COBOL for OS/390 & VM

Customization under OS/390

Version 2 Release 2

GC26-9045-02

COBOL for OS/390 & VM

Customization under OS/390

Version 2 Release 2

GC26-9045-02

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix, “Notices” on page 68.

Second Edition (September 2000)

This edition applies to:

IBM® COBOL for OS/390® & VM Version 2 Release 2 (Program Number 5648-A25)

and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, HHX/H3
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

About this book . . . vi
How to read the syntax diagrams . . . vi
Using the macro planning worksheets . . . vii

Summary of changes . . . ix
Major changes to COBOL for OS/390 & VM . . . ix
Version 2 Release 2, September 2000 . . . ix
Service update to Version 2 Release 1, April 1998 . . . ix
Version 2 Release 1, May 1997 . . . x
Version 1 Release 2, October 1995 . . . x
Major changes to the documentation . . . xi
Second edition, September 2000 . . . xi

Chapter 1. Planning to customize COBOL for OS/390 & VM . . . 1
Making changes after installation—why customize? . . . 1
Planning to modify compiler option default values . . . 2
Why make compiler options fixed? . . . 2
Modifying compiler options and phases . . . 3
Planning to place compiler phases in shared storage . . . 5
Why place the compiler phases in shared storage? . . . 6
Compiler phases and their defaults . . . 7
Planning to create an additional reserved word table . . . 11
Why create additional reserved word tables? . . . 11
Controlling use of nested programs . . . 11
Reserved word tables supplied with COBOL for OS/390 & VM . . . 11
COBOL for OS/390 & VM compiler options . . . 13
Specifying COBOL compiler options . . . 13
Options in support of the COBOL 85 standard . . . 13
Conflicting compiler options . . . 14
Compiler options syntax and descriptions . . . 15
ADATA . . . 15
ADEXIT . . . 16
ADV . . . 16
ALOWCBL . . . 17
ANALYZE . . . 17
ARITH . . . 18
AWO . . . 18
BUF . . . 19
CMPR2 . . . 19
COMPILE . . . 20
CURRENCY . . . 21
DATA . . . 22
DATEPROC . . . 23
DBCS . . . 23
DBCSXREF . . . 24
DECK . . . 25
DIAGTRUNC . . . 25
DLL . . . 26
DYNAM . . . 26
EXPORTALL . . . 27

FASTSRT . . . 27
FLAG . . . 28
FLAGMIG . . . 29
FLAGSTD . . . 29
IDLGEN . . . 31
INEXIT . . . 32
INTDATE . . . 32
LANGUAGE . . . 33
LIB . . . 34
LIBEXIT . . . 34
LINECNT . . . 35
LIST . . . 35
LITCHAR . . . 36
LVLINFO . . . 36
MAP . . . 36
NAME . . . 37
NUM . . . 37
NUMCLS . . . 38
NUMPROC . . . 38
OBJECT . . . 39
OFFSET . . . 40
OPTIMIZE . . . 40
OUTDD . . . 41
PGMNAME . . . 41
PRTEXIT . . . 42
RENT . . . 42
RMODE . . . 43
SEQ . . . 44
SIZE . . . 45
SOURCE . . . 45
SPACE . . . 46
SQL . . . 46
SSRANGE . . . 47
TERM . . . 47
TEST . . . 48
TRUNC . . . 49
TYPECHK . . . 50
VBREF . . . 51
WORD . . . 51
XREFOPT . . . 52
YRWINDOW . . . 53
ZWB . . . 54

Chapter 2. Customizing COBOL for OS/390 & VM . . . 55
Summary of user modifications . . . 55
Change compiler options defaults . . . 56
Changing compiler options default module . . . 56
Creating an options module to override options specified as fixed . . . 57
Creating or modifying additional reserved word tables . . . 58
Procedure for creating or modifying a reserved word table . . . 58
Coding control statements . . . 59
Rules for coding control statements . . . 60
Coding operands in control statements . . . 60
Rules for coding control statement operands . . . 60

ABBR statement . . . 61
Example of an ABBR statement . . . 61
INFO statement . . . 61
Example of an INFO statement . . . 61
RSTR statement . . . 61
Examples of RSTR statements . . . 62
Modifying and running JCL to create a new reserved word table . . . 62
Procedure for modifying and running non-SMP/E JCL . . . 63
Placing COBOL for OS/390 & VM modules in shared storage . . . 63

Chapter 3. Customizing Debug Tool . . . 65
Placing Debug Tool modules in shared storage . . . 65
Setting up CICS with Debug Tool . . . 66

Appendix. Notices . . . 68
Programming interface information . . . 69
Trademarks . . . 69

Bibliography . . . 70
IBM COBOL for OS/390 & VM . . . 70
Language Environment for OS/390 & VM . . . 70
Related publications . . . 70
Softcopy publications . . . 70

Index . . . 71

About this book

This book is for systems programmers who are responsible for customizing IBM COBOL for OS/390 & VM for their location. This book provides information needed to plan for and customize COBOL for OS/390 & VM under OS/390. In addition, this book can help to assess the value of COBOL for OS/390 & VM to your organization.

In this book, the generic term operating system is used when referring to OS/390. Planning and customizing information specific to the OS/390 operating system will be indicated where applicable.

You should have a knowledge of COBOL for OS/390 & VM and of your system's operating environment to use this book and ensure a successful customization of COBOL for OS/390 & VM.

How to read the syntax diagrams

Throughout this book, syntax for the compiler options is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following table shows the meaning of symbols at the beginning and end of syntax diagram lines.

Symbol	Indicates
▸▸—	the syntax diagram starts here
—▸	the syntax diagram is continued on the next line
▸—	the syntax diagram is continued from the previous line
—▸◂	the syntax diagram ends here

  Diagrams of syntactical units other than complete statements start with the ▸— symbol and end with the —▸ symbol.

- Required items appear on the horizontal line (the main path).

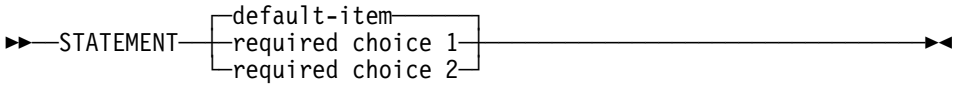
```
▸▸—STATEMENT—required item———————————————————————————▸◂
```

- Optional items appear below the main path.

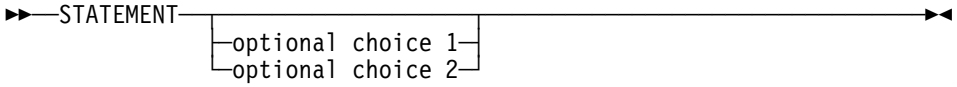
```
▸▸—STATEMENT———————————————————————————————————————▸◂
              └—optional item—┘
```

- When you can choose from two or more items, they appear vertically in a stack.

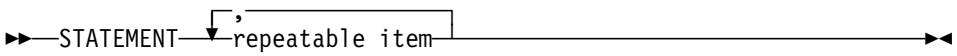
  If you **must** choose one of the items, one item of the stack appears on the main path. The default, if any, appears above the main path and is chosen by the compiler if you do not specify another choice. In some cases, the default is affected by the system in which the program is being run or the environmental parameters specified.



If choosing one of the items is optional, the entire stack appears below the main path.



- An arrow returning to the left above the main line indicates an item that can be repeated.



A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

- Keywords appear in uppercase letters (for example, PRINT). They must be spelled exactly as shown. Variables appear in all lowercase letters (for example, `item`). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, they must be entered as part of the syntax.
- Use at least one blank or comma to separate parameters.

Using the macro planning worksheets

The planning worksheets in this book will help you prepare to customize COBOL for OS/390 & VM. By completing them, you will be able to easily identify those options that you want to change from the IBM-supplied default values. You might also want to use the worksheets as a source from which to actually customize the IBM-supplied default values.

The headings in each worksheet might differ slightly from each other. Refer to the following list of definitions for an explanation of each specific column heading. Worksheets are located on pages 3 and 10.

Option
The OPTION column identifies the options contained within a specific installation macro. This column represents the options exactly as they are in the macro.

Fixed
The FIXED column is used to identify the options that can not be overridden by an application programmer. Enter an asterisk [*] into the **Enter * for Fixed** only for those options that you want to be fixed.

Selection
The SELECTION column is for you to identify the value associated with each option. In the space provided, enter the value you want to assign to each option. Use the page reference column to locate the specific information about the option that will assist you in selecting the appropriate value.

IBM-Supplied Default

The IBM-SUPPLIED DEFAULT column identifies the value supplied for the specified installation macro if the option is not altered. If the IBM-supplied default is identical to the value you desire for installation, you need not modify that option within that specific macro.

Page Reference

The PAGE REFERENCE column identifies the page on which you will find the syntax diagram for and more specific information about the option.

Once you have completed the worksheets, identify those options that are different from the IBM-supplied defaults. These are the items you must code in the installation macros. The worksheet entries have been positioned such that the order of the entries will remain consistent with the actual coding semantics.

Summary of changes

This section lists the major changes that have been made to the IBM COBOL for OS/390 & VM product and this manual since IBM COBOL for OS/390 & VM Version 2 Release 1. Technical changes are marked in the text by a change bar in the left margin.

Major changes to COBOL for OS/390 & VM

Version 2 Release 2, September 2000

- Millennium Language Extension tool has been incorporated into the compiler for this release.
- Support for linking of COBOL applications using the DFSMS binder alone, with the prelinker required only in exceptional cases under CICS.
- Support for compiling, linking, and execution in the OS/390 UNIX System Services (USS) environment, with COBOL files able to reside in the HFS (hierarchical file system.) A new cob2 command is included with the product, for compiling and linking COBOL programs in the USS shell environment. This command is itself installed into an HFS directory that is dedicated to COBOL components.
- New compiler option in support of decimal data:
  - ARITH
- New compiler option in support of DB2 coprocessing:
  - SQL
- Enhanced production debugging using overlay hooks rather than compiled-in hooks, with symbolic debugging information optionally in a separate file:
  - TEST
- New compiler option in support of diagnosis of moves (implicit or explicit) that result in numeric truncation:
  - DIAGTRUNC

Service update to Version 2 Release 1, April 1998

- When used in conjunction with IBM VisualAge COBOL Millennium Language Extensions for OS/390 & VM, support for automatic date processing (century windowing for dates containing 2-digit years).
- New language elements in support of automatic date processing:
  - DATE FORMAT clause in data description entries
  - Intrinsic functions:
    - DATEVAL
    - UNDATE
    - YEARWINDOW
- New compiler options in support of automatic date processing:
  - DATEPROC/NODATEPROC

  - YEARWINDOW

- New date intrinsic functions to cover the recommendation in the *Working Draft for Proposed Revision of ISO 1989:1985 Programming Language COBOL*.
  - DATE-TO-YYYYMMDD
  - DAY-TO-YYYYDDD
  - YEAR-TO-YYYY
- Extension of the ACCEPT statement to cover the recommendation in the *Working Draft for Proposed Revision of ISO 1989:1985 Programming Language COBOL*:
  - ACCEPT FROM DATE YYYYMMDD
  - ACCEPT FROM DAY YYYYDDD

Version 2 Release 1, May 1997

- INTDATE is now a standard compiler option, not just an installation option.
- New compiler options for DLL support: DLL and EXPORTALL.
- The IBM Debug Tool tape shipped with the Full Function Feature now contains the same three FMIDs as the Debug Tool tapes shipped with IBM COBOL for MVS & VM (5688-197), IBM C++ for MVS & VM (5655-121), OS/390 C++ optional feature (5645-001), or IBM PL/I for MVS & VM (5688-235). The additional FMIDs shipped with the Debug Tool in COBOL for MVS & VM (V1R2M0) are not required in Version 2.
- The cataloged procedure that was used to install COBOL for MVS & VM has been removed. Installers must define DDDEFs for all target, distribution, and SMP/E data sets needed for RECEIVE, APPLY, and ACCEPT commands in the appropriate SMP/E zones. Sample job IGYWFDDF (or IGYWDDDF) can be used to define DDDEFs for target and distribution data sets. Sample job IGYWFSMI (or IGYWSMPI) has been modified to define DDDEFs for all necessary SMP/E data sets in the newly created global, target, and distribution zones.
- Instructions for changes you may need to make in the sample jobs before running them have been moved to a list in the comments at the beginning of the JCL.

Version 1 Release 2, October 1995

- New installation option: INTDATE(LILIAN|ANSI)
- New compiler options:
  - IDLGEN/NOIDLGEN
  - OPTIMIZE(FULL)
  - PGMNAME(LONGMIXED|LONGUPPER|COMPAT)
  - RMODE(AUTO|24|ANY)
  - TYPECHK/NOTYPECHK
- FLAGSAA compiler option no longer supported.
- IGYCSAW compiler phase no longer supported.

Major changes to the documentation

Second edition, September 2000

- Installation information is now located in the COBOL for OS/390 & VM Program Directory.

Chapter 1. Planning to customize COBOL for OS/390 & VM

This chapter provides the following information for planning the customization of COBOL for OS/390 & VM:

- Making changes after installation—Why customize?
- Planning to modify compiler default values.
- Planning to place COBOL for OS/390 & VM in shared storage.
- Planning to create an additional reserved word table.

If you're installing IBM Debug Tool, you can decide whether to place its modules in the Shared Storage and whether you want to set up your CICS® environment to work with Debug Tool.

The information in this chapter helps you plan your customization. For the actual customization procedures, see Chapter 2, “Customizing COBOL for OS/390 & VM” on page 55 or Chapter 3, “Customizing Debug Tool” on page 65.

This chapter also contains worksheets to help you plan modifications to the IBM-supplied default values within macros. See “Using the macro planning worksheets” on page vii for an explanation of the planning worksheets in this manual.

Important

Please confer with the application programmers at your site while you plan the customization of COBOL for OS/390 & VM. Doing so will ensure that the modifications you make serve their needs and support the applications being developed.

Making changes after installation—why customize?

When you install COBOL for OS/390 & VM, you receive IBM-supplied defaults for compiler options and phases and the reserved word table. You might want to customize COBOL for OS/390 & VM to better suit the needs of application programmers at your site.

After you install COBOL for OS/390 & VM, you can:

- Create additional reserved word tables
- Make the compiler options fixed
- Modify the compiler option default values
- Modify the compiler phases residency values

The following sections discuss the planning you must do to customize COBOL for OS/390 & VM for your site.

Planning to modify compiler option default values

Under OS/390, compiler option defaults and residency for compiler phases are set in the IGYCDOPT program, as shown in Table 1 on page 4 and Table 2 on page 10. The default options module, IGYCDOPT, is link-edited with AMODE(31) and RMODE(ANY) during installation.

When assembling COBOL customization parts, such as IGYCDOPT, you need access to a system MACLIB. Typically, the MACLIB is found in SYS1.MACLIB. You also need access to the COBOL MACLIB IGY.V2R2M0.SIGYMAC.

The IGYCDOPT program has a dual purpose: it allows you to select and fix the compiler options defaults and to specify which compiler phases are in shared storage. This means that you can accept the IBM-supplied compiler option values you receive when you install COBOL for OS/390 & VM, or you can modify them to better suit the needs of programmers at your location. You can also choose whether or not your application programmers will have the ability to override these options.

By identifying compiler phases that reside in shared system storage, the compiler can use the storage in the region for work areas. For a more detailed description on why you might want to modify the phase defaults, see “Why place the compiler phases in shared storage?” on page 6.

Why make compiler options fixed?

COBOL for OS/390 & VM can help you to set up your site's unique programming standards. For example, many sites select RENT as the preferred compiler option, but have no easy way to enforce its use.

With COBOL for OS/390 & VM, you can use the IGYCDOPT program to specify that an option is fixed and cannot be changed or overridden at compile time. Then, at compile time, an attempt to override a fixed option is not allowed and results in a diagnostic message with a nonzero compiler return code.

When certain options are fixed for consistent usage, there might be times when special conditions require the ability to bypass such a fixed option. This can be done by assembling a temporary copy of the IGYCDOPT program with different parameters. At compile time, if you use a JOBLIB or STEPLIB containing the required IGYCDOPT module, you can bypass the fixed option. For example, if you select the OPT (OPTIMIZE) option to be fixed—indicating that you always want the COBOL compiler to generate optimized object code—and then need to exempt an application from this requirement, you must reassemble the IGYCDOPT program after you remove the asterisk parameter from the option. You then place the resulting IGYCDOPT module in a temporary library to be accessed as a JOBLIB or STEPLIB at compile time.

Sample installation jobs

COBOL for OS/390 & VM provides two sample installation jobs that you can modify and then use to change compiler option defaults. One sample job provides an example of how to change the IBM-supplied compiler option defaults. The other sample job provides an example of how to override compiler options that have been fixed.

IGYWDOPT
Use this sample installation job to change the IBM-supplied defaults using SMP/E.

IGYWUOPT
Use this sample installation job to create a module outside of SMP/E in which you can specify different defaults if it becomes necessary to override compiler options that have been fixed with the IGYCDOPT program.

These jobs are located in the COBOL sample data set IGY.V2R2M0.SIGYSAMP.

Modifying compiler options and phases

If you plan to modify compiler option values and compiler phases, use the IGYCOPT syntax format shown in Figure 1. The IBM-supplied default values are shown both on the planning worksheets and immediately following each syntax diagram. The syntax diagrams also show the default as explained in "How to read the syntax diagrams" on page vi.

Compiler options and phases, and their defaults, are discussed in the following sections. Review these options and phases and their default values to determine the values most suitable for your applications.

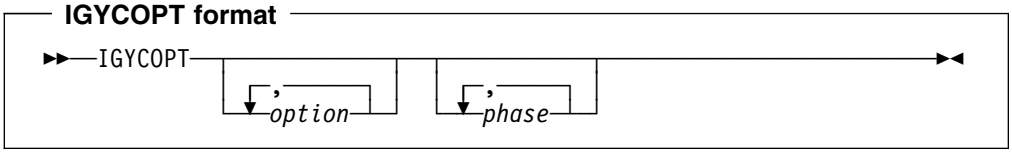


Figure 1. Syntax Format for IGYCOPT compiler options and phases macro

IGYCDOPT worksheet for compiler options

The following worksheet will help you to plan and code the compiler options portion of the IGYCDOPT program. To complete the worksheet, fill in the 'Enter * for Fixed' and the 'Enter Selection' columns.

The IGYCDOPT worksheet also includes a section for compiler phases. That section of the worksheet can be found in "IGYCDOPT worksheet for compiler phases" on page 10, following the discussion on compiler phases.

