SC28-6469-5
File No. S370-24

Program Product

IBM CMS User's Guide for COBOL

IBM

SC28-6469-5
File No. S370-24

Program Product

IBM CMS User's Guide for COBOL

Program Numbers **5740-CB1**
5740-LM1
5746-CB1
5746-LM4

Release 2.4

Sixth Edition (August 1983)

This is a major revision of, and makes obsolete, SC28-6469-4.

This edition applies to OS/VS COBOL and DOS/VS COBOL, Release 2.4 of Program Products 5740-CB1, 5740-LM1, 5746-CB1, 5746-LM4, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

PREFACE

This publication is intended for the COBOL programmer who is using or is planning to use the following program products under the control of the Conversational Monitor System (CMS) in the virtual machine environment of Virtual Machine Facility/370 (VM/370) and Virtual Machine/System Product (VM/SP):

- *IBM OS/VS COBOL Compiler and Library* (5740-CB1)
- *IBM DOS/VS COBOL Compiler and Library* (5746-CB1)

In this publication, the term COBOL applies to the COBOL program products listed above, unless a more restrictive wording is used. Similarly, the term OS is used to refer to both OS and OS/VS; DOS to DOS and DOS/VS.

It is assumed that the user has a basic understanding of the CMS component and that user of VM/370 or VM/SP, also knows how to write COBOL programs, and is familiar with the contents of the latest editions of the following prerequisite publications.

OS/VS COBOL:

- *IBM OS/VS COBOL Compiler and Library Programmer's Guide,* SC28-6483
- *IBM Virtual Machine Facility/370: CMS User's Guide,* GC20-1819
- *IBM Virtual Machine Facility/370: CMS Command and Macro Reference,* GC20-1818
- *IBM Virtual Machine Facility/370: Terminal User's Guide,* GC20-1810
- *IBM Virtual Machine Facility/370: System Messages,* GC20-1808
- *IBM Virtual Machine Facility/370: Operating Systems in a Virtual Machine,* GC20-1821
- *IBM Virtual Machine Facility/370: CP Command Reference for General Users,* GC20-1820

DOS/VS COBOL:

- *IBM DOS/VS COBOL Compiler and Library, Programmer's Guide,* SC28-6478
- *IBM Virtual Machine/System Product: CMS User's Guide*, SC19-6210
- *IBM Virtual Machine/System Product: System Product Editor Command and Macro Reference*, SC24-5221
- *IBM Virtual Machine/System Product: System Product Editor User's Guide,* SC24-5220.
- *IBM Virtual Machine/System Product: EXEC2 Reference*, SC24-5219

The purpose of this publication is to provide the COBOL programmer with a fundamental understanding of how to properly enter the pertinent CMS commands to invoke the OS or DOS COBOL compiler under CMS.

Under CMS, the OS/VS COBOL compiler can accept and compile any COBOL source program that it can accept and compile under OS/VS1 and OS/VS2. The object code generated by the OS/VS COBOL compiler under

CMS can be executed under control of OS/VS1 and OS/VS2 or it can be executed under CMS with the restrictions noted under "Restrictions on Using OS COBOL Under CMS."

Under CMS, the DOS/VS COBOL compiler can accept and compile any COBOL source program that it can accept and compile under DOS/VSE. The object code generated by the DOS/VS compiler under CMS can be executed under control of DOS/VSE or it can be executed under CMS with the restrictions described under "Restrictions on Using DOS COBOL Under CMS."

Although this publication reviews some of the concepts, terminology, and procedures that are introduced in the prerequisite publications, it does not attempt to explain everything you must know about CMS, VM, and COBOL to program successfully in this environment. You are expected to extrapolate a great deal of information from other publications.

- *IBM OS COBOL Interactive Debug Terminal User's Guide and Reference,* SC28-6465, contains information about how to use the program product IBM OS COBOL Interactive Debug (5734-CB4) to debug an OS COBOL program under CMS.
- *IBM VS COBOL for OS/VS*, GC26-3857, describes the COBOL language, its rules and restrictions, as implemented for OS/VS (Release 2).
- *IBM VS COBOL for DOS/VSE,* GC26-3998, describes the COBOL lanaguage, its rules and restrictions, as implemented for DOS/VSE.

This publication is divided into the following sections:

- An Introduction, which summarizes the facilities of CMS and includes sample terminal sessions, showing the commands necessary to create, compile, link-edit, and execute an OS or a DOS COBOL program in CMS.
- A section for OS/VS COBOL programmers, describing the commands necessary to compile, load and execute OS/VS COBOL programs, as well as the commands necessary to identify and manipulate OS data sets and CMS files.
- A short section describing how to prepare a COBOL program using CMS.
- A section for DOS COBOL programmers, describing the commands necessary to compile, link-edit, and execute DOS COBOL programs, as well as the commands available to manipulate DOS and CMS files.

The sections on OS and DOS COBOL programming each contain a list of the CMS error messages produced by the commands that invoke the compilers, as well as a summary of the restrictions on executing COBOL programs in CMS.

References to Virtual Machine Facility/370 (often abbreviated VM/370) are for OS users; Virtual Machine/System Product (VM/SP) for DOS users.

SUMMARY OF AMENDMENTS

August 1983

New: Programming Feature

The MIGR option of the COBOL command is described. MIGR flags major COBOL language elements that are no longer supported or are supported differently by the VS COBOL II compiler, Program Number 5668-958.

Maintenance: Documentation

Other corrections and clarifications have been made.

DOS/VS COBOL Release 3, 15 May 1981

New: Programming Feature

The text has been updated to reflect changes required for DOS/VS COBOL Release 3. Major features include:

- Enhanced VSAM support
- Expanded physical sequential file capabilities
- Expanded library facilities
- Powerful data manipulation
- Extended computational facilities
- User-defined collating sequences
- Eased programming rules
- COBOL source program debug language
- 1974 ANS Standard language

References to the *IBM VS COBOL for DOS/VSE* publication have been added.

Miscellaneous Changes

Maintenance: Documentation Only

Minor clarifications and corrections have been made, primarily in the OS/VS areas of passing execution-time parameters and using VSAM.

CONTENTS

FIGURES

INTRODUCTION

The Conversational Monitor System (CMS) is a time-sharing system that provides an extensive range of conversational programming capabilities at a remote terminal. The CMS command language uncomplicates file and data handling through the use of simple terminal commands.

Using CMS commands, COBOL source programs can be processed in the CMS environment:

- In OS/VS COBOL under CMS, you can use the COBOL command to both invoke and control the compilation of OS/VS COBOL source programs. The COBOL command invokes the OS/VS COBOL Compiler, and processes the OS/VS COBOL source program in the CMS file you specify. You can specify compiler options in the operand field of the COBOL command. Execution of the compiled program can be in either the OS environment or, with restrictions, in the CMS environment.
- In DOS/VS COBOL, you can invoke and control the compilation of DOS/VS COBOL source programs under CMS through the FCOBOL and OPTION commands. The FCOBOL command invokes the DOS/VS COBOL compiler, which then processes the DOS/VS COBOL source program in the CMS file you specify. You can specify compiler options in the operand field of the OPTION command or use the default system options. Execution of the compiled program can be either in the DOS/VS environment, or, with restrictions, in the CMS environment.

The Control Program (CP) component of VM/370 or VM/SP creates a simulated computer for the programmer to use on a time-sharing basis. The environment of VM/370 or VM/SP is called a "virtual machine environment" because there is a functional simulation of a real computer and its associated input/output devices.

What You Must Know to Use CMS

Before you begin a terminal session, you should know how to perform the operations that are described briefly in this part of the publication. A quick overview of the CMS conventions and procedures presented here may bring to your attention those areas of CMS that deserve greater study in the prerequisite publications.

How to Start and End a Terminal Session

Before you can use the facilities of the CMS component of VM/370 or VM/SP, you must have a valid user identification (userid) and a valid password. These required identifiers are assigned to you by the person responsible for VM/370 or VM/SP at your installation. In a process that is called logon, you identify yourself to the system by entering your userid and password. When you are finished using the system, you signal your wish to stop by performing a logoff. The period between logging on and logging off is called a terminal session.

Note: Throughout this text, entries you make at the terminal are shown in lowercase letters, and OUTPUT THE SYSTEM TYPES AT THE TERMINAL IS SHOWN IN UPPERCASE. These conventions are used to illustrate clearly what you must type in each example.

VM Logon

First, turn on your terminal and establish a connection to your computer. For detailed information regarding how to operate your terminal, or how to dial a multiaccess system to establish communications with a computer, see *IBM VM/SP CMS User's Guide* or *IBM VM/370 Terminal User's Guide.*

Once a communications line is established, your system types a message with contents and format that depend upon your type of terminal. Assuming that you have an IBM 2741 Communications Terminal with a standard IBM SELECTRIC ® character set, the system prints this message for example:

```
xxxxxxxxxxxx            vm/370 online
```

The Xs are meaningless characters which you should ignore. VM/370 ONLINE tells you that a communications line is established. You press the attention (ATTN) key once, and the system responds by unlocking your keyboard.

In this example, where your userid is USER00, you identify yourself to the system and press the RETURN key. Your entry and the system response follows:

```
login user00
ENTER PASSWORD:
```

The system response shows it has accepted your userid. You must type your password and press the RETURN key. Assuming your 2741 terminal is equipped with the Print Inhibit feature, your password is not printed.

If your password is accepted and your logon is complete, the system issues the following message:

```
LOGON AT 10:45:15 EST ON TUESDAY 03/29/76
```

This gives you the time and date you have entered the VM environment.

Now you must initialize CMS. You enter the following:

```
ipl cms
CMS...VERSION 3.0 03/29/76  10:46:15
```

In this example, your entry, followed by pressing the RETURN key, causes a copy of the CMS nucleus to be brought into storage from disk. Once the system issues the CMS...message with the version and modification level, the keyboard is unlocked, and you may enter any CMS command.

If you are going to use the DOS simulation capabilities of CMS, and are going to use the DOS/VS COBOL compiler, you must enter the following commands:

```
access 192 c
set dos on c
```

where "c" refers to the DOS system residence volume.

VM Logoff

To end your terminal session, enter the following:

```
logoff
LOGOFF AT 12.15.30 03/29/76
```

The system gives you the logoff time and date. Logging off ends your terminal session, and you may turn off the power at your terminal.

How to Enter Information at the Terminal

Although every terminal has a typewriter-like keyboard through which you enter information to the system, the features of each keyboard vary from terminal to terminal. For a complete description of features, variations, and restrictions as they apply to VM/370, see *IBM VM/370 Terminal User's Guide*; as they apply to VM/SP, see *IBM VM/SP CMS User's Guide.*

Figure 1 shows how various operations are performed when entering information at an IBM 2741 Communications Terminal:

Operation	How it is accomplished
enter a line	Type a line of input in upper- or lowercase. (Input is shown in lowercase in this publication to distinguish it from system output.) A line may not exceed 130 characters.
end a line	Press the RETURN key.
delete a character	Type the character-delete symbol (@) to delete the preceding character in an input line.
delete characters	Type n character-delete symbols to delete the preceding n characters in an input line.
delete a line	Type the line-delete symbol (¢) to delete the entire input line.

Figure 1. Entering information at the terminal involves various operations.

How to Enter the CMS Commands

CMS commands are merely requests for work, and they are your way to communicate to the system. You enter a command by typing it, along with its operands, at the terminal. Figure 2 on the next page lists the CMS commands according to their uses.

Syntax of a Command

A command consists of a command name followed, usually, by one or more operands. A command name is usually a familiar English word, or combination of words, that describes the function of the command. For example, the COBOL command invokes the OS/VS COBOL Compiler, a language processing program that translates OS COBOL source statements into machine language object code. The format of the command is described fully later in the part of this publication called "The COBOL Command." The format of the DOS COBOL command (FCOBOL) is described in the section entitled "The FCOBOL Command."

Each CMS command has a general purpose and a specific use. Figure 2 lists CMS commands, and their capabilities, that the COBOL programmer will find useful.

Operands provide the specific information that the command requires to perform the work that you request. For example, the operands of the COBOL command identify which file contains the source program you want to compile and which compiler options you wish to select. Figure 3 illustrates the syntax of a CMS command, using the COBOL command as an example.

General Use	CMS Command (OS COBOL)	CMS Command (DOS/VS COBOL)	Specific Purpose
Initialization		SET DOS ON	Initialize CMS/DOS Environment
	MACLIB TXTLIB	SSERV, DSERV, PSERV, RSERV, DOSLIB	Manipulate libraries
File handling control	AMSERV COPYFILE EDIT ERASE FILEDEF LISTDS LISTFILE MOVEFILE PRINT PUNCH READCARD RENAME STATE TYPE	AMSERV COPYFILE EDIT ERASE ASSGN/DLBL LISTDS LISTFILE MOVEFILE PRINT PUNCH READCARD RENAME STATE TYPE	File creation—input from terminal, card, disk, or tape File maintenance—adding, changing or deleting data; processing an existing file against an update file; copying, combining, moving, splitting, and listing disk files
COBOL compilation control	COBOL	OPTION FCOBOL	Compiling source programs written in COBOL language
Program execution		DOSLKED	Link-editing DOS object modules
control	EXEC GENMOD GLOBAL LOAD LOADMOD RUN START	EXEC GLOBAL FETCH START	Loading and running COBOL programs
Debugging control	DEBUG SVCTRACE TESTCOB	DEBUG SVCTRACE	Monitor program execution; examine and change storage

Figure 2. CMS commands grouped according to their uses.

