.

MP/OS Interactive COBOL Utilities 069-705019

Serves as the reference document for the COBOL runtime and CLI system utilities, excluding the text and screen format editors.

IC/EDIT: Interactive COBOL Editor 055-004

Describes the Interactive COBOL text editor used to write source code and documentation. Explains how to enter and execute IC/EDIT commands useful for creating, modifying, or deleting source code. An alphabetized command reference and command summary table are also provided.

SCREEN: Screen Format Editor 055-006

Describes how to use the IC/SCREEN or CLI/SCREEN programs (the latter not available with MP/OS) to design, code and document COBOL display sceen formats. Using SCREEN, you compose the screen image, typing in literal and data fields exactly as they will appear to the program operator. The source code for this image is automatically generated for inclusion in your COBOL program.

MP/OS Documentation

Microproducts Hardware Systems Reference 014-000636

Provides a detailed functional description of the Microproducts line of microcomputers and related peripherals, board by board.

Microproducts Assembly Language Programmer's Reference 014-000653

Serves as the main source of information for the assembly language programmer. Describes the instruction sets in detail.

An Introduction to Microproducts and Micron 069-400000

Describes the hardware and software in general terms, providing an overview of your system and its capabilities.

Installing Your Microproducts System 069-400001

Provides instruction both for installing MP/OS hardware and for beginning to run the MP/OS software and making backup copies of it.

Learning to Use the MP/OS Operating System 093-400000

Written for anyone who has never used a Microproducts computer. Introduces the MP/OS file system and the command line interpreter (CLI). Provides hands-on, step-by-step instruction.

MP/OS System Programmer's Reference 093-400001

Describes the MP/OS system in detail and tells you how to call system routines from your programs. Includes a dictionary of system calls and library routines.

MP/OS Utilities Reference 093-400002

Describes the utility programs available with the MP/OS system. Provides operating instructions for the macro facility, the text editor SPEED, the macroassembler, the binder and library editor, the symbolic debugger, disk initialization, the utility MOVE (used for making backups), the AOS file transfer utility, and the file display and comparison utility.

MP/Pascal Programmer's Reference 093-400003

Describes Data General's extended version of Pascal.

MP/FORTRAN IV Programmer's Reference 093-400004

Covers all of the features of this powerful high-level language.

MP/BASIC Programmer's Reference 093-400005

Provides the inexperienced programmer with the detailed discussion, numerous examples, and hands-on practice necessary to master the rudiments of programming in BASIC. Reference materials and a complete summary are useful to the experienced programmer as well.

MP/OS File Management Utilities Reference 093-400009

Written for experienced assembly language and system programmers who want to use the file management programs available with MP/OS, this manual describes MP/ISAM (an ISAM file creation utility), and the SORT/MERGE utility.

MSCP Programmer's Reference 093-400012

Describes the MP/OS synchronous communications package, a set of program calls that allow your MP/OS system to communicate with a remote station over a synchronous line. The manual is written for applications programmers familiar with MP/OS and assembly language.

--End of Preface--

Chapter 1
The File System

This chapter provides an overview of the MP/OS file system. It describes file-names, file creation, and file organization.

1.1 Introduction

On the Interactive COBOL system on MP/OS there are several situations in which the name of a file is used:

* TO CREATE OR EDIT A COBOL SOURCE FILE--COBOL source text is written into a file using a text editor. This file-name is the one used when compiling, debugging, and executing the program.

* TO IDENTIFY A DATA FILE--The SELECT entry identifies the data file which is to be used by the program.

* TO IDENTIFY A CALLED PROGRAM--In the CALL PROGRAM statement, the called program is identified by specifying the file-name of the object program.

1.2 Naming Files

Files in Interactive COBOL are identified in two ways: COBOL file-names, referred to in this manual as **internal** file-names, and operating system file-names, referred to as **external** file-names.

1.2.1 Internal (COBOL) File-names

An internal file-name is the name by which a file is referenced within a COBOL program, as in READ TESTFILE. An internal file-name is formed according to the standard COBOL rules for any programmer-created word.

1.2.2 External File-names

The external file-name is the name by which a file is known to the operating system. An external file-name is formed according the following rules:

* The name may contain up to 15 characters from the set A-Z (either upper or lowercase; the MP/OS system does not distinguish between the two), 0-9, dollar sign ($), question mark (?), period (.), and the underscore(_). Examples

of valid file-names are:

```
CUST_CODE2
INVENTORY?AMT
EMPL$FICA.QTR
```

* Optionally, a pathname (see below) may precede the file-name.

Note: All CS file-names (except some device-names) are legal MP/OS file-names.

1.3 Creating External Files

You may create a source file with the CLI command "X SPEED file-name" or by selecting the proper choice from the IC/EDIT Main Menu. The system then opens the file and you may enter source text. See the MP/OS Utilities Reference, DGC No. 093-400002, for a description of SPEED, or IC/EDIT: Interactive COBOL Editor, DGC No. 055-004, for instructions on how to use this editor.

You may also create a file with the CLI command "CREATE file-name." In order to edit the file, however, you have to use SPEED.

Data files are usually created and modified by using COBOL programs.

1.4 The Directory Structure

On the MP/OS system, files are grouped into directories. A **directory** is a file which is a catalogue of other files, including other directories.

Each disk or diskette device has a **root directory,** identified by the device name followed by a colon. The root directory contains all other directories and files residing on that device.

When you boot the system, you are placed in the Command Line Interpreter (CLI) in the root directory. The root directory is the **current** or **working** directory until you enter another directory.

You may create a subordinate directory within any directory with the "CREATE/DIRECTORY directory-name" command. You enter the new directory with the command "DIRECTORY directory-name." This directory then becomes your working directory. Another directory that is subordinate to the one you are in may then be created, and so forth. The result, as shown in Figure 1-1, is a hierarchical file system.

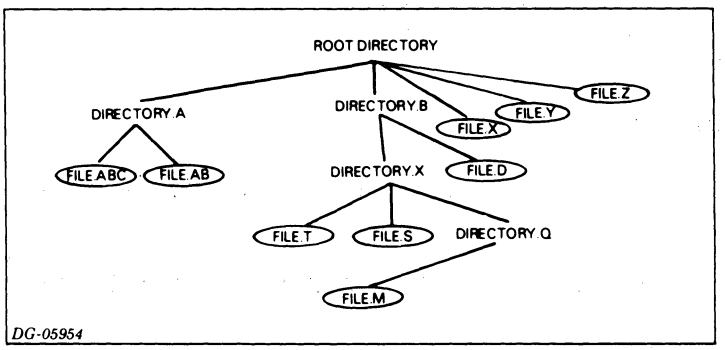


Figure 1-1. Hierarchical File System

1.4.1 Pathnames

A file in the working directory may be referenced simply by mentioning the desired file-name. However, a file located in a directory other than the working directory must be referenced either by qualifying the name of the file with a **pathname** or by using a **searchlist** (see below). A pathname represents the unique path through the file system to a specific file. The pathname consists of one or more file-names separated by colons. All file-names except the last one must be the names of directories, and each directory named must be a subdirectory of the preceding one. The maximum number of characters permitted in a pathname is 127. Using the file system in Figure 1-1 as an example, the full pathname from the root directory to FILE.M is:

```
:DIRECTORY.B:DIRECTORY.X:DIRECTORY.Q:FILE.M
```

When a pathname begins with a subdirectory of the working directory, it is called a partial pathname. The following example shows a partial pathname to FILE.M from DIRECTORY.B:

```
DIRECTORY.X:DIRECTORY.Q:FILE.M
```

1.4.2 Searchlists

As long as all of the files desired are in the working directory, accessing the files is a simple matter since the system automatically searches the current directory when it encounters a file-name with no pathname. However, referencing several files in different directories by pathnames becomes tedious. Creating a searchlist relieves this problem. The searchlist is an ordered list of directories to be searched any time a file referenced with just a file-name or partial pathname is not in the working directory. The searchlist may include from one to five

directories. It typically names the directories that are accessed most frequently.

The searchlist can be set using the CLI SEARCHLIST command. The format is:

SEARCHLIST [pathname(1)] [pathname(2)] . . . [pathname(n)]

As an example, using Figure 1-1, suppose that FILE.ABC is a COBOL program, but the program uses files in the root directory and also in DIRECTORY.X. The system automatically searches the working directory, DIRECTORY.A. The following searchlist directs the system to search both the root directory and DIRECTORY.X when a file-name is encountered:

SEARCHLIST : :DIRECTORY.B:DIRECTORY.X

For a detailed discussion of creating and working with files using the CLI, see MP/OS Interactive COBOL Utilities, DGC 069-705019.