Notes:

1. Coding the asterisk [*], when modifying a compiler option default value, indicates that the option is to be fixed and cannot be overridden by an application programmer.
2. The ALOWCBL, DBCSXREF, LVLINFO, and NUMCLS options cannot be overridden at compile time. Therefore, the 'Enter * for Fixed' worksheet entries for these options are blank.
3. The IBM-supplied default value for ADEXIT, INEXIT, LVLINFO, LIBEXIT, and PRTEXIT is null. Therefore, the 'IBM-Supplied Default' entries for these options are blank.
4. The DUMP compiler option cannot be set through the IGYCDOPT program. Unless changed at compile time, DUMP is always set to NODUMP.

Table 1 (Page 1 of 2). IGYCDOPT Worksheet for options

Compiler Option	Enter * for Fixed	Enter Selection	IBM-Supplied Default	Syntax Description
ADATA=	____	_____	**NO**	Page 15
ADEXIT=	____	_____		Page 16
ADV=	____	_____	**YES**	Page 16
ALOWCBL=		_____	**YES**	Page 17
ANALYZE=	____	_____	**NO**	Page 17
ARITH=	____	_____	**COMPAT**	Page 18
AWO=	____	_____	**NO**	Page 18
BUF=	____	_____	**4K**	Page 19
CMPR2=	____	_____	**NO**	Page 19
COMPILE=	____	_____	**NOC(S)**	Page 20
CURRENCY=	____	_____	**NO**	Page 21
DATA=	____	_____	**31**	Page 22
DATEPROC=	____	_____	**NO**	Page 23
DBCS=	____	_____	**NO**	Page 23
DBCSXREF		_____	**NO**	Page 24
DECK=	____	_____	**NO**	Page 25
DIAGTRUNC=	____	_____	**NO**	Page 25
DLL=	____	_____	**NO**	Page 26
DYNAM=	____	_____	**NO**	Page 26
EXPORTALL=	____	_____	**NO**	Page 27
FASTSRT=	____	_____	**NO**	Page 27
FLAG=	____	_____	**(I)**	Page 28
FLAGMIG=	____	_____	**NO**	Page 29
FLAGSTD=	____	_____	**NO**	Page 29
IDLGEN=	____	_____	**NO**	Page 31
INTDATE=	____	_____	**ANSI**	Page 32
INEXIT=	____	_____		Page 32
LANGUAGE=	____	_____	**ENGLISH**	Page 33
LIB=	____	_____	**NO**	Page 34
LIBEXIT=	____	_____		Page 34
LINECNT=	____	_____	**60**	Page 35
LIST=	____	_____	**NO**	Page 35
LITCHAR=	____	_____	**QUOTE**	Page 36
LVLINFO=		_____		Page 36
MAP=	____	_____	**NO**	Page 36
NAME=	____	_____	**NO**	Page 37
NUM=	____	_____	**NO**	Page 37
NUMCLS=		_____	**PRIM**	Page 38

Table 1 (Page 2 of 2). IGYCDOPT Worksheet for options

Compiler Option	Enter * for Fixed	Enter Selection	IBM-Supplied Default	Syntax Description
NUMPROC=	____	_____	**NOPFD**	Page 38
OBJECT=	____	_____	**YES**	Page 39
OFFSET=	____	_____	**NO**	Page 40
OPT=	____	_____	**NO**	Page 40
OUTDD=	____	_____	**SYSOUT**	Page 41
PGMNAME=	____	_____	**COMPAT**	Page 41
PRTEXIT=	____	_____		Page 42
SQL=	____	_____	**NO**	Page 46
RENT=	____	_____	**NO**	Page 42
RMODE=	____	_____	**AUTO**	Page 43
SEQ=	____	_____	**YES**	Page 44
SIZE=	____	_____	**MAX**	Page 45
SOURCE=	____	_____	**YES**	Page 45
SPACE=	____	_____	**1**	Page 46
SSRANGE=	____	_____	**NO**	Page 47
TERM=	____	_____	**NO**	Page 47
TEST=	____	_____	**NO**	Page 48
TRUNC=	____	_____	**STD**	Page 49
TYPECHK=	____	_____	**NO**	Page 50
VBREF=	____	_____	**NO**	Page 51
WORD=	____	_____	***NO**	Page 51
XREFOPT=	____	_____	**NO**	Page 52
YRWINDOW=	____	_____	**1900**	Page 53
ZWB=	____	_____	**YES**	Page 54

Planning to place compiler phases in shared storage

In an OS/390 environment, you might want to make some load modules resident in a link-pack area in order to minimize the search for them when a COBOL for OS/390 & VM program is run or when the modules will be shared. You might also want to make some or all of the compiler phases resident. Note that the term “shared storage” is used generally to describe the Link-Pack Area (LPA), the Extended Link-Pack Area (ELPA), or Modified Link-Pack Area (MLPA). Except where specifically stated, all three terms are referenced when **Link-Pack Area** is used in the following sections.

Why place the compiler phases in shared storage?

All compiler modules, except the rundump modules (IGYCRDPR and IGYCRDSC) and the reserved word utility (IGY8RWTU), are eligible for placement in the shared storage of OS/390 machines. All compiler phases except IGYCRCTL and IGYCSIMD have RMODE(ANY)and AMODE(31). Since IGYCRCTL and IGYCSIMD have RMODE(24), they can be placed in the LPA or MLPA, but not in the ELPA. Shared storage is an area of storage that is the same for each virtual address space. Information stored there can be shared and does not have to be loaded in the user region because it is the same space for all users. By sharing the information, more space is made available for the compiler work area.

Note: The COBOL for OS/390 & VM, COBOL for MVS & VM, and VS COBOL II compilers use the same module names; thus, only one set of phases can be placed in the LPA for any given initialization of the operating system.

The IGYCDOPT program indicates where each compiler phase is loaded—either inside (IN) or outside (OUT) of the user region. By placing compiler phases in the MLPA, the compiler has more storage available for the user's program.

If you indicate that a phase will not reside in the user region, you must ensure that you actually place the phase in shared storage. This information is used by the compiler to determine how much storage to leave for the system to load compiler phases in the user region. For a description of how to place a phase in shared storage, see *Initialization and Tuning* manual for your particular OS/390 environment, as listed under “Related publications” on page 70.

We recommend that the following four phases be placed in a shared system area:

IGYCRCTL because it is resident in the user region throughout compilation.

IGYCSIMD because it is resident in the user region throughout compilation.

IGYCPGEN because it is one of the two largest compiler phases.

IGYCSCAN because it is the other of the two largest compiler phases.

You can select any or all compiler phases to be placed in shared storage based on frequency of concurrent use and phase size. If your facility seldom uses the compiler, there might be no advantage to installing any phases in shared storage. However, if there are frequent compilations and sufficient MLPA storage is available, making the entire compiler resident might be advantageous. If sufficient shared storage is not available, priority should be given to IGYCRCTL and IGYCSIMD, the two phases that are always resident in the user region during compilation. Also, if sufficient shared storage is not available, priority should be given to, IGYCPGEN and IGYCSCAN, the largest compiler phases.

Another advantage of placing compiler phases in shared storage is that, at compile time, the initialization logic allocates in the user region a storage block of sufficient size to contain the largest phase not resident in shared storage. Minimizing the space allocation for any given user region size means more space for the compilation process, (which allows larger programs to be compiled within a given user region) and possibly a more efficient compile. The IGYCPGEN and IGYCSCAN compiler phases are approximately 250KB larger than the next largest compiler phase. This can make a significant difference if you are compiling using the 760KB minimum region size.

Compiler phases and their defaults

To indicate where each compiler phase is loaded in relation to the user region, specify either IN or OUT. See “Why place the compiler phases in shared storage?” on page 6, for more information on why you might or might not want to change these defaults.

IN Indicates that the compiler phase is loaded into the user region from a library available at compile time. The compiler reserves storage for the phase from the value specified in the SIZE option.

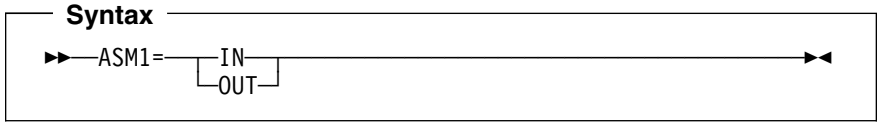
Even though IN is specified for a compiler phase, it still can be placed into the shared system area. However, the compiler control phase ensures that the main storage area reserved for compiler phases is large enough to contain the largest phase for which IN is specified. This will cause some storage to be unused.

OUT Indicates that the compiler phase is not loaded into the user region from the library, and therefore must reside in a shared system area, such as the MLPA.

IGYCASM1

The Assembly 1 phase that determines the object module storage, allocates the permanent and temporary registers, and optimizes addressability for data and procedure references. It also creates object text for data areas.

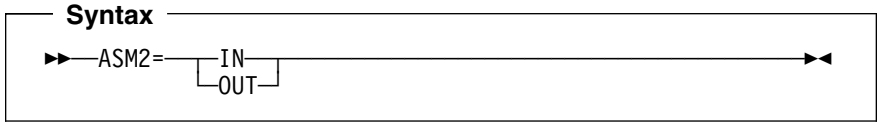
Syntax



IGYCASM2

The Assembly 2 phase that completes preparation of the object program and creates object text, listings, punch data sets, and tables for the debugging feature.

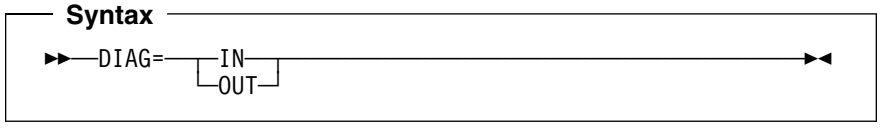
Syntax



IGYCDIAG

The diagnostic phase that processes E-form text and generates compiler diagnostics for source program errors. It includes IGYCDIAG plus the following message modules: IGYCxx$D, IGYCxx$1, IGYCxx$2, IGYCxx$3, IGYCxx$4, IGYCxx$5, and IGYCxx$8, where xx is EN, UE, or JA.

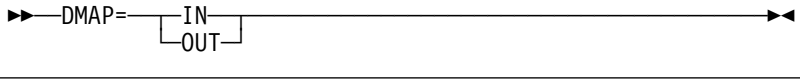
Syntax



IGYCDMAP

The DMAP phase that prepares text for output requested by the MAP option.

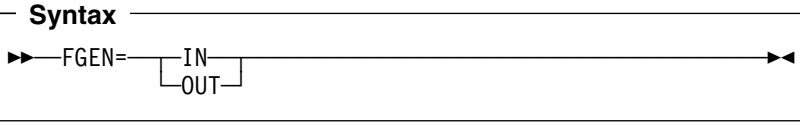
Syntax



IGYCFGEN

The file generation phase that generates the control blocks for the FDs and SDs defined in the program.

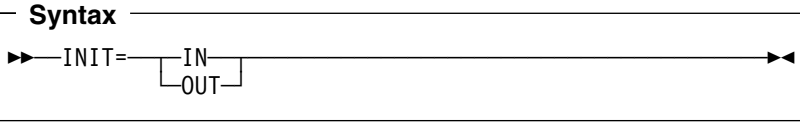
Syntax



IGYCINIT

The initialization phase that does housekeeping to prepare for running of the processing phases.

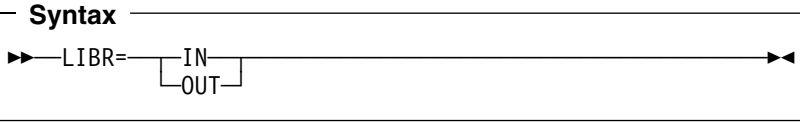
Syntax



IGYCLIBR

The COPY phase that processes library source text and does a syntax check of the COPY, BASIS, and REPLACE statements.

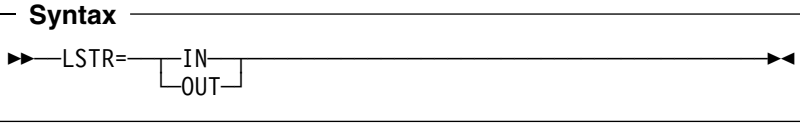
Syntax



IGYCLSTR

The source listing phase that prints the source listing with embedded cross reference and diagnostic information.

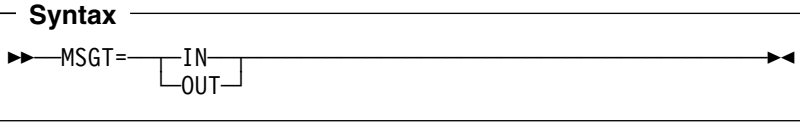
Syntax



IGYCMSGT

Represents the header text table and diagnostic message level tables. It includes the following modules: IGYCxx$R, IGYCLVL0, IGYCLVL1, IGYCLVL2, IGYCLVL3, and IGYCLVL8, where xx is EN, UE, or JA.

Syntax



IGYCOPTM

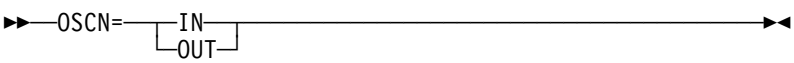
The optimizer phase that restructures the PERFORM statements and eliminates duplicate computations.

Syntax

IGYCOSCN

The option scanning phase that determines the default options, processes the EXEC PARM options, and processes the PROCESS (CBL) statements.

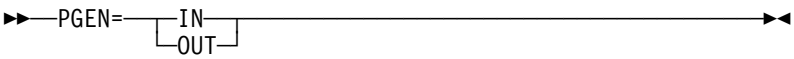
Syntax



IGYCPGEN

The procedure generation phase that supplies code for all procedure source verbs.

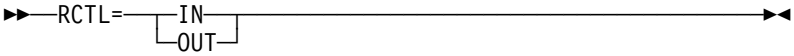
Syntax



IGYCRCTL

The resident control phase that establishes the size of compiler common and working storage, and performs initialization of program common storage.

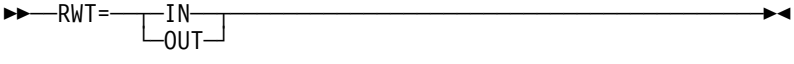
Syntax



IGYCRWT

The normal reserved word table.

Syntax



IGYCSCAN

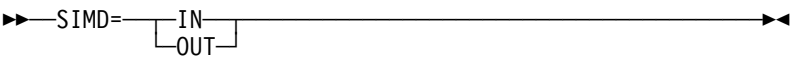
The scanning phase that does syntax and semantic analysis of the source program and translates the source to intermediate text.

Syntax

IGYCSIMD

The system interface phase for the COBOL for OS/390 & VM compiler. This phase is called by all other compiler phases to perform system-dependent functions.

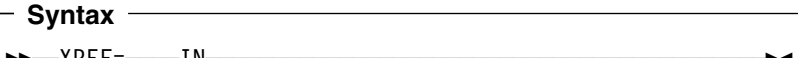
Syntax



IGYCXREF

The XREF phase that sorts user-names and procedure-names in EBCDIC collating sequence.

Syntax



IGYCDOPT worksheet for compiler phases

The following worksheet will help you to plan and code the phases portion of the IGYCDOPT program. Circle the value you plan to assign to each phase. For more information on the values that can be assigned to each phase, see “Compiler phases and their defaults” on page 7.

Note: All phase defaults are initially set to **IN**.

Table 2. IGYCDOPT program worksheet for compiler phases

Phase	Circle Selection	Syntax Description
ASM1=	**IN** / OUT	Page 7
ASM2=	**IN** / OUT	Page 7
DIAG=	**IN** / OUT	Page 7
DMAP=	**IN** / OUT	Page 7
FGEN=	**IN** / OUT	Page 8
INIT=	**IN** / OUT	Page 8
LIBR=	**IN** / OUT	Page 8
LSTR=	**IN** / OUT	Page 8
MSGT=	**IN** / OUT	Page 8
OPTM=	**IN** / OUT	Page 8
OSCN=	**IN** / OUT	Page 9
PGEN=	**IN** / OUT	Page 9
RCTL=	**IN** / OUT	Page 9
RWT=	**IN** / OUT	Page 9
SCAN=	**IN** / OUT	Page 9
SIMD=	**IN** / OUT	Page 10
XREF=	**IN** / OUT	Page 10

Planning to create an additional reserved word table

When you install COBOL for OS/390 & VM, you have access to the following reserved word tables:

- IGYCRWT—The default reserved word table provided for your entire facility.
- IGYCCICS—A CICS-specific reserved word table, provided as an alternate reserved word table (See "CICS reserved word table (IGYCCICS)" on page 12).
- IGYCNOOO—An alternate reserved word table provided for compatibility with COBOL programs that use the reserved words introduced in COBOL for MVS & VM Version 1 Release 2. (See "NOOO reserved word table (IGYCNOOO)" on page 12).

You can create additional reserved word tables after installation. (During compilation, the value of the WORD compiler option determines which reserved word table is used.)

Why create additional reserved word tables?

You can create additional reserved word tables to do the following:

- Translate the reserved words into another language, such as French or German.
- Prevent application programmers from using a particular COBOL for OS/390 & VM instruction, such as GO TO.
- Control the usage of nested programs.
- Flag those COBOL verbs that do not have a run-time function on CICS, such as READ/WRITE.

Controlling use of nested programs

To restrict the use of nested programs, without restricting any other COBOL language features, modify the reserved word table. Do this with the INFO and RSTR control statements. For instructions on how to make these modifications, see "Procedure for creating or modifying a reserved word table" on page 58.

Reserved word tables supplied with COBOL for OS/390 & VM

Three reserved word tables are on the installation tape:

- Default reserved word table
- Non-OO compatibility reserved word table
- CICS reserved word table

Default reserved word table (IGYCRWT)

The default reserved word table is described in an appendix of *COBOL Language Reference*.

NOOO reserved word table (IGYCNOOO)

COBOL for OS/390 & VM provides an alternate reserved word table that does **not** reserve the object-oriented COBOL words and function listed below.

CLASS-ID	METACLASS	REPOSITORY
END-INVOKE	METHOD	RETURNING
FACTORY	METHOD-ID	SELF
INHERITS	OBJECT	SUPER
INVOKE	OVERRIDE	
LOCAL-STORAGE	RECURSIVE	

IGYCNOOO is used for compatibility with COBOL programs that use these words reserved in COBOL for OS/390 & VM.

Using the table: To use the NOOO reserved word table, you must specify the WORD(NOOO) compiler option. To have the NOOO reserved word table used as the default, you must set the default value of the WORD compiler option to `WORD=NOOO`.

Location of the table: The data used to create the NOOO reserved word table is in member IGY8NOOO in IGY.V2R2M0.SIGYSAMP.

Note: The high-level qualifiers of this name might have been changed when COBOL for OS/390 & VM was installed.

CICS reserved word table (IGYCCICS)

COBOL for OS/390 & VM provides an alternate reserved word table specifically for CICS application programs. It is set up so that COBOL words not supported under CICS are flagged by the compiler with an error message.