Positional Operands

Positional operands are values that follow the command name in a prescribed sequence. The value may be one or more names, symbols, or integers. Examples of positional operands are filename and yyyyyyy in Figure 3.

Command	Operands
COBOL	*filename* ([SIZE*yyyyyyy*] [DMAP | NODMAP] [SXREF | NOSXREF])

Figure 3. COBOL command example.

The COBOL command has many more operands than are shown here. But Figure 3 illustrates how the format of a CMS command is presented. You must replace "filename," a positional operand, with the name of the actual CMS file that contains the source program you want to compile. Similarly, you must replace "yyyyyyy" with the actual value of the SIZE option. Positional operands are presented in lowercase in CMS formats to show that you must replace the name with a value when you enter the command and its operands at the terminal.

Keyword Operands

Keyword operands are presented in uppercase in CMS formats to show that you must type them exactly as they are shown. In some cases you may specify a value with a keyword (such as "yyyyyyy" with "SIZE").

Delimiters

When you type a command, separate the command name from the operand by one or more blanks, and also separate operands by one or more blanks.

Notation Conventions

Figure 4 summarizes the notations used to define CMS command syntax:

Name	Symbol	Meaning
parentheses	( )	Parentheses must be typed exactly as they appear whenever any of the operands enclosed within them are specified. The closing parenthesis is optional.
underscore	_	This indicates the default option, which you need not type if it is the one you want.
braces	{ }	These group related items; you must choose one of the items.
brackets	[ ]	These also group related items; you may choose one of the items.
ellipsis	...	This indicates that the preceding item(s) may be repeated one or more times in succession.
suffixes 1,2,N	1,2,N	These suffixes denote the first, second, and Nth items respectively.
or symbol	|	This symbol indicates you must choose one of two (or three) operands.

Figure 4. CMS notation defines how to select and use COBOL command operands.

According to the notation in Figure 3, you *must* type the parentheses if you specify any of the operands enclosed within them; you *may* specify the SIZE option; you *may* specify DMAP or NODMAP, and SXREF or NOSXREF; and the defaults NODMAP AND NOSXREF are assumed if you specify neither choice.

Typing the Command

Once you have initialized CMS according to procedures explained earlier in "VM Logon," you may enter any CMS command merely by typing it at the terminal. After you have typed the command name plus the required operands and any optional operands you want to include, correct any mistakes and press the RETURN key.

Note: Although you can type your commands in upper- or lowercase, you may want to choose lowercase so that you can easily distinguish your input at the terminal from the system's output. The examples in this publication show input in lowercase AND OUTPUT IN UPPERCASE for this reason.

How to Use the CMS File Conventions

You must understand the CMS file conventions in order to use the various file-handling facilities of CMS. Whereas you may never need to manipulate files very much, you will always need to specify a file in the operand field of the COBOL or FCOBOL command. This file, which contains the source program you want to compile, must be available to CMS before you invoke COBOL compilation. Because CMS files may be stored on disk, cards, or magnetic tape, you must make CMS aware of where the file resides. The default medium is disk.

In OS, you use the FILEDEF command (and/or DLBL in the case of VSAM) to achieve this function.

In DOS/VS, you use the ASSGN and DLBL commands.

All of these commands are explained fully in *VM/370 CMS User's Guide* and in *VM/SP CMS User's Guide*. Examples are shown in the sample terminal session at the end of this section.

File Identifier

Each file that resides on disk must have a unique identifier that consists of three components: filename, filetype, and filemode. You must specify this identifier, or a portion of it, in the operand fields of various CMS commands that must access user and system files. For example, you must specify the filename in the operand field of COBOL.

The *filename* must be any combination of eight-or-fewer alphanumeric characters (A-Z, 0-9, #, @, $). This user-supplied name has no special implications for CMS.

The *filetype* may be any combination of eight-or-fewer alphanumeric characters. Certain filetypes imply specific characteristics to CMS. Although you may assign any of the reserved filetypes to any file you create, you must ensure that the contents of your file conform to the required format for each filetype. Commands that use particular filetypes do not execute successfully when the contents of the file are not in the expected, and required, format.

Files that contain COBOL source code must have a filetype of COBOL. How to create such a file and ensure that the filetype, as well as the other fields of the identifier, are correct is described later in "Preparing a COBOL Source Program with the CMS Editor."

The *filemode* consists of two characters, one which indicates the disk directory in which the file resides and the other which specifies certain user-determined attributes such as whether the file may be written or read.

Filemodes are described in detail in *VM/370 CMS User's Guide* and in *VM/SP CMS User's Guide.*

Using the CMS Features

The following features are included here to provide a brief overview of the various operations you can perform, along with COBOL compilation, under CMS. See the prerequisite CMS publications for additional information on the topics covered here.

CMS Commands

A CMS command is the name of a program resident in a discontiguous saved segment (DCSS), in the nucleus or on any CMS disk, or the name of a file containing CMS commands. The CMS commands are summarized according to their uses in Figure 2 and are explained in detail in *VM/370: CMS Command and Macro Reference* and in *IBM VM/SP: System Product Editor Command and Macro Reference*. Usage information is contained in *VM/370: CMS User's Guide* and in *VM/SP User's Guide.*

Execution Control

The execution control commands allow you to load your object programs from data sets called TEXT files (the required filetype for object programs is TEXT) and then to execute the object program. To simplify execution control, you can create your own command language or command procedures and place logic statements in the file with the commands so that the order of command execution can be dynamically set or altered. You may use the EXEC command to execute the command procedures.

Debugging Facilities

A permanent, nucleus-resident debugging facility is available which enables you to interrupt program execution at predetermined points and then to examine and modify if necessary the registers, program status word (PSW), and storage.

In OS, the program product IBM OS COBOL Interactive Debug (5734-CB4) is also available for debugging OS COBOL programs under CMS. (The TEST compiler option and the TESTCOB command are explained later in this publication.)

Utilities

The utility functions in CMS enable you to copy tapes, compare and sort CMS disk files, create VSAM catalogs, data spaces and files, copy the contents of a 3270 screen on a 3284 printer, and dump files either by name onto the console or by cylinder location onto the offline printer. You can also dump files to tape and reload them onto disk.

Control Commands

The control commands enable you to suppress and restore the typed output at your terminal, redefine the line-end, character-delete, and line-delete symbols, rename commands, and define abbreviations.

Library Facilities

OS users can create macro libraries, containing COBOL source statement copy files. These libraries, called MACLIBs in CMS, can be manipulated using the MACLIB command. You can specify macro libraries to be searched at compilation time with the GLOBAL command. Similarly, you can use text libraries, or TXTLIBs, to contain object modules that are referenced by other programs.

DOS users can directly access the DOS/VS system residence file, and use the system or private source statement, relocatable, and core image libraries. By using the ASSGN and DLBL commands, you can copy from the source statement library at execution time. The CMS commands DSERV, SSERV, PSERV, and RSERV allow you to access and copy files from the DOS system or from private libraries.

PREPARING A COBOL SOURCE PROGRAM WITH THE CMS EDITOR

When you create a COBOL source file using the CMS Editor, you must assign a filetype of COBOL to the file. You can assign any eight-character filename you wish. If you are creating a new file, you must be sure not to assign a name that is the same as that of an existing COBOL file on the same disk.

To create a COBOL file named MYFILE, you would enter

```
edit myfile cobol
```

The CMS Editor recognizes the filetype of COBOL, and assigns the following characteristics to the file:

- Tab settings are 1, 8, 12, 20, 28, 36, 44, 68, 72, and 80. When you press the tab key on your terminal (or equivalent), the Editor will use these internal tab settings to provide the proper number of spaces on the input line.
- All input is translated to uppercase characters, regardless of how you enter it, unless you first specify CASE M, to enter lowercase letters. For example, if you need to enter a program response or special identification in lowercase, you can specify

  ```
  case m
  ```

  in edit mode, insert the line, and then specify CASE U to continue having your input translated to uppercase.
- All input lines are truncated in column 72. If you try to enter a line longer than 72 characters, the Editor truncates the line and issues the message

  ```
  TRUNCATED.
  ```

  followed by a display of the line as it was entered into the file. If you want to modify the line before continuing to enter input, you must enter a null line to return to edit mode.
- Input records are automatically serialized in columns 76 through 80.

Entering Input Lines

When you are creating a source file, or adding lines to an existing file, you use the EDIT subcommand INPUT. For example,

```
edit myfile cobol
NEW FILE.
EDIT:
input
```

You must press the tab key (or equivalent) in front of every line to ensure the proper spacing. In the examples below, the percent sign (%) indicates pressing of the tab key:

```
%identification division.
%program-id. myprog.
%environment division.
```

If you display these lines, they appear as they have been interpreted by the Editor:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MYPROG.
ENVIRONMENT DIVISION.
```

Note that the pressing of the tab key resulted in the input lines beginning in column 8. Pressing the tab key twice before entering text on an input line would result in the text being aligned in column 12. Thus, the first two COBOL tab settings correspond to Area A and Area B for a COBOL source file.

The publications *IBM VS COBOL for OS/VS* and *IBM VS COBOL for DOS/VSE* describe how to write COBOL programs in standard format. If you are using a 3270 terminal, there is no functional tab key supported by VM.

Filing a Source File

When you have finished creating a source file, you may want to use the CMS Editor to make corrections, additions, and deletions. Then, when you are ready to save the file on a CMS disk, issue the subcommand

```
file
```

To store the file on disk.

Serializing Records

You can place serial or line numbers in columns 1 through 5 of your COBOL source files by using the EDIT subcommand LINEMODE. After you enter the EDIT environment, and before you enter input mode, enter the subcommand

```
linemode left
```

You can then use line-number editing to create the file. When you are in input mode, the Editor prompts you to enter an input line:

```
10%This is the first line.
20%This is the second line.
```

The Editor right-aligns the serial numbers, and pads with blanks to the left.

When you are in edit mode, and using EDIT subcommands to make changes to the file, you can use the LINEMODE subcommand to refer to lines by line number. For example, if you enter

```
30
```

the Editor's current line pointer is positioned at the line with a line number of 30.

Note: If you use line-number editing for a COBOL file, you might wish to use the PROMPT subcommand to specify increments greater than 10 for input lines in the file you are creating. Once a COBOL file has been created with line numbers on the left, there is no way for the Editor to resequence the file numbers.

If you want to place sequence numbers in columns 76 through 80, you can use the SERIAL subcommand.

Renaming Existing Source Files

Because all disk files to be used as input to the compiler must have a filetype of COBOL, if you have a COBOL source file that exists with some other filetype, you can rename it using the CMS RENAME command:

```
listfile myfile *
MYFILE SOURCE   A1
R;
rename myfile source * myfile cobol *
R;
```

The input file, MYFILE SOURCE, must be in the proper format for the COBOL compiler.

Sample Terminal Session for an OS User

A terminal session is everything that happens at your terminal between logon and logoff. The sample terminal session shown in Figure 5 shows you how to do the following, where the numbers refer to the numbers in the sequence below:

1. Create the source program
2. Compile the program and create a text file of the object program
3. Create an input file and identify it
4. Load the COBOL subroutine library and program
5. Execute the program

The entire terminal session is shown below to provide an overview of the program creation, compilation, and development process. Details are contained in the section "Using OS COBOL Under CMS."

```
edit myfile cobol
NEW FILE.
EDIT:
input
INPUT:
00010%identification division.
00020%program-id. myprog.                                  1
00030%environment division.
   .
   .
   .
   (enter a null line by pressing the RETURN KEY)
EDIT:
file
R;
cobol myfile quote
   .
   .
 (Progress, diagnostic, and compiler messages              2
         appear here.)
   .
   .
R;
edit infile data
NEW FILE.
EDIT:
input
INPUT:
alpha
bravo
charlie                                                    3
   .
   .
   (enter a null line by pressing the RETURN KEY)
EDIT:
file
R;
filedef ddfile disk infile data
R;
global txtlib vscoblib
R;
load myfile                                                4
R;
start
EXECUTION BEGINS...
   .                                                       5
   .
R;
```

Figure 5. Sample terminal session for an OS user.

Sample Terminal Session for a DOS User

A terminal session is something that happens at your terminal between logon and logoff. The sample terminal session in Figure 6 shows how to do the following, where the numbers refer to the sequence below:

1. Create the source program file
2. Access the DOS system core/image library as file mode C and initiate the CMS/DOS enrivonment
3. Set compiler options (such as LISTX), assign the source input device, in this case, your CMS A-disk, and compile the program
4. Examine the compiler output listing
5. Create an input file
6. Link-edit the object program and identify the DOS phase library
7. Identify the input file
8. Load the program into virtual storage
9. Execute the program

To provide an overview of the program creation, compilation, and execution process, the entire terminal session is shown below. Details are contained in the section "Using DOS COBOL Under CMS."