1.5 File Access Privileges

The MP/OS system provides protection against improper deletion of a file, reading of a file, or writing to a file. The CLI "ATTRIBUTES" command sets or displays a file's attributes. The attributes can be:

P: File is permanent and cannot be deleted.
R: File is read-protected.
W: File is write-protected.

Note: Do not assign the "R" attribute to .PD and .DD files, since the system needs to read these.

For more information, see MP/OS Utilities Reference, DGC 093-400002.

1.6 Creating Files in a COBOL Program

MP/OS Interactive COBOL file operations are both simple and convenient. Files are automatically created by the system when a file that does not exist is opened for OUTPUT or I-O by a COBOL program. No system commands are necessary, and no information specifying file length need be given. File space is allocated randomly by the system on an as-needed basis, limited only by the physical size of the device; thus the user need not specify the size of a file. All blocking is handled by the system, using a fixed 512-byte block size.

Instructions relating to file creation are given in the Environment and Procedure divisions of a COBOL program. The file-control entry in the Input-Output Section of the Environment Division contains a SELECT statement for each file to be processed by the program. This statement gives the file's internal and external names, organization, and access mode; SELECT also identifies the primary key and any alternate keys for indexed files, names the relative record key for relative files, and identifies an optional file status item. In addition, it assigns the file to a device (most often a disk or a printer).

In the Procedure Division, the OPEN statement controls file creation according to the parameters specified in the SELECT statement. These instructions contain all the information the operating system needs to create files. The DELETE FILE statement removes a file from the user's operating environment.

1.7 Assigning Files to System Devices

In the SELECT statement, Interactive COBOL allows an optional identifier or literal to be supplied by the programmer to identify a specific device file-name. The syntax for the SELECT statement is:

SELECT file-name ASSIGN TO { PRINTER / DISPLAY / KEYBOARD / DISK } , id-lit (identifier or literal for external file-name)

Example:

```
SELECT CUST_FILE ASSIGN TO DISK, "CUST_CODE2".
```

When using an identifier, the full value of the identifier must be a valid external file-name, or the file-name must be left-justified in the identifier and terminated by a null (LOW-VALUE).

If the external file-name option is omitted from the SELECT statement, the compiler generates default file-names. These generic file-names are listed in Table 1-1.

Table 1-1. Generic Device File-names

Device	file-name
PRINTER	$LPT
DISPLAY	$TTO
KEYBOARD	$TTI
DISK	The first 10 characters of the internal (COBOL) file-name with "$" replacing "-".

The MP/OS runtime system converts these generic device file-names to system file-names. These system file-names are listed in Table 1-2.

Table 1-2. System Device File-names

Device	file-name
PRINTER	@LPT
DISPLAY	@TTO
KEYBOARD	@TTI
DISK	The first 10 characters of the internal (COBOL) file-name with "$" replacing "-".

For more details, see the SELECT statement in the Interactive COBOL Programmer's Reference, 093-705013.

1.8 System Files

Interactive COBOL automatically creates a number of special files that are exclusively for system use. These files have file-name **extensions.** A period and a suffix are added to the name of each file being processed. The suffix identifies the file usage. System files are placed in the directory in which the parent file resides. Table 1-3 lists the system extensions.

Table 1-3. System File-name Extensions

Extension	System Use
.PD	The Procedure Division of a COBOL object program
.DD	The Data Division of a COBOL object program
.NX	The Index portion of an indexed or relative file
.XD	The Data portion of an indexed or relative file
.CLI	System macro files
.SR	COBOL source files (CRT format)
.PR	Save files
.OL	Overlay files
.CO	COBOL source files (card format)
.QK	Compiler statistics files
.TX	Text files
.SO	CRT format output files
.LS	Listing files
.DL	Delete files (ICEDIT)
.CU	Cut files (ICEDIT)
.SS	SCREEN source file
.AX	SCREEN descriptor file
.SX	SCREEN descriptor file

These system file-name extensions should be avoided when naming files for use with Interactive COBOL.

--End of Chapter--

Chapter 2
System Calls

Certain system functions can be executed from within a COBOL program. These system calls have two-character names (a number sign [#] followed by a single uppercase letter), with an additional argument in certain cases. The programmer invokes the function as a literal argument in a CALL PROGRAM statement. The general syntax is:

```
CALL PROGRAM id-lit.
```

The functions available on the MP/OS system are:

#S Stop. Shuts down the runtime system and returns control to the CLI. The value of id-lit is "#S".

#L LOGON. The LOGON program is invoked. The value of id-lit is "#L".

#H Hang up. Shuts down the runtime system and returns control to the CLI. The value of id-lit is "#H".

#D Enter the debugger. Runs the specified program under the interactive debugger's control. The value of id-lit is "#Dprogram-name", where program-name is the name of the program to be debugged. The debug option will automatically bring up the debugger across CALL PROGRAMS; thus it is not necessary to recompile and use the #D chain. For a complete description of the Interactive COBOL debugger, see Chapter 5, "The Interactive COBOL Debugger".

#I Mount a device. Directories need not be initialized as under RDOS. The value of id-lit is "#Idevice-name".

#R Dismount a device. Directories need not be released as under RDOS. The value of id-lit is "#Rdevice-name".

#W Wait. Creates a pause in the program. It might be used to allow a message to be read before the screen is blanked. If no other argument is specified, the pause is 3 seconds. An argument specifies the amount of time to pause in tenths of a second. The maximum value of the time argument is 65,535; thus, the longest possible delay is 1 hour, 49 minutes, 13.5 seconds. After the pause, execution continues with the next statement. The value of id-lit is either "#W" or "#Wcount".

Examples:

The following command results in a 15-second delay:

```
CALL PROGRAM "#W150".
```

The following technique can be used to produce a delay whose length is specified interactively. These statements appear in the Working-Storage Section:

```
01 WAIT.
   03 FILLER PIC X(2) VALUE "#W".
   03 TIMER PIC 9(5).
```

The Procedure Division contains a DISPLAY-ACCEPT sequence through which the operator enters a length of time for the delay. This value is moved to TIMER. The statement CALL PROGRAM WAIT produces the delay for the specified time.

Note: The USING clause is not valid when executing system functions.

The following system calls have no meaning under MP/OS: #A, #P, #M, #T, #F, and #C. If they are called, the system will return an exception status code of 203, indicating the program was not found. (See Chapter 3 in this module and the ACCEPT verb in the Interactive COBOL Programmer's Reference for information about the EXCEPTION STATUS item.)

--End of Chapter--

Chapter 3
The MP/OS Interactive COBOL Runtime System

The MP/OS Interactive COBOL Runtime System is a system program that monitors the execution of COBOL programs. The runtime system performs the following functions:

* Begins execution of Interactive COBOL programs via LOGON
* Recognizes control characters entered at the terminal keyboards
* Accepts data and validates data entries in screen fields
* Informs operators of runtime errors
* Provides utility functions which support system operations
* Provides an interactive symbolic debugger for applications programmers

3.1 Runtime System and Compiler Compatibility

The runtime system and the COBOL compiler are both system programs with revision levels. The runtime system cannot execute a program unless the code revision level agrees with its own. The compiler writes its revision level in the program's object code. The code revision level can be inspected by running the COBOL program CREV. (See MP/OS Interactive COBOL Utilities, DGC 069-705019.)

3.2 The Interactive COBOL Command Line

The runtime system is called via the CLI. To bring up the runtime system, type ICX <NL>. The following files must be in the current directory or in the searchlist: ICX.PR, ICX.OL, ICX.ER, LOGON.PD, and LOGON.DD. LOGON will come up and ask for a program to run or a program to debug. The runtime system then locates the object program files, known to the operating system as program-name.DD and program-name.PD, and execution begins.

The LOGON display can be modified to suit the needs of the application. Since LOGON is a COBOL program itself, an editor should be used to make changes. LOGON must then be recompiled.

The maximum PERFORM depth is 30. The /P switch is not supported.

3.3 Pathnames in Program Names

If the object files are not in the current directory or in the searchlist, a pathname (e.g., @DPH0:ACCTS.PAYABLE) must be given as a prefix to the program name. (For information about the current directory, the searchlist, and pathnames, see Chapter 1.)

3.4 Program Switches

The program name may be followed by logical switches which indicate an ON or OFF status for program-defined switches. (Switches are defined in the SPECIAL-NAMES paragraph of the Environment Division of the COBOL program; see the Interactive COBOL Programmer's Reference.) A logical switch is any uppercase alphabetic character. For example, typing PAYROLL/D in response to the Run Program prompt would indicate ON status for the /D switch.

3.5 Program Size

If you accept the default values used to generate the operating system, up to 15kb of space is available for COBOL programs. If you generate your own system, less space may be available.