Contents of the table: The CICS reserved word table is the same as the default reserved word table except that the following COBOL words are marked as restricted (RSTR):

CLOSE	I-O-CONTROL	SD
DELETE	MERGE	SORT
FD	OPEN	START
FILE	READ	WRITE
FILE-CONTROL	RERUN	
INPUT-OUTPUT	REWRITE	

SORT Users

If you intend to use the SORT statement under CICS (COBOL for OS/390 & VM supports an interface for the SORT statement under CICS), you must modify the CICS reserved word table before using it. The words underlined above must be removed from the list of words marked as restricted, because they are required for the SORT function.

Using the table: To use the CICS reserved word table, you must specify the WORD(CICS) compiler option. To have the CICS reserved word table used as the default, you must set the default value of the WORD compiler option to WORD=CICS.

Location of the table: The data used to create the CICS reserved word table is in member IGY8CICS in IGY.V2R2M0.SIGYSAMP.

Note: The high-level qualifiers of this name might have been changed when COBOL for OS/390 & VM was installed.

COBOL for OS/390 & VM compiler options

This section describes those compiler options whose default values you can change. The notes that accompany some of the descriptions provide additional information about these options, such as how they interact with other options during compilation. This information might help you to make decisions about which default values are appropriate for your installation. For more information on using the compiler options, see *COBOL for OS/390 & VM Programming Guide*.

Important

Please confer with the application programmers at your site while you plan the customization of COBOL for OS/390 & VM. Doing so will ensure that the modifications you make serve their needs and support the applications being developed.

Specifying COBOL compiler options

When you specify compiler options in the IGYCOPT or IGYCINS macros, both the option name and its value must be specified in uppercase. If you don't specify the option name in uppercase, both the option name and its value are ignored and the default value is used instead. No error message is issued. If only the option value is not in uppercase, an error message will be issued indicating that an invalid option value has been specified.

Options in support of the COBOL 85 standard

The following option values are required to conform to the COBOL 85 Standard:

ADV=YES
CMPR2=NO
CURRENCY=NO
DATEPROC=NO
DBCS=NO
DLL=NO
DYNAM=YES
FASTSRT=NO
FLAGMIG=NO
INTDATE=ANSI
LIB=YES
LITCHAR=QUOTE
NAME=ALIAS or NOALIAS
NUM=NO
NUMPROC=NOPFD or MIG
SEQ=NO
TRUNC=STD
WORD=NO
ZWB=YES

Note: The term "COBOL 85 Standard" is used in this book to refer to the combination of the following standards:

1. ISO 1989:1985, Programming Languages - COBOL.

   ISO 1989/Amendment 1, Programming Languages - COBOL - Amendment 1: Intrinsic Function Module.

2. X3.23-1985, American National Standard for Information Systems - Programming Language - COBOL.

   X3.23a-1989, American National Standard for Information Systems - Programming Language - Intrinsic Function Module for COBOL.

X3.23b-1995, American National Standard for Information Systems - Programming Language - Corrections Amendment for COBOL.

Conflicting compiler options

If you specify certain compiler option values, a conflict with other compiler options might result. Table 3 will help you resolve possible conflicts between compiler options.

Table 3 (Page 1 of 2). Conflicting compiler options

Compiler Option	Conflicts with:
ANALYZE=YES	ADATA=NO LIB=NO SQL=YES
CMPR2=NO	FLAGMIG=YES
CMPR2=YES	ADATA=YES ANALYZE=YES ARITH=EXTEND CURRENCY=(other than NO) DATEPROC=YES DBCS=YES DIAGTRUNC=YES DLL=YES EXPORTALL=YES FLAGSTD=(other than NO) IDLGEN=YES INTDATE(LILIAN) OPT(FULL) PGMNAME=(other than COMPAT) RMODE=(other than AUTO) SQL=YES TYPECHK=YES YRWINDOW=(other than 1900)
COMPILE=NOC	DECK=YES LIST=YES OBJECT=YES OFFSET=YES OPT=STD or OPT=FULL TEST=(other than NO) VBREF=YES
DATEPROC=YES	CMPR2=YES FLAGSTD=(other than NO)
DBCS=YES	CMPR2=YES FLAGMIG=YES FLAGSTD=(other than NO)
DBCSXREF=(other than NO)	XREFOPT=NO
DLL=YES	DYNAM=YES RENT=NO
DYNAM=YES	DLL=YES
EXPORTALL=YES	DLL=NO DYNAM=YES RENT=NO

Table 3 (Page 2 of 2). Conflicting compiler options

Compiler Option	Conflicts with:
FLAGMIG=YES	CMPR2=NO DBCS=YES FLAGSTD=(other than NO)
FLAGSTD=(other than NO)	ADV=NO CMPR2=YES DATEPROC=(other than NO) DBCS=YES DYNAM=NO FLAGMIG=YES LIB=NO LITCHAR=APOST NUM=YES NUMPROC=PFD SEQ=YES TRUNC=OPT or BIN WORD=(other than NO or RWT) ZWB=NO
LIST=YES	OFFSET=YES
OBJECT=NO	TEST=(other than NO)
OFFSET=YES	LIST=YES
OPT=STD or OPT=FULL	TEST=(other than NONE for hook location)
SQL=YES	LIB=NO
TEST=(other than NO)	OBJECT=NO
TEST=(other than NONE for hook-location suboption)	OPT=STD or OPT=FULL
WORD=xxxx	FLAGSTD=(other than NO)
XREFOPT=NO	DBCSXREF=(other than NO)

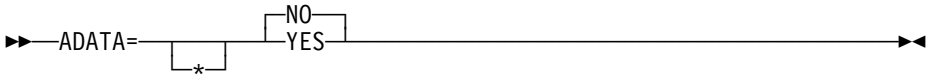
Compiler options syntax and descriptions

The syntax diagrams on the following pages describe each modifiable compiler option. The text below each diagram describes the effect of selecting a specific parameter.

Note: Notice the absence of the DUMP option. Unless you change it at compile time, DUMP is always set to NODUMP. This option is not for general use; it is only used at the request of an IBM representative.

ADATA

Syntax



Default

ADATA=NO

YES
Produces the Associated Data file with the appropriate records.

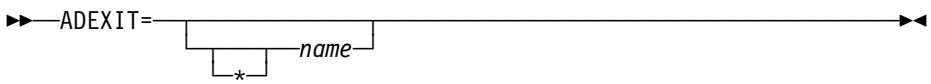
NO
Does not produce the Associated Data file.

Notes:

1. The ADATA option can be specified only at invocation via the option list, on the PARM field of JCL, as a command option, or as an installation default.
2. Selection of the Japanese language option might result in DBCS characters written records in the Associated Data file.
3. Specification of NOCOMPILE(W|E|S) might stop compilation prematurely, resulting in a loss of specific Associated Data Records.
4. Specification of INEXIT prohibits identification of the compilation source file for the Associated Data file.

ADEXIT

Syntax

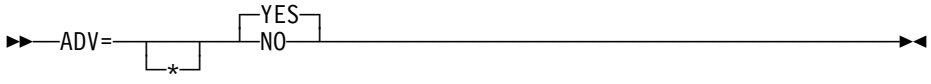


Default
No exit specified.

name
Identifies a module to be used with the EXIT compiler option. When the sub-option for this user exit is specified, the compiler loads that module and calls it for each record that is written to the Associated Data file. For more specific information, see *COBOL for OS/390 & VM Programming Guide*.

ADV

Syntax



Default
ADV=YES

YES
Adds one byte to the record length for the printer control character. This option might be useful to programmers who use WRITE...ADVANCING in their source files. The first character of the record does **not** have to be explicitly reserved by the programmer.

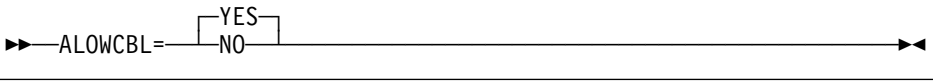
NO
Does not adjust the record length for WRITE...ADVANCING. The compiler uses the first character of the specified record area to place the printer control character. The application programmer must ensure that the record description allows for this additional byte.

Notes:

1. ADV=YES conforms to the COBOL 85 Standard. With ADV=YES, the record length on the physical device is one byte larger than the record description length in the source program.
2. If the record length for the output file is not defined in the source code, COBOL ensures that the DCB parameters are appropriately set.
3. If ADV=YES is specified, and the record length for the output file has been defined in the source code, the programmer ***must*** specify the record description length as one byte larger than the source program record description. The programmer ***must*** also specify the block size in correct multiples of the larger record size.
4. If the LINAGE clause is specified in a file description (FD), the compiler treats that file as if ADV=YES has been specified.

ALOWCBL

Syntax



Default
ALOWCBL=YES

YES
Allows the use of the PROCESS (or CBL) statements in COBOL programs.

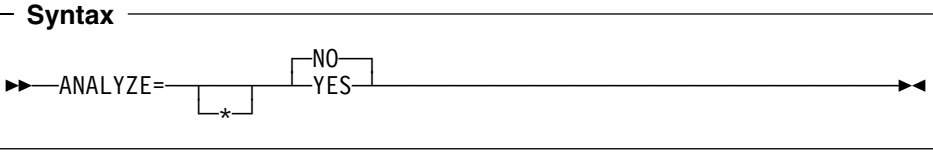
NO
Diagnoses the use of PROCESS (or CBL) statements in a program as an error.

Notes:

1. ALOWCBL cannot be overridden at compile time because it cannot be included in the PROCESS (or CBL) statement or, additionally for CMS, on the COBOL2 command line.
2. The PROCESS (or CBL) statement specifies compiler option parameters within source programs. If your installation requirements do not allow compiler options to be specified in a source program, specify ALOWCBL=NO.

ANALYZE

Syntax



Default
ANALYZE=NO

YES
Checks the syntax of imbedded SQL and CICS statements in addition to native COBOL statements.

NO
Does not check the syntax of imbedded SQL and CICS statements.

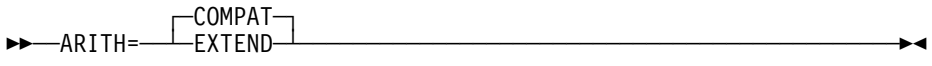
Notes:

1. ANALYZE can be set as the installation default option or as a compiler invocation option, but cannot be set on a CBL or PROCESS statement.
2. No executable code is generated when this compiler option is specified, regardless of the COMPILE option setting.
3. The ANALYZE option enables COPY/BASIS/REPLACE processing, regardless of the LIB option setting.
4. You can specify the ADATA=YES option with this option to create a SYSADATA file for later analysis by program understanding tools, such as the Year 2000 tool.
5. Specifying ANALYZE=YES forces the handling of the following character strings as reserved words:

   CICS
   EXEC
   END-EXEC
   SQL

ARITH

Syntax



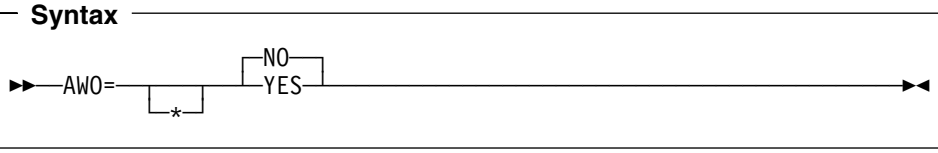
Default
ARITH=COMPAT

COMPAT
Specifies 18 digits as the maximum precision for decimal data.

EXTEND
Specifies 31 digits as the maximum precision for decimal data.

AWO

Syntax



Default
AWO=NO

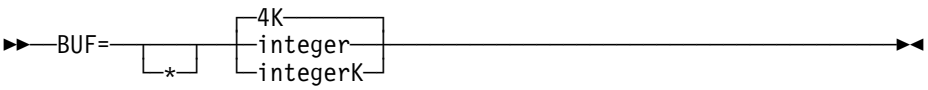
YES
Activates the APPLY-WRITE-ONLY clause for any file within the program that is physical sequential with Variable Block format regardless of whether or not the APPLY-WRITE-ONLY clause is specified in the program.

Performance consideration: Using AWO=YES generally results in fewer calls to Data Management Services for run-time files when handling I/O.

NO
Does not activate the APPLY-WRITE-ONLY clause for any file within the program that is physical sequential with Variable Block format unless the APPLY-WRITE-ONLY clause is specified in the program.

BUF

Syntax



Default
BUF=4K

integer
Specifies the amount of dynamic storage, in bytes, to be allocated to each compiler work file buffer. The minimum value is 256 bytes.

Performance consideration: Using BUF=integer generally improves compile-time performance by reducing the number of I/Os to the work files.

integerK
Specifies the amount of dynamic storage to be allocated to buffers in increments of 1K (1024) bytes.

Notes:

1. BUF and SIZE values are used by the compiler to determine how much storage to use during compilation. The amount allocated to the buffers is included in the amount of main storage available to the compiler for the SIZE option.
2. BUF cannot exceed the track capacity for the device used, nor can it exceed the maximum allowed by data management services.

CMPR2

Syntax

Default
CMPR2=NO

YES
Generates code that is run-time compatible with valid VS COBOL II Version 1 Release 2 programs. See *COBOL for OS/390 & VM Compiler and Run-Time Migration Guide* for details of language elements that are sensitive to CMPR2.

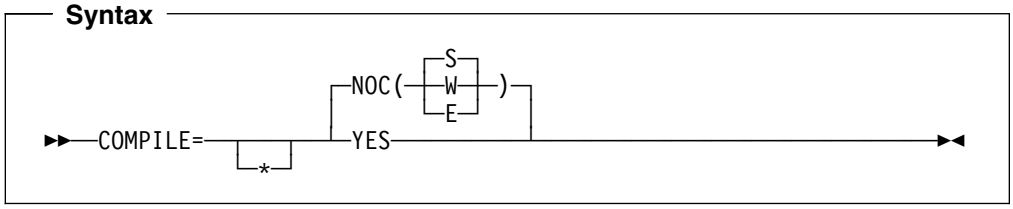
NO
Generates code that might not be run-time compatible with valid VS COBOL II, Release 2 programs for some 1985 COBOL standard language constructs.

Notes:

1. Functions are **NOT** allowed under CMPR2. The word **function** might be a dataname under CMPR2, but under NOCMPR2 it is a reserved word.
2. New language extensions for defining and manipulating procedural pointers are supported only with NOCMPR2.
3. Object orientation is supported only with NOCMPR2.
4. CMPR2=YES cannot be specified with certain other compiler options. See "Conflicting compiler options" on page 14 for more information on conflicting compiler options.
5. Specification of this option might result in different attribute information being produced and passed back to IBM Dictionary Services.
6. CMPR2=NO supports the COBOL 85 standard.

COMPILE

Syntax



Default
COMPILE=NOC(S)

YES
Indicates that you want full compilation, including diagnostics and object code.

NOC
Indicates that you want only a syntax check.

NOC(W)
NOC(E)
NOC(S)
Specifies an error message level: W is warning; E is error; S is severe. When an error of the level specified or of a more severe level occurs, compilation stops, and only syntax checking is done for the balance of the compilation.

Notes:

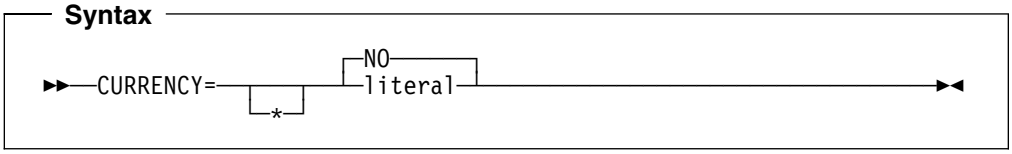
1. If the following compiler options are explicitly or implicitly specified in a program, COMPILE=NOC issues a warning message during compile time:

   DECK=YES
   LIST=YES
   OFFSET=YES
   OPT=STD or OPT=FULL
   OBJECT=YES
   TEST=(other than NO)
   VBREF=YES

   Specifying these options together with COMPILE=NOC, while attempting to assemble the customization macro, results in a nonzero return code.
2. Specifying NOCOMPILE might affect the Associated Data file by stopping compilation prematurely, resulting in loss of specific messages.

CURRENCY

Syntax



The COBOL default currency symbol is the dollar sign ($). The CURRENCY option allows you to define an alternate default currency symbol.

Default
CURRENCY=NO

literal
Represents the default currency symbol you want to use in your program.

The literal must be a nonnumeric literal representing a one-byte EBCDIC character that must not be any of the following:

- Digits zero (0) through nine (9)
- Uppercase alphabetic characters: A B C D P R S V X Z
- Lowercase alphabetic characters a through z
- The space
- Special characters: * + - / , . ; ( ) = "
- Uppercase alphabetic character G, if the COBOL program defines a DBCS item with the PICTURE symbol G. The PICTURE clause will be invalid for that DBCS item because the symbol G is considered to be a currency symbol in the PICTURE clause.
- Uppercase alphabetic character N, if the COBOL program defines a DBCS item with the PICTURE symbol N. The PICTURE clause will be invalid for that DBCS item because the symbol N is considered to be a currency symbol in the PICTURE clause.
- Uppercase alphabetic character E, if the COBOL program defines an external floating point item. The PICTURE clause will be invalid for the external floating-point item because the symbol E is considered to be a currency symbol in the PICTURE clause.

The literal (including hex literal) syntax rules are as follows:

- The literal delimiters can be either quotes or apostrophes regardless of any option setting for literal delimiters.
- When an apostrophe (') is to be the currency sign, the embedded apostrophe must be "doubled", that is, two apostrophes must be coded to represent one apostrophe within the literal. For example:

  ```
  '''' or "'"
  ```

- The format for a hex literal specification is as follows:

  ```
  X'H1H2' or X"H1H2"
  ```

  where H1H2 is a valid hexadecimal value representing a one-byte EBCDIC character conforming to the rules for the currency sign literal as described above. Alphabetic characters in the hex literal must be in uppercase.

Note: Hex values of X'20' or X'21' are not allowed.

NO
Indicates that no alternate default currency sign is provided by way of the CURRENCY option, and the dollar sign will be used as the default currency sign for the program if the CURRENCY option is not specified at compile time.

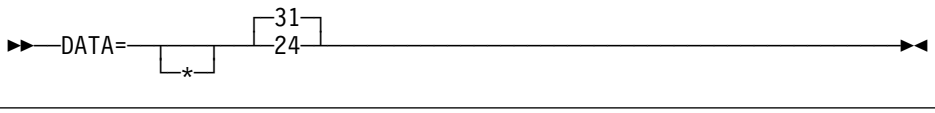
The value NO provides the same results for the source program as omitting the CURRENCY SIGN clause in the COBOL source program.

Notes:

1. You can use the CURRENCY option as an alternative to the CURRENCY SIGN clause (which is specified in the COBOL source program) for selecting the currency symbol you use in the PICTURE clause of your COBOL program.
2. When both the CURRENCY option and the CURRENCY SIGN clause are used in a program, the symbol specified in the CURRENCY SIGN clause is the currency symbol in a PICTURE clause when that symbol is used (even if the CURRENCY option is fixed {*}).