```
edit myfile cobol
NEW FILE.
EDIT:
input
INPUT:
 cbl quote                                                  1
00010%identification division.
00020%program-id. myprog.
00030%environment division.
   .
   (enter a null line by pressing the RETURN key)
EDIT:
file
R;
access 192 c
C (192) R/O_DOS                                             2
R;
Set dos on c
R;
option listx
R;                                                          3
assgn sysipt a
R;
fcobol myfile
R;
edit myfile listing
EDIT:
locate/message
 (Progress, diagnostic, and compiler messages               4
         appear here.)
   .
R;
edit infile data
NEW FILE.
EDIT:
input
INPUT:                                                      5
alpha
bravo
charlie
   .
   (enter a null line by pressing the RETURN key)
EDIT:
file
R;
doslked myfile mylib
R;
global doslib mylib                                         6
R;
assgn sys009 a
R;
dlbl ddfile a cms infile data (sys009                       7
R;
fetch myfile
PHASE MYFILE ENTRY POINT AT LOCATION 020000                 8
R;
start
EXECUTION BEGINS...
   .                                                        9
   .
R;
```

Figure 6. Sample terminal session for a DOS user.

USING OS COBOL UNDER CMS

When you enter the COBOL command at your terminal, it invokes the OS/VS COBOL Compiler, a language processing program that translates COBOL source statements into machine-language object code.

When you invoke the COBOL command, you must specify the filename of the COBOL source file, and, optionally, any compiler options that you want in effect for the compilation. These options are equivalent to the options you would specify on the PARM parameter of an EXEC job control statement, if you were invoking the compiler under OS.

When the compiler finishes executing, it displays any messages indicating compilation errors on your terminal, as well as writing them into the

LISTING file, which is described below. You can usually identify a problem from the diagnostic message you see displayed at the terminal, so that you do not have to examine a printed listing file. When you have an error, you can correct the source file immediately and recompile it.

Under CMS, object files created by the compiler have a filetype of TEXT. TEXT files can be loaded into your virtual storage area and executed in CMS using the LOAD and START commands.

The COBOL Command

The format of the OS COBOL command is shown below (defaults are underscored):

```
COBOL | CO   filename([SIZEyyyyyyy] [BUFyyyyy] [LINECNTnn])
             [ADV | NOADV]
             [BATCH | NOBATCH]
             [CDECK | NOCDECK]
             [CLIST | NOCLIST]
             [COUNT | NOCOUNT]
             [CSYNTAX | NOCSYNTAX]
             [DECK | NODECK]
             [DISK | PRINT | NOPRINT]
             [DMAP | NODMAP]
             [DUMP | NODUMP]
             [DYNAM | NODYNAM]
             [ENDJOB | NOENDJOB]
             [FDECK | NOFDECK]
             [FLAGE | FLAGW]
             [FLOWnn | NOFLOW]
             [LANGLVL1 | LANGLVL2]
             [LCOL1 | LCOL2]
             [LIB | NOLIB]
             [LOAD | NOLOAD]
             [LSTCOMP | NOLST | LSTONLY]
             [LVLy | NOLVL]
             [L120 | L132]
             [MIGR | NOMIGR]
             [NAME | NONAME]
             [NUM | NONUM]
             [OPTIMIZE | NOOPT]
             [OSDECK]
             [PMAP | NOPMAP]
             [QUOTE | APOST]
             [RESIDENT | NORES]
             [SEQ | NOSEQ]
             [SOURCE | NOSOURCE]
             [SPACE1 | SPACE2 | SPACE3]
             [STATE | NOSTATE]
             [SUPMAP | NOSUPMAP]
             [SXREF | NOSXREF]
             [SYMDMP | NOSYMDMP]
             [SYNTAX | NOSYNTAX]
             [SYST | SYSx]
             [TERM | NOTERM]
             [TEST | NOTEST]
             [TRUNC | NOTRUNC]
             [VBREF | NOVBREF]
             [VBSUM | NOVBSUM]
             [VERB | NOVERB]
             [XREF | NOXREF]
             [ZWB | NOZWB]
```

COBOL | CO
is the unique command word that must be entered in one of its two forms to invoke the OS COBOL compiler you're using.

filename
specifies the filename of the file to be compiled. The file must have a filetype of COBOL, which implies that the file contains fixed length records.

You can compile a source file from a medium other than a CMS disk, for example, an OS disk, a tape, or your virtual card reader. If you need to do this, you must use the FILEDEF command to identify the COBOL input, and you must use a ddname of COBOL. For example, if the source file is in your card reader, enter the commands

```
filedef cobol reader
cobol testfile
```

If the source file was on an OS disk accessed as your C-disk, the following sequence of commands would be used:

```
filedef cobol c dsn test cds
cobol testfile
```

Output files produced by the compiler will have a CMS filename of TESTFILE, because that is the name specified on the COBOL command line. You can also compile a source file from an OS disk, using the FILEDEF command to identify the OS data set name of the file. The options that follow the filename control the COBOL compiler operation and output. You may specify these options in any order within a set of parentheses. If an option has an acceptable abbreviation, its abbreviation is included in the following pages.

SIZE*yyyyyyy* | SIZE*yyy*K
indicates the amount of virtual storage, in bytes, available for compilation. *yyyyyyy* is an integer from 131072 to 9999999 for OS/VS COBOL. 9999999 instructs the compiler to obtain as much virtual storage as possible. The SIZE parameter can be given in multiples of K, where K = 1024 decimal bytes.

Default: 131072 is the default for OS/VS COBOL.

BUF*yyyyy* | BUF*yyy*K
indicates the amount of virtual storage to be allocated to buffers. If both SIZE and BUF are specified, the amount allocated to buffers is included in the amount of virtual storage available for compilation. *yyyyy* is an integer from 4096 to 99999 for OS/VS COBOL. The BUF parameter can be given in multiples of K, where K = 1024 decimal bytes.

If BUF is omitted, and SIZE is specified, the value of BUF for OS/VS COBOL is calculated as:

$$\frac{\textbf{SIZE-98304}}{4} + 4096$$

Defaults: For OS/VS COBOL, 4096 is the default if both SIZE and BUF are omitted.

LINECNT*nn* | CNT
indicates the number of lines to be printed on each page of the output listing. *nn* is a two digit number from 01 to 99.

Default: If the parameter is omitted a value of 60 is assumed.

ADV
specifies that records for files with WRITE...ADVANCING need not reserve the first byte for the control character.

NOADV
specifies that records for files with WRITE...ADVANCING need to reserve the first byte for the control character.

BATCH | BAT
specifies that the file named 'filename' consists of multiple programs and/or subprograms to be compiled with a single invocation of the COBOL compiler.

NOBATCH | NOBAT
specifies that a multiple program compilation is not being performed.

CDECK | CDE
specifies that COPY statements are to be expanded in the reformatted deck requested through FDECK.

NOCDECK | NOCDE
specifies that no COPY members will be expanded in the reformatted deck.

CLIST | CLI
specifies that a condensed listing of the compiler generated object code is to be produced.

NOCLIST | NOCLI
specifies that a condensed listing is not to be produced.

COUNT | COU
specifies that code is to be generated to produce verb execution summaries at the end of problem program execution. Each verb is identified by procedure-name and by statement number, and the number of times it was used is indicated. In addition, the percentage of verb execution for each verb with respect to the execution of all verbs is given. Also, a summary of the executable verbs used in a program and the number of times they are executed is provided. If COUNT is specified, SYSCOUNT is dynamically allocated.

NOCOUNT | NOCOU
specifies that no COUNT information is to be provided.

CSYNTAX | CSYN
specifies that conditional syntax checking and object code generation of the COBOL source file are to be performed. A full compilation is produced as long as no messages exceed the W or C level. If one or more E-level or higher severity messages are produced, the compiler generates the messages but does not generate the object text.

NOCSYNTAX | NOCSYN
specifies (if your installation has not changed the defaults) that the NOSYNTAX option is in effect. If your installation's default is CSYNTAX, then you must specify NOCSYNTAX in order to specify SYNTAX.

DECK | DEC
specifies that the output object module is to be punched on the spooled punch.

NODECK | NODEC
specifies that the object module is not to be punched on the spooled punch.

DISK | DI
specifies that a program listing is to be produced as specified under the PRINT option with the exception that the listing will be written to the appropriate read/write disk with a filetype of LISTING.

PRINT | PRI
specifies that a program listing is to be printed containing page headings, line numbers of the statements in error, message identification numbers, severity levels, and message texts (as well as any other output requested by CLIST, DMAP, LSTCOMP, LSTONLY, PMAP, XREF, SOURCE, SXREF, VBREF, or VBSUM). The listing will be printed at the offline printer.

NOPRINT | NOPRI
no LISTING file will be produced.

DMAP | DMA
specifies that the glossary, global tables, literal pools, and register assignments are to be listed.

NODMAP | NODMA
specifies that a DMAP is not to be produced.

DUMP | DUM
specifies that there is to be an abnormal termination at the point of failure if a disaster situation occurs.

NODUMP | NODUM
specifies that there is to be no abnormal termination at the point of failure if a disaster situation occurs. However, the D-level message is produced.

DYNAM | DYN
causes subprograms invoked through the 'CALL literal' statement to be dynamically loaded, and through the 'CANCEL' statement to be dynamically deleted at object time instead of automatically loaded when the calling program is loaded.

NODYNAM | NODYN
causes subprograms to be statically loaded with calling programs at the time the calling program is loaded.

ENDJOB | END
specifies that, at the end of a COBOL run unit, subroutines are to be called to close DCBs opened by subroutines and free their associated buffers, free storage acquired through GETMAINs, and delete modules (RES). Specifying ENDJOB prevents fragmentation of storage for programs executed after the COBOL program in the same job step. This option takes effect at a GOBACK statement in a main program or at a STOP RUN statement in any program. If any program executed in a run unit specifies ENDJOB, ENDJOB is in effect for the entire run unit.

NOENDJOB | NOEND
specifies that the ENDJOB procedures described above are not to occur.

FDECK | FDE
specifies that a copy of the reformatted listing is to be written on the SYSPUNCH data set. Since FDECK has meaning only with either LSTONLY or LSTCOMP, the lister output will be both a reformatted listing and a reformatted deck.

NOFDECK | NOFDE
specifies that no copy of the reformatted listing is to be written on the SYSPUNCH data set.

FLAGE | LAG
specifies that only error messages are to be printed.

FLAGW | LAGW
specifies that all compilation error and warning messages are to be printed.

FLOW*nn* | FLO
specifies that a formatted trace of a variable number of procedures executed before an abnormal termination is to be listed. Where *nn* specifies the number of procedures.

NOFLOW | NOFLO
specifies that a trace is not to be produced.

LANGLVL1 | LANGLVL2
LANGLVL applies only to those few language elements where the 1968 and the 1974 standards have different interpretations. LANGLVL1 specifies the COBOL source will be interpreted using the 1968 American National Standard COBOL (X4.23-1968). LANGLVL2 specifies that COBOL source will be interpreted using the 1974 American National Standard COBOL (X3.23-1974).

LCOL1 | OL1
specifies that the Procedure Division part of the reformatted listing is to be in single column format.

LCOL2 | OL2
specifies that the Procedure Division part of the lister output listing is to be in double column format.

LIB
specifies that data requested by a COPY statement in the source or a BASIS card in the input stream will be searched for in the macro definition libraries now in effect. If the COPY or BASIS statements are used, LIB must be specified.

NOLIB
specifies that, because COPY or BASIS statements will not be encountered, the macro definition libraries will not have to be searched.

LOAD | LOA
specifies that a TEXT file is to be produced.

NOLOAD | NOLOA
specifies that a TEXT file is not to be produced.

LSTCOMP | LSTC
specifies that the lister feature is to be used and that both a reformatted listing is to be produced and compilation is to occur in the same job step.

NOLST
specifies that the lister feature is not to be used.

LSTONLY | LSTO
specifies that the lister feature is to be used and that a reformatted listing is to be produced but no compilation is to occur.

LVL*y*

specifies that the standards deviations from American National Standard COBOL are to be flagged. If flagging is specified, the value of *y* must be one of the following levels of FIPS (Federal Information Processing Standard): A, B, C, or D. Entering any character other than A, B, C, or D will be flagged, and the user will be prompted to reenter a valid character. LVL *y* indicates that source clauses and statements that do not conform to the specified level of FIPS are to be identified. No listings will be produced. The LANGLVL compiler option, discussed later, causes FIPS flagging to be done according to either the 1975 FIPS (LANGLVL 2) for 1974 ANS COBOL, or to the 1972 FIPS (LANGLVL 1) for 1968 ANS COBOL. A complete list of statements flagged at each level is contained in *IBM VS COBOL for OS/VS*.

Note: If LVL *y* is the SYSGEN default, the default level y, whatever it is, can be overridden at compile time with any other level, but not overridden by NOLVL. Also, LVL *y* requires the SYSUT6 data set, which is dynamically allocated under CMS.

NOLVL

specifies that standards deviations are not to be flagged.

L120 | L12

specifies that the length of each line of the reformatted listing is to be 120 characters.

L132 | L13

specifies that the length of each line of the reformatted listing is to be 132 characters.

MIGR | NOMIGR

specifies whether or not migration flagging is to occur. If MIGR is specified, both syntax and semantics as well as specific language elements are flagged.

MIGR issues I-level (informational) messages for those COBOL statements that are no longer supported or are supported differently by the VS COBOL II compiler.

Notes:

- The SYNTAX option can be used with the MIGR option. The MIGR option does not affect CSYNTAX. That is, if CSYNTAX is in effect and only migration messages are generated, the compilation will *not* be terminated with listing and object output inhibited.
- See *IBM VS COBOL for OS/VS* for a list of the statements that are flagged and those that are not flagged.
- Informational messages issued when you specify MIGR do not affect the compiler return code.

NAME | NAM

specifies that TEXT files generated from the batch compilation environment will contain 'NAME' records. The program names on the 'NAME' records will be formed according to the rules for forming module names from the program-name.

NONAME | NONAM
: specifies that 'NAME' records will not be produced for the 'TEXT' files generated from batch compilations.

  Note: If the BATCH option is not specified, NONAME will be in effect.

NUM
: specifies that user-supplied line numbers recorded in the input are to be used instead of compiler-generated statement numbers wherever such numbers are listed.

NONUM
: specifies that compiler generated line numbers are to be used.

  Note: If when the NUM option is in effect the compiler discovers a nonnumeric character in a line number or if ascending numeric sequence is broken and one or more of the debugging options are in effect, the compiler begins generating card numbers at that point. The increment of these card numbers is one.

OPTIMIZE | OPT
: specifies that the compiler generate optimized object code, considerably reducing the use of object program main storage.

NOOPT
: causes no optimized object code generation.

  Note: The OPTIMIZE feature is automatically in effect when the SYMDMP option is specified. In general, the greater the number of COBOL PROCEDURE DIVISION statements, the greater the percentage of reduction in the amount of main storage required.

OSDECK | OSD
: specifies that the object program is to be executed under OS or as a subprogram under CMS. If OSDECK is not specified, it is assumed that the object program is to be executed as a main program under CMS; this is the default.

PMAP | PMA
: specifies that global tables, literal pools, register assignments, and assembler language expansion of the source program are to be listed.

NOPMAP | NOPMA
: specifies that a PMAP is not to be produced.

QUOTE | QUO
: specifies that the double quote (") character is to be used to delineate literals, and also used by the compiler to delineate figurative constants.

  Note: If double quote is the default logical escape character for your CMS Editor, you may wish to use APOST instead.

APOST | APO
: specifies that the apostrophe (') is to be used in the above situations.

RESIDENT | RES
: requests the COBOL Library Management feature to be in effect. COBOL Library subroutines will be dynamically loaded at execution time rather than statically loaded by the LOAD command.

NORES
specifies no COBOL Library Management feature.

Note: If both NORES and NODYNAM are either specified or implied by default, and a 'CALL identifier' statement occurs in the source statements being compiled, the RESIDENT option is automatically in effect.

SEQ
specifies that the compiler is to check the sequence of the source module statements.

NOSEQ
specifies that sequence checking is not to be performed.

SOURCE | SOU
specifies that the source module is to be listed.

NOSOURCE | NOSOU
specifies that the source module is not to be listed.

SPACE1 | ACE1
specifies that single spacing is to be used on the source card listing generated when SOURCE is specified.

SPACE2 | ACE2
specifies that double spacing is to be used in the above condition.

SPACE3 | ACE3
specifies that triple spacing is to be used in the above condition.

STATE | STA
specifies that during execution the number of the statement and the verb being executed at the time of abnormal termination is to be listed.

NOSTATE | NOSTA
specifies that the above listing is not to be produced.

SUPMAP | SUP
specifies that object code listing, and object modules are to be suppressed if an E-level or D-level message is generated by the compiler.

NOSUPMAP | NOSUP
specifies that the above items are not suppressed when an E-level or D-level message is generated by the compiler.

SXREF | SXR
specifies that a sorted cross-reference listing is to be produced.

NOSXREF | NOSXR
specifies that a sorted cross-reference listing is not to be produced.

SYMDMP | SYM
requests a formatted dump of the data area of the object program at abnormal termination. With this option the programmer can request dynamic dumps of specified data names at strategic points during program execution.

NOSYMDMP | NOSYM
specifies that no formatted dumps be produced.

Note 1: If the SYMDMP option is in effect, the SYSUT5 utility file will be produced. If the BATCH option is specified, the SYMDMP option is negated. Specification of the SYMDMP option automatically yields the OPTIMIZE feature, and negates the STATE option.

Note 2: If WITH DEBUGGING MODE and USE FOR DEBUGGING declaratives are both specified, TEST will be canceled.

SYNTAX | SYN
specifies that the COBOL source file is to be scanned for syntax errors only and appropriate error messages are to be generated.

NOSYNTAX | NOSYN
specifies that normal compilation with both syntax checking and object code generation is to be performed.

Note: When the SYNTAX option is in effect, all of the following compile-time options are suppressed:

LOAD	PMAP	FLOW
XREF	DECK	STATE
SXREF	SYMDMP	NAME
CLIST	TRUNC	RESIDENT
NOSUPMAP	OPTIMIZE	COUNT
LSTCOMP	LSTONLY	VBREF
VBSUM		

If both SYNTAX and OPTIMIZE are specified, no object code is produced.

Unconditional syntax checking is assumed if all of the following compile time options are specified:

NOLOAD	NOCLIST	SUPMAP
NOXREF	NOPMAP	NODECK
NOSXREF	NOLST	NOVBREF
NOVBSUM		

SYST
specifies that SYSOUT is the ddname of the file to be used for debug output and for data when SYSOUT is specified (implicitly or explicitly) in a DISPLAY statement.

SYS*x*
specifies that SYSOU*x* (where '*x*' is an alphanumeric character) is the ddname of the file to be used for debug output and for data when SYSOUT is specified (implicitly or explicitly) in a DISPLAY statement.

Note: The user must issue a FILEDEF with ddname 'SYSOU*x*' at execution time if the SYS*x* compiler option is selected.

TERM | TER
specifies that progress and diagnostic messages and compiler statistics are to be printed at the terminal. No listings are produced.

NOTERM | NOTER
specifies that progress and diagnostic messages and compiler statistics are not to be printed at the terminal.

TEST
specifies that the program can be debugged at the terminal using the program product *IBM OS COBOL Interactive Debug* (Program Number 5734-CB4).

Note: TEST has the following effects on other compiler options: TEST overrides FLOW, STATE, and SYMDMP. However, BATCH overrides TEST.

If WITH DEBUGGING MODE and USE FOR DEBUGGING declaratives are both specified, TEST will be canceled.

Complete information on OS COBOL Interactive Debug, which includes the TESTCOB command and its subcommands, is contained in *IBM OS COBOL Interactive Debug Terminal User's Guide and Reference.*

NOTEST
specifies that the program cannot be debugged at the terminal using the program product OS COBOL Interactive Debug.

Note: A program compiled with NOTEST may be executed under OS COBOL Interactive Debug in combination with programs compiled with TEST, provided all programs are compiled with the RES option.

TRUNC | TRU
specifies that a computational (binary) is moved to a receiving field according to the specification in the PICTURE clause during a move operation.

NOTRUNC | NOTRU
specifies that the item is moved according to the size of the field in storage.

VBREF | VBR
specifies that a cross reference of all verbs used in the program is to be provided. VBREF gives the user a quick index to any verb used in the program.

NOVBREF | NOVBR
specifies that a cross reference of all verbs is not to be produced.

VBSUM | VBS
specifies that there is to be provided a brief summary of verbs used in the program and a count of how often each verb was used.

NOVBSUM | NOVBS
specifies that there is to be no summary and count of verb use.

VERB
specifies that procedure-names and verb-names are to be listed with the associated code on the object program listing. VERB has meaning only if PMAP or CLIST is in effect.

NOVERB
specifies that no procedure-names and verb-names are to be listed. NOVERB yields more efficient compilation.

XREF | XRE
specifies that an unsorted cross-reference listing is to be produced.

NOXREF | NOXRE
specifies that an unsorted cross-reference listing is not to be produced.

ZWB
specifies that the compiler will generate code to strip the sign from a signed external decimal field when comparing this field to an alphanumeric field.

NOZWB
specifies that the compiler will not generate this code.

Entering Command Options

You can enter as many options as you wish on a command line, to a maximum of 100 characters, including blanks. If you specify conflicting options, such as XREF and NOXREF, the last one you entered is in effect. Exceptions to this rule are noted in the operand descriptions. Accordingly, you may change one or more options without reentering the entire command line.

If any of the following mutually exclusive options are specified, the last to appear in the command will have priority:

```
CLIST - PMAP
XREF - SXREF
PRINT - DISK
```

You can abbreviate any command options to the minimum truncation shown in the option descriptions. The only exceptions are that for the options NOOPTIMIZE and NORESIDENT, only the abbreviations of NOOPT and NORES are acceptable.

The compiler options, default values, and alternate names are summarized in Figure 7.

Option	**Default**	**Alternate Names**
APOST	QUOTE	APO,QUO
BATCH	NOBATCH	BAT,NOBAT
BUF *yyyyy*	4096	BUF *yyy* K
CDECK	NOCDECK	CDE,NOCDE
CLIST	NOCLIST	CLI,NOCLI
COUNT	NOCOUNT	COU,NOCOU
CSYNTAX	NOCSYNTAX	CSYN,NOSYN
DECK	NODECK	DEC,NODEC
DMAP	NODMAP	DMA,NODMA
DUMP	NODUMP	DUM,NODUM
DYNAM	DODYNAM	DYN,NODYN
ENDJOB	NOENDJOB	END,NOEND
FDECK	NOFDECK	FDE,NOFDE
FLAGE	FLAGW	LAG,LAGW
FLOW *nn*	NOFLOW	FLO,NOFLO
LANGLVL1	LANGLVL2	
LCOL1	LCOL2	OL1,OL2
LIB	NOLIB	
LINECNT *nn*	60	CNT
LSTCOMP	NOLST	LSTC
LSTONLY	NOLST	LSTO
LVL *y*	NOLVL	
L120	L132	L12,L13
MIGR	NOMIGR	MIG,NOMIG
NAME	NONAME	NAM,NONAM
NOADV	ADV	
NOLOAD	LOAD	NOLOA,LOA
NOPRINT	DISK	DI,NOPRI
NOSEQ	SEQ	
NOSOURCE	SOURCE	NOSOU,SOU
NOTERM	TERM	NOTER,TER
NOTRUNC	TRUNC	NOTRU,TRU
NOVERB	VERB	
NOZWB	ZWB	
NUM	NONUM	
OPTIMIZE	NOOPT	OPT
OSDECK	NOOSDECK	OSD
PMAP	NOPMAP	PMA,NOPMA
PRINT	DISK	PRI,DI
RESIDENT	NORES	RES
SIZE *yyyyyy*	131072	SIZE*yyy* K
SPACE2	SPACE1	ACE2,ACE1
SPACE3	SPACE1	ACE3,ACE1
STATE	NOSTATE	STA,NOSTA
SUPMAP	NOSUPMAP	SUP,NOSUP
SXREF	NOSXREF	SXR,NOSXR
SYMDMP	NOSYMDMP	SYM,NOSYM
SYNTAX	NOSYNTAX	SYN,NOSYN
SYS *x*	SYST	
TEST	NOTEST	TES
VBREF	NOVBREF	VBR,NOVBR
VBSUM	NOVBSUM	VBS,NOVBS
XREF	NOXREF	XRE,NOXRE

Figure 7. OS compiler option defaults.

Caution: Where you intend to execute your object programs is a compile-time consideration. If you are compiling a COBOL source program for execution under OS, you *must* specify the OSDECK option. If you compile it for execution under CMS, you should *not* specify the OSDECK option.

Copying OS COBOL Files from CMS MACLIBs

If you are using the OS COBOL compiler COPY statement, and you have files that contain COBOL code that you use frequently, you can place them in a CMS file called a MACLIB, and then identify the MACLIB to be searched during compilation.

A MACLIB is similar to an OS partitioned data set. It has individual members, which you can create using the CMS Editor, or which you can copy from other sources. In order for a member to be added to a MACLIB, it must be in a CMS file with a filetype of COPY or MACRO.

The MACLIB command in CMS is used to create a macro library, to add or delete members, or to list or compress the members in a library. For example, to create a MACLIB from files named DATA COPY, CREATE COPY, and LIST COPY, you would enter

```
maclib gen mylib data create list
```

This command creates the file MYLIB MACLIB, which has the members DATA, CREATE, and LIST.

In order to make the MACLIB available, both the GLOBAL and FILEDEF CMS commands must be issued:

```
GLOBAL MACLIB mylib
FILEDEF SYSLIB DISK mylib MACLIB
```

Note that up to eight MACLIBs can be specified on a GLOBAL command and that the FILEDEF command must define the DDNAME SYSLIB.

The FILEDEF command will be issued by the CMS interface for COBOL, as a default, if the library has a filetype of MACLIB and is on your A-disk or a disk accessed as an extension of your A-disk. If you fail to follow these guidelines, your COPY library members will not be found during compilation of your COBOL program.

If the enhanced COPY facility of OS/VS COBOL is desired, you must issue a FILEDEF command for each library. The DDNAME field of the FILEDEF command is identical to the library-name in your program. For these additional libraries, you must not issue a GLOBAL command.

You can also copy library text directly from your OS partitioned data sets at compilation time. You must have the OS disk that contains the data set attached to your virtual machine and accessed. Then, you can use the FILEDEF command to assign a ddname of SYSLIB to the data set, and use the GLOBAL command to identify the library. For example, to use the data set COBTEST.COPYDATA, you would enter:

```
access 195 d
D(195) R/O - OS
R;
filedef syslib disk syslib maclib d dsn cobtest copydata
global maclib syslib
```

You can also concatenate additional OS libraries, or CMS MACLIBs, at the same time.

Additional information on using MACLIBs is contained in the *VM/370: CMS User's Guide.*

Files Created by the OS COBOL Compiler

When the OS COBOL compiler is executing, it creates workfiles, and assigns them filenames that are the same as the source file, and filetypes of SYSUT1, SYSUT2, SYSUT3, SYSUT4, and SYSUT6. They are erased at the end of compilation, but if you should ever have occasion to terminate a compilation before it is finished, these files might still reside on one of your disks, and you should erase them.

When the compiler writes an output disk file, it places it on a read/write CMS disk. If your COBOL source file is on a read/write disk, the output files are written to that disk. If your source file is on a read-only disk that is an extension of a read/write disk, the files are written on the read/write parent disk. If your COBOL source file is on a read-only CMS disk or an OS disk, then the output files are written on the first available read/write CMS disk in the standard search order.

According to options that you specify, the compiler also creates files with the filetypes of TEXT, LISTING, and SYSUT5. Again, their filenames are the same as the source file. Figure 8 lists the types of output produced as a result of the various compiler options.

TEXT Files

TEXT files contain the machine-language object code generated by the COBOL compiler. In CMS, this TEXT file is executable. If you want to punch the TEXT file on cards, you can specify the option DECK when you invoke the compiler:

```
cobol newfile (deck
```

Or, you can use the CMS PUNCH command to punch the TEXT file:

```
punch newfile text
```

In both cases, the disposition of the output punch file depends on the spooling characteristics of your virtual card punch. See *VM/370: CMS User's Guide* for information on spooling virtual punch files.

LISTING Files

In addition to TEXT files, the compiler produces a compilation listing for each COBOL source file that you compile. The listing is placed on your A-disk (unless LISTING was assigned to some other read/write disk) in a file with a filetype of LISTING.

Compiler Option	Listing File	Test File	Debug File	Terminal Response
CDECK		file is punched		
CLIST	object module is listed			
DECK		file is punched		
DISK	writes the LISTING file to disk			
DMAP	glossary, global tables, literal pools, register assignments	file is punched		
FDECK				
FLAGE				only error compiler messages are printed
FLAGW				all compile error and warning warning messages are printed
LOAD		object module to disk		
LSTCOMP	reformatted source listing is produced			
LSTONLY	reformatted source listing is produced			
LVL *y*	FIPS messages listed			
NUM	user-supplied line numbers are used here			
PMAP	global tables, literal pools, register assignments, assembler expansion of source module			
PRINT	prints the listing file on the virtual printer			
SOURCE	source module is listed			
SPACE 1	single spacing for this file			
SPACE 2	double spacing for this file			
SPACE 3	triple spacing for this file			
SUPMAP	suppresses listing of object module if D or E level compiler messages			
SXREF	sorted cross-reference listing is listed			
SYMDMP			written to disk, filetype SYSUT5	
TERM				progress and diagnostic messages
TEST				written to disk, filetype SYSUT5
VBREF	verb references listed			
VBSUM	verb references listed			
XREF	unsorted cross-reference listing is listed			

Figure 8. OS compiler output.

Error Messages from the COBOL Command

When you use the COBOL command in CMS, the TERM option is in effect by default. This specifies that all diagnostic messages issued by the compiler are to be displayed at your terminal, as well as being written into a LISTING file. COBOL messages all have prefix IKF. A complete listing of compiler diagnostic messages, along with their explanations can be generated by compiling a program with PROGRAM-ID of ERRMSG. For further information, see *IBM OS/VS COBOL Compiler and Library Programmer's Guide*, or *IBM VS COBOL for OS/VS*.

In addition to the messages from the compiler, there are messages issued by the COBOL command. They have the identification DMSCOB, followed by a message number and a message text. Whether or not the message identification or text, or both, is displayed depends on the setting in your virtual machine of EMSG, which is a CP SET command function. Many of the DMSCOB messages issued by CMS are self-explanatory. For example:

```
cobol
DMSCOB001E NO FILENAME SPECIFIED
R(00024);
```

The error was issued because no filename was specified to identify the source file, and you would reenter the command, specifying a filename.

The error messages issued by the COBOL command are listed below.

DMSCOB001E   NO FILENAME SPECIFIED

Explanation: You must specify the filename of the COBOL source file.

System Action: RC = 24. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Retype the command, supplying a filename in the command line.

DMSCOB002E   FILE 'fn COBOL' NOT FOUND

Explanation: The specified input file was not found on any disks accessed.

System Action: RC = 28. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Reissue the command, specifying the correct input file.

DMSCOB003E   INVALID OPTION 'option'

Explanation: The option(s) specified are not valid for this command.

System Action: RC = 24. The compilation is terminated.

User Action: Reissue the corrected command.

DMSCOB004W   WARNING MESSAGES ISSUED

Explanation: Minor errors were detected during compilation.

System Action: RC = 4. Compilation is completed.

User Action: If execution is not successful, correct the minor errors identified through the compiler warning messages.

DMSCOB006E   NO READ/WRITE DISK ACCESSED

Explanation: No read/write disk is available to contain compiler output and work files.

System Action: RC = 36. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Access a CMS disk in read/write status.

DMSCOB008W ERROR MESSAGES ISSUED

Explanation: Errors were detected in the compiled program.

System Action: RC = 8. Compilation is complete. Execution may be possible.

User Action: Correct errors identified by error messages.

DMSCOB012W SEVERE ERROR MESSAGES ISSUED

Explanation: Serious errors were detected in the compiled program.

System Action: RC = 12. Compilation is complete. Successful execution is not probable.

User Action: Correct the serious errors identified by error messages.

DMSCOB016W DISASTER ERROR MESSAGES ISSUED

Explanation: Very serious errors were detected during compilation.

System Action: RC = 16. Compilation is incomplete. Results are unpredictable.

User Action: Correct the very serious errors identified through the compiler disaster error messages.

DMSCOB034E FILE 'fn COBOL' IS NOT FIXED LENGTH

Explanation: The specified file must have fixed length records to be acceptable to this command.

System Action: RC = 32. Compilation is terminated.

User Action: Convert the file to fixed length records, or reissue the command with the correct file specified.

DMSCOB038E FILEID CONFLICT FOR DDNAME 'COBOL'

Explanation: A previously issued FILEDEF for ddname COBOL to a disk device did not contain the same filename and/or filetype as specified and implied by the COBOL command.

System Action: RC = 40. Compilation is terminated.

User Action: Reissue the corrected FIELDEF and/or COBOL command.

DMSCOB052E MORE THAN 100 CHARACTERS OF OPTIONS SPECIFIED

Explanation: The COBOL command options, including a delimiting blank between each option, totaled more then the permissible maximum of 100 characters.

System Action: RC = 24. Compilation is terminated.

User Action: Reisuue the COBOL command using fewer options.

DMSCOB070E INVALID PARAMETER 'parameter'

Explanation: The specified parameter is not expected in the command line.

System Action: RC = 24. Compilation is terminated.

User Action: Reissue the command in the correct format.

DMSCOB075E DEVICE 'device type' ILLEGAL FOR INPUT

Explanation: The specified device type is invalid for input to the COBOL compiler.

System Action: RC = 40. Compilation is terminated.

User Action: Reissue the FILEDEF command for the ddname of COBOL with the device type DISK, READER, or TAPE as input source file device. Reissue the FILEDEF command, then the COBOL command.

Note: If the COBOL command is issued while CMS/DOS is active, the following LOADMOD error message is issued:

DMSMOD114E 'COBOL MODULE fn' NOT LOADED; CMS/DOS ENVIRONMENT ACTIVE

Explanation: The COBOL command cannot be invoked while CMS/DOS is active.

System Action: RC = 40 or -5. Execution of the command terminates.

User Action: Issue the SET DOS OFF command, then reissue the COBOL command.

How to Load and Execute an OS COBOL Program

In order to execute OS COBOL programs under CMS, you must have access to the COBOL library, which is contained in a CMS TXTLIB. To identify the COBOL library, use the GLOBAL command:

```
global txtlib coblibvs
```

In this example, the TXTLIB is named COBLIBVS TXTLIB. The filename is assigned at system installation time, so you should find out from your installation personnel what the name is in your installation.

To load your program into storage, you can use the LOAD command, and then use the START command to begin execution:

```
load myfile
R;
start
EXECUTION BEGINS...
```

In this example, the file MYFILE TEXT contains the object code from a COBOL compilation. The message EXECUTION BEGINS... indicates that the program is executing.

You can also load more than one routine into storage by naming them both on the LOAD command line, or by using the INCLUDE command:

```
load prog1 prog2
R;
include prog3
R;
start
```

You can, if you want, use the START option on the LOAD or INCLUDE commands to specify that execution is to begin immediately:

```
load myfile (start
```

The RUN command performs like a cataloged compile, load, and execute procedure, if you invoke it for a COBOL source file:

```
run oldfile
```

When you use the RUN command, however, you cannot specify compiler options, nor can you indicate the inclusion of more than one TEXT file for execution.

Passing Execution-Time Parameters

The CMS START and RUN commands can be used to pass parameters to a COBOL program when it is invoked for execution. The RUN command can be used to pass user-defined parameters (which will be accessible in the program through the COBOL USING statement). The START command can be used to pass user-defined parameters and/or COBOL-defined execution-time options. The formats of the two commands are:

```
START {entry | *} [user-defined parameters] [/ COBOL options]
RUN fn [ft [fm]] [(user-defined parameters)]
```

CMS requires all items in the parameter list to be separated by blanks. Also, each item ("token," in CMS terms) cannot exceed eight characters. The delimiting blanks are removed by CMS when the parameters are passed, and the concatenated list arrives in the COBOL program as a single character string. For example, if you entered

```
START * ONE TWO THREE
```

your program would receive the variable ONETWOTHREE. The parameter list MAINE, MASSACHU SETTS, NORTHDAK OTA would enter the COBOL program as MAINE,MASSACHUSETTS,NORTHDAKOTA. The maximum permissible length of a parameter string is 100 characters.

The COBOL-defined execution-time options are explained in your *COBOL Programmer's Guide*. If you include any of these options on the START command, a slash must precede the first such option. Remember that to meet CMS requirements, any individual option that exceeds eight characters must be split with a blank. For example, to pass UPSI switches and debugging notification, one might code:

```
START * / UPSI(nnn nnnnn) DEBUG
```

Defining Program Input and Output Files

The FILEDEF command simulates the functions of the Job Control Language (JCL) data definition (DD) statement. You must use it whenever you are going to execute a program that performs I/O. You must establish file definitions, as well, for any input or output performed by a verb such as ACCEPT, DISPLAY, and so on.

The FILEDEF command relates a ddname (filename) in your program with an I/O device. The ddname in your program is specified with the SELECT or ASSIGN clause in the FILE CONTROL paragraph. If the device is a disk device, then you must specify a file identification. For example, if your file contains the following:

```
FILE-CONTROL.
      SELECT INFILE
      ASSIGN TO UR-2540R-S-CARDIN
      SELECT OUTFILE
      ASSIGN TO DA-3330-S-OUTDD.
           .
           .
           .
  FD INFILE
           .
           .
           .
  FD OUTFILE
           .
           .
           .
```

you would specify the following to execute the program:

```
filedef cardin reader
filedef outdd disk test4 output a4
load myprog (start
```

By specifying a filemode number of 4, you ensure that CMS will write the file in simulated OS data set format.

You can also read input files directly from an OS disk, if it is attached to your virtual machine. For example, to read an input file named COBOL.TEST.FILE from an OS disk at virtual address 198, enter:

```
access 198 c
C(198) R/O - OS
R;
filedef input c dsn ?
ENTER DATA SET NAME:
cobol.test.file
R;
load cobtest2 (start
```

The FILEDEF command is described in greater detail in *VM/370: CMS User's Guide* and *VM/370: CMS Command and Macro Reference.*

Although you can read data sets from OS disks when you execute OS COBOL programs in CMS, you cannot write files on OS disks, unless you are writing VSAM files. You can, however, write OS simulated data sets on CMS disks, which retain file characteristics of OS sequential data sets.

If you attempt to execute a program without issuing FILEDEF commands to define any input or output files, CMS creates a default file definition when the file is opened. The file defaults to a filename of FILE, and a filetype that is the same as the external name specified in the ASSIGN clause in your program. Any file definitions that are created by default are cleared when the file is closed.

All other file definitions in effect are cleared when the program finishes executing, except those that were specified with the PERM option.

When a program ABENDs, all file definitions are cleared.

Be aware that the COBOL-CMS interface subroutine (DMSILB) issues its own FILEDEF commands for the files SYSOUT, SYSIN, SYSDBOUT, SYSPUNCH, SYSDBG, and SYSUT5. If the user is to access these files in a program, correct FILEDEF commands must be issued for them. If this requirement is not observed, the program will execute with no apparent error, but the data being read will be unreliable.

Using VSAM Files in CMS

You can execute OS/VS COBOL programs that read and write VSAM files in CMS. The VSAM being used will be DOS VSAM (not OS VSAM). Nevertheless, the VSAM files written in CMS have the same format as those created under OS, and are fully compatible with OS VSAM data sets. VSAM files created in CMS can be read by an OS system, and vice versa.

The fact that DOS VSAM is being used has no effect on your coding requirements within the OS/VS COBOL program. These are the same as when using OS VSAM. However, instead of using the FILEDEF command to identify the VSAM files, you must use the DOS DLBL command. The DLBL command has the same basic format as the FILEDEF command:

```
dlbl input c dsn cobtest data (vsam
```

This file might be identified in your program as follows:

```
FILE-CONTROL.
      SELECT INVSAM
      ASSIGN TO INPUT
      ORGANIZATION IS INDEXED...
          .
          .
          .
  FD INVSAM
          .
          .
          .
```

More information on the DLBL command can be found in *VM/370: CMS User's Guide*.

If you are executing a program that uses a VSAM file and a non-VSAM file, then you must use the DLBL command to identify the VSAM file and the FILEDEF command to identify your non-VSAM file. There are additional options you may specify if the VSAM data set is a multivolume file, or if it is cataloged in a user catalog. If you use any of these special options, then you do not need to use the VSAM option.

There is a special ddname provided for you to identify the VSAM master catalog you will be using during a terminal session:

```
dlbl ijsysct f dsn mastcat (perm
```

Entering this command makes the VSAM master catalog available to you for the remainder of your terminal session.

Note: Even if PERM has been specified, if a program abnormally terminates or if an HX Immediate command is issued during execution, all DLBL definitions are cleared.

Using Access Method Services under CMS

You must use DOS/VS Access Method Services to define VSAM catalogs, data spaces, and clusters, and to perform REPRO, EXPORT/IMPORT, LISTCAT, and other Access Method Services functions using the CMS command AMSERV. How to use AMSERV command is described in *VM/370: CMS User's Guide* and in *VM/370 CMS Command and Macro Reference*. The Access Method Services control statements are described in *DOS/VS Access Method Services*.

Special Considerations

The programmer's guide that corresponds to your version of the compiler explains how to assign and use ddnames. That publication explains the relationships between DD statements and various COBOL compiler options and language statements. For example, the programmer's guide explains how the operation of the SYMDMP option is dependent upon the object-time control data placed in the SYSDBG data set.

Under CMS, if you specify the SYMDMP compiler option, you must create a file with the ddname of SYSDBG and with the required contents that are described in the programmer's guide. You can use the CMS Editor to create the file. The execution-time interface does a default FILEDEF for SYSDBG to disk: you do not have to enter the FILEDEF command.

If the source file is compiled with the SYMDMP option, it should be given the same name as the program-name specified in the PROGRAM-ID paragraph; in this case, the execution-time interface does a default FILEDEF for SYSUT5. If the program name is not used, then you must issue a FILEDEF command for the debug file before you execute the program.

Restrictions on Using OS COBOL under CMS

There are several restrictions placed on compiling and/or executing COBOL programs in CMS.

Compiler Restrictions

At compile time, the "nn" subparameter of the FLOW option must be specified; it is not optional.

The "*" and "dsname" subparameters of the PRINT option and the "dsname" subparameter of the LIB option are not valid; you must not specify them when you are compiling a COBOL program in CMS.

Execution-Time Restrictions

The following restrictions apply to executing OS COBOL programs in CMS. You can, however, compile these programs in CMS and then execute them in an OS virtual machine, or under a real OS system.

- Indexed files (BISAM and QISAM) are not supported. Therefore, the following clauses and statements associated with these access methods are invalid:

```
RECORD KEY        TRACK-AREA
START             APPLY REORG-CRITERIA
APPLY CORE-INDEX
```

- Creating direct files is restricted as follows:
  - For U and V recording modes, access must be sequential.
  - For ACCESS IS SEQUENTIAL, track identifier must not be modified.
  - You must specify the XTENT option of the FILEDEF command to indicate the number of logical records to be written.
- There is no Checkpoint/Restart feature. Therefore, the RERUN clause is not supported.
- The positioning options of the OPEN (EXTEND) and CLOSE statements are ignored.

- There is no multivolume data set support. Therefore, the CLOSE statement with the REEL or UNIT option is invalid.
- None of the user label handling functions are supported. Therefore, the label handling format of USE is invalid. The data-name option of the LABEL RECORDS clause is invalid.
- There is no TCAM support; however, the BSAM test facility will function only for single level queues.
- There is no Sort/Merge feature. Therefore, the SORT verb is not supported.
- ASCII-encoded tape files are not supported.
- Spanned record (S-mode) processing is not available under QSAM, BDAM, and BSAM. This means that the S-mode default (block size smaller than record size) cannot be specified, and that the RECORDING MODE IS S clause cannot be specified.
- No support for the IBM 3886 Optical Character Reader is provided.
- Neither the LISTING nor the SYSUT5 data set can be used under other systems.
- The GIVING option of the USE statement in the error declarative section is not supported for VSAM data sets.
- The AIXBLD execution-time option is not supported. Therefore, the dynamic building of alternate indexes and the dynamic completion of VSAM relative record data sets (RRDS) record information is not supported.
- The CALL...ON OVERFLOW statement is not available.

Executing Programs under OS, OS/VS1, and OS/VS2

You can execute your COBOL programs under an OS system, either in a virtual machine, or on a real machine. In either case, you can use CMS to prepare your job stream, and then punch a card deck containing your JCL statements, TEXT files, and so on. Use the CMS Editor to create a file with fixed-length, 80-character card images, exactly as you would punch them on a real card punch. Then, use the CMS PUNCH command to punch them:

```
punch job stream (noh
```

The NOHEADER option on the PUNCH command eliminates the punching of a CMS header card.

To execute your program in an OS virtual machine, spool your virtual card punch to the reader of the OS virtual machine, which can then read and execute your job. To load the OS system into your own virtual machine, you can spool your virtual punch to your own card reader. To execute the program on another real OS system, punch your file onto real cards, and submit to the OS system in the normal manner. For more information on these techniques, consult *VM/370: Operating Systems in a Virtual Machine.*

USING DOS COBOL UNDER CMS

In order to use the DOS/VS COBOL compiler under CMS, you must first place your virtual machine in the CMS/DOS environment. CMS/DOS is a part of the normal CMS system, but it accepts, in addition to the regular CMS commands, a number of commands and routines that apply specifically to DOS functions.

You enter the CMS/DOS environment by specifying the command

```
set dos on
```

Accessing Disks under CMS

In addition to disks defined in your VM/SP directory, you can temporarily obtain disks via use of the CP LINK and DEFINE commands. CMS must also know about these disks, and you must use the CMS ACCESS command to establish a filemode letter for them:

```
ACCESS 197 f
```

CMS uses the filemode letter to manage your files during your terminal session. By using the ACCESS command, you can:

- Control whether the disk is to be read-only (that is, you cannot write on it), or is read/write.
- Control the minidisk search sequence used by CMS.
- Control which disks are to contain new files that you create.

For the most part, you will use your primary 191 minidisk, that is, your A-disk, your DOS and/or OS read-only disks, and, if needed, your VSAM disks. For more detailed information on the CMS ACCESS command and the CP LINK and DEFINE commands, refer to the *VM/SP CMS User's Guide.*

If you want access to the DOS system residence volume during your terminal session, you can access the volume and name it as the system residence by entering its filemode letter on the SET command line:

```
access 198 C
C(198) R/O - DOS
R;
set dos on c
R;
```

When you have accessed the system residence, you can copy files from the source statement, relocatable, and procedure libraries with the CMS commands SSERV, RSERV, and PSERV. The command DSERV displays the directories of the libraries, including the core-image library, from which you can fetch phases for execution, and from which the DOS/VS COBOL compiler is fetched.

Also available to you in the CMS/DOS environment are the commands ASSGN and DLBL, which are familiar to you as DOS/VS control statements. In CMS, they perform similar functions, though their format varies somewhat. You must use the ASSGN command to relate a system or programmer logical unit specified in a program with an external device. If the device is a DASD, then you must use the DLBL command to establish the file identificaiton. In

CMS, you do not need to specify an EXTENT statement, since the CMS file system can locate and determine the extents of existing files.

The CMS commands DOSLKED and FETCH allow you to link-edit DOS COBOL object programs and load the core image phases into virtual storage for execution.

Compiling DOS Source Files

Under DOS/VS, entering the FCOBOL command invokes the DOS/VS COBOL compiler, which translates your COBOL source programs into machine-language object code.

Before you can use the compiler, you must be sure that you have accessed the DOS disk on which the compiler resides (if it is not the system residence volume) and identified the private core image and relocatable libraries containing the compiler phases. If the COBOL compiler library is on the system residence disk, and you have entered the CMS/DOS environment specifying the system residence, then you do not have to identify it as the core image library. If the COBOL compiler is on a private library, you must access the disk, and use the special ddnames IJSYSCL and IJSYSRL with the system logical units SYSCLB and SYSRLB to identify the libraries:

```
access 195 g
assgn sysclb g
dlbl ijsyscl g dsn cobol clib (sysclb
assgn sysrlb g
dlbl ijsysrl g dsn cobol rlib (sysrlb
```

The FCOBOL Command

The format of the FCOBOL command (which is actually an EXEC procedure) is:

FCOBOL	*filename*

filename
specifies the filename of the file to be compiled. If the file is on a CMS disk, it must have a filetype of COBOL. The file must have fixed-length, 80-character records. If this command is executed from within an EXEC procedure, it must be specified as

```
exec fcobol filename
```

because FCOBOL, itself, is an EXEC procedure.

Note 1: FCOBOL erases any existing $LISTIO EXEC on your A-disk. The A-disk must be accessed in read/write mode.

Note 2: If the DOS COBOL compiler resides in the DOS system core image library instead of in the private core image library, the time required to compile your COBOL program will be significantly increased if any DOSLIB libraries were previously activated via a GLOBAL DOSLIB libename-1... libename-n command. This is because CMS's fetch routine uses the following search order to find the COBOL compiler:

1. In a DOS/VS private core image library, if one had been previously assigned via a DLBL command such as the following:

   ```
   dlbl ijsyscl fm dsn? (sysclb)
   ```

2. If a GLOBAL DOSLIB command with up to eight DOSLIB libraries had been previously issued, all such libraries are searched in the order specified in the GLOBAL command.
3. In the DOS/VS system core image library, if the CMS disk mode letter (fm) was specified in the SET DOS ON fm command.

   Therefore, if the COBOL compiler is in the DOS system core image library, you should issue the GLOBAL DOSLIB command with no additional operands to eliminate the searching of the DOSLIB libraries prior to invoking the FCOBOL command.

When you invoke the FCOBOL command, you must make an assignment for SYSIPT, specifying the filemode letter of the disk that contains the source file. For example, if your COBOL source statements are in the CMS file NEWFILE COBOL, on your A-disk, then you would enter the commands:

```
assgn sysipt a
fcobol newfile
```

In this example, because the source file is on a CMS disk, the FCOBOL command will automatically issue a DLBL command for you.

You can also compile a sequential source file directly from a DOS/VS disk. In this case, you must specify the DLBL command, as well as the ASSGN command, to identify the file name. For example, to compile a source program that is located on a DOS disk that has been accessed as your C-disk, you would enter:

```
assgn sysipt c
dlbl ijsysin c dsn dostest file (sysipt
fcobol test2
```

When this FCOBOL command executes, the source program is read from the DOS disk with a file-id of DOSTEST.FILE. Any output files created have filenames of TEST2, since that is the filename specified on the FCOBOL command line.

Specifying Compiler Options—OPTION Command

The OPTION command allows you to specify the compiler options to be used while the DOS/VS COBOL compiler is being executed. These options are similar to those you would enter on the OPTION control statement if you were compiling a source file on a DOS/VS system.

The command defaults in CMS/DOS are not necessarily the same as the defaults on the DOS/VS system being used.

The format of the OPTION command is:

OPTION	[*option1 option2...optionn*]

Option can be omitted or can be one or more of the following:

no option
resets the options to the initial default settings.

DUMP
dumps the registers and the virtual partition on the virtual SYSLST device in the case of abnormal program end.

<u>**NODUMP**</u>
suppresses the DUMP option.

DECK
: punches the resulting object module on the virtual SYSPCH device. If you have not assigned SYSPCH, CMS stores the object module on your A-disk with a filetype of TEXT.

NODECK
: suppresses the DECK option; therefore, no CMS TEXT file is created.

LIST
: writes the output listing of the source module on the SYSLST device.

NOLIST
: suppresses the LIST option. This option overrides the XREF option as it does in DOS/VS.

LISTX
: produces a Procedure Division map on the SYSLST device.

NOLISTX
: suppresses the LISTX option.

SYM
: prints a Data Division map on SYSLST.

NOSYM
: suppresses the SYM option.

XREF
: writes the output symbolic cross-reference list on SYSLST.

NOXREF
: suppresses the XREF option.

ERRS
: writes an output listing of all errors in the source program on SYSLST.

NOERRS
: suppresses the ERRS option.

48C | 60C
: Uses the 48-character set. Uses the 60-character set.

The OPTION command causes bits in the partition communication region to be modified. If an invalid option is specified on the command line, an error message is issued for that option. Other valid options on the command line are processed. Only those options specified are altered. All other options remain unchanged. If you enter conflicting options, only the last one entered is in effect.

Options entered on the OPTION command stay in effect throughout a terminal session, unless they are specifically changed. Thus, to compile many different source programs, if you wanted the same options in effect for each compilation, you would only need to enter the OPTION command once.

If the OPTION command is issued with no operands, the options are reset to the initial default settings that were in effect when the CMS/DOS environment was entered.

Specifying Options with the CBL Statement

In addition to the options you can specify with the OPTION command, you can use the CBL statement in your source file to specify additional options. The options you can specify are:

ADV | NOADV
APOST | QUOTE
BUF=*nnnnn*
CATALR | NOCATALR
CLIST | NOCLIST
COUNT | NOCOUNT
FLAGE | FLAGW
FLOW=*nn*
LANGLVL (1) | LANGLVL(2)
LIB | NOLIB
LVL=A | B | C | D | NOLVL
OPTIMIZE | OPT | NOOPTIMIZE | NOOPT
PMAP=*h*
SEQ | NOSEQ
SPACE*n*
STATE | NOSTATE
STXIT | NOSTXIT
SUPMAP | NOSUPMAP
SXREF | NOSXREF
SYMDMP=*filename*
SYNTAX | CSYNTAX | NOSYNTAX
TRUNC | NOTRUNC
VERB | NOVERB
VERBREF | NOVERBREF
VERBSUM | NOVERBSUM
ZWB | NOZWB

Note: For pre-DOS/VS Release 3, the APOST | QUOTE and TRUNK | NOTRUNK defaults are reversed and LVL and ADV are not available.

These options are explained in *DOS/VS COBOL Compiler and Library Programmer's Guide.*

If filename is not specified, the default filename of IJSYS05 is used by the compiler for the DEBUG file. If you are invoking the compiler via the FCOBOL command, this default *must* be in effect. After compilation, the following DEBUG file is created for use at execution time:

```
filename SYMDMP filemode
```

SYMDMP is a reserved CMS filetype. The above is a permanent CMS file. You must provide a DLBL for it when executing the program. FCOBOL erases any existing DEBUG file on your disk before writing a new one.

The filename parameter of the SYMDMP option should *never* be specified *unless* the DLBL in FCOBOL EXEC is changed to specify a matching filename.

If you use a CBL statement, it must be the first input record in your source file, and it must begin in column 2. If you are using line-number editing, enter the EDIT subcommand

```
linemode off
```

in order to insert the CBL statement beginning in column 2. (Column 1 must be blank.) And then issue the subcommand LINEMODE LEFT to resume line-number editing.

Copying COBOL Files from DOS Source Statement Libraries

If your COBOL source programs contain COPY statements that reference books cataloged in DOS source statement libraries, you can identify the libraries to be searched during compilation. If you specified a DOS system residence when you entered the CMS/DOS environment, then the system source statement library is searched for files named in a COPY statement.

Additionally, you can identify a private library using the ddname IJSYSSL and the logical unit SYSSLB. For example, if your COBOL source file MYTEST references a file contained in the private library COBTEST.COPYFILE, then you can enter:

```
access 195 c
C(195)  R/O - DOS
assgn sysslb c
dlbl ijsyssl c dsn cobtest copyfile (sysslb
fcobol mytest
```

COBOL programs being compiled under CMS/DOS cannot copy files from CMS MACLIBs.

Files Used by the Compiler

During compilation, the compiler uses a number of files. These files are allocated by the FCOBOL command, according to the logical units and filenames listed below.

Logical Unit	Filename
SYSIPT/SYSIN	IJSYSIN
SYSLST	IJSYSLS
SYSPCH	IJSYSPH
SYSSLB	IJSYSSL
SYS001	IJSYS01
SYS002	IJSYS02
SYS003	IJSYS03
SYS004	IJSYS04
SYS005	IJSYS05
SYS006	IJSYS06

When you use the COBOL compiler in CMS/DOS, if SYSLST, SYSPCH, and SYS001 through SYSnnn are not assigned, they are assigned to the disk on which the input source file resides, if it is a read/write CMS disk. If the source file is on a read-only disk, then the ASSGN is made for the disk's read/write parent, if it is an extension; otherwise the assignment is made to the A-disk, or to the first available read/write disk in the standard CMS search order.

Note: If you already have any or all of the above system logical units assigned for some other purpose, DOS/VS COBOL will attempt to use those assignments even if the device is inappropriate or cannot be used. For example, if SYS001 is a read-only DOS disk, the FCOBOL compiler will attempt to write a work file on that disk and will terminate with an error message. Also if SYSLST had been previously assigned to 00E, then the listing will be printed rather than written to disk. You should use the ASSGN SYSxxx UA command to unassign any system logical units that should not be used by the FCOBOL command. FCOBOL will then make its own assignments.

Error Messages from the DOS COBOL Compiler

When you execute the FCOBOL command in CMS, there are no diagnostic messages from the compiler automatically displayed at the terminal. You must examine the listing produced by the compilation to determine whether your program compiled successfully.

There are, in addition to the DOS/VS COBOL compiler messages, messages produced by the CMS FCOBOL command, which are identified with the prefix DMSFCO and a message number. Whether the identification or the text of the message, or both, is displayed depends on the setting of the EMSG function of the CP SET command.

Most of the messages produced by the FCOBOL command are self-explanatory; for example, the sequence:

```
fcobol myfile
DMSFCO027E INVALID DEVICE ' UA ' FOR ' SYSIPT '
```

indicates that SYSIPT is unassigned. You must use the ASSGN command to specify the mode letter of the disk containing your COBOL source file and then reenter the FCOBOL command.

The error messages produced by the FCOBOL command are listed below.

DMSFCO001E NO FILENAME SPECIFIED

Explanation: You must specify the filename of the COBOL source file.

System Action: RC = 24. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Retype the command, supplying a filename in the command line.

DMSFCO002E FILE 'fn COBOL' NOT FOUND

Explanation: The specified input file was not found on any of the accessed disks.

System Action: RC = 28. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Reissue the command, specifying the correct input file.

DMSFCO005E INVALID PARAMETER 'parameter'

Explanation: More than one positional parameter was specified in the command line.

System Action: RC = 24. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Retype the command in the correct format.

DMSFCO006E NO READ/WRITE DISK ACCESSED

Explanation: No read/write disk is available to contain compiler output and work files.

System Action: RC = 36. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Access a CMS disk in read/write status.

DMSFCO027E INVALID DEVICE 'device' for 'SYSxxx'

Explanation: The device associated with the specified logical unit is not supported by the processor.

System Action: RC = 28. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Use the 'LISTIO SYSxxx' command to verify the device to which the logical unit is assigned. Reassign the logical unit to a proper device, and retype the command.

DMSFCO037E DISK 'mode' is READ ONLY

Explanation: Logical units for compiler output files have been assigned to a read-only disk.

System Action: RC = 36. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Assign the output files to a read/write disk, or unassign them and let the compiler assign them to read/write disks.

DMSFCO099E CMS/DOS ENVIRONMENT NOT ACTIVE

Explanation: In order to execute the command, the CMS/DOS environment must be active.

System Action: RC = 24. Execution of the command terminates. The system remains in the same status as before the command was entered.

User Action: Use the 'SET DOS ON' command to activate the CMS/DOS environment and retype the command.

Because the FCOBOL command is actually a CMS EXEC procedure, you may receive error messages as a result of other CMS commands being executed within the EXEC procedure.

Output from the Compiler

When you invoke the compiler with certain options in effect, output files are written to the virtual devices assigned to SYSLST and SYSPCH. The options that result in output files, and the files they produced, are listed in Figure 9.

Compiler Option	Listing File (on SYSLST)	Text File (on SYSPCH)
DUMP	dump printed	
DECK		deck is punched
ERRS		all source program errors printed
LIST		source program printed
LISTX		Procedure Division map printed
SYM		Data Division map printed
XREF		cross-reference listing printed

Figure 9. DOS compiler output.

If you are going to use any of these options when you compile a source file, you must be sure to issue the ASSGN command for the appropriate device before entering the FCOBOL command.

The default output files created on your CMS A-disk have filetypes of LISTING and TEXT, which contain the compiler listing and the object module produced by the compiler. If you specifically assign SYSLST to PRINTER, your listing is spooled to your virtual printer and the CMS listing file is *not* produced. Likewise, if SYSPCH is assigned to PUNCH, your object module is spooled to your virtual punch and the CMS file for TEXT is not created.

The Output LISTING File

The LISTING file contains the compiler output listing, which you must examine to find out if the compiler completed successfully. When you are intially compiling a new source file, you may have compilation errors indicated in the listing. Because the LISTING file is written to disk if SYSLST is not assigned, you can use the CMS Editor to determine whether or not the compilation completed successfully. If not, then you can edit the source file to correct the error and attempt to recompile. If the compilation was successful, then you can print the LISTING file so that you will have a copy of your program available when you are debugging.

As noted above, the assignment of SYSLST defaults to disk. If, however, you do not want to retain a disk copy of the listing, but merely want a printed copy, then you can assign SYSLST to your virtual printer:

```
assgn syslst printer
fcobol myfile
```

When the compiler finishes executing, the file MYFILE LISTING is spooled to your virtual printer.

The CMS TEXT File

SYSPCH, if unassigned, defaults to your CMS A-disk. The CMS TEXT file contains the output object file, which is relocatable. This file must be link-edited into a relocatable phase and placed in a special CMS file, called a DOSLIB, before you can execute it.

A TEXT file corresponds to an output object deck, which is sometimes punched onto cards. Whether or not an object module (text) file is created is controlled by the option DECK. If DECK is in effect, the text file is written to the virtual device assigned to SYSPCH. When the compiler is invoked, this assignment is made to disk, if it had not been assigned before. Therefore, if you do not want a CMS TEXT file created, but want a text file punched as a result of compilation, you must enter the following before you compile:

```
assgn syspch punch
fcobol myfile
```

If you have an existing CMS TEXT file that you want punched, you can use the CMS PUNCH command.

Link-Editing and Executing DOS COBOL Programs

You can link-edit CMS TEXT files, as well as relocatable phases from a DOS relocatable library, with the CMS command DOSLKED. Input to the linkage editor can be

- In a CMS file with a filetype of DOSLNK, which contains linkage editor control statements, such as INCLUDE, ACTION, and so on, and optionally, one or more object decks
- In a CMS file with a filetype of TEXT, which can also be edited to contain linkage editor control statements
- In a DOS relocatable library, which must be identified in a DLBL command specifying a ddname of IJSYSRL, with a logical unit of SYSRLB

Link-edited output phases, in CMS/DOS, are placed in a special CMS file called a DOSLIB. The DOSLKED command must specify the name of the input DOSLNK or TEXT file, or phase, as well as the output library name. If no DOSLIB name is specified, a DOSLIB is created with the same name as the input file.

For example, if you have a CMS TEXT file named DOSTEST, and you want it link-edited and placed in a DOSLIB with a filename of TESTLIB, you would enter the DOSLKED command as follows:

```
doslked dostest testlib
```

If the file, TESTLIB DOSLIB, does not exist when this command is executed, it is created.

Note: Each DOSLKED command invoked extends the space used in the DOSLIB. Thus, fetch time is increased. If possible, a separate DOSLIB should be created for each program. This DOSLIB should be erased before another DOSLKED command is issued for the same program. If several programs must reside in the same DOSLIB, that DOSLIB should be condensed periodically with a "DOSLIB COMP libename" command.

Executing DOS COBOL Programs

To execute a previously link-edited executable phase in CMS/DOS, you must use the FETCH command to load the phase into your virtual storage area, and then issue the START command to begin execution. If you are executing a phase from a CMS DOSLIB, you must first use the GLOBAL command to identify the DOSLIB. For example, to execute the program MYPROG, which has been link-edited into the file TESTLIB DOSLIB, you must enter the commands:

```
global doslib testlib
fetch myprog
start
```

Instead of issuing a separate START command, you can use the START option of the FETCH command to indicate that execution is to begin immediately. For example,

```
fetch myprog2 (start
```

You can also fetch a core-image phase directly from a DOS core-image library and execute it in CMS/DOS. To fetch a phase from the system core-image library, you must have entered the CMS/DOS environment by specifying the mode letter of the system residence. If you want to fetch a phase from a private core-image library, you must define the library with an ASSGN command and a DLBL command specifying a ddname of IJSYSCL with an assignment of SYSCLB to the mode letter at which the disk is accessed:

```
assgn sysclb d
dlbl ijsyscl d dsn private corelib (sysclb
fetch prog10 (start
```

Setting the UPSI Byte

If you use the special names UPSI-0 through UPSI-7 in your programs to set switches in the DOS/VS Communications Region, you can set these UPSI switches prior to executing your program. In CMS, the SET command has the UPSI operand, which allows you to set the UPSI byte. For example, to turn on switches 0, 2, and 7, you would enter:

```
set upsi 10100001
```

Defining Program Input and Output Files

When you are executing a COBOL program that performs input/output functions, you must identify the I/O devices by associating a system or programmer logical unit in your program with a virtual device. You can do this with the ASSGN command; for example:

```
assgn sys008 reader
```

You might issue this ASSGN command if you had an input file that defined an input reader device as SYS008.

You must establish file definitions as well, for any input or output performed by verbs such as ACCEPT, DISPLAY 6, 7, and so on.

If your input and output files are disk files, then you must use the DLBL command to establish a definition for the file, and associate the file with a logical unit. For example, the commands

```
assgn sys009 a
dlbl file1 a dsn cobtest output (sys009
```

establish the correspondence between an output file, named COBTEST OUTPUT, to be written on your CMS A-disk, with the file identified in your program as follows:

```
FILE CONTROL.
     SELECT OUTREC
     ASSIGN TO SYS100-DA-3330-S-FILE1.
```

An input file might reside on a DOS disk, and you can enter the DLBL command specifying the DOS file-id. For example, if your program contains the following:

```
FILE CONTROL.
     SELECT INREC
     ASSIGN TO SYS013-DA-3330-S-INDD.
     SELECT OUTLIST
     ASSIGN TO SYS014-UR-1403-S-PRINT.
```

Then, to use the DOS file PRIMARY DATA as an input file, and to direct your output to your virtual printer, enter

```
assgn sys013 c
dlbl indd c dsn ? (sys013
DMSDLB220R ENTER DATA SET NAME:
primary data
assgn sys014 printer
```

You must enter the file-id using the DSN ? form of the DLBL command because the file-id contains embedded blanks.

When you are executing DOS COBOL programs in CMS, you cannot write or update DOS files on DOS disks, except for VSAM files. If you are writing sequential disk files, you must write them to CMS disks.

Identifying VSAM Files

If you are going to be using VSAM files in CMS, you should specify the VSAM option, as well, when you enter the CMS/DOS environment:

```
set dos on (vsam
```

Entering the command this way makes all of the VSAM functions of CMS available to your virtual machine.

You can identify VSAM files for program input and output in the same way as other DOS files, except that you must specify the VSAM option of the DLBL command, to indicate that the file is a VSAM file:

```
assgn sys202 d
dlbl test3 d dsn vsamtest cobol (vsam sys202
```

There are additional options you may specify if the VSAM data set is a multivolume file, or if it is cataloged in a private catalog. If you use any of these special options, then you do not need to use the VSAM option.

There is a special ddname provided for you to identify the VSAM master catalog you will be using during a terminal session:

```
assgn syscat f
dlbl ijsysct f dsn mastcat (syscat perm
```

Entering these commands makes the VSAM master catalog available to you for the remainder of your terminal session.

Using Access Method Services under CMS

You can use Access Method Services to define VSAM catalogs, data spaces, and clusters, and to perform REPRO, EXPORT/IMPORT, LISTCAT, and other Access Method Services functions using the CMS command AMSERV. How to use the AMSERV command is described in *VM/SP: CMS User's Guide.*

Restrictions on Using DOS COBOL in CMS

The following restrictions apply to executing DOS/VS COBOL programs in CMS. You can, however, compile these programs in CMS and then execute them in a DOS virtual machine, or under a DOS system.

- Indexed files (DTFIS) are not supported. The following clauses and statements are therefore invalid:

  ```
  NOMINAL KEY
  TRACK-AREA
  APPLY CYL-OVERFLOW
  APPLY MASTER/CYL-INDEX
  APPLY CORE-INDEX
  ```

- Creating direct files is restricted as follows:
  - For U and V recording modes, access mode must be sequential.
  - For ACCESS IS SEQUENTIAL, track identifier must not be modified.
- None of the user label-handling functions are supported. Therefore, the label handling format of USE is invalid. The data-name option of the LABEL RECORDS clause is invalid.
- The positioning options of the OPEN (EXTEND) and CLOSE statements are ignored.
- There is no multivolume data set support. Therefore, the CLOSE statement with the REEL or UNIT option is invalid.
- The segmentation feature can only be used with LANGLVL(2).
- Use of SYSPARM as an execution-time option is not available.
- Neither AIXBLD nor NODEBUG may be set.
- ASCII-encoded tape files are not supported.
- Spanned records (S-mode) processing is not available. This means that the S-mode default (block size smaller than record size) cannot be specified, and that the RECORDING MODE IS S clause cannot be specified.
- There is no Checkpoint/Restart feature. Therefore, the RERUN clause is not supported.
- Multitasking, multipartition operation, and teleprocessing functions are not supported when executing under CMS.

Executing Programs under DOS/VS

You can execute your COBOL programs under a DOS system, either in a virtual machine, or on a real machine. In either case, you can use CMS to prepare your job stream, and then punch a card deck containing your control statements, TEXT files, and so on. Use the CMS Editor to create a file with fixed-length, 80-character card images, exactly as you would punch them on a real card punch. Then, use the CMS PUNCH command to punch them:

```
punch job stream (noh
```

The NOHEADER option on the PUNCH command eliminates the punching of a CMS header card.

If you are going to execute your program in a DOS virtual machine, you can spool your virtual card punch to the reader of the DOS virtual machine, which can then read and execute your job. Or, if you are going to load the DOS system into your own virtual machine, you can spool your virtual punch to your own card reader.

If you are going to execute the program on an DOS system, then you can have your file punched onto real cards, which you can submit to the DOS system in the usual manner.

APPENDIX A. SAMPLE EXEC PROCEDURES FOR DOS USERS

This section contains sample compile, compile and link-edit, and compile, link-edit, and execute EXEC procedures for DOS/VS COBOL users in CMS. The RUN command performs a similar function for OS COBOL users.

Sample EXEC for a COBOL Compile

```
&CONTROL OFF
&HIGH = 0
CP SET EMSG ON
* DEVICE 350 IS ASSUMED TO BE DOS/VS SYSRES
CP LINK DOSRES 350 350 RR PSWD
ACCESS 350 F
* DEVICE 353 IS ASSUMED TO BE A PRIVATE RELOCAT LIBRARY
* CONTAINING COBOL  LIBRARY MODULES
CP LINK LIBRARY 353 353 RR PSWD
* DEVICE 352 IS ASSUMED TO BE A PRIVATE CORE IMAGE LIBRARY
* CONTAINING THE COBOL  COMPILER PHASES
CP LINK LIBRARY 352 352 RR PSWD
SET DOS ON F
ASSGN SYSIPT A
ACCESS 352 E
ACCESS 353 G
ASSGN SYSCLB E
ASSGN SYSRLB G
DLBL IJSYSCL E DSN PRIVAT CORE IMAGE LIB ( SYSCLB PERM
DLBL IJSYSRL G DSN PRIVAT RELOCAT LIB ( SYSRLB PERM
&IF &INDEX EQ 0 &GOTO  -NOSOURC
&IF &INDEX GT 1 &GOTO -TOOMANY
&FN = &1
OPTION DUMP NODECK LIST XREF ERRS 60C
ASSGN SYSLST PRINTER
EXEC FCOBOL &FN
&IF &RETCODE EQ 0 &SKIP 1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-EOJ &CONTINUE
&TYPE END OF JOB  &FN  CHECK SPOOLED  OUTPUT TO VERIFY-
RESULTS
&TYPE THE HIGHEST UN EXPECTED RETURN CODE WAS " &HIGH "
&TYPE ALL ERRORS WILL APPEAR ON THE SPOOL PRINTER
CP SET EMSG TEXT
&EXIT &HIGH
-NOSOURC &CONTINUE
&BEGTYPE
* NO SOURCE FILE SPECIFIED
*  PLEASE RE-ENTER THE COMMAND WITH THE SOURCE FILE NAME-
AS ARGUMENT ONE
&END
&HIGH = 213
&GOTO -EOJ
-TOOMANY &CONTINUE
&BEGTYPE
* THE ONLY ARGUMENT ALLOWED IS THE SOURCE FILE NAME
&END
&HIGH = 22
&GOTO -EOJ
```

Sample COBOL Compile and Link-Edit EXEC Procedure

```
&CONTROL OFF
&HIGH = 0
CP SET EMSG ON
* DEVICE 350 IS ASSUMED TO BE DOS/VS SYSRES
CP LINK DOSRES 350 350 RR ALL
ACCESS 350 F
* DEVICE 353 IS ASSUMED TO BE A PRIVATE RELOCAT LIBRARY
* CONTAINING FCOBOL LIBRARY MODULES
CP LINK LIBRARY 353 353 RR PASS
* DEVICE 352 IS ASSUMED TO BE A PRIVATE CORE IMAGE LIBRARY
* CONTAINING THE FCOBOL COMPILER PHASES
CP LINK LIBRARY 352 352 RR PASS
SET DOS ON F
ASSGN SYSIPT D
ACCESS 352 E
ACCESS 353 G
ASSGN SYSCLB E
ASSGN SYSRLB G
DLBL IJSYSCL E DSN PRIVAT CORE IMAGE LIB ( SYSCLB PERM
DLBL IJSYSRL G DSN PRIVAT RELOCAT LIB ( SYSRLB PERM
&IF &INDEX EQ 0 &GOTO  -NOSOURC
&IF &INDEX GT 1 &GOTO -TOOMANY
&FN = &1
OPTION DUMP DECK LIST XREF ERRS 60C
ASSGN SYSLST PRINTER
EXEC FCOBOL &FN
&IF &RETCODE EQ 0 &SKIP 1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-LINK &CONTINUE
&STACK LIFO FILE
&STACK LIFO C / ,/,/ 5
&STACK LIFO TOP
&STACK LIFO C / DUMMY / &FN / 5
&STACK LIFO TOP
&BEGSTACK LIFO
I  INCLUDE
I  PHASE DUMMY ,S
I  ACTION REL,MAP
TOP
&END
EDIT &FN   TEXT *
DOSLKED &FN
&IF &RETCODE EQ 0 &SKIP  1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-EOJ &CONTINUE
&TYPE END OF JOB &FN CHECK SPOOLED OUTPUT TO VERIFY-
RESULTS
&TYPE THE HIGHEST UN EXPECTED RETURN CODE WAS " &HIGH "
&TYPE ALL ERRORS WILL APPEAR ON THE SPOOL PRINTER
CP SET EMSG TEXT
&EXIT & HIGH
-NOSOURC & CONTINUE
&BEGTYPE
* NO SOURCE FILE SPECIFIED
*  PLEASE RE-ENTER THE COMMAND WITH THE SOURCE FILE NAME-
AS ARGUMENT ONE
&END
&HIGH = 213
&GOTO -EOJ
-TOOMANY &CONTINUE
&BEGTYPE
* THE ONLY ARGUMENT ALLOWED IS THE SOURCE FILE NAME
&END
&HIGH = 22
&GOTO -EOJ
```

Sample COBOL Compile, Link-Edit, and Execute EXEC Procedure

```
&CONTROL OFF
&HIGH = 0
CP SET EMSG ON
* DEVICE 350 IS ASSUMED TO BE DOS/VS SYSRES
CP LINK DOSRES 350 350 RR PSWD
ACCESS 350 F
* DEVICE 353 IS ASSUMED TO BE A PRIVATE RELOCAT LIBRARY
* CONTAINING COBOL LIBRARY MODULES
CP LINK LIBRARY 353 353 RR PASS
* DEVICE 352 IS ASSUMED TO BE A PRIVATE CORE IMAGE LIBRARY
* CONTAINING THE COBOL COMPILER PHASES
CP LINK LIBRARY 352 352 RR PASS
SET DOS ON F
ASSGN SYSIPT D
ACCESS 352 E
ACCESS 353 G
ASSGN SYSCLB E
ASSGN SYSRLB G
DLBL IJSYSCL E DSN PRIVAT CORE IMAGE LIB ( SYSCLB PERM
DLBL IJSYSRL G DSN PRIVAT RELOCAT LIB ( SYSRLB PERM
&IF & INDEX EQ 0 & GOTO  -NOSOURC
&IF &INDEX GT 1 &GOTO -TOOMANY
&FN = &1
OPTION DUMP DECK LIST XREF ERRS 60C
ASSGN SYSLST PRINTER
EXEC FCOBOL &FN
&IF &RETCODE EQ 0 &SKIP 1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-LINK &CONTINUE
&STACK LIFO FILE
&STACK LIFO C / ,/,/ 5
&STACK LIFO TOP
&STACK LIFO C / DUMMY / &FN / 5
&STACK LIFO TOP
&BEGSTACK LIFO
I  INCLUDE
I  PHASE DUMMY ,S
I  ACTION REL,MAP
TOP
&END
EDIT &FN   TEXT *
DOSLKED &FN
&IF &RETCODE EQ 0 &SKIP 1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-EXECUTE &CONTINUE
GLOBAL DOSLIB &FN
FETCH  &FN  ( START
&IF &RETCODE EQ 0 &SKIP 1
&IF &RETCODE GT &HIGH &HIGH = &RETCODE
-EOJ &CONTINUE
&TYPE END OF JOB  &FN  CHECK SPOOLED  OUTPUT TO VERIFY-
RESULTS
&TYPE THE HIGHEST UN EXPECTED RETURN CODE WAS " &HIGH "
&TYPE ALL ERRORS WILL APPEAR ON THE SPOOL PRINTER
CP SET EMSG TEXT
&EXIT &HIGH
-NOSOURC &CONTINUE
&BEGTYPE
*  NO SOURCE FILE SPECIFIED
*  PLEASE RE-ENTER THE COMMAND WITH THE SOURCE FILE NAME-
AS ARGUMENT ONE
&END
```

```
&HIGH = 213
&GOTO -EOJ
-TOOMANY &CONTINUE
&BEGTYPE
* THE ONLY ARGUMENT ALLOWED IS THE SOURCE FILE NAME
&END
&HIGH = 22
&GOTO -EOJ
```

APPENDIX B. RESERVED FILETYPE DESCRIPTIONS PERTINENT TO COBOL USERS

OS COBOL Reserved Filetype Descriptions

Filetype	Command	Usage	Filename	Format	Contents
COBOL	COBOL	input	fn	fixed-length	COBOL source statements
LISTING	ASSEMBLE	output	fn	fixed-length (121)	processor printed output
	COBOL	output	fn	variable-length	
SYSUT1 SYSUT2 SYSUT3 SYSUT4	ASSEMBLE COBOL	work	fn		
SYSUT5	COBOL	output	fn	fixed-length	debug file
SYSUT6	COBOL (OS/VS only)	work	fn		
TEXT	ASSEMBLE COBOL INCLUDE LOAD output TXTLIB	output output input input fn input	fn fn fn fn fn	fixed-length	object file

DOS/VS COBOL Reserved Filetype Descriptions

Filetype	Command	Usage	Filename	Format	Contents
COBOL	FCOBOL	input	fn	fixed-length	COBOL source statements
LISTING	FCOBOL ESERV SSERV PSERV DSERV RSERV	output	fn	variable-length	processor printed output
SYS001 SYS002 SYS003 SYS004 SYS006	FCOBOL ESERV	work	fn		
SYS005	FCOBOL	output	fn	fixed-length	debug file
TEXT	FCOBOL RSERV	output	fn	fixed-length	object file
	DOSLKED phases	output	fn	fixed-length	CMS/DOS executable
DOSLNK	DOSLKED	output	fn	fixed-length phases	CMS/DOS executable
DOSLIB	DOSLIB DOSLKED FETCH GLOBAL QUERY	input output input library input	fn fn fn fn fn	variable-length fixed-length fixed-length fixed-length fixed-length	CMS/DOS executable phases

INDEX

(Where more than one page reference is given, the major reference appears first.)

IBM CMS User's Guide
for COBOL
SC28-6469-5

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ______________________

Previous TNL ___________________

Previous TNL ___________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC28-6469-5

Reader's Comment Form

Fold and tape — Please do not staple — Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape — Please do not staple — Fold and tape

IBM®

IBM CMS User's Guide for COBOL (File No. S370-24) Printed in U.S.A. SC28-6469-5

SC28-6469-5

IBM CMS User's Guide for COBOL (File No. S370-24) Printed in U.S.A. SC28-6469-5