If you do decide to generate your own system, answer NO to the question "Should the default system configuration parameters be used?" in the SYSGEN dialogue, and provide the necessary parameters. Every extra item makes the operating system larger and, consequently, maximum program size smaller. Conversely, decreasing the number of items makes the operating system smaller and program size larger. Table 3-1 gives sizes for each item, the normal default values, and the range for values if you do generate another system. For more information on SYSGEN, see MP/OS System Programmer's Reference, DGC 093-400001.

Table 3-1. Available Sizes for Operating System Parameters

	Size	Default	Range
System buffer	512 bytes	7	7-20
File information buffer	128 bytes	4	2-10
Task control block	60 bytes	1	0-100
System stack	260 bytes	2	1-10
Channel	38 bytes	16	8-64

3.6 Normal Exit from a Program

When a program terminates normally, the message STOP RUN is displayed. Pressing **ESC, NL** or **CR** returns the LOGON display.

3.7 Control Keys and the ACCEPT Statement

Field terminator keys, the ESC key, and function keys drive the ACCEPT statement. The ACCEPT statement transfers data from the keyboard to buffer areas associated with data items defined in the Data Division. The data transfer occurs when one of these keys is pressed. The positions of the various keys on the keyboard are shown in Figure 3-1.

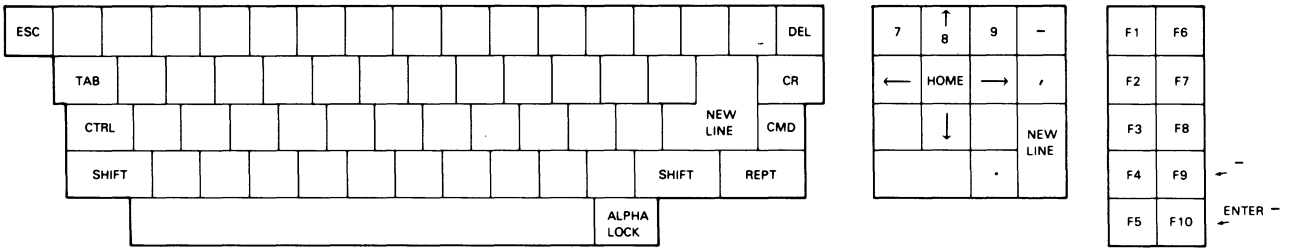


Figure 3-1. Keyboard

Note: To use the up-arrow, down-arrow, left arrow, right arrow, and HOME keys, cursor mode must be turned on. To do this, hold the CMD key down and press 9 on the numeric control keypad. When cursor mode is on, the five direction keys cannot be used for their numeric values. To turn cursor mode off, hold the CMD key down and press 7 on the numeric keypad.

A single field or a group of fields may be transferred with the ACCEPT statement. When one of the field terminator keys (except the **ESC** key) is pressed at a field within a group, the data in that field is transferred, but the execution of the ACCEPT statement may not be complete. The descriptions below of each of the keys give the circumstances in which ACCEPT is completed.

3.7.1 Field Terminator Keys

The following keys advance the cursor to the beginning of the next field in a group.

* Both **NEW LINE** keys

* **CR**

* **TAB**

* Down-arrow on the cursor control pad

In the following instances, pressing one of the field terminator keys listed above at a field also results in the completion of the ACCEPT statement:

1) The field is the only one being accepted.

2) The field is the last one in a group being accepted.

The up-arrow on the cursor control pad accepts a single field. In a group of fields, it does not complete ACCEPT; it simply positions the cursor to the beginning of the previous field in the group. If the cursor is within the first field in the group, the up-arrow repositions the cursor to the beginning of that field.

3.7.2 Escape Key: ESC

This key aborts the ACCEPT statement. When **ESC** is pressed at a single field, the data is not transferred. When pressed at a field within a group, data entered in previous fields is transferred, but the data in the current field is not. If the **ESC** key is pressed and an ON ESCAPE clause is included in the ACCEPT statement, the imperative statement of the ON ESCAPE clause is executed; otherwise, the program continues executing at the next statement.

3.7.3 Function Keys

Pressing a function key completes an ACCEPT statement without aborting data transfer. Data from a single field is transferred. When a function key is pressed at a field within a group, data entered in previous fields and in the current field (up to the current cursor position) is transferred. The following keys and key combinations act as function keys:

* Function keys 1 through 8

* Function keys 1 through 10 with **SHIFT**

* Function keys 1 through 10 with **CTRL**

* Function keys 1 through 10 with **SHIFT** and **CTRL**

3.7.4 Execution of the ACCEPT Statement

A COBOL program can determine which function key was struck. To do so, the program requires two ACCEPT statements. The first has the form:

ACCEPT { screen-name / id } ON ESCAPE imperative-statement.

This statement accepts data into the screen. The ON ESCAPE clause applies to function keys as well as the **ESC** key. When a function key is pressed, the imperative statement is executed. In order to determine which key terminated the ACCEPT, the program must include a standard ACCEPT in the form:

ACCEPT id FROM ESCAPE KEY.

The ESCAPE KEY item contains the two-digit code generated by a termination key. Its PICTURE is 9(2). When the operator presses any of the termination keys, a two-digit value is moved to the ESCAPE KEY item. The item can then be interrogated by the program to determine which key was struck. The codes generated are listed in Table 3-2. (For further information, see the ACCEPT verb in the Interactive COBOL Programmer's Reference.)

Table 3-2. Field Termination Codes

Terminator Key	Value			
	Key Only	Key with SHIFT	Key with CTRL	Key with SHIFT and CTRL
CR, NL, TAB, ↑, ↓	00			
ESC	01			
F1	02	10	18	26
F2	03	11	19	27
F3	04	12	20	28
F4	05	13	21	29
F5	06	14	22	30
F6	07	15	23	31
F7	08	16	24	32
F8	09	17	25	33
F9	(-)	41	48	55
F10	(ENTER -)	42	49	56

3.7.5 Other Control Keys

Other control keys which are used for editing and cursor control are discussed below.

Key	Action
DEL	Moves the cursor back one space, deletes the character in that position, and replaces it with a space. (It is ignored if the cursor is at the beginning of the field.)
←	The left arrow on the cursor control pad moves the cursor to the left without erasing characters. (It is ignored if the cursor is at the beginning of the field.)
→	The right arrow on the cursor control pad moves the cursor forward without erasing characters. (It is ignored if the cursor is at the last character present in the field or if the next character is an underscore.)
ER EOL	Erases the field, displays underscores in all its character positions, and positions the cursor at the beginning of the field.
ER PG	Erases the entire page and positions the cursor at line 1, column 1.
REPT	When pressed at the same time with another key, repeats that character or function.
_	The underscore character terminates a field when an input field is accepted and interpreted by the runtime system. In effect, the underscore erases to the end of the field. For example:

ACCEPT Field Contents	Effective Entry
ACCOUNTS_ECEIVABLE	ACCOUNTS
_ELETE THIS STUFF	<empty line>
ONE_ TWO_ THREE _	ONE

3.8 Validation of Entered Data

Pressing a field terminator key initiates the runtime system's data validation function. If the data entry does not match the PICTURE for the field, an error message is displayed at the bottom of the screen and the cursor is positioned to the first invalid character of the field. The error message describes the discrepancy with the PICTURE clause and shows the correct form of the data. The data validation error messages and an explanation of each are given in Appendix A.

3.9 Interprogram Communications

Interactive COBOL provides interprogram communications via the CALL PROGRAM...USING statement. When an application makes use of this feature, the runtime system stores data to be passed from program to program in a system file called ICX.UF. This file is maintained by the system. Its capacity is 12K bytes; the compiler checks for this limit.

When a CALL PROGRAM...USING statement is executed, the data items specified in the USING phrase are written to the using file, ICX.UF. The data items remain in this file until another CALL PROGRAM...USING statement is executed, or until the runtime system is brought down and brought back up.

3.10 Runtime Interrupts

On the MP/OS system it is not possible to abort a running COBOL program. If a program must be terminated, either turn the terminal off and reboot or reboot by holding down the CMD and ESC keys. There is no need to clear files; MP/OS does this automatically.

3.11 Runtime Errors

In the event of a fatal error in the execution of a COBOL program, the runtime system displays an error message on the bottom line of the terminal screen and halts program execution. When the runtime system halts a program, the operator should copy the error message and the data on the screen, and then press **ESC** to return to LOGON.

The error messages have the form:

```
ERROR: error text      relative-procedure-address [COBOL PC =]
```

The **relative-procedure-address** is a five-digit number that the Interactive COBOL compiler inserts in an output listing in place of a procedure name's line number. It gives the approximate location of the statement that caused the error.

The text of the runtime system error messages and an explanation of each are given in Appendix A.