DATA

Syntax



Default
DATA=31

31 Causes allocation of user data areas, such as working storage and FD record areas, from unrestricted storage or in space acquired by a GETMAIN with the LOC=ANY option. Specifying this option can result in storage being acquired in virtual addresses either above or below 16 megabytes. The operating system generally satisfies the request with space in virtual addresses above 16 megabytes, if it is available.

24 Causes allocation of user data areas in virtual addresses below 16 megabytes in storage acquired by a GETMAIN with the LOC=BELOW option.

Specify DATA=24 for programs compiled with the RENT option that are passing data parameters to programs in 24-bit mode. This includes the following cases:

- A COBOL for OS/390 & VM program is passing items in its working storage to an AMODE(24) program.
- A COBOL for OS/390 & VM program is passing, by reference, data items received from its caller to an AMODE(24) program. DATA=24 is required even when the data received is below the 16-megabyte line.

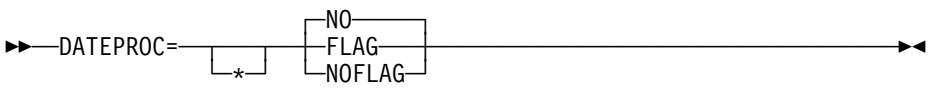
Otherwise, the data might not be addressable by the called program.

Notes:

1. When a program is compiled with the RENT option, the DATA option controls how dynamic storage is acquired.
2. The DATA option has no effect on programs compiled with the NORENT option.

DATEPROC

Syntax



The DATEPROC option determines whether the compiler will perform date processing using the DATE FORMAT clause and other language constructs.

Default
DATEPROC=NO

FLAG
Recognizes the DATE FORMAT clause and performs automatic date processing. In addition, specifying DATEPROC=FLAG flags, either with an information-level message or a warning-level message as appropriate, each statement that uses or is affected by date processing.

NOFLAG
Recognizes the DATE FORMAT clause and performs automatic date processing. Statements that use or are affected by date processing are not flagged with information-level or warning-level messages.

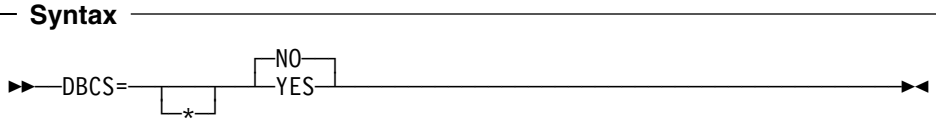
NO
Treats the DATE FORMAT clause as comments and disables automatic date processing. In the case of the new intrinsic functions, specifying DATEPROC=NO generates object code that returns a default value whenever a new intrinsic function is used.

Notes:

1. Error-level and severe-level messages are issued regardless of whether DATEPROC=FLAG or DATEPROC=NOFLAG is specified.
2. DATEPROC=NO supports the COBOL 85 standard.

DBCS

Syntax



Default
DBCS=NO

YES
Recognizes X'0E' and X'0F' in a nonnumeric literal and treats them as Shift-Out and Shift-In control characters for delimiting DBCS data.

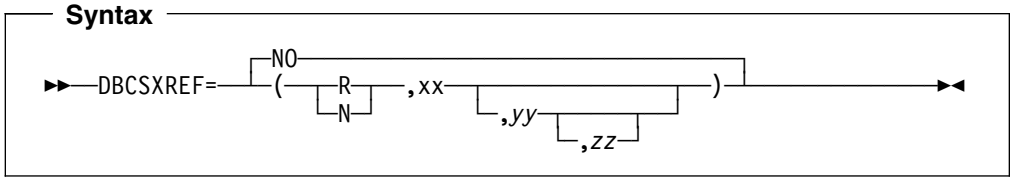
NO
Does not recognize X'0E' and X'0F' in a nonnumeric literal.

Notes:

1. The presence of DBCS data inside the nonnumeric literal might cause the compiler to disallow certain uses of that literal. For example, DBCS characters are not allowed as program names or DDNAMES.
2. DBCS=YES can conflict with other compiler options. See “Conflicting compiler options” on page 14 for conflict resolution information.
3. DBCS=NO supports the COBOL 85 standard.
4. Specification of this option affects the treatment of SHIFT OUT/SHIFT IN characters in nonnumeric literals passed back to IBM Dictionary Services.

DBCSXREF

Syntax



OS/390 only: This option is not applicable under VM/CMS.

Default
DBCSXREF=NO

NO
Specifies that no ordering program is used for cross reference of DBCS names. If the XREF phase is specified, a DBCS names cross-reference listing is provided based on their physical order in the program.

R Specifies that the DBCS Ordering Support Program (DBCSOS) is loaded into the user region.

N Specifies that the DBCS Ordering Support Program (DBCSOS) is loaded into a shared system area such as the MLPA.

xx Names a load module of the relevant ordering program to produce DBCS cross references. It must be 8 characters in length.

yy Names an ordering type. It must be 2 characters in length. The default ordering type defined by the specified ordering program occurs if this parameter is omitted.

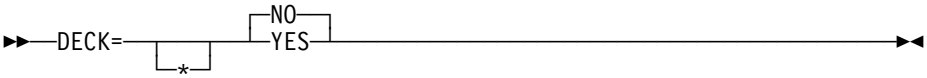
zz Names the encode table used by the specified ordering type. It must be 8 characters in length. The default encode table associated with the particular ordering type occurs if this parameter is omitted.

Notes:

1. The DBCS Ordering Support Program (DBCSOS) must be installed to specify anything other than DBCSXREF=NO.
2. If R is specified and the SIZE value is anything other than MAX, ensure that the user region is large enough to accommodate both the compiler and ordering program.
3. Specifying both XREFOPT=NO and DBCSXREF with an ordering program results in a nonzero return code while attempting to assemble the customization macro.
4. The assembly process terminates when validation diagnoses:
   - An invalid parameter length
   - Characters other than 'R' and 'N'
   - Missing parameters after a comma
   - Missing 'yy' when 'zz' is specified

DECK

Syntax



Default
DECK=NO

YES
Places the generated object code in a file defined by SYSPUNCH.

NO
Sends no object code to SYSPUNCH.

DIAGTRUNC

Syntax

Default
DIAGTRUNC=NO

YES
Causes the compiler to issue a severity-4 (Warning) diagnostic message for MOVE statements with numeric receivers when the receiving data item has fewer integer positions than the sending data item or literal.

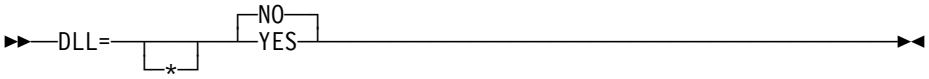
NO
Does not cause the compiler to produce a severity-4 message.

Notes:

1. The diagnostic is also issued for moves to numeric receivers from alphanumeric data names or literal senders, except when the sending field is reference modified.
2. There is no diagnostic for COMP-5 receivers, nor for binary receivers when you specify the TRUNC(BIN) option.

DLL

Syntax



OS/390 only: This option is not applicable under VM/CMS.

Default
DLL=NO

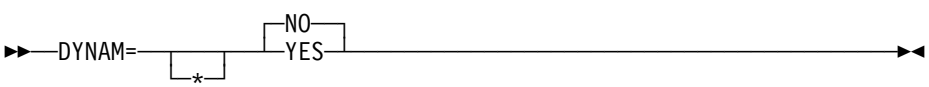
YES
Generates an object module that is enabled for Dynamic Link Library (DLL) support. DLL enablement is required if the program is part of a DLL, references DLLs, or if the program contains object-oriented COBOL syntax, for example, INVOKE statements, class definitions, or CALLs to SOMobjects™ API functions.

Specification of the DLL option requires that the NODYNAM and NOCMPR2 options are also used.

NO
Generates an object module that is not enabled for DLL usage.

DYNAM

Syntax



Default
DYNAM=NO

YES
Dynamically loads subprograms that are invoked through the CALL literal statement.

Performance Consideration: Using DYNAM=YES eases subprogram maintenance since the application is not relink-edited if the subprogram is changed. However, individual applications with CALL literal statements can experience some performance degradation due to a longer path length.

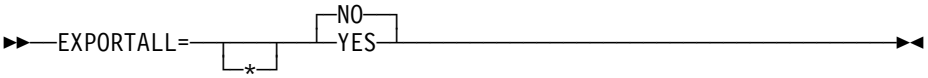
NO
Includes, in the calling program, the text files of subprograms called with a CALL literal statement into a single module file.

Notes:

1. DYNAM=YES is used in support of the COBOL 85 Standard.
2. NODYNAM must be used for programs containing CALLs to SOM API functions.
3. The DYNAM option has no effect on the CALL identifier statement at compile time. The CALL identifier statement always compiles to a dynamic call.
4. For **OS/390 only**, do not specify DYNAM=YES for applications running under CICS.

EXPORTALL

Syntax



OS/390 only: This option is not applicable under VM/CMS.

Default
EXPORTALL=NO

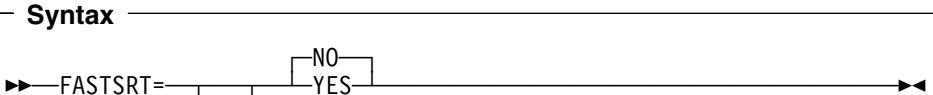
YES
Automatically exports certain symbols when the object deck is link-edited to form a DLL.

Specification of EXPORTALL requires that the DLL, NODYNAM, and NOCMPR2 options are also used.

NO
Does not export any symbols.

FASTSRT

Syntax



Default
FASTSRT=NO

YES
Specifies that the IBM DFSORT™ licensed program or comparable product performs I/O when using either the USING or GIVING option.

Performance Consideration: Using FASTSRT=YES eliminates the overhead, in terms of CPU time usage, of returning to COBOL for OS/390 & VM after each record is processed. However, there are restrictions you must follow if you choose to use this option. (For a detailed description of the restrictions, see *COBOL for OS/390 & VM Programming Guide*.)

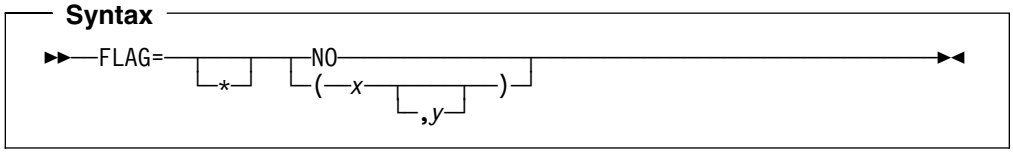
NO
Specifies that COBOL for OS/390 & VM does the I/O for the sort/merge.

Notes:

1. If FASTSRT is in effect at compile time, the compiler verifies that the FASTSRT interface can be used for all restrictions except:
   - those requiring use of a device other than a direct-access device for sort work files; and
   - the DCB parameter of the DD statement for the input/output file must match the File Description (FD) of the file.
2. If FASTSRT cannot be used, the compiler generates a diagnostic message and prevents the sort program from performing I/O when using either the USING or GIVING options. Therefore, it might be to your advantage to specify YES as the default.

FLAG

Syntax



Default
FLAG=(I)

Note: The second severity level used in this syntax must be equal to or higher than the first.

x I|W|E|S|U

Specifies that errors at or above the severity level specified are flagged and written at the end of the source listing.

ID	Type	Return Code
I	Information	0
W	Warning	4
E	Error	8
S	Severe error	12
U	Unrecoverable error	16

y I|W|E|S|U

The optional second severity level specifies the level of syntax messages embedded in the source listing in addition to being at the end of the listing.

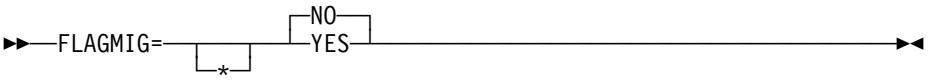
NO
Indicates that no error messages are flagged.

Notes:

1. If the messages are to be embedded, SOURCE must be specified at compile time. Embedded messages enhance productivity because they are placed after the referenced source statement.
2. Specification of FLAG(W|E|S) might result in the loss of entire classes of messages from the Events records in the Associated Date file. See *COBOL for OS/390 & VM Programming Guide* for more information.

FLAGMIG

Syntax



Default
FLAGMIG=NO

YES
Flags those VS COBOL II Release 2 language elements that might have different semantics than COBOL for OS/390 & VM.

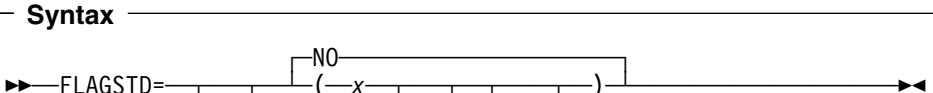
NO
Does not flag those VS COBOL II Release 2 language elements that have different semantics than COBOL for OS/390 & VM.

Notes:

1. FLAGMIG=YES has no effect when specified with CMPR2=NO.
2. FLAGSTD=YES and DBCS=YES all will override the FLAGMIG option.
3. FLAGMIG=NO supports the COBOL 85 standard.

FLAGSTD

Syntax



Default
FLAGSTD=NO

x Can be M, I, or H to specify flagging for a FIPS COBOL subset or standard.

M = ANS minimum subset of Standard COBOL.

I = ANS intermediate subset, composed of those additional intermediate subset language elements that are not part of the ANS minimum subset.

H = ANS high subset, composed of those additional high subset language elements that are not part of the ANS intermediate subset.

y Can be any one or two combinations of D, N, or S to further define the level of flagging produced.

D Specifies ANS Debug module Level 1

N Specifies ANS Segmentation Module Level 1

S Specifies ANS Segmentation Module Level 2 (S is a superset of N.)

O Specifies that obsolete elements occurring in any of the above sets are flagged.

NO
Specifies that no FIPS flagging is accomplished.

Notes:

1. The following new elements are flagged as nonconforming nonstandard IBM extensions to the COBOL 85 Standard:
   - Language syntax, such as the DATE FORMAT clause, used by the COBOL automatic date processing facilities (as enabled by the DATEPROC compiler option)
   - Language syntax for object orientation and improved interoperability
   - Use of the PGMNAME=LONGMIXED compiler option.
2. When FIPS flagging is specified, informational messages in the source program listing identifies:
   - Whether the language element is obsolete, nonconforming standard, or nonconforming nonstandard (language elements that are both obsolete and nonconforming are flagged as obsolete only)
   - The clause, statement, or header containing the nonconforming or obsolete syntax
   - The source program line and an indication of the starting column within that line
   - The level or optional module to which the language element belongs
3. FIPS flagging is suppressed when:
   - Any error diagnosed as level E or higher occurs.
   - The ABBR function of the WORD option is used, because standards conformance requires the standard set of reserved words.
   - The FLAGSTD option can conflict with other compiler options. See “Conflicting compiler options” on page 14 for conflict resolution information.
   - If the following compiler options are explicitly or implicitly specified in a program, FLAGSTD=(other than NO) issues a warning message during compile time:

ADV=NO
CMPR2=YES
DATEPROC=(other than NO)
DBCS=YES
DYNAM=NO
FASTSRT=YES
FLAGMIG=YES
LIB=NO
LITCHAR=APOST
NUM=YES
NUMPROC=PFD
SEQ=YES
TRUNC=OPT or BIN
WORD=(other than NO or RWT)
ZWB=NO

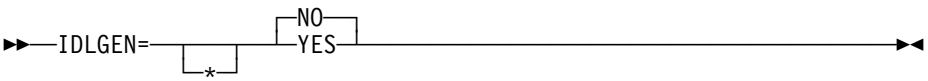
- Specifying the following options together with FLAGSTD=(other than NO), while attempting to assemble the customization macro, results in a nonzero return code.

  ADV=NO
  CMPR2=YES
  DATEPROC=(other than NO)
  DBCS=YES
  DYNAM=NO
  FLAGMIG=YES
  LIB=NO
  LITCHAR=APOST
  NUM=YES
  NUMPROC=PFD
  SEQ=YES
  TRUNC=OPT or BIN
  WORD=(other than NO or RWT)
  ZWB=NO

- FLAGSTD might produce Events records in the Associated Data file for FIPS standard conformation messages. Error messages are not guaranteed to be sequential in regard to source record numbers.

IDLGEN

Syntax



Default
IDLGEN=NO

YES
Requests that, in addition to the normal compile of the COBOL source file, SOM Interface Definition Language (IDL) definitions be generated for classes defined in the file.

The IDLGEN option has no effect on non-OO programs.

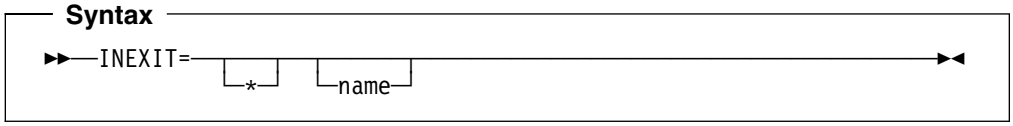
NO
Requests that IDL definitions not be generated.

Notes:

1. The Interface Definition Language (IDL) is a formal language that defines the interface for a class.
2. The Interface Repository (IR) is a database providing storage for objects that represent the major elements of interface definitions.

INEXIT

Syntax



Default
No exit specified.

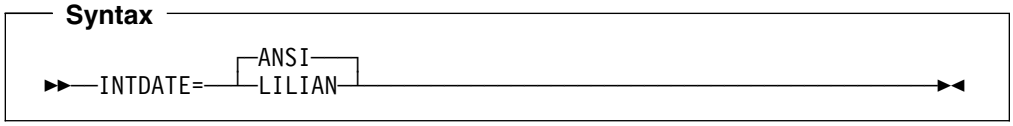
name
Identifies a module to be used with the EXIT compiler option. When the sub-option for this user exit is specified, the compiler loads that module and calls it to obtain source statements instead of reading the SYSIN data set. When the option is supplied, the SYSIN data set is not opened. For more specific information, see *COBOL for OS/390 & VM Programming Guide*.

Notes:

1. Specification of INEXIT prohibits identification of the compilation source file.
2. This option determines the compiler input for source library text being passed back to IBM Dictionary Services.

INTDATE

Syntax



Default
INTDATE=ANSI

ANSI
Uses the ANSI COBOL Standard starting date for Integer date format dates used with date intrinsic functions. Day 1 = Jan 1, 1601.

With INTDATE(ANSI), the date intrinsic functions return the same results as in COBOL/370™ Release 1.

LILIAN
Uses the Language Environment Lilian starting date for integer date format dates used with date intrinsic functions. Day 1 = Oct 15, 1582.

With INTDATE(LILIAN), the date intrinsic functions return results compatible with the Language Environment® date callable services. These results are different from those in COBOL/370 Release 1.

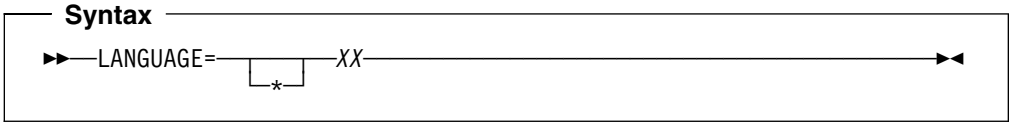
Notes:

1. When INTDATE(LILIAN) is in effect, CEECBLDY is not useable since you have no way to turn an ANSI integer into a meaningful date using either intrinsic functions or callable services. If you code a CALL literal statement with CEECBLDY as the target of the call with INTDATE(LILIAN) in effect, the compiler diagnoses this and converts the call target to CEEDAYS.
2. If you set your installation option to INTDATE(LILIAN), you should recompile all of your COBOL/370 Release 1 programs that use Intrinsic Functions to ensure

that all of your code uses the LILIAN integer date standard. This method is the safest, because you can store integer dates, pass them between programs, and even pass them from PL/I to COBOL to C programs and have no problems.

LANGUAGE

Syntax



Default

LANGUAGE=EN

XX

Specifies the language for compiler output messages. Entries for this parameter might be selected from the following list.

Table 4. Entries for the LANGUAGE compiler option

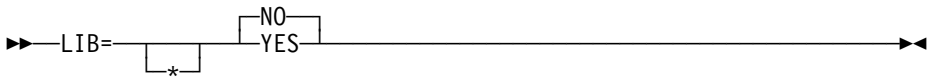
Entry	Language
EN or **ENGLISH**	Mixed case US English
JA, **JP**, or **JAPANESE**	Japanese
UE or **UENGLISH**	Uppercase US English

Notes:

1. The LANGUAGE option name must consist of at least the first two identifying characters. Other characters following the first two identifiers can be used; however, only the first two are used to determine the language name.
2. This compiler option does not affect the language in which run-time messages are displayed. For more information on run-time options and messages, refer to *Language Environment for OS/390 & VM Programming Guide*.
3. Some printers use only uppercase and might not accept output in mixed case (LANGUAGE=ENGLISH).
4. To specify the Japanese language option, the Japanese National Language Feature must be installed.
5. To specify the English language option (mixed-case English), the US English Language Feature must be installed.
6. If your installation provides a language other than those listed above, and you select it as your installation's default, you must specify at least the first two characters of the language name. These two characters must be alphanumeric.
7. The selection of Japanese, together with specification of the EVENTS option and/or the ADATA option, might result in DBCS characters being written to Error Identification records in the Associated Data file.

LIB

Syntax



Default
LIB=NO

YES
Indicates that the source program contains COPY or BASIS statements.

NO
Indicates that the source program doesn't contain COPY or BASIS statements.

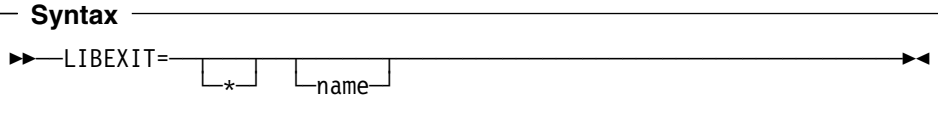
Notes:

1. LIB=YES conforms to the COBOL 85 Standard.
2. LIB=YES is used when compiling a program containing COPY or BASIS statements. Specifying LIB=YES when there are no COPY or BASIS statements in a source program results in an unnecessary invocation of the COPY processing phase, unnecessary allocation of a buffer for SYSLIB, and an unnecessary pass of the source text. However, compilation results are not affected.
3. Do not specify LIB=NO for application programs that run under CICS, if the programs will be translated.
4. LIB=YES might cause insertions or deletions of records. This might interfere with the association of Events records in the Associated Data file with the correct source records.

   Also, library messages are generated as a separately sorted group preceding any message produced by the compiler. They are not merged into the remaining messages; therefore, specification of this option might cause the Events records in the Associated Data file to be out-of-sequence with regard to source record numbers.
5. If you are using IBM Dictionary Service for data extraction services, specification of LIB involves library processing before syntax/semantic checking of Working Storage. Therefore, if a COPY statement includes part of a data item, the compiler provides the data information but does not provide indication that any part of the data item is an expansion of the COPY. LIB messages are produced for the entire SYSIN file.

LIBEXIT

Syntax



Default
No exit specified.

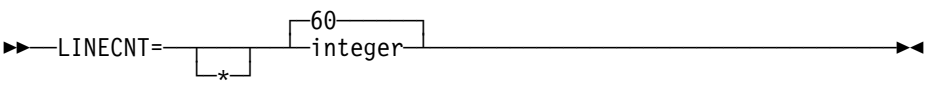
name
Identifies a module used with the EXIT compiler option. When the suboption for this user exit is specified, the compiler loads that module and calls it to

obtain copy statements instead of reading the SYSLIB or library-name data set. When the option is supplied, the SYSLIB and library-name data sets are not opened. For more specific information, see *COBOL for OS/390 & VM Programming Guide*.

This option determines the compiler input for source library text being passed back to IBM Dictionary Services.

LINECNT

Syntax



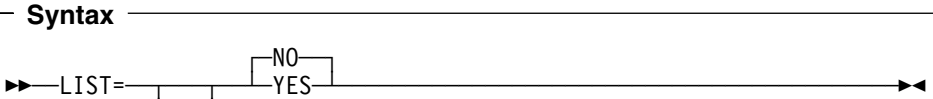
Default
LINECNT=60

integer
Specifies the number of lines to be printed on each page of the compiler source code listing. Three of the lines are used to generate headings. For example, if you specify LINECNT=60, 57 lines of source code are printed on each page of the output listing, and 3 lines are used for headings.

Note: The LINECNT installation option is equivalent to the LINECOUNT compile-time option.

LIST

Syntax



Default
LIST=NO

YES
Produces a listing that includes:

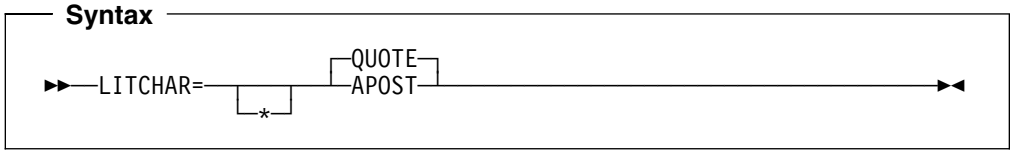
- The assembler-language expansion of source code
- Information about working storage
- Global tables
- Literal pools

NO
Suppresses this listing.

Note: The LIST and OFFSET compiler options are mutually exclusive. Setting OFFSET=YES and LIST=YES results in a nonzero return code and an error message during assembly of the customization macro.

LITCHAR

Syntax



Default
LITCHAR=QUOTE

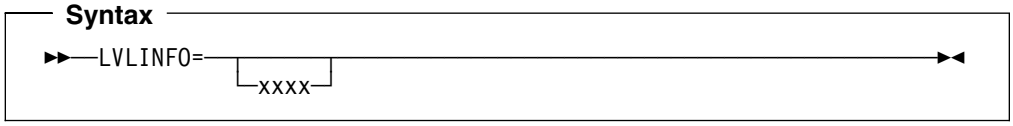
QUOTE
Use QUOTE if you want the figurative constant [ALL] QUOTE or [ALL] QUOTES to represent one or more quotation mark (") characters. QUOTE conforms to the COBOL 85 Standard.

APOST
Use APOST if you want the figurative constant [ALL] QUOTE or [ALL] QUOTES to represent one or more apostrophe (') characters.

Note: Either quotes or apostrophes can be used as literal delimiters, regardless of whether the APOST or QUOTE option is in effect. The delimiter character used as the opening delimiter for a literal must be used as the closing delimiter for that literal.

LVLINFO

Syntax

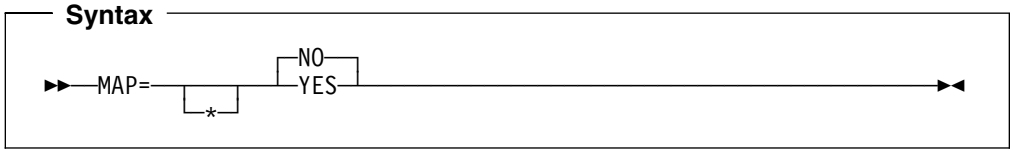


Default
No characters are specified.

xxxx
Identifies the one to four alphanumeric characters that are inserted into the listing header following the Release number (the last four bytes of the signature area). This option might be used to identify “compiler level” information within the listing header.

MAP

Syntax



Default
MAP=NO

YES
Maps items declared in the DATA DIVISION. Map output includes:

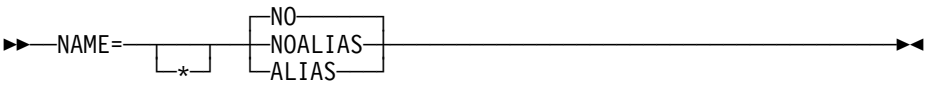
- DATA DIVISION map
- Global tables

- Literal pools
- Program statistics
- Size of the program's working storage and its location in the object code if the program is compiled without the RENT compiler option.

NO
Mapping is not performed.

NAME

Syntax



Default
NAME=NO

NOALIAS
Appends a linkage editor NAME statement (NAME modname(R)) to each object module created in a batch compilation. The module name (modname) is derived from the PROGRAM-ID statement according to the rules for forming external module names.

ALIAS
Precedes the NAME statement corresponding to the PROGRAM-ID with a linkage editor ALIAS statement for each ENTRY statement in the program.

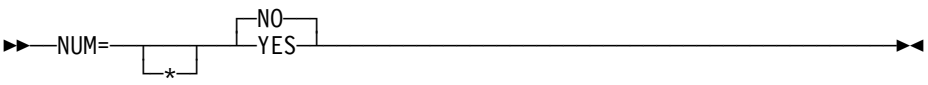
NO
Does not append linkage editor NAME statements.

Notes:

1. The NAME option allows you to create multiple modules in a program library with a single batch compilation. This can be useful for dynamic calls.
2. NAME=NOALIAS or NAME=ALIAS supports the COBOL 85 Standard.

NUM

Syntax



Default
NUM=NO

YES
Uses the line numbers from the source program rather than compiler-generated line numbers for error messages and procedure maps.

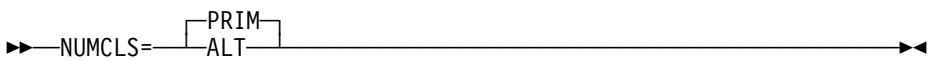
NO
Uses the compiler-generated line numbers for error messages and procedure maps.

Notes:

1. If COBOL programmers use COPY statements and NUM=YES is in effect, they must ensure that the source program line numbers and the COPY member line numbers are coordinated.
2. NUM=NO supports the COBOL 85 Standard.
3. Specification of this option might cause duplicate line numbers (with respect to the expanded source), possibly causing the Events records in the Associated Data file to be associated with the wrong source record.

NUMCLS

Syntax



Default
NUMCLS=PRIM

ALT
Specifies the sign representations that are recognized as valid by the numeric class test for data items that are defined:

- As signed (with an “S” in the PICTURE clause)
- Using DISPLAY or COMPUTATIONAL-3 (packed-decimal)
- Without the SEPARATE phrase on any SIGN clause

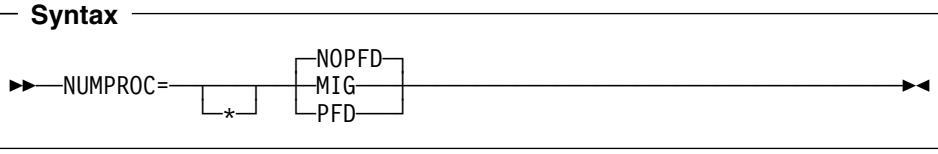
Processing with ALT accepts hexadecimal A through F as valid.

PRIM
Processing with PRIM accepts hexadecimal C, D, and F as valid.

Note: The numeric class test is affected by how both the NUMPROC and the NUMCLS options are specified. The NUMCLS option is effective only for NUMPROC=MIG or NUMPROC=NOPFD. NUMPROC=PFD specifies more strict rules for valid sign configuration.

NUMPROC

Syntax



Default
NUMPROC=NOPFD

MIG
Aids in migrating OS/VS COBOL application programs to COBOL for OS/390 & VM. Processing with MIG:

- Uses existing signs for comparison and arithmetic operations.
- Generates preferred signs for the results of MOVE and arithmetic operations. (These results meet the criteria for using NUMPROC=PFD.)
- Performs numeric rather than logical comparisons.

NOPFD
Repairs signs on input. After repair is performed, the signs meet the criteria for NUMPROC=PFD.

PFD
Optimizes the generated code, especially when OPT=STD or OPT=FULL. No explicit sign repair is performed. Note that NUMPROC=PFD has stringent criteria to produce correct results. To use NUMPROC=PFD:

- The sign position of unsigned numeric items must be X'F'.
- The sign position of signed numeric items must be either X'C' if positive or zero, or must be X'D' if negative.
- The sign position of separately signed numeric items must either be '+' if positive or zero, or '-' if otherwise.

Elementary MOVE and arithmetic statements in COBOL for OS/390 & VM always generate results with these preferred signs; however, group MOVEs and redefinitions might produce nonconforming results. The numeric class test can be used for verification. With NUMPROC=PFD, a numeric item fails the numeric class test if the signs do not meet the preferred sign criteria.

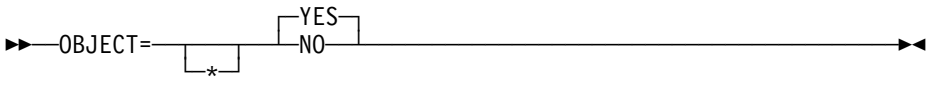
Performance Consideration: Using NUMPROC=PFD generates significantly more efficient code for numeric comparisons. For most references to COMP-3 and DISPLAY numeric data items, using NUMPROC=MIG and NUMPROC=NOPFD generates extra code because of sign “fix-up” processing. This extra code might also inhibit some other types of optimizations. Before setting this option, please consult with your application programmers to determine the effect on the application program's output.

Notes:

1. NUMPROC=NOPFD or NUMPROC=MIG supports the COBOL 85 Standard.
2. NUMPROC=NOPFD and NUMPROC=PFD are equivalent to the VS COBOL II, Release 2 options PFDSGN=NO and PFDSGN=YES, respectively.
3. Both the NUMPROC and NUMCLS options affect the numeric class test. With NUMPROC=MIG or NUMPROC=NOPFD, the results of the numeric class test are controlled by how NUMCLS is set. When NUMPROC=PFD, a data item must meet the preferred sign criteria to be considered numeric.

OBJECT

Syntax



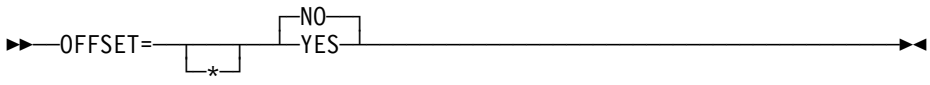
Default
OBJECT=YES

YES
Places the generated object code in a file defined by SYSLIN.

NO
Places no object code in SYSLIN.

OFFSET

Syntax



Default
OFFSET=NO

YES
Produces a condensed PROCEDURE DIVISION listing. The procedure portion of the listing will contain line numbers, verb references, and the location of the first instruction generated for each verb. In addition, the following are produced:

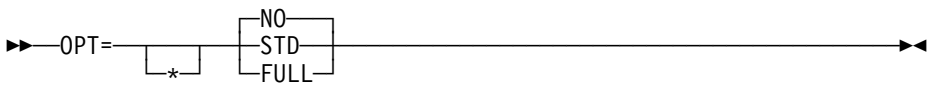
- Global tables
- Literal pools
- Program statistics
- Size of the program's working storage, and its location in the object code if the program is compiled with the NORENT compiler option

NO
Does not condense the listing or produce the items listed above.

Note: The LIST and OFFSET compiler options are mutually exclusive. Setting OFFSET=YES and LIST=YES results in a nonzero return code when attempting to assemble the customization macro. See “Conflicting compiler options” on page 14 for more information on conflict resolution.

OPTIMIZE

Syntax



Default
OPT=NO

NO
Specifies that your code is not optimized.

STD
Generates optimized object code.

Generates optimized object code. ***Performance Consideration:*** Using OPT=STD or OPT=FULL generally results in more efficient run-time code.

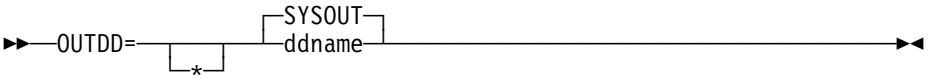
FULL
Discards any unused data items and does not generate code for any VALUE clauses for these data items. If the OPT(FULL) and MAP option are both specified, discarded data items will have a BL number of XXXX in the MAP output indicating that the number is not used.

Notes:

1. If attempting to debug optimized object code using Debug Tool, the only hook-location subparameter allowed for the TEST option is NONE. Any other combination results in a nonzero return code and an error message during installation.
2. Optimization is turned off if an S-level error or higher occurs.

OUTDD

Syntax



Default
OUTDD=SYSOUT

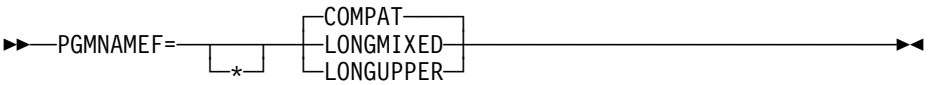
ddname
Specifies the ddname of the file used for run-time DISPLAY output.

Notes:

1. See *Language Environment for OS/390 & VM Programming Reference* description of the MSGFILE run-time option to see how OUTDD interacts with MSGFILE.
2. Change the default for this option if, at run time, you expect to have a conflict with another product that requires SYSOUT as a ddname.

PGMNAME

Syntax



Default
PGMNAME=COMPAT

LONGMIXED
Program and method names are processed as is, without truncation, translation, or folding to upper case.

LONGUPPER
Method names and nested-program names are folded to upper case by the compiler but otherwise are processed as is, without truncation or translation.

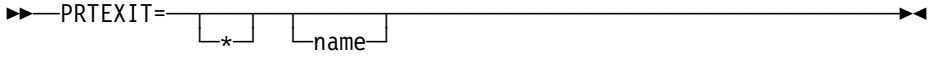
COMPAT
Program names are processed in a manner compatible with COBOL/370 Release 1.