3.12 Runtime System Failure

The message FATAL RUNTIME SYSTEM ERROR is displayed in the event the runtime system itself fails. This may indicate an attempt to run a program which has a compiler error. If this is not the case, run the program again. Key **CTRL-C CTRL-E** when this message appears. Retain the ?ICX.BRK file and contact a Data General systems engineer.

3.13 Printing Files

To print a file on MP/OS, enter TYPE/L from the CLI. Alternately, you may open the printer directly from the COBOL program.

3.14 System Utilities

The following utilities are provided for the Interactive COBOL user on the MP/OS system:

COBOL Programs	CLI Programs
LOGON	MESSAGES
CALC	COLLAPSE
CREV	REORG
ANALYZE,	CSSORT
FILESTATS	RDUMP
IC/SCREEN	RLOAD
IC/EDIT	

Note: IC/SCREEN and IC/EDIT are available only on development systems.

For details on these utilities, see MP/OS Interactive COBOL Utilities, DGC 069-705019.

3.15 Program Status Indicators

During the execution of Interactive COBOL programs, there are two indications of the status of the program: the File Status Code and the Exception Status Code.

3.15.1 File Status Codes

After an I/O operation is executed, the system generates a File Status Code indicating the outcome of the operation. If the "FILE STATUS IS data-name" clause is specified in a FILE-CONTROL entry, a value indicating the status of the I/O operation is placed into the specified two-character data item before any applicable USE procedure, AT END phrase, or INVALID KEY phrase is executed. That item may then be tested to determine the condition that terminated the I/O operation. The File Status Codes are most commonly examined in the USE procedures of the DECLARATIVES portion of a program, but they may be examined anywhere within the Procedure Division.

The 2-byte codes listed in Table 3-3 are assigned after execution of I/O operations.

Table 3-3. File Status Codes

Code	Meaning
00	Successful I/O operation.
10	AT END condition.
21	INVALID KEY: in sequential access mode, the primary key on a WRITE to an ISAM file is not greater than the value of the primary key of the previous record, or the primary key value has been changed between the successful execution of a READ statement and the execution of the next REWRITE statement for that file.
22	INVALID KEY: an attempt has been made to write or rewrite a record that would create a duplicate primary key.
23	INVALID KEY: the record was not found.
24	Boundary error: the index depth has been exceeded.
30	Permanent hardware error.
34	Permanent error: the disk space has been exhausted.
91	Open error: the file does not exist; an illegal file-name has been used; the file is already open during an attempted OPEN; the file is not open during an attempted CLOSE; or there are not enough I-O channels.
92	Mode error: the file is not open; there is an attempted WRITE operation to an input file or a READ operation of an output file; or a DELETE/REWRITE operation for a sequential access file is not preceded by a READ statement.
94	Use error: the file is in use on OPEN.
96	The device is not mounted.
9A	File descriptor inconsistency: the record length, key length, or key positions do not agree with the file description.
9B	Bad index of an ISAM or relative file; the file should not be used. The utility REORG may be able to rebuild the file.

3.15.2 Exception Status Codes

The syntax of the CALL PROGRAM...USING statement includes the optional ON EXCEPTION clause. If a CALL PROGRAM statement fails, an exception code is written in the exception status register. If an ON EXCEPTION clause exists, its imperative statement is executed. If no ON EXCEPTION clause has been provided, execution of the calling program continues at the next statement. There is no system-generated error message which informs the operator that the CALL PROGRAM statement has failed.

The exception status code can be examined via a system ACCEPT. The EXCEPTION STATUS item contains a three-character code that identifies the type of exception condition which has occurred during the execution of the CALL PROGRAM statement. An "ACCEPT...FROM EXCEPTION STATUS" clause must immediately follow the CALL PROGRAM statement or the code found in the EXCEPTION STATUS field will be undefined. The PICTURE of the EXCEPTION STATUS item is 9(3). Table 3-4 lists the Exception Status Codes. For more information, see the ACCEPT and CALL PROGRAM verbs in the Interactive COBOL Programmer's Reference.

Table 3-4. Exception Status Codes

Code	Meaning
000	CALL program completed, no exception.
200	CALLed program exceeds size limitation.
201	Revision incompatibility in CALLed program.
202	CALLed program is not a legal COBOL program file.
203	COBOL program not found.

--End of Chapter--

Chapter 4
Operating the Compiler

The Interactive COBOL compiler operates in one pass on the source code. It produces up to four files as output: a "PROCEDURE DIVISION" file with the extension .PD, a "DATA DIVISION" file with the extension .DD, a listing file, and an optional error file.

4.1 The Command Line

The command line used to call the compiler is:

ICOBOL [/global-switch] ... s-file-name [/local-switch] ...

where

s-file-name identifies the source file.

Global Switches

C

Card format source. The compiler treats columns 73-80 of a card format source line as a comment field. Omission of the C switch specifies CRT format. All of the files which have been copied by a COPY command in the program must be in the same format (CARD or CRT) as the program that uses them.

E=e-file-name

The error file is e-file-name. Error messages will be written to this file. If the error file does not exist, it is created; otherwise, the errors are appended to the end of the file.

E

Suppresses error messages. This switch is ignored if an error file is specified.

N

No object program is produced. If this switch is not present, the object program will be produced in the files s-file-name.PD and s-file-name.DD.

L=l-file-name

The listing file is l-file-name. If the listing file does not exist, it is created; otherwise, the source listing is appended to the end of the file.

L

The listing file is s-file-name.LS.

X

The Cross Reference Table is output to the listing file at the end of compilation.

S

The symbol table, compiler statistics, and additional data are output to the listing file at the end of compilation.

U

Decompilation of object program is output to the listing file at the end of compilation.

D

Add symbol table to object data file for runtime debugging. This switch will be ignored if a global N switch is also present.

Note: The use of the /N and /U switches together in a command line is illegal and produces an error message.

If no error file is specified, the listing file will be used for error messages. If no listing file is specified, error messages will be output to the console.

The following example shows the command line to compile the source file INVENTORY.SR. (The .SR is a system file-name extension; see Chapter 1.) The global switches request a symbol table, prepare the program for execution under the Interactive COBOL debugger (see Chapter 5), and request the creation of a listing file with the name INVENT_LIS.

```
ICOBOL/S/D/L=INVENT_LIS INVENTORY.SR <NL>
```

Local Switches

W Suppress warning messages from the listing file.

I The source file is indexed.

4.2 Command Line Error Messages

If the compiler finds any errors in the command line it will terminate execution with one of the following error messages on the console:

DUPLICATE SWITCH

L or E switch appears with more than one file, or more than one source file is specified.

ILLEGAL LOCAL SWITCH

A switch other than L or E appears with a file, or the L and E switches appear with the same file.

ILLEGAL SWITCH COMBINATION /N/U

These two switches may not be used together in a command line.

INPUT IS NOT AN ISAM FILE

The specified indexed file is not in the disk directory, or it was specified with a file-name extension.

ISAM SOURCE FILE HAS AN INCOMPATIBLE REVISION NUMBER

The source file's organization is that of an out-of-date ISAM revision.

NO SOURCE FILE SPECIFIED

A source file argument does not appear in the command line.

SOURCE FILE NOT FOUND

The specified source file is not in the disk directory.

Other errors that may occur at initialization (e.g., specifying a read-protected file as the source file) will cause termination with the appropriate system error message.

4.3 Source Listing

The source program is listed to the listing file, if one is specified. The top line of each listing page includes a page number and the program name from the PROGRAM-ID entry. A new page occurs at a form feed character, at a slash (/) character in the command field, or after 60 lines. Line numbers within each page appear at the left margin of the listing file. COPY file lines are listed with their COPY file line numbers, followed by a "C".

4.4 Compiler Warning Messages

These messages are printed only in the listing file. If no listing file is produced, they will not be seen.

WARNING: "n" ITEMS CORRESPOND

In an ADD CORRESPONDING, SUBTRACT CORRESPONDING, or MOVE CORRESPONDING statement it is not always clear how many items will be affected. This message reports the number of items which do not correspond.

WARNING: NO "WITH DUPLICATES" CLAUSE. DUPLICATES ARE ALWAYS PERMITTED

In Interactive COBOL, the WITH DUPLICATES phrase is not necessary in a SELECT statement, but duplicate key values are still allowed. This warning reminds the programmer of this fact.

4.5 Compiler Statistics

A list of compiler statistics is available to the applications programmer. They are provided at the end of the listing file if the /S switch is used in the command line. The statistics provided are:

SYMBOL TABLE: Amount of core memory occupied by the symbol table. This includes all user-defined words, literals, edit masks, and the screen fields, but excludes the symbols themselves.

NAME TABLE: An adjunct to the symbol table which holds the actual symbols.

SYSTEM OVERHEAD: Amount of core memory used for stacks and other program-dependent temporary storage.

TOTAL MEMORY USED: Sum of the above three quantities.

MEMORY % USED: Total memory used as a percentage of initial free space.

SYMBOL COUNT: Number of symbols entered into the symbol table.

DATA REFERENCES: Largest data reference number.

COMPUTE TEMPS: Number of temporary working data items used by COMPUTE for intermediate computational results. Each temporary data item occupies 18 decimal digits.

PROCEDURE NAMES: Largest procedure reference number.

Note: The compiler error messages are listed in Appendix A of this manual.

--End of Chapter--

Chapter 5
The Interactive COBOL Debugger

Using the Interactive COBOL debugger, a programmer can run a program under conditions which are controlled interactively. The programmer can begin execution at the start or end of a procedure (i.e., paragraph or section). Any identifier, except one defined as a computational item, can be set to any legal value. The programmer can specify procedure names at which execution is to be halted. These **trap points** may be set at either the beginning or the end of a procedure. Then, while program execution is suspended, the current value of an identifier can be examined or modified.

The debugger enables the programmer to "step through" programs one paragraph at a time. In this way, logical errors and problems in program design can be isolated to specific areas of a program's Procedure Division.

5.1 Operation of the Debugger

Before a COBOL program can be debugged, it must be compiled with the global switch /D. When this option is specified in the command line, the compiler adds a symbol table to the data portion of the object program (i.e., the file program-name.DD). This table must be present for the debugger to operate.

The debugger runs under the control of the runtime system. A suggested procedure for its efficient use is given below. The debugger has its own set of commands; these are described in the Debugger Command Reference below. Some examples of the use of the debugger are also provided.

5.1.1 Entering Debugger Commands

The debugger accepts commands in the form shown in the Debugger Command Reference and in abbreviated form. The first three characters of each word in a debugger command are sufficient; for example, DIS is equivalent to DISPLAY. Debugger commands are terminated by **NL** or **CR**.

The **DEL** key erases the previous character in a line, and the key labeled **ER EOL** erases the entire line. The up-arrow cursor control key positions the cursor to the first character position in the line. An **NL** can be used in response to the debugger prompt (**!**) to display the value of the identifier that was most recently referenced.

The debugger keeps the programmer informed about where it is in the Procedure Division. Whenever the execution of a program is suspended, the debugger displays a message giving the procedure name where it is stopped. Error messages are displayed if unacceptable commands are entered.

The procedure for running the debugger is as follows:

1) Before compiling the program for debugging, be sure that procedure names have been inserted near the COBOL statements where execution is to be halted. The debugger commands that suspend program execution require a "procedure-name" argument.

2) Compile the program, inserting the global switch /D in the command line. Be sure to get a compilation listing (i.e., a copy of the file PROGRAM-NAME.LS) by using the local or global /L switch. For example, the following command produces a compilation listing at the printer:

```
COBOL/D program-name @LPT/L
```

3) Respond to Debug Program prompt on the LOGON menu with the program name given in the command line for compilation. When the debugger's prompt (!) is displayed, the interactive debugger is ready to accept commands. See the command reference below.

4) To restart debugging after the program reaches a STOP RUN, press **ESC** to return to LOGON and go to step 3.

5) To end a debugging session, issue the debugger command STOP. The runtime system's message STOP RUN is displayed. Pressing **ESC** returns you to LOGON.

6) Once a program is debugged, recompile it without the /D switch. (The symbol table created by this option adds to the program's size.) Recompilation deletes the old object program files and creates new object code.

5.2 Debugger Command Reference

Arguments in Debugger Commands

procedure-name

A section name, paragraph name, or paragraph name qualified by a section name. (OF or IN may be used as a connector.)

identifier

Any identifier in the Data Division, except those in the Screen Section and those whose USAGE is COMPUTATIONAL or INDEX. This includes group items. An identifier may be qualified and/or subscripted (see example below).

literal

A nonnumeric or numeric literal. A nonnumeric literal must be enclosed in quotation marks.

Commands That Control Execution of a Program

RUN

Runs program from the current relative-procedure-address. If this is the first command issued, the program runs from the beginning. The program runs until it reaches a trap point or a STOP RUN.

RUN procedure-name

Runs the program, starting at the given procedure-name. This command should not be used if the procedure name is the object of a PERFORM statement; it may interfere with the normal program flow.

RUN START

Runs program from the current relative-procedure-address to the beginning of the next procedure.

RUN END

Runs program from the current relative-procedure-address to

the end of the next procedure.

RUN procedure-name START

Runs program from the current relative-procedure-address to the beginning of the given procedure.

RUN procedure-name END

Runs program from the current relative-procedure-address to the end of the given procedure.

STOP

Halts the debugger. The message STOP RUN is displayed.

Commands That Control Trap Points

The trap commands are different from the "run and stop" commands. When trap commands are used, the program halts at a trap point every time the specified procedure is reached. The trap point remains until it is explicitly removed by a CLEAR command.

TRAP procedure-name

Sets a trap to suspend execution at the beginning of the given procedure.

TRAP procedure-name END

Sets a trap at the end of the given procedure.

CLEAR

Removes all traps.

UNCLEAR

Recovers from a CLEAR command issued in error, i.e., it retains the traps. This command must immediately follow the erroneous CLEAR command.

CLEAR procedure-name

Removes any traps at the beginning and end of the specified procedure. UNCLEAR does not recover from this command.

LIST

Displays all procedure-names that have trap points set at

the beginning of the procedure.

LIST END

Displays all procedure-names that have trap points set at the end of the procedure.

Commands That Set and Display the Value of an Identifier

DISPLAY identifier

Displays the value of the given identifier. If the identifier argument is omitted, the value of the most recently referenced identifier (whether named in a SET or DISPLAY command) is displayed.

SET identifier TO literal

Puts the value given by the literal into the identifier. When the identifier is omitted, the value of the most recently referenced identifier (whether named in a SET or DISPLAY command) is set.

The following sequence illustrates the debugger's retention of an identifier referenced in a SET or DISPLAY command.

```
! DISPLAY BALANCE <NL>
! 1000
! SET TO 800 <NL>
! <NL> (implied DISPLAY)
! 800
```

The following equivalent sequences illustrate the use of the commands with subscripted items:

Short Form	Long Form
! DISPLAY ITEM <NL>	! DISPLAY ITEM(1) <NL>
! DISPLAY (3) <NL>	! DISPLAY ITEM(3) <NL>
! SET TO 10 <NL>	! SET ITEM(3) TO 10 <NL>

Debugging a Calling Program and Its Called Program

The debugger can be used to debug a program and a program it CALLS. To do this, compile both programs with the /D switch. Debug the first program; when the CALL PROGRAM for the second program, is executed, you will come up in the debugger for the second program, and the debugger will be ready to accept commands.

If the second program was not compiled with the /D switch, that program will be run, but you will not come up in the debugger for that program.

--End of Chapter--

Appendix A
Error Messages

This appendix contains error messages produced by the MP/OS compiler, the screen management data validation function, and the runtime system.

A.1 Compiler Error Messages

An error message is written to the error file when an error in a program statement is detected. If the error file is the same as the listing file then the error messages appear below the lines containing the error (in some cases two lines below; in other cases an error message will not appear until the end of the division). Included in the error file at the end of compilation are:

* PROGRAM CONTAINS EXTENDED FEATURES (This notification appears whenever extended features are used in the source program.)
* Source line count
* Error count
* Compilation time statistics
* Object program size statistics

The error messages and their meanings are:

ACCEPT STATEMENT

Illegal syntax for an ACCEPT statement.

ACCESS MODE CLAUSE

Illegal syntax for an ACCESS MODE clause; or the ACCESS MODE is specified as RANDOM or DYNAMIC for a non-disk device.

ALPHABETIC OPERAND NOT PERMITTED

An alphabetic operand is specified for a NUMERIC class test.

AT END CLAUSE

Illegal syntax for an AT END clause; or an AT END clause is illegal for an operation, e.g., a random read.

AT END OR INVALID KEY REQUIRED

Statement requires either an AT END or INVALID KEY clause when no USE procedure is defined for the file.

BLANK CLAUSE

Illegal syntax for a BLANK LINE or BLANK SCREEN clause in the Screen Section.

BLANK WHEN ZERO CLAUSE

Illegal syntax for a BLANK WHEN ZERO clause; or it was specified for a group item or a nonnumeric (edited) item; or the item's picture string contains the asterisk character.

BLOCK CONTAINS CLAUSE

Illegal syntax for a BLOCK CONTAINS clause.

CALL STATEMENT

Illegal syntax for a CALL PROGRAM statement.

CHARACTER

A character not belonging to the 51-character COBOL set has been detected in a statement. (The full 96-character ASCII set is permitted only in comments and nonnumeric literals.)