Notes:

1. The PGMNAME option controls the handling of names used in the following contexts:
   - Program names defined in the PROGRAM-ID paragraph
   - Program entry point names on the ENTRY statement
   - Program name references in CALLs to nested programs and static CALLs to separately compiled programs
   - Program name references in the static SET procedure-pointer TO ENTRY literal statement
   - Program name references in CANCEL of a nested program
2. Class and method names are not affected by the PGMNAME option.
3. COBOL source files containing class definitions or method invocations must be complied with either the PGMNAME=LONGUPPER or PGMNAME=LONGMIXED options.
4. COBOL source files containing calls to SOM API interfaces must be compiled with PGMNAME=LONGMIXED, as the SOM API names are case sensitive and can be longer than eight characters.
5. For more specific information, see *COBOL for OS/390 & VM Programming Guide*.

PRTEXIT

Syntax

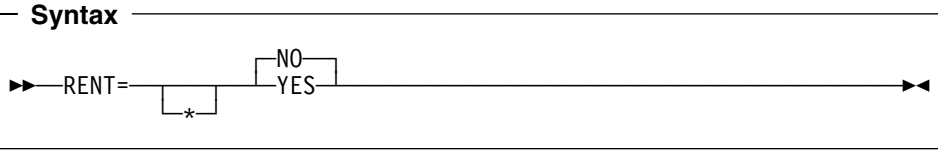


Default
No exit specified.

name
Identifies a module to be used with the EXIT compiler option. When the sub-option for this user exit is specified, the compiler loads that module and calls it instead of writing to the SYSPRINT data set. When the option is supplied, the SYSPRINT data set is not opened. For more specific information, see the *COBOL for OS/390 & VM Programming Guide*.

RENT

Syntax



Default
RENT=NO

YES
Indicates that the object code produced for a COBOL program is reentrant. Using RENT=YES enables the program to be placed in shared storage for running above the 16-megabyte line. However, this option causes the compiler to generate additional code to ensure that the application program is reentrant.

NO
Indicates that the object code produced for a COBOL program is to be nonreentrant.

Notes:

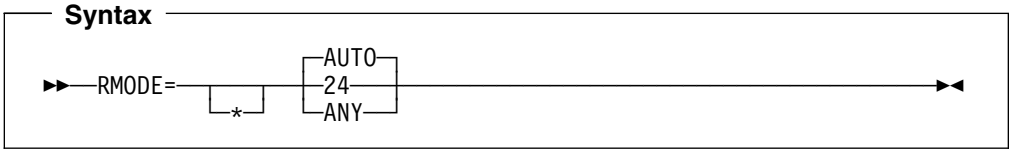
1. Programs must be compiled with RENT=YES or RMODE(ANY), if they will be run in virtual storage addresses above 16 megabytes.
2. The RENT compiler option is required for programs that are run on CICS.
3. Programs compiled with COBOL for OS/390 & VM always have AMODE(ANY). The RMODE assigned to a program depends on the RENT/NORENT and RMODE compiler options. Valid combinations include:

Table 5. Effect of RENT and RMODE on residency mode

RENT/NORENT Setting	RMODE Setting	Residency Mode Assigned
RENT	AUTO	RMODE(ANY)
RENT	ANY	RMODE(ANY)
RENT	24	RMODE(24)
NORENT	AUTO	RMODE(24)
NORENT	ANY	RMODE(ANY)
NORENT	24	RMODE(24)

RMODE

Syntax



Default
RMODE=AUTO

AUTO
Specifies that a program will have RMODE(24) if NORENT is specified, and RMODE(ANY) if RENT is specified.

24 Specifies that a program will have RMODE(24) whether NORENT or RENT is specified.

ANY
Specifies that a program will have RMODE(ANY) whether NORENT or RENT is specified.

Notes:

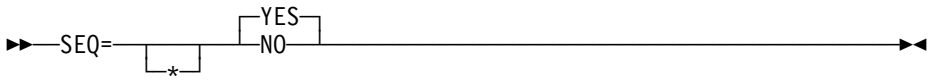
1. COBOL for OS/390 & VM NORENT programs that are required to pass data to programs running in AMODE(24) must either be compiled with the RMODE(24) option, or link-edited with RMODE(24). The data areas for NORENT programs will be above the line or below the line depending on the RMODE of the program, even if DATA(24) has been specified. DATA(24) applies to programs compiled with the RENT option only.
2. Programs compiled with COBOL for OS/390 & VM always have AMODE(ANY). The RMODE assigned to a program depends on the RMODE and RENT/NORENT compiler options. Valid combinations include:

Table 6. Effect of RMODE and RENT/NORENT on residency mode

RMODE Setting	RENT/NORENT Setting	Residency Mode Assigned
AUTO	RENT	RMODE(ANY)
AUTO	NORENT	RMODE(24)
ANY	RENT	RMODE(ANY)
ANY	NORENT	RMODE(ANY)
24	RENT	RMODE(24)
24	NORENT	RMODE(24)

SEQ

Syntax



Default
SEQ=YES

YES
Checks that the source statements are in ascending alphanumeric order by line number.

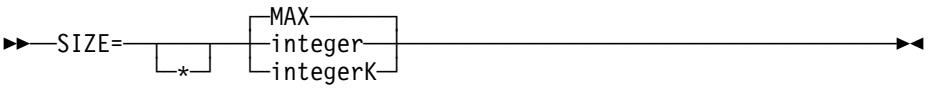
NO
Does not perform sequence checking.

Notes:

1. If both SEQ and NUM are in effect at compile time, the sequence is checked according to numeric, rather than alphanumeric, collating sequence.
2. SEQ=NO supports the COBOL 85 Standard.

SIZE

Syntax



Default
SIZE=MAX

integer
Specifies the amount of virtual storage available, in bytes. The minimum acceptable value is 778240.

integerK
Specifies the amount of virtual storage available in 1024-byte (K) increments.

The minimum acceptable value is 760K.

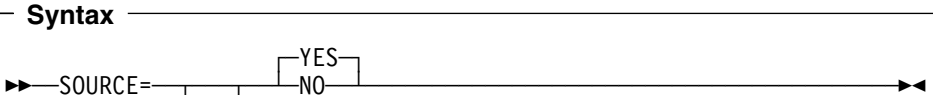
MAX
Requests all available space in the user region for use during the compilation. For extended architecture, the compiler obtains the largest contiguous block of free storage above the 16M line.

Notes:

1. Using SIZE=MAX simplifies compiler invocation by eliminating the need to determine a specific value for the SIZE option.
2. Do not use SIZE=MAX if, when you invoke the compiler, you require it to leave a specific amount of unused storage available in the user region.
3. SIZE=MAX in Extended Architecture allows the compiler to obtain all available above-the-line storage in the user region and below-the-line storage for work file buffers and compiler modules that must be below the 16M line. This occurs unless tuning is done for the extended architecture environment. Therefore, you might not want to fix this option at SIZE=MAX at installation.

SOURCE

Syntax



Default
SOURCE=YES

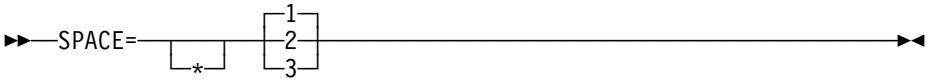
YES
Indicates that you want a listing of the source statements in the compiler-generated output. This listing also includes any statements embedded by COPY.

NO
Source statements do not appear in the output.

Note: The SOURCE compiler option must be in effect at compile time if you want embedded messages in the source listing.

SPACE

Syntax



Default
SPACE=1

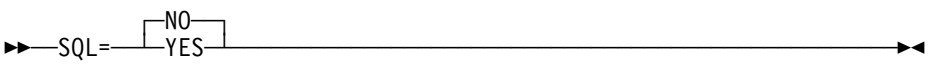
1 Provides single spacing for the source statement listing.

2 Provides double spacing for the source statement listing.

3 Provides triple spacing for the source statement listing.

SQL

Syntax



OS/390 only: This option is not applicable under VM/CMS.

Default
SQL=NO

NO
Specify to have any SQL statements found in the source program diagnosed and discarded.

Use SQL=NO if your COBOL source programs do not contain SQL statements, or if the SQL preprocessor will be used to process SQL statements prior to invocation of the COBOL compiler.

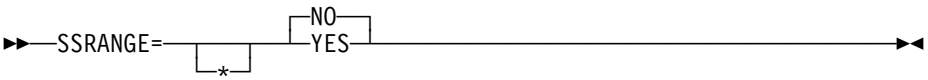
YES
Use to enable the DB2 coprocessor capability and to specify DB2 suboptions. You must specify the SQL option if your COBOL source program contains SQL statements and if it has not been processed by the DB2 precompiler.

Notes:

1. You can specify the SQL option in any of the compiler option sources: compiler invocation, PROCESS/CBL statements, or installation defaults.
2. Use either quotes or apostrophes to delimit the string of DB2 suboptions.
3. Note that DB2 suboptions cannot be specified as part of customizing the COBOL SQL option (DB2 suboptions are only supported when the SQL compiler option is specified as an invocation option or on a CBL/PROCESS card). However, default DB2 options may be specified when customizing the DB2 product installation defaults.

SSRANGE

Syntax



Default
SSRANGE=NO

YES
Generates code that checks subscripts, reference modifications, variable length group ranges, and indexes in the source program at run time to ensure that they do not refer to storage outside the area assigned. It also verifies that a table with ALL subscripting, specified as a function argument, contains at least one occurrence in the table.

The generated code also checks that a variable-length item does not exceed its defined maximum length as a result of incorrect setting of the OCCURS DEPENDING ON object.

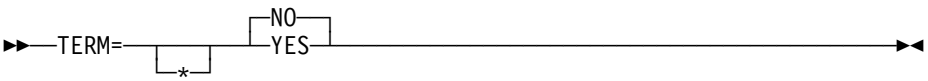
Performance Consideration: If SSRANGE=YES at compile time, object code size is increased and there will be an increase in run-time overhead to accomplish the range checking.

NO
No code is generated to perform subscript or index checking at run time.

Note: If the SSRANGE option is in effect at compile time, the range-checking code is generated. Range-checking can be inhibited at run time by specifying the Language Environment run-time option CHECK(OFF). However, the range-checking code still requires overhead and is dormant within the object code. The range-checking code can be used optionally to aid in resolving any unexpected errors without recompilation.

TERM

Syntax



Default
TERM=NO

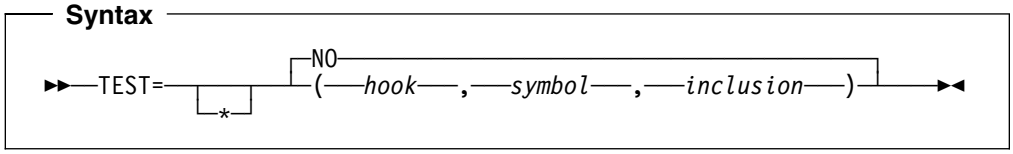
YES
Specifies that the progress and diagnostic messages are sent to the SYSTERM file, which defaults to the user's terminal unless specified otherwise.

NO
Specifies that no messages are sent to the SYSTERM file.

Note: If TERM is specified in the source program, a SYSTERM DD statement must also be specified for each application program.

TEST

Syntax



Default

TEST=NO

Other than NO

Produces object code that can be run using Debug Tool.

You must specify values for the hook, symbol and inclusion suboptions:

hook values:

ALL Activates the generation of all compiled-hooks. Hooks are generated at all statements, all path points, and at all program entry and exit points. In addition, if either the DATEPROC=FLAG option or DATEPROC=NOFLAG option is in effect, hooks are generated for all date processing statements.

NONE Suppresses the generation of all compiled-hooks. TEST(NONE) is compatible with the OPT compiler option.

STMT Hooks are compiled at every statement and label, as well as at all program entry and exit points. In addition, if either the DATEPROC=FLAG option or DATEPROC=NOFLAG option is in effect, hooks will be generated for all date processing statements.

PATH Hooks are compiled at all path points, including program entry and exit points.

BLOCK Hooks are compiled at all program entry and exit points.

symbol values:

SYM Activates generation of symbolic dictionary information tables.

NOSYM Deactivates generation of symbolic dictionary information tables.

inclusion values:

SEPARATE Generate the debugging information tables in a data set separate from your object program. You can only specify SEPARATE if you also specify SYM as the value for the symbol sub-option.

NOSEPARATE Include the debugging information tables in your object program.

Performance Consideration: Because TEST=(a hook-location suboption other than NONE) generates additional code, it can cause significant performance degradation at run time when used in a production environment.

NO
Produces object code that cannot be run using Debug Tool.

Notes:

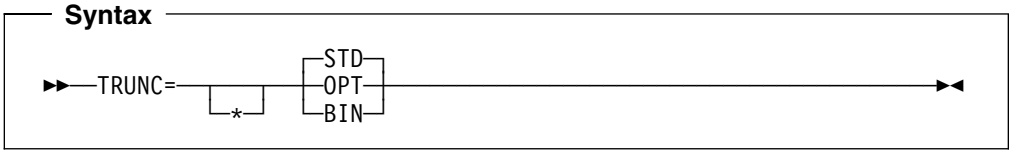
1. If you specify TEST= (other than NONE for the hook-location suboption), the following options are put into effect at compilation time:

   OBJ=YES
   OPT=NO

2. Only TEST(NONE) is compatible with the OPT compiler option. See “Conflicting compiler options” on page 14 for more conflict resolution information.
3. Programmers might notice an increase in run time for programs compiled with any hook-location suboption other than NONE.
4. A date processing statement is any statement that references a date field, or any EVALUATE or SEARCH statement WHEN phrase that references a date field.
5. To get the full capability of Debug Tool, compile your program with TEST(ALL,SYM).
6. For production programs, compile your programs with TEST(NONE,SYM,SEPARATE) if you do not want to increase the size of your production modules, but still get minimum debugging capability.

TRUNC

Syntax



Default
TRUNC=STD

STD
Conforms to the COBOL 85 Standard.

Controls the way arithmetic fields are truncated during MOVE and arithmetic operations. The TRUNC option applies only to binary (COMP) receiving fields in MOVE statements and in arithmetic expressions. When TRUNC=STD is in effect, the final intermediate result of an arithmetic expression, or of the sending field in the MOVE statement, truncates to the number of digits in the PICTURE clause of the binary receiving field.

OPT
The compiler assumes that the data conforms to PICTURE and USAGE specifications. The compiler manipulates the result based on the size of the field in storage (halfword or fullword).

TRUNC=OPT is recommended, but it should be specified only when data being moved into binary areas does not have a value with larger precision than that defined by the binary item PICTURE clause. Otherwise, truncation of high-order digits might occur. The truncation results are dependent on the particular code sequence generated and might not be the same in OS/VS COBOL and COBOL for OS/390 & VM.

BIN

Should not be used as an installation default. Specifies that:

1. Output binary fields are truncated only at the S/370™ half/full/doubleword boundaries, rather than at COBOL base 10 picture limits.
2. Input binary fields are treated as S/370 half/full/double words, and no assumption is made that the values are limited to those implied by the base 10 PICTURE clause.
3. DISPLAY converts and outputs the full content of binary fields with no truncation to the PICTURE description.

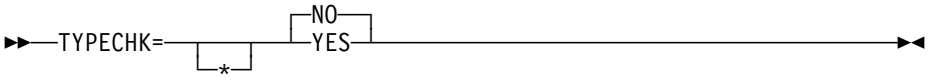
Performance Consideration: Using TRUNC=OPT does not generate extra code and generally improves performance. However, both TRUNC=BIN and TRUNC=STD generate extra code whenever a BINARY data item is changed. TRUNC=BIN is usually the slowest of these options.

Notes:

1. TRUNC=STD and TRUNC=OPT are equivalent to TRUNC=YES and TRUNC=NO (respectively) in VS COBOL II Release 2.
2. Setting this option affects program run-time logic; that is, the same COBOL source program can give different results, depending on the option setting. Verify whether the COBOL for OS/390 & VM source programs assume a particular setting for correct running.
3. TRUNC=BIN is the recommended option when interfacing with other products that have S/370-format binary data (such as CICS, DB2®, FORTRAN, and PL/I). This is especially true if there is a possibility of having more than 9 digits in a fullword or more than 4 digits in a halfword.

TYPECHK

Syntax



Default

TYPECHK=NO

YES

Enforces the rules for object-oriented type checking, and generates diagnostics for any violations.

NO

Does not enforce rules for object-oriented type checking or generate diagnostics for violations.

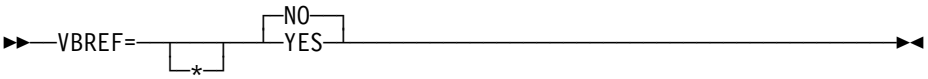
Notes:

1. When TYPECHK is specified, there must be entries in the SOM Interface Repository (IR) for each class that is referenced in the COBOL source being compiled.
2. For details on type conformance requirements, see *COBOL Language Reference*. For details on creating IR entries, SOM, and type checking, see *COBOL for OS/390 & VM Programming Guide*.

Do not use SIZE(MAX) when compiling large object-oriented applications with TYPECHK. Storage space must be left in the user region for Interface Repository data.

VBREF

Syntax



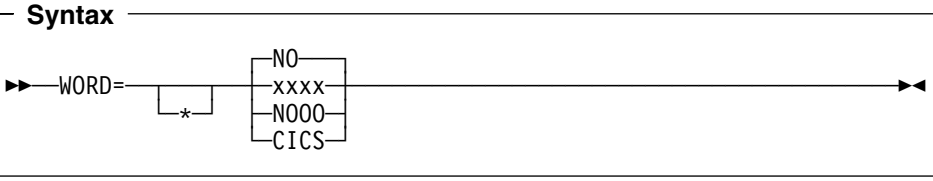
Default
VBREF=NO

YES
Produces a cross reference of all verb types in a source program to the line numbers where they are found. VBREF=YES also produces a summary of how many times each verb was used in the program.

NO
Does not produce a cross-reference or verb-summary listing.

WORD

Syntax



Default
WORD=*NO

NO
Indicates that no alternative reserved word table is to be used as the default.

xxxx
Specifies an alternative default reserved word table to be used during compilation. **xxxx** represents the ending characters (can be 1 to 4 characters in length) of the name of the reserved word table used. The first 4 characters are IGYC. The last 4 characters cannot be any one of the character strings listed below, nor can any of them contain the dollar sign character ($).

ASM1	LIBO	LVL8	RDSC
ASM2	LIBR	OPTM	RWT
DIAG	LSTR	OSCN	SAW
DMAP	LVL0	PGEN	SCAN
DOPT	LVL1	RCTL	SIMD
FGEN	LVL2	RDPR	XREF
INIT	LVL3		

NOOO
A non object-oriented-specific word table, IGYCNOOO, is provided as an alternative reserved word table. For a description, see “NOOO reserved word table (IGYCNOOO)” on page 12.

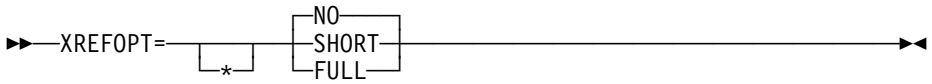
CICS
a CICS-specific word table, IGYCCICS, is provided as an alternate reserved word table. For a description, see “CICS reserved word table (IGYCCICS)” on page 12.