CLOSE STATEMENT

Illegal syntax for a CLOSE statement.

COLUMN CLAUSE

Illegal syntax for a COLUMN clause in the Screen Section.

CONDITIONAL

Illegal syntax for a conditional statement or sentence.

CONFIGURATION SECTION MISSING

The Configuration Section is required in the Environment Division.

COPY FILE NOT FOUND

The file named in a COPY statement is unknown to the system.

COPY NESTED

A COPY statement exists within a COPY file.

COPY STATEMENT

Illegal syntax for a COPY statement.

DATA DESCRIPTOR CLAUSE DUPLICATE

The same clause has been used more than once in describing a data item.

DATA DIVISION OVERFLOW

Size of the data file exceeds 65,536 bytes.

DATA NAME

1) A data name contains ilegal characters.

2) In a PROCEDURE DIVISION USING header, at least one data name must be present.

3) In a CALL PROGRAM USING statement, at least one data name must be present.

DATA NAME AMBIGUOUS: data-name

This data-name requires further qualification.

DATA NAME UNDEFINED: data-name

1) No description for this name exists in the DATA DIVISION.

2) LINKAGE SECTION items are referenced in the SCREEN SECTION FROM, TO, and USING phrases, or as ALTERNATE RECORDS KEYS in the SELECT statement, but have not been specified in the PROCEDURE DIVISION USING list. These names are all flagged following the PROCEDURE DIVISION USING statement.

3) Data names are referenced in the Procedure Division that were defined in the Linkage Section but were not used in the PROCEDURE DIVISION USING statement.

DATA NAME UNDEFINED: data-name

No description for this name exists in the Data Division.

DATA RECORDS CLAUSE

Illegal syntax for a DATA RECORDS clause.

DEVICE

The device is not DISK, PRINTER, KEYBOARD, or DISPLAY in a SELECT statement.

DISPLAY STATEMENT

Illegal syntax for a DISPLAY statement.

DIVIDE STATEMENT

Illegal syntax for a DIVIDE statement.

DIVISION MISSING

An expected division header was not found.

ELSE WITHOUT IF

An ELSE clause has been encountered without a preceding IF statement.

ENVIRONMENT DIVISION MISSING

The Environment Division header is required.

EXIT STATEMENT

The EXIT sentence is not the only statement in the paragraph.

EXPONENT MUST BE AN UNSIGNED INTEGER WITH 5 OR FEWER DIGITS

The first operand following an exponentiation operator is a literal or identifier with sign or fraction, or has too many digits.

FATAL ERROR--COMPILATION ABORTED

The previous error is considered catastrophic and compilation cannot continue. The source file will continue to be read and listed.

FD DUPLICATE: file-name

An FD has already been processed for this file.

FD CLAUSE DUPLICATE

The same clause has been used more than once in describing a file.

FIELD CLAUSE MISSING

A picture string is specified for a screen field, but no FROM, TO, or USING clause is present.

FIGURATIVE CONSTANT

ALL is not followed by a nonnumeric literal or figurative constant.

FILE-CONTROL CLAUSE DUPLICATE

The same clause has been used more than once in a SELECT statement.

FILE-CONTROL CLAUSES INCONSISTENT

ACCESS MODE, ORGANIZATION, and RECORD (RELATIVE) KEY clauses have been used in an illegal combination.

FILE-CONTROL ENTRY AMBIGUOUS: data-name

The FILE STATUS or KEY data-name requires further qualification.

FILE-CONTROL ENTRY SUBSCRIPTED: data-name

The FILE STATUS or KEY data-name has, or is subordinate to, an OCCURS clause.

FILE-CONTROL ENTRY UNDEFINED: data-name

The FILE STATUS or KEY data-name was not defined in the Data Division.

FILE-CONTROL PARAGRAPH MISSING

The FILE-CONTROL paragraph is required.

FILE DESCRIPTION UNDEFINED: file-name

No FD was encountered for a file named in a SELECT statement.

FILE NAME

A file-name contains an illegal character.

FILE NAME UNDEFINED: file-name

No SELECT statement exists for this file-name.

FILE STATUS: data-name

The File Status item must be a two-character alphanumeric item defined in the Data Division.

FILLER NOT ELEMENTARY

Filler items must not be further subdivided.

IDENTIFICATION DIVISION MISSING

The Identification Division header is required.

IDENTIFIER

An identifier is expected, but has not been found.

IDENTIFIER OR LITERAL

An identifier or literal is expected, but has not been found.

IF STATEMENT

Illegal syntax for an IF statement.

IMPERATIVE STATEMENT REQUIRED

An imperative statement must appear after AT END, ON SIZE ERROR, ON ESCAPE, ON EXCEPTION, etc.

INDEX NAME

An index-name is expected, but has not been found.

INSPECT STATEMENT

Illegal syntax for an INSPECT statement.

INTEGER

A non-integer data item has been encountered in a PICTURE string parentheses; or an integer in an ADVANCING clause has a value greater than 80.

INVALID KEY CLAUSE

Illegal syntax for an INVALID KEY clause.

I-O CONTROL CLAUSE

Illegal syntax for an I-O CONTROL clause.

I-O STATEMENT ILLEGAL FOR ACCESS MODE

Inconsistency in an I/O statement, e.g., READ from a printer.

I/O STATEMENT ILLEGAL FOR DEVICE

Inconsistency in an I/O statement, e.g., random READ from a file opened in sequential mode.

I/O STATEMENT ILLEGAL FOR ORGANIZATION

Inconsistency in an I/O statement, e.g., OPEN EXTEND of a relative file.

JUSTIFIED CLAUSE

JUSTIFIED has been specified for a group, numeric, or numeric edited item.

KEY CLAUSE

Illegal syntax for a KEY clause in a START statement.

LABEL RECORDS CLAUSE

Illegal syntax for a LABEL RECORDS clause.

LABEL RECORDS CLAUSE MISSING

A LABEL RECORDS clause is required for every FD entry.

LEVEL NUMBER

1) A level number is expected, but has not been found.

2) All data items referenced in a PROCEDURE DIVISION USING header must have a level number of 77 or 01.

3) All data items used in a CALL PROGRAM USING statement must have a level number of 77 or 01.

LEVEL 77 NOT ELEMENTARY

A level 77 entry may not be subdivided.

LITERAL CATEGORY

The category of the literal in a VALUE clause does not match the category of the data item.

LITERAL EXCEEDS ITEM SIZE

The size of the literal in a VALUE clause exceeds the size of the data item.

LITERAL IS SIGNED

The numeric literal in a VALUE clause was signed but the data item specified no operational sign.

LINE CLAUSE

Illegal syntax for a LINE clause in the Screen Section.

LINE EXCEEDS 132 CHARACTERS

The source line read had at least 133 characters without a terminator.

LINE/COLUMN LIMITS EXCEEDED

Data has been defined past line 24 or column 80 in the Screen Section.

MARGIN A MUST BE BLANK

On a continuation line, area A must be blank.

MAY NOT BE USED IN A "CALL PROGRAM": data-name

The named item may not be passed to a called program.

MEMORY CAPACITY EXCEEDED

The symbol table (user-defined names and literals) has exceeded the available memory.

MISSING "=" SIGN

Illegal syntax for a COMPUTE statement.

MOVE STATEMENT

Illegal syntax for a MOVE statement.

MOVE UNDEFINED FOR OPERANDS

Category of operands is illegal, e.g., moving a numeric edited item to a numeric item.

MULTIPLY STATEMENT

Illegal syntax for a MULTIPLY statement.

NAME DUPLICATE: name

1) A user-defined word has already been defined as the same type (file-name, data-name, etc.) and cannot be distinguished by qualification.

2) A data-name has been used more than once in a PROCEDURE DIVISION USING header.

NO DATA ITEMS DEFINED IN LINKAGE SECTION

1) When a LINKAGE SECTION occurs in a program, at least one data item must be defined within the section.

2) When a PROCEDURE DIVISION USING header occurs in a program, a LINKAGE SECTION and at least one subordinate item must be defined in the calling program.

NONNUMERIC LITERAL EXCEEDS 132 CHARACTERS

Nonnumeric literals must not be longer than 132 characters.

NOT A LINKAGE SECTION ITEM

In a PROCEDURE DIVISION USING header, all items in the list must be defined in the Linkage Section of the calling program.

NUMERIC LITERAL EXCEEDS 18 DIGITS

Numeric literals must not be longer than 18 digits.

NUMERIC (EDITED) ITEM EXCEEDS 18 DIGITS

Numeric items must not be longer than 18 digits.

NUMERIC OPERAND NOT PERMITTED

Numeric operands are illegal for this statement.

NUMERIC OPERAND REQURED

A numeric operand was expected, but was not found.

NUMERIC OR NUMERIC EDITED OPERAND REQUIRED

A numeric or numeric edited item was expected, but was not found.

OBJECT-COMPUTER PARAGRAPH MISSING

The OBJECT-COMPUTER paragraph is required.

OCCURS CLAUSE

Illegal syntax for an OCCURS clause.

OCCURS DEPTH

OCCURS clauses are only permitted to a depth of 3.

OCCURS NOT PERMITTED AT LEVEL 01 OR 77

OCCURS may only be specified from level 02 to 49.

OPEN STATEMENT

Illegal syntax for an OPEN statement.

OPERAND NOT INTEGER

An integer was expected, but was not found.

OPERAND NOT LENGTH 1

The operand must be one character long.

OPERAND NOT USAGE DISPLAY: data-name

A USAGE DISPLAY operand is required by this statement.

OPERAND MUST NOT BE SIGNED NUMERIC OR COMPUTATIONAL

A signed or computational item cannot be used in this statement.

OPERATIONAL SIGN MISSING

A SIGN clause has been specified but the picture string contains no S character.

ORGANIZATION CLAUSE

Illegal syntax for an ORGANIZATION clause.

PERFORM STATEMENT

Illegal syntax for a PERFORM statement.

PERIOD MISSING

A period is required.

PICTURE MISSING

Field clauses (FROM, TO, or USING) have been specified for a screen item, but no picture was specified.

PICTURE STRING

Illegal syntax for a picture string.

PREVIOUS ITEM WAS ELEMENTARY

The current data or screen item has a higher level number than the previous one, yet the previous item was not a group item.

PREVIOUS ITEM WAS GROUP

The current data or screen item has an equal or lower level number than the previous one, yet the previous item was not elementary.

PROCEDURE DIVISION MISSING

The Procedure Division header is required.

PROCEDURE DIVISION NOT SECTIONED

The Procedure Division did not begin with a section name, but one is encountered later.

PROCEDURE DIVISION OVERFLOW

The size of the procedure file exceeds 65,536 bytes.

PROCEDURE NAME

Procedure name contains an illegal character.

PROCEDURE NAME AMBIGUOUS: procedure name

Reference is made to an unqualified paragraph name, but none exists in the current section, and two or more exist in other sections.

PROCEDURE NAME DUPLICATE: procedure name

Two procedure names exist which cannot be differentiated by qualification.

PROCEDURE NAME MISSING

The Procedure Division must start with a procedure name (after the DECLARATIVES, if any). A section name must be followed by a paragraph name.

PROCEDURE NAME UNDEFINED: procedure-name

The procedure name referenced does not exist.

PROGRAM-ID PARAGRAPH MISSING

The PROGRAM-ID paragraph is required.

QUOTE MISSING

A line ends with an open nonnumeric literal, and the next line is not a continuation line or does not begin with a quote in area B.

RECORD CONTAINS CLAUSE

Illegal syntax for a RECORD CONTAINS clause.

RECORD KEY MUST BE ALPHANUMERIC: data-name

The RECORD KEY of an indexed file must be alphanumeric.

RECORD KEY MUST BE DEFINED IN FD: data-name

The RECORD KEY of an indexed file must be defined in one of the record descriptions of that file.

RECORD NAME

The data-name referred to is not a record name, which is required by this statement.

RECORD KEY CLAUSE

Alternate key clause used illegally, or more than four alternates have been specified.

RECORD SIZE EXCEEDS CONTAINS CLAUSE

The actual record size exceeds the maximum specified in the RECORD CONTAINS clause or is less than the minimum.

RECORD SIZE EXCEEDS DEVICE LIMITATIONS

Files asigned to PRINTER, KEYBOARD, or DISPLAY may have a maximum record size of 132 characters; indexed or relative files may have a maximum record size of 4096 characters.

RECORDING MODE

The RECORDING MODE may only be specified for disk files; the VARIABLE clause is only permitted for sequential disk files.

REDEFINED AREA SIZE

The size of the redefining area must equal the size of the redefined area except at level 01.

REDEFINES NOT PERMITTED: data-name

The data-name being redefined either itself has a REDEFINES clause, has an OCCURS clause, or is subordinate to an an item containing a REDEFINES or OCCURS clause.

REDEFINES NOT PERMITTED THIS LEVEL

REDEFINES is not allowed at level 01 in the File Section.

REFERENCE LIMIT EXCEEDED

More than 764 procedure, file, or data references have been made.

RELATIONAL UNDEFINED FOR OPERANDS

The relational condition is not permitted for certain categories of operands.

RELATIVE KEY MUST NOT BE DEFINED IN FD: data-name

The RELATIVE KEY data-name must not be defined within any of the record descriptors of the file.

RELATIVE KEY MUST BE PIC 9(4) COMP: data-name

Relative keys must be defined as PIC 9(4) COMP in the Data Division.

REWRITE STATEMENT

Illegal syntax for a REWRITE statement.

SCREEN DESCRIPTOR CLAUSE DUPLICATE

The same clause has been used twice in one data item's description.

SCREEN HAS NO INPUT FIELDS

A screen-name appears in an ACCEPT statement, but no input fields were defined in it.

SCREEN NAME MISSING

A level 01 screen item must define a screen-name.

SCREEN NOT PERMITTED

A screen-name may not be used as the identifier in a statement of the form ACCEPT id FROM...

SECTION MISSING

A section header was expected, but was not found.

SEPARATOR

Separators must be followed by a space.

SET STATEMENT

Illegal syntax for a SET statement.

SIGN CLAUSE

Illegal syntax for a SIGN clause; or conflict with a group SIGN clause.

SIGN ERROR CLAUSE

Illegal syntax for an ON SIZE ERROR clause.

SOURCE-COMPUTER PARAGRAPH MISSING

The SOURCE-COMPUTER paragraph is required.

START STATEMENT

Illegal syntax for a START statement.

STOP STATEMENT

Illegal syntax for a STOP statement.

SUBSCRIPT IS NOT INTEGER

All subscript data items must be integers.

SUBSCRIPT NOT PERMITTED: data-name

This item neither contains nor is subordinate to an item with an OCCURS clause.

SUBSCRIPT REQUIRED: data-name

The referenced data-name must have a subscript to indicate which occurrence of the item is to be accessed.

SUBTRACT STATEMENT

Illegal syntax for a SUBTRACT statement.

SWITCH LITERAL

Illegal syntax for a switch literal; or the literal is not A to Z.

TERMINATOR

Words must be terminated by a space, period, comma, semicolon, or parenthesis. A word may not end with a hyphen.

UNMATCHED PARENTHESES

Parentheses must appear in paired sets.

UNRECOGNIZABLE WORD

The word separator or terminator is not legal in the current context of the program.

USAGE CLAUSE

Illegal syntax for a USAGE clause; or conflict with a group USAGE clause.

USE PROCEDURE DUPLICATE

A USE procedure has already been defined for this file or class of files.

USE STATEMENT

Illegal syntax for a USE statement.

VALUE CLAUSE

Illegal syntax for a VALUE clause.

VALUE NOT PERMITTED IN LINKAGE SECTION

A VALUE clause must not be used for any Linkage Section data item.

VALUE NOT PERMITTED IN OCCURS

A VALUE clause must not be used in an item with an OCCURS clause or an item subordinate to an OCCURS clause.

VALUE NOT PERMITTED IN REDEFINES

A VALUE clause must not be used in an item with a REDEFINES clause or an item subordinate to a REDEFINES clause.

VALUE SPECIFIED FOR GROUP

A VALUE clause has already appeared for an item to which this is subordinate.

WORD DUPLICATE: name

This user-defined word has already been defined (as another type; i.e., file-name, data-name, etc.). Both definitions are accepted and correct usage is determined by context.

WORD EXCEEDS 30 CHARACTERS

A programmer-defined word exceeds 30 characters. The word is truncated on the right to 30 characters.

WRITE STATEMENT

Illegal syntax for a WRITE statement.

A.2 Data Validation Error Messages

Interactive COBOL provides a data validation function for interactive screen management. If the data entry does not match the PICTURE for the field, an error message is displayed at the bottom of the screen, the cursor is positioned to the first invalid character of the field, and the system waits for reentry of the data. Below is a list of the data validation error messages and an explanation of each.

CHARACTER MUST BE ALPHABETIC

The permissible characters are the uppercase letters A-Z and the blank.

CHARACTER MUST BE ALPHANUMERIC

An alphanumeric field can contain only graphic (noncontrol) characters.

CHARACTER MUST BE NUMERIC

Permissible characters in a numeric field are digits, a positive or negative sign, and the decimal point.

DATA ENTRY IS REQUIRED

At least one character must be entered.

FIELD DOES NOT PERMIT A DECIMAL POINT

This field must contain an integer.