Notes:

1. The default for the WORD option is specified with an asterisk. When the option is installed with the default (WORD=*NO), the application programmer cannot override the option at compile time to specify an alternative reserved word table.
2. WORD=NO supports the COBOL 85 Standard.
3. Specification of WORD affects the interpretation of input reserved words. System names (such as UPSI and SYSPUNCH) and the 42 intrinsic function names should not be used as aliases for reserved words. If a function name is specified as an alias via the reserved word table ABBR control-statement, that function name will be recognized and diagnosed by the compiler as a reserved word and the intrinsic function will not be performed.
4. Changing the default value of the WORD option could cause compiler option conflicts. See “Conflicting compiler options” on page 14 for information on conflict resolution.
5. Specification of WORD affects the interpretation of input reserved words passed back to IBM Dictionary Services.

XREFOPT

Syntax



Default
XREFOPT=NO

SHORT
Produces only the explicitly referenced variables in the cross-reference listing.

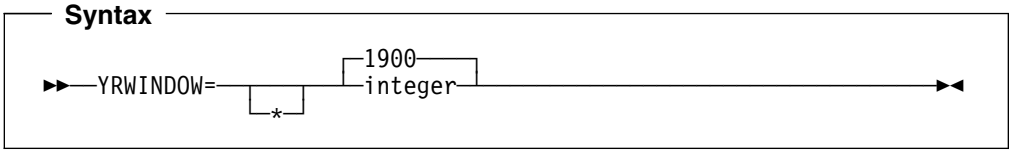
FULL
Produces both a sorted and embedded cross-reference listing. If SOURCE=YES is also specified, the listing line numbers cross-reference recurrences of particular data-names.

NO
Suppresses the cross-reference listing.

Note: The XREFOPT option sets the default value for the compiler option XREF.

YRWINDOW

Syntax



Default
YRWINDOW=1900

integer
Specifies the first year of the 100-year window used by COBOL windowed date fields, and may be one of the following:

- An unsigned decimal integer from 1900 through 1999.
- A negative decimal integer from -1 through -99, representing an offset from the current year at run time. The current year is determined for each compilation unit when it is first initialized, or reinitialized after execution of a CANCEL statement that refers to the compilation unit.

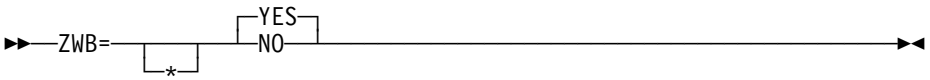
Notes:

1. YRWINDOW has no effect unless DATEPROC=FLAG or DATEPROC=NOFLAG is specified.
2. At run time, two conditions must be true:
   a. The 100-year window must have its beginning year in the 1900s.
   b. The current year must lie within the 100-year window for the compilation unit.
3. All windowed dates have a year relative to the base year. For example, if the base year were specified as 1965, all windowed year values would be interpreted as years within the 100-year window of 1965 to 2064, inclusive. So, a windowed year value of 67 would represent the year 1967, whereas a windowed year value of 05 would represent the year 2005.

   If the base year were specified as -30, then the window would depend on when the application were run. Running it during 2000 would give a 100-year window of 1970 through 2069. So, a windowed year value of 70 would represent the year 1970, whereas a windowed year value of 69 would represent the year 2069.
4. The YRWINDOW installation option is equivalent to the YEARWINDOW compile-time option.

ZWB

Syntax



Default

ZWB=YES

YES

Removes the sign from a signed external decimal (DISPLAY) field when comparing this field to an alphanumeric field during run time.

NO

Does not remove the sign from a signed external decimal (DISPLAY) field when comparing this field to an alphanumeric field during run time.

Notes:

1. Setting this option affects program run-time logic; that is, the same COBOL source program can give different results, depending on the option setting. Verify whether your COBOL for OS/390 & VM source programs assume a particular setting to run correctly.
2. ZWB=YES supports the COBOL 85 Standard.
3. Application programmers use ZWB=NO to test input numeric fields for SPACES.

Chapter 2. Customizing COBOL for OS/390 & VM

You can make modifications to COBOL for OS/390 & VM only after installation of the product is complete. One of the modifications is made using an SMP/E USERMOD. If you do not ACCEPT COBOL for OS/390 & VM into the distribution libraries before applying the USERMOD, you will not be able to use the SMP/E RESTORE statement to remove your USERMOD. Do not accept your USERMOD into the distribution libraries. You might want to remove your USERMOD if you find it does not suit the needs of the programmers at your site. You would have to remove your USERMOD before applying service to the modules it changes. In this case, you would probably want to reapply your USERMOD after successful installation of the service.

Important

Make sure that COBOL for OS/390 & VM serves the needs of the application programmers at your site. Please confer with them while you plan the customization of COBOL for OS/390 & VM. Doing so will ensure that the modifications you make at install time best support the application programs being developed at your site.

Note: All information for installing COBOL for OS/390 & VM is included in the Program Directory provided with the product.

Summary of user modifications

Installation of COBOL for OS/390 & VM places a number of sample modification jobs in the target data set IGY.V2R2M0.SIGYSAMP.Table 7 shows the names of the sample modification jobs, which are described in detail in the following sections.

The sample modification jobs that IBM provides are not customized for your particular system. You must customize them.

Copy members from IGY.V2R2M0.INSTALL into one of your personal data sets before you modify and submit them so that you have an unmodified backup copy if you make changes that you want to abandon.

Descriptions of possible modifications appear in the comments in the JCL. You can use TSO to modify and submit the job. Save the modified JCL for future reference.

Table 7. Summary of user modification jobs for COBOL for OS/390 & VM

Description	Customization Job	Page
Change Compiler Options Default Module	IGYWDOPT	56
Create an Options Module to Override Compiler Options Specified as Fixed	IGYWUOPT	57
Create Additional Reserved Word Table	IGYWRWD	58
Place COBOL for OS/390 & VM Modules in Shared Storage	IGYWMLP2	63

Change compiler options defaults

To change the compiler options defaults or create a compiler options module to override fixed options, copy the source of options module IGYCDOPT from IGY.V2R2M0.SIGYSAMP into the appropriate job in place of the two-line comment. Then change the parameters on the IGYCOPT macro call to match the compiler options you have selected for your installation. Observe the following requirements in coding your changed IGYCOPT macro call.

Notes on IGYCOPT Invocation

1. A continuation character (X in the source) must be present in column 72 on each line of the IGYCOPT invocation except the last line. The continuation line must start in column 16. You can break the coding after any comma.
2. Do not put a comma in front of the first option in your macro.
3. Options and suboptions must be specified in uppercase. Only suboptions that are strings can be specified in mixed-case or lowercase. For example both LVLINFO=(Fix1) and LVLINFO=(FIX1) are acceptable.
4. If one of the string suboptions contains a special character, for example, an embedded blank or unmatched right or left parenthesis, the string must be enclosed in apostrophes ('), not in quotes ("). (A null string can be specified with either contiguous apostrophes or quotes.

   To obtain an apostrophe (') or a single ampersand (&) within a string, two contiguous instances of the character must be specified. The pair is counted as only one character in determining whether the maximum allowable string length has been exceeded and in setting the effective length of the string.
5. Avoid unmatched apostrophes in any string that uses apostrophes. The error cannot be captured within IGYCOPT itself. Instead, the assembler produces a message such as:

   ```
   IEV03  *** ERROR ***  NO ENDING APOSTROPHE
   ```

   This message bears no spatial relationship to the offending suboption. Furthermore, none of the options are properly parsed if this error is committed.
6. Code only the options whose default value you want to change. The IGYCOPT macro supplies the IBM supplied defaults for any option you do not code. See “IGYCDOPT worksheet for compiler options” on page 3 for a worksheet to help you plan your default compiler options. See “COBOL for OS/390 & VM compiler options” on page 13 for descriptions of the options.
7. Place an END statement after the macro instruction.

Changing compiler options default module

Use the sample job IGYWDOPT to change the defaults for the COBOL for OS/390 & VM compiler options. Use the information in “COBOL for OS/390 & VM compiler options” on page 13 to select your default values.

If you coded OUT as the value for any compiler phase options, be sure to place those phases in shared storage before compiling a program using your new compiler options default module. See “Compiler phases and their defaults” on page 7 and “Placing COBOL for OS/390 & VM modules in shared storage” on page 63 for more information.

Modifying the JCL for IGYWDOPT

Add a job card appropriate for your site.

Add a JES ROUTE card if required for your site.

Replace the two comment lines in IGYWDOPT with a copy of the source for IGYCDOPT found in IGY.V2R2M0.SIGYSAMP.

Code parameters on the IGYCOPT macro statement in IGYCDOPT to reflect the values you have chosen for your installation-wide default compiler options.

Change #GLOBALCSI to the global CSI name.

Change #TZONE in the SET BDY statement to the target zone name.

After you modify the IGYWDOPT job, submit it. You will get a condition code of 0 if the job runs correctly. Also check the IGYnnnn informational messages in your listing to verify the defaults that will be in effect for your installation.

Creating an options module to override options specified as fixed

If you have specified some options as fixed in your compiler default options module, you might occasionally find an application that needs to override a fixed option. You can do this by creating a temporary copy of the options module in a separate data set that can be accessed as a STEPLIB or JOBLIB (ahead of the IGY.V2R2M0.SIGYCOMP data set) when the application is compiled. Sample job IGYWUOPT creates a default options module that is link-edited into a user-specified data set. The assembly and link-editing take place outside SMP/E control, so the standard default options module is not disturbed.

Modify the JCL for IGYWUOPT

Add a job card appropriate for your site.

Add a JES ROUTE card if required for your site.

Replace the two comment lines in IGYWUOPT with a copy of the source for IGYCDOPT found in IGY.V2R2M0.SIGYSAMP.

Change the parameters on the IGYCOPT macro statement in IGYWUOPT to reflect the values you have chosen for this fixed option override compiler options module.

If you chose to use a different prefix than the IBM supplied one for the COBOL for OS/390 & VM target data sets, check the SYSLIB DD statement (marked with '<<<<<') to ensure that the data set names are correct.

Change DSNAME=YOURLIB in the SYSLMOD DD statement to the name of the partitioned data set you want your IGYCDOPT module linked into. Note that an IGYCDOPT module currently in the chosen data set will be replaced by the new version.

After you modify the IGYWUOPT job, submit it. Both steps should return a condition code of 0 if the job runs successfully. Also check the IGYnnnn informational messages in your listing to verify the defaults that are in effect when this module is used in place of the standard default options module.

Creating or modifying additional reserved word tables

The reserved words used by the COBOL for OS/390 & VM compiler are maintained in a table (IGYCRWT) provided with the product. A non object-oriented-specific word table (IGYCNOOO) is provided as an alternative reserved word table. See "NOOO reserved word table (IGYCNOOO)" on page 12. A CICS-specific reserved word table (IGYCCICS) is provided as an alternate reserved word table. See "CICS reserved word table (IGYCCICS)" on page 12. You can change the reserved words using the reserved word table utility (IGY8RWTU), thereby creating additional reserved word tables. You can also modify tables that you previously created.

The reserved word table utility accepts control statements you can use to create or modify a reserved word table. The new table then contains the reserved words from the IBM-supplied table with all the changes you have applied.

You can make the following types of changes to reserved word tables:

- Add an alternative form of an existing reserved word.
- Add words to be flagged with an informational message whenever they are used in a program.
- Add words to be flagged with an error message whenever they are used in a program.
- Indicate that words currently flagged with an informational or error message should no longer be flagged.

Each reserved word table you create must have a unique 1- to 4-character identifier. For a list of 1- to 4-character strings that cannot be used, see the WORD compiler option on page 51.

At compile time, the value of the compiler option **WORD(xxxx)** identifies the reserved word table to be used. **xxxx** is the unique 1- to 4-character identification you specified in the member name IGYCxxxx. You can create multiple reserved word tables, but only one can be specified at compile time.

Note: The total number of entries in a reserved word table should not exceed 1536 or 1.5K.

Procedure for creating or modifying a reserved word table

To create or modify a reserved word table, you must edit a copy of the appropriate source file:

- Member IGY8RWRD in IGY.V2R2M0.SIGYSAMP (IBM supplied default reserved word table)
- Member IGY8CICS in IGY.V2R2M0.SIGYSAMP (IBM supplied CICS reserved word table)
- A user file (user-supplied reserved word table).

You must also modify and invoke the appropriate non-SMP/E JCL.

Your file should have four parts: Parts I, II, III, and IV. Modify the file and non-SMP/E JCL as follows:

1. Make a private copy of the file.
2. Do not edit Part I (all lines up to and including the line with the keyword MOD).
3. Edit Part II by placing asterisks in column 1 of the lines that contain CODASYL reserved words for which you do not want informational messages issued.
4. Edit Part III by placing asterisks in column 1 of the lines that contain obsolete reserved words for which you do not want severe messages issued.
5. Edit Part IV by coding additional reserved word control statements that create the modifications you want, as described under "Coding control statements."
6. Modify and run the JCL, as discussed under "Modifying and running JCL to create a new reserved word table" on page 62. You also must create a unique 1- to 4-character identification for the new reserved word table and supply it in the JCL.

Coding control statements

To create a reserved word table, you must understand the syntax rules for the control statements and for the operands within control statements.

Figure 2 illustrates the format for coding reserved word processor control statements.

```
ABBR   reserved-word: user-word [comments]
      [reserved-word: user-word [comments]]

⋮
INFO   COBOL-word [(0|1)] [ comments ]
      [COBOL-word [(0|1)] [comments]]

⋮
RSTR   COBOL-word  [(0|1)] [ comments ]
      [COBOL-word [(0|1)] [comments]]

⋮
```

Figure 2. Syntax format for reserved word processor control Statements

As shown in Figure 2, keywords you can use are:

ABBR Specifies an alternative form of an existing reserved word

INFO Specifies words that are to be flagged with an informational message whenever they are used in a program

RSTR Specifies words that are to be flagged with an error message whenever they are used in a program

Note: All words you identify with the control statement keywords INFO and RSTR are flagged with a message in the source listing of the COBOL for OS/390 & VM program that uses them. Words that are abbreviated are not flagged in the source

listing unless you have also specified them on the INFO or RSTR control statements.

Rules for coding control statements

When you code your control statements, do the following:

1. Begin the control statement in column 1.
2. Place one or more spaces between the keyword and the first operand.
3. When specifying a second operand, include a colon (:) and one or more spaces after the first operand.
4. Continue a control statement by putting blanks in columns 1 through 5, followed by the operand or operands, to make additional specifications.
5. Specify comments by putting an asterisk (*) in column 1 of the control statements. You can also place comments on the same line as the control statement. In that case, however, there must be at least one space following the operand(s) before a comment begins.
6. To specify more than one change within a single control statement, put each additional specification on a separate line.
7. Do not add any blank lines.

Coding operands in control statements

The following list shows the types of operands you will be coding in the control statements:

reserved-word An existing reserved word.

user-word A user-defined COBOL word that is not a reserved word.

comments Any comments you want to put on the same line with the control statement, or on a separate line that has an asterisk in column 1.

COBOL-word A word of up to 30 characters that can be a system name, a reserved word, or a user-defined word.

Rules for coding control statement operands

When you are coding the control statement operands, follow these rules:

1. A user-word can be used in only one ABBR statement in any particular reserved word table.
2. A reserved-word specified in an ABBR statement can also be specified in either a RSTR or an INFO statement.
3. A particular reserved-word can be specified only once in an ABBR statement.
4. A particular COBOL-word can be specified only once in either a RSTR or an INFO statement.

The remaining sections provide examples for coding each type of control statement.

ABBR statement

ABBR reserved-word: user-word [comments]

Defines an alternative symbol for the reserved word specified. The symbol can be used when coding a program.

Notes:

1. User-word becomes a reserved word and can be used in place of the reserved word specified in this statement.
2. Reserved-word remains a reserved word with its original definition.
3. The source listing shows the original source—the symbol as you coded it.
4. The reserved word is used in compiler output—other listings, some messages, and so forth.

Example of an ABBR statement

```
ABBR  REDEFINES:  REDEF
      SEPARATE:  SEP
```

REDEF and SEP become abbreviations that can be used in source programs. The appropriate reaction to the use of REDEFINES and SEPARATE takes place when the source program is compiled.

INFO statement

INFO COBOL-word [(0|1)] [comments]

This statement specifies the COBOL words that are to be flagged by the compiler.

This statement can also be used to control the use of nested programs. By selecting either 1 or 0, you can indicate whether a specific COBOL-word can be used only once, or not at all.

0 Indicates that whenever the specified COBOL-word is used, the 0086 informational message is issued.

1 Indicates that the specified COBOL-word can be used once. If it is used more than once, informational message 0195 is issued.

The messages are handled as information (I) messages. The compilation condition is not changed.

Example of an INFO statement

```
INFO  ENTRY
```

When the IBM extension reserved word ENTRY is used in a program, it is flagged with message 0086.

RSTR statement

RSTR COBOL-word [(0|1)] [comments]

This statement specifies COBOL words that cannot be used in a program.

This statement can also be used to control the use of nested programs. By selecting either 1 or 0, you can indicate whether a specific COBOL-word can be used only once, or not at all.

0 Indicates that whenever the specified COBOL-word is used, message 0084 is issued.

1 Indicates that the specified COBOL-word can be used once. If it is used more than once, severe message 0194 is issued.

Note: The following reserved words can be restricted with the 1 option only:

IDENTIFICATION
FD
ENVIRONMENT
DATA
WORKING-STORAGE
PROCEDURE
DIVISION
SECTION
PROGRAM-ID

Examples of RSTR statements

Example 1

```
RSTR  BOOLEAN
      XD
      PARENT
```

The use of BOOLEAN, XD, and PARENT causes error messages.

Example 2

```
RSTR  GO
      ALTER
```

The use of GO TO and ALTER causes error messages.

Example 3

```
RSTR IDENTIFICATION(1)  only allow 1 program per compilation unit
RSTR ID(1)              same for the short form
RSTR PROGRAM-ID(1)      only allow 1 program per compilation unit
RSTR GLOBAL             do not allow this phrase at all
```

The reserved word table generated allows usage of all COBOL 85 Standard language, except nested programs.

Modifying and running JCL to create a new reserved word table

The JCL you use to create a new reserved word table contains STEP1, STEP2, and STEP3, that do the following, respectively:

Run the reserved word table utility with your modified table
Assemble your modified reserved word table
Produce a run-time load module from the object module.

After you run the job, a new reserved word table is created, the library you specified contains the new table, and the table has IGYC plus the 1- to 4-character identification you specified.