FIELD DOES NOT PERMIT A SIGN

The PICTURE statement does not allow a sign.

FULL FIELD IS REQUIRED

A character must be entered in each character position of the field.

ILLEGAL EMBEDDED BLANKS

Blanks preceded and followed by other characters are not permitted in a numeric field.

NO DIGITS ENTERED

The field is numeric and cannot be null.

SIGN MUST BE LEFTMOST CHARACTER

The sign must be in the first character position of the field.

SIGN MUST BE RIGHTMOST CHARACTER

The sign must be in the last character position of the field.

TOO MANY DECIMAL PLACES ENTERED

The number of digits to the right of the decimal point exceeds fhe number specified in the PICTURE statement.

TOO MANY DECIMAL POINTS

Only one decimal point is permitted in a numeric field.

TOO MANY INTEGER PLACES ENTERED

The number of digits to the left of the decimal point exceeds the number specified in the PICTURE statement.

TOO MANY SIGNS ENTERED

Only one sign is allowed in a numeric field.

TRANSMISSION ERROR, REENTER LAST CHARACTER

If the terminal has a dial-up connection, this could be because the character has not been successfully transmitted over the line. If the terminal has a local connection, the error may be because the BREAK key has been pressed or the setting of the PARITY switch on the terminal is incorrect.

A.3 Runtime Error Messages

In the event of a fatal error in the execution of a COBOL program, the runtime system displays an error message on the bottom line of the terminal screen and halts program execution.

Error messages have the form:

ERROR: error text relative-procedure-address [COBOL PC =]

The relative-procedure-address is a five-digit number that the Interactive COBOL compiler inserts in an output listing in place of a procedure name's line number. It gives the approximate location of the statement that caused the error. The text of the runtime system error messages and an explanation of each are given below.

CONVERSION ERROR. COBOL PC =

An error has occurred in converting an index or subscript.

FATAL COBOL PROGRAM I/O ERROR. PROGRAM ABORTED. COBOL PC =

An I/O error, for which no USE procedure applies, has occurred. See the documentation of the USE statement in the Interactive COBOL Programmer's Reference, DGC 093-705013.

INDEX REGISTER OVERFLOW. COBOL PC =

An index or subscript is negative or exceeds 65,535.

INDEXED/RELATIVE FILE INCOMPATIBILITY. COBOL PC =

The record size or key specified for a relative or indexed file, or the key position specified for an indexed file, does not agree with that of the existing file; or the ISAM revision levels are incompatible.

NOT A LEGAL COBOL PROGRAM FILE.

The object program files are unacceptable. Check the COBOL source code and recompile the program.

PERFORM <n> TIMES. COBOL PC =

The value of n exceeds 32,768.

PERFORM STACK OVERFLOW. COBOL PC =

30 active PERFORM statements are permitted. The number of active PERFORM statements has exceeded this limit.

PROGRAM TOO LARGE.

The program size exceeds the maximum for the given runtime system and system configuration.

REVISION INCOMPATIBILITY.

The revision number of the called program's compiler does not agree with that of the runtime system.

SUBSCRIPT OUT OF RANGE. COBOL PC =

An index or subscript is zero or greater than the maximum occurrence value of the item.

UNDEFINED PROCEDURE. COBOL PC =

An attempt has been made to execute an undefined procedure.

--End of Appendix--

Index

DataGeneral

READERS COMMENT FORM

MP/OS Interactive COBOL User's Guide

No. 069-705013-00

Your comments will help us improve the quality of this publication. If your answer to a question is "NO" or requires qualification, please explain.

HOW DID YOU USE THIS PUBLICATION?

- ☐ As an introduction to the subject
- ☐ As an aid for advanced knowledge
- ☐ For information about operating procedures
- ☐ To instruct in a class
- ☐ As a student in a class
- ☐ As a reference manual
- ☐ Other *(please explain)*:

DID YOU FIND THE MATERIAL:

	YES	NO		YES	NO
• Useful	☐	☐	• Well illustrated	☐	☐
• Complete	☐	☐	• Well indexed	☐	☐
• Accurate	☐	☐	• Easy to read	☐	☐
• Well organized	☐	☐	• Easy to understand	☐	☐
• Well written	☐	☐			

We would appreciate any other comments. For specific annotations please reference page numbers. Your comments will be carefully reviewed by the persons responsible for the writing and publishing of this document.

COMMENTS:

FROM:

Name _____ Title _____

Firm _____ Division _____

Address _____

City _____ State _____ Zip _____

Telephone _____ Date _____

First fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MASS 01772

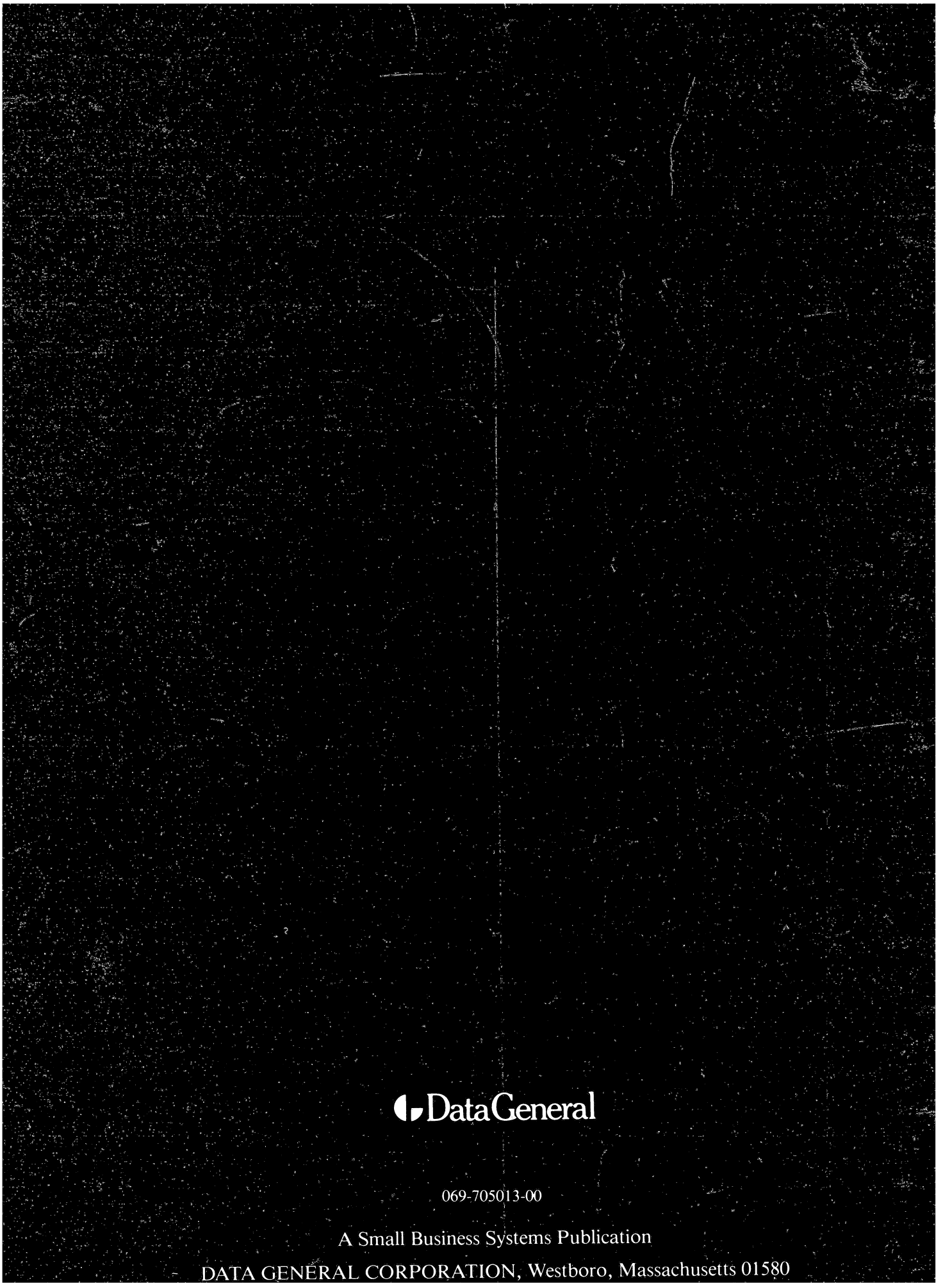
POSTAGE WILL BE PAID BY ADDRESSEE:

DataGeneral

ATTN: Small Business Systems Documentation
62 Alexander Drive
Research Triangle Park, NC 27709
USA

Second fold

TAPE CLOSED HERE



069-705013-00

A Small Business Systems Publication

DATA GENERAL CORPORATION, Westboro, Massachusetts 01580