Procedure for modifying and running non-SMP/E JCL

Use sample job IGYWRWD in IGY.V2R2M0.SIGYSAMP to create your new reserved word table. The sample job is distributed with the name IGY8TBL1 as the input to the reserve word utility, and creates load module IGYCTBL1 in IGY.V2R2M0.SIGYCOMP if you supplied your input as member IGY8TBL1 in IGY.V2R2M0.SIGYSAMP. Before you run the job, do the following:

Modify the JCL for IGYWRWD

- Modify the job statement for your site.
- Add JES ROUTE records if desired.
- Change the data set name on the STEPLIB DD statement in STEP1 to match the compiler target data set name you used during installation.
- Either
  - Change the data set name in //RSWDREAD DD DSN=... to the data set name and member name of your modified reserved word table, or
  - Replace the RSWDREAD DD with //RSWDREAD DD * and insert your modified reserved table immediately following that line.

  Note: For specific instructions, see comments in job IGYWRWD.
- Change the name of the data set on the SYSLMOD DD statement in STEP3 to match the name of the data set to which you are adding your modified reserved word table. (The data set name on the SYSLMOD DD statement should be the name of the compiler target data set.) Also, you must specify the name of your modified reserved word table in the parentheses that follow the data set name on the SYSLMOD DD statement.

If, after you run IGYWRWD, you receive a nonzero return code from the table utility, use the error messages in the output data set specified on the RSWDPRNT DD statement to correct any mistakes and rerun the job.

Placing COBOL for OS/390 & VM modules in shared storage

All of the modules in IGY.V2R2M0.SIGYCOMP that are reentrant can be included in shared storage on OS/390. This is done by:

1. Authorizing the data set IGY.V2R2M0.SIGYCOMP
2. Including IGY.V2R2M0.SIGYCOMP in the LNKLSTnn concatenation (optional)
3. Creating an IEALPAnn member in SYS1.PARMLIB that lists the modules to be made resident in the MLPA when the system is IPLed.

IGYWMLP2 is installed in IGY.V2R2M0.SIGYSAMP for you to use as an example in creating your IEALPAnn member.

Under OS/390, you do not need to place IGY.V2R2M0 in the LNKLSTnn concatenation to be able to load modules into the LPA. If you choose not to add it to the LNKLSTnn concatenation, you must make the modules that are not included in the LPA available to steps that compile COBOL for OS/390 & VM applications by either:

- Copying the non-LPA modules to a data set that is in the LNKLSTnn concatenation, or
- Copying the non-LPA modules to a data set that can be used as a STEPLIB or JOBLIB.

Notes:

1. Using the entire IGY.V2R2M0.SIGYCOMP data set as a STEPLIB or JOBLIB defeats the purpose of placing the modules in the LPA because modules are loaded from a STEPLIB or JOBLIB before the LPA is searched.
2. Modules you copy into another data set are not serviced automatically by SMP/E in that data set. You must rerun your copy job after you apply service to COBOL for OS/390 & VM to make the updated modules available in the LNKLSTnn data set or in the STEPLIB.

Refer to the following publications for more information on including modules in the LPA:

- *OS/390 MVS Initialization and Tuning Guide*, SC28-1751
- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752

If you are placing compiler phases in shared storage, code the corresponding phase options with the value OUT when you run the sample job IGYWDOPT to change the compiler options defaults. See “Change compiler options defaults” on page 56 for more information.

Chapter 3. Customizing Debug Tool

This chapter describes how you can customize Debug Tool. It contains information on:

- Placing Debug Tool modules in shared storage
- Setting up CICS with Debug Tool

You can customize Debug Tool only after installation is complete.

Placing Debug Tool modules in shared storage

All of the modules in EQAW.V1R2M0.SEQAMOD that are reentrant (except EQAWEC62, EQAWEM62, EQAWVDBG, and EQAWECAL, which are only used by SMP/E for service) can be included in shared storage on OS/390 by:

1. Authorizing the data set EQAW.V1R2M0.SEQAMOD
2. Including EQAW.V1R2M0.SEQAMOD in the LNKLSTnn concatenation (optional)
3. Creating an IEALPAnn member in SYS1.PARMLIB that lists the modules to be made resident in the Modified Link Pack Area (MLPA) when the system is IPLed

The EQAWMLP2 member in EQAW.V1R2M0.SEQASAMP can be used as an example of how to create your own IEALPAnn member.

Under OS/390, you do not need to place EQAW.V1R2M0.SEQAMOD in the LNKLSTnn concatenation to be able to load modules into the LPA. Instead, you can make those modules not included in the LPA available to steps that compile Debug Tool applications by either:

- Copying the non-LPA modules to a data set that is in the LNKLSTnn concatenation or
- Copying the non-LPA modules to a data set that can be used as a STEPLIB or JOBLIB.

Notes:

1. Using the entire EQAW.V1R2M0.SEQAMOD data set as a STEPLIB or JOBLIB would defeat the purpose of placing the modules in the LPA because modules are loaded from a STEPLIB or JOBLIB before the LPA is searched.
2. Modules you copy into another data set are not serviced automatically by SMP/E in that data set. You must rerun your copy job after you apply service to Debug Tool in order to make the updated modules available in the LNKLSTnn data set or in the STEPLIB.

If you do not want to place all eligible modules in the LPA, use the following information to help you decide which modules to remove from the LPA lists. You should remove the module names for a national language you did not install. You can also remove the module names for a programming language you use infrequently or not at all.

Module Name	Function Support
EQA10xx0	U.S. English Mixed Case
EQA10xx2	Japanese
EQA13xxx	C/370™
EQA14xxx	COBOL for OS/390 & VM
EQAx2xxx	PL/I for OS/390 & VM

If you do not want to put any of the modules in the LPA, you can either:

- Add **SEQAMOD** to LNKLST00 or
- Make **SEQAMOD** available via TSO LOGON PROC.

Refer to the following publications for more information on including modules in the LPA:

- *OS/390 MVS Initialization and Tuning Guide*, SC28-1751
- *OS/390 MVS Initialization and Tuning Reference*, SC28-1752

Setting up CICS with Debug Tool

Both CICS/ESA® (Version 4) and IBM OS/390 Language Environment must already be installed. After doing the SMP/E APPLY of Debug Tool, you must update the CICS CSD and DCT with definitions for Debug Tool as follows:

1. **Update the CICS CSD**
   - Merge the supplied Debug Tool definitions into the existing job you have for maintaining the CICS CSD. The statements required to define Debug Tool are in the member, **EQACCSD**, which is contained in the sample library, **EQAW.V1R2M0.SEQASAMP**.
   - The CSD group list used during CICS startup must include the CSD group associated with the Debug Tool library routines (the group name for the Debug Tool routines is EQA in the sample **EQACCSD**).
2. **Update the CICS DCT**
   - Merge the required Debug Tool definitions into the existing job you have for maintaining the CICS DCT. The statements required to define Debug Tool are in the member, **EQACDCT**, which is contained in the sample library, **EQAW.V1R2M0.SEQASAMP**.
   - **EQACDCT** contains definitions to create DCT entries for the transient data queues used by Debug Tool. They are defined as extra partition data queues.

     Important

     Do not add DD statement to the CICS JCL startup deck for these queues.

   - *CICS/ESA Resource Definition (Macro)* manual contains further information on the DFHDCT macro, as well as on defining the queues and associated buffers.

   Note: When DFHDCT encounters the entry names CINSPIN, CINSPLS, CINSPOT, CINL, and CINO, it generates a warning message stating that queue names beginning with the letter 'C' are reserved for CICS. These messages do not indicate errors in this case.

3. **Update the CICS Startup Job**
   - All of the Debug Tool modules defined in **EQACCSD**, as well as modules for language libraries (like IBM OS/390 Language Environment) and any user modules that might be called during the debug session, must be in libraries that are accessible to CICS. The libraries can be made accessible by including them in the concatenation for the DFHRPL DD in the CICS startup JCL. *CICS/ESA Operations Guide* provides an example of a CICS startup job.
   - Place the **SEQAMOD** library in the DFHRPL concatenation.
4. **Update the CICS Authorized Library**
   - The Debug Tool module **EQA00DYN** must reside in one of the authorized libraries accessible to CICS. This could be via authorized libraries in the STEPLIB concatenation, via a JOBLIB, or via an OS/390 LINKLST data set.
   - Depending on the alternative you choose, it might be necessary to copy **EQA00DYN** from **SEQAMOD** to one of the authorized libraries.

Appendix. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming interface information

IBM COBOL for OS/390 & VM provides no macros that allow a customer installation to write programs that use the services of IBM COBOL for OS/390 & VM.

Attention: Do not use as programming interfaces any IBM COBOL for OS/390 & VM macros.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States and/or other countries or both:

C/370	DB2	OS/390
CICS	DFSORT	S/370
CICS/ESA	IBM	SOMobjects
COBOL/370	Language Environment	

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

IBM COBOL for OS/390 & VM

Compiler and Run-Time Migration Guide, GC26-4764

Customization under OS/390, GC26-9045

Debug Tool User's Guide and Reference, SC09-2137

Diagnosis Guide, GC26-9047

Fact Sheet, GC26-9048

Language Reference, SC26-9046

Licensed Program Specifications, GC26-9044

Programming Guide, SC26-9049

Language Environment for OS/390 & VM

Concepts Guide, GC28-1945

Debugging Guide and Run-Time Messages, SC28-1942

Installation & Customization, SC28-1941

Programming Guide, SC28-1939

Programming Reference, SC28-1940

Run-Time Migration Guide, SC28-1944

Writing Interlanguage Applications, SC28-1943

Related publications

American National Standard Programming Language COBOL, X3.23 – 1985

CICS/ESA Version 4 Installation Guide, SC33-1163

CICS/ESA Version 4 Operations and Utilities Guide, SC33-1167

DBCS Ordering Support Program (DBCSOS): Program Description, N:SH18-0144 (in Japanese)

ISO 1989 — 1985 Standard of the International Organization for Standardization (ISO)

OS/390 MVS Initialization and Tuning Guide, SC28-1751

OS/390 MVS Initialization and Tuning Reference, SC28-1752

OS/390 SMP/E User's Guide, SC28-1740

OS/390 SMP/E Reference, SC28-1806

OS/390 SOMobjects Getting Started, GA22-7248

OS/390 SOMobjects Configuration and Administration, GC28-1851

SMP/E Release 8.1 User's Guide, SC28-1302

SMP/E Release 8.1 Reference, SC28-1107

Softcopy publications

The following collection kits contains IBM COBOL and related product publications.

- *OS/390 collection*, SK2T-6700
- *VM collection*, SK2T-2067

Index

A

ADATA compiler option 15
ADEXIT compiler option 16
ADV compiler option 16
ALOWCBL compiler option 17
ANALYZE compiler option 17
ANSI Standard
  See COBOL 85 Standard
ARITH compiler option 18
ASM1 phase 7
ASM2 phase 7
AWO compiler option 18

B

BUF compiler option 19

C

CBL statement 17
CICS reserved word table 12
CICS with Debug Tool, setting up 66
CMPR2 compiler option 19
COBOL 85 standard 13
COBOL for OS/390 & VM
  job modification 55
COMPILE compiler option 20
compiler options
  COBOL 85 standard 13
  conflicting options 14
  default values 2
  description of
    ADATA 15
    ADEXIT 16
    ADV 16
    ALOWCBL 17
    ANALYZE 17
    ARITH 18
    AWO 18
    BUF 19
    CMPR2 19
    COMPILE 20
    CURRENCY 21
    DATA 22
    DATEPROC 23
    DBCS 23
    DBCSXREF 24
    DECK 25
    DIAGTRUNC 25
    DLL 26
    DYNAM 26
    EXPORT 27
    FASTSRT 27
compiler options *(continued)*
  description of *(continued)*
    FLAG 28
    FLAGMIG 29
    FLAGSTD 29
    IDLGEN 31
    INEXIT 32
    INTDATE 32
    LANGUAGE 33
    LIB 34
    LIBEXIT 34
    LINECNT 35
    LIST 35
    LITCHAR 36
    LVLINFO 36
    MAP 36
    NAME 37
    NUM 37
    NUMCLS 38
    NUMPROC 38
    OBJECT 39
    OFFSET 40
    OPTIMIZE 40
    OUTDD 41
    PGMNAME 41
    PRTEXIT 42
    RENT 42
    RMODE 43
    SEQ 44
    SIZE 45
    SOURCE 45
    SPACE 46
    SQL 46
    SSRANGE 47
    TERM 47
    TEST 48
    TRUNC 49
    TYPECHK 50
    VBREF 51
    WORD 51
    XREFOPT 52
    YRWINDOW 53
    ZWB 54
  fixed options 2
  modifying 3, 56
  planning worksheet 3
  setting defaults for 2, 56
  storage allocation 19
compiler phases
  defaults for 7
  description of
    ASM1 7
    ASM2 7

compiler phases *(continued)*
description of *(continued)*
DIAG 7
DMAP 7
FGEN 8
INIT 8
LIBR 8
LSTR 8
MSGT 8
OPTM 8
OSCN 9
PGEN 9
RCTL 9
RWT 9
SCAN 10
SIMD 10
XREF 10
fixed phases 6
IN|OUT parameters 7
macro worksheet 10
modifying 3
placing in shared storage 6
CURRENCY compiler option 21
customization
compiler options 13, 56
Debug Tool 65
installation jobs
COBOL for OS/390 & VM 55
planning for 1

D

DATA compiler option 22
DATEPROC compiler option 23
DBCS compiler option 23
DBCSXREF compiler option 24
Debug Tool
customizing 65
modules, placing in shared storage 65
producing object code for 48
setting up CICS with 66
DECK compiler option 25
default reserved word table 11
default values
compiler options 2
compiler phases 7
DIAG phase 7
DIAGTRUNC compiler option 25
DLL compiler option 26
DMAP phase 7
DUMP compiler option 15
DYNAM compiler option 26

E

ELPA
See shared storage
error messages
flagging 28
EXPORT compiler option 27

F

FASTSRT option 27
FGEN phase 8
fixed compiler options 2
FLAG compiler option 28
FLAGMIG compiler option 29
FLAGSTD compiler option 29
format notation, description vi

I

IDLGEN compiler option 31
IGYCCICS (CICS reserved word table) 12
IGYCDOPT
AMODE 31 and 2
planning worksheet 3
RMODE ANY and 2
using with IGYCDOPT program 2
IGYCNOOO reserved word table 12
IGYCOPT
syntax format 3
IGYCRWT (default reserved word table) 11
index checking 47
INEXIT compiler option 32
INIT phase 8
INTDATE compiler option 32
ISO Standard
See COBOL 85 Standard

K

keywords vii

L

LANGUAGE compiler option 33
LIB compiler option 34
LIBEXIT compiler option 34
LIBR phase 8
LINECNT compiler option 35
LIST compiler option 35
LITCHAR compiler option 36
LSTR phase 8
LVLINFO compiler option 36

M

macro worksheets
  See planning worksheets
macros
  IGYCDOPT (compiler options)
    planning worksheet 3
    syntax format 3
  IGYCDOPT (compiler phases)
    planning worksheet 10
    syntax format 3
MAP compiler option 36
messages, flagging 28
MLPA
  See shared storage
MSGT phase 8

N

NAME compiler option 37
nested programs 11
notation, syntax vi
notice information 68
NUM compiler option 37
NUMCLS compiler option 38
NUMPROC compiler option 38

O

object code, reentrant 42
OBJECT compiler option 39
OFFSET compiler option 40
OPTIMIZE compiler option 40
optional words vi
options
  See compiler options
OPTM phase 8
OSCN phase 9
OUTDD compiler option 41

P

PGEN phase 9
PGMNAME compiler option 41
phases, compiler
  defaults for 7
  macro worksheet 10
  modifying 3
  placing in shared storage 6
placing
  Debug Tool modules in shared storage 65
planning worksheets
  description of vii
  IGYCDOPT (compiler options) 3
  IGYCDOPT (compiler phases) 10
preface vi
PROCESS (CBL) statement 17
PRTEXIT compiler option 42
publications 70

R

RCTL phase 9
reentrant object code 42
RENT compiler option 42
required words vi
reserved word table
  contents of 12
  creating or modifying 58
  nested programs 11
  planning for 11
  specifying an alternative table 51
  supplied with COBOL for OS/390 & VM
    IGYCCICS 11
    IGYCNOOO 11
    IGYCRWT (default reserved word table) 11
residency mode 43
RMODE compiler option 43
rules for syntax notation vi
RWT phase 9

S

sample installation jobs 2
SCAN phase 10
sequence checking of line numbers 44
SEQUENCE compiler option 44
shared storage
  compiler phases in 5, 7
  placing COBOL for OS/390 & VM modules in 63
  placing Debug Tool modules in 65
  planning for 5
SIMD phase 10
SIZE compiler option 45
SOURCE compiler option 45
SPACE compiler option 46
SQL compiler option 46
SSRANGE compiler option 47
stacked words vi
subscript checking 47
symbols for syntax notation vi
syntax checking 20
syntax notation
  COBOL keywords vii
  description of vi
  repeat arrows vii
  rules for vi
  symbols used in vi
SYSLIN 39
SYSOUT 41
SYSPUNCH 25

SYSTERM 47

T

TERM compiler option 47
TEST compiler option 48
TRUNC compiler option 49
TYPECHK compiler option 50

U

user exit routine
ADEXIT compiler option 16
LIBEXIT option 34
PRTEXIT option 42

V

values, default
See default values
VBREF compiler option 51
virtual storage

W

WORD compiler option 51
worksheets
See planning worksheets

X

XREF compiler option 52
XREF phase 10
XREFOPT option 52

Y

YRWINDOW compiler option 53

Z

ZWB compiler option 54

We'd Like to Hear from You

COBOL for OS/390 & VM
Customization under OS/390
Version 2 Release 2

Publication No. GC26-9045-02

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - IBMLink: COBPUBS at STLVM27
  - Internet: COBPUBS@VNET.IBM.COM

  Be sure to include the following with your comments:
  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

COBOL for OS/390 & VM
Customization under OS/390
Version 2 Release 2

Publication No. GC26-9045-02

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

May we contact you to discuss your comments? Yes No

Would you like to receive our response by E-Mail?

_____

Your E-mail address

Name

Address

Company or Organization

Phone No.

Readers' Comments
GC26-9045-02

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department HHX/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5648-A25

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

COBOL for OS/390 & VM

GC26-4764	Compiler and Run-Time Migration Guide
SC09-2137	Debug Tool Reference Summary
SX26-3840	Debug Tool User's Guide and Reference
GC26-9048	Fact Sheet
GC26-9045	Customization under OS/390
SC26-9046	Language Reference
GC26-9044	Licensed Program Specifications
SC26-9049	Programming Guide

GC26-9045-02

Spine information:

COBOL for OS/390 & VM

Customization under OS/390

Version 2 Release 2