

IBM z/OS Debugger



# Reference Summary

*Version 14.2.0*



IBM z/OS Debugger



# Reference Summary

*Version 14.2.0*

**Note!**

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 75.

**First Edition (September 2019)**

- | This edition applies to IBM z/OS Debugger, Version 14.2.0 (Program Number 5724-T07), which supports the following compilers:
- AD/Cycle C/370™ Version 1 Release 2 (Program Number 5688-216)
  - C/C++ for MVS/ESA Version 3 (Program Number 5655-121)
  - C/C++ feature of OS/390 (Program Number 5647-A01)
  - C/C++ feature of z/OS Version 1 (Program Number 5694-A01)
  - C/C++ feature of z/OS Version 2 (Program Number 5650-ZOS)
  - OS/VS COBOL, Version 1 Release 2.4 (5740-CB1) - with limitations
  - VS COBOL II Version 1 Release 3 and Version 1 Release 4 (Program Numbers 5668-958, 5688-023) - with limitations
  - COBOL/370 Version 1 Release 1 (Program Number 5688-197)
  - COBOL for MVS & VM Version 1 Release 2 (Program Number 5688-197)
  - COBOL for OS/390 & VM Version 2 (Program Number 5648-A25)
  - Enterprise COBOL for z/OS and OS/390 Version 3 (Program Number 5655-G53)
  - Enterprise COBOL for z/OS Version 4 (Program Number 5655-S71)
  - Enterprise COBOL for z/OS Version 5 (Program Number 5655-W32)
  - | • Enterprise COBOL for z/OS Version 6 Release 1, Release 2, and Release 3 (Program Number 5655-EC6)
  - High Level Assembler for MVS & VM & VSE Version 1 Release 4, Version 1 Release 5, Version 1 Release 6 (Program Number 5696-234)
  - OS PL/I Version 2 Release 1, Version 2 Release 2, Version 2 Release 3 (Program Numbers 5668-909, 5668-910) - with limitations
  - PL/I for MVS & VM Version 1 Release 1 (Program Number 5688-235)
  - VisualAge® PL/I for OS/390 Version 2 Release 2 (Program Number 5655-B22)
  - Enterprise PL/I for z/OS and OS/390 Version 3 (Program Number 5655-H31)
  - Enterprise PL/I for z/OS Version 4 (Program Number 5655-W67)
  - | • Enterprise PL/I for z/OS Version 5 Release 1, Release 2, and Release 3 (Program Number 5655-PL5)

This edition also applies to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

You can access publications online at [www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss)

You can find out more about IBM z/OS Debugger by visiting the following IBM Web sites:

- IBM Debug for z/OS: <https://www.ibm.com/us-en/marketplace/debug-for-z-systems>
- IBM Developer for z/OS: <https://www.ibm.com/us-en/marketplace/developer-for-z-systems>
- IBM Z Open Development: <https://www.ibm.com/us-en/marketplace/z-open-development>
- IBM Z Open Unit Test: <https://www.ibm.com/us-en/marketplace/z-open-unit-test>

© Copyright IBM Corporation 1992, 2019.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this document</b> . . . . .	<b>vii</b>	CALL entry_name (COBOL) . . . . .	9
Who might use this document . . . . .	vii	CALL procedure . . . . .	9
Accessing z/OS licensed documents on the Internet . . . . .	vii	CC command. . . . .	10
How to read syntax diagrams . . . . .	viii	CHKSTGV. . . . .	10
Symbols . . . . .	viii	CLEAR command . . . . .	10
Syntax items. . . . .	viii	CLEAR AT (remote) . . . . .	11
Syntax examples . . . . .	ix	CLEAR prefix (full-screen mode) . . . . .	12
How to provide your comments. . . . .	x	COMMENT command. . . . .	12
		COMPUTE command (COBOL) . . . . .	12
		CURSOR command (full-screen mode) . . . . .	12
<b>Summary of changes.</b> . . . . .	<b>xi</b>	Declarations (assembler, disassembly, and LangX COBOL) . . . . .	13
		Declarations (C and C++). . . . .	13
<b>IBM z/OS Debugger as a component</b> . . . . .	<b>xiii</b>	Declarations (COBOL). . . . .	15
		DECLARE command (PL/I). . . . .	15
<b>Chapter 1. z/OS Debugger commands.</b> . . . . .	<b>1</b>	DESCRIBE command . . . . .	16
? command. . . . .	1	DISABLE command . . . . .	17
ALLOCATE command . . . . .	1	DISABLE prefix (full-screen mode) . . . . .	17
ANALYZE command (PL/I) . . . . .	1	DO command (assembler, disassembly, and LangX COBOL) . . . . .	17
Assignment command (assembler and disassembly) . . . . .	1	DO command (PL/I) . . . . .	17
Assignment command (LangX COBOL) . . . . .	2	do/while command (C and C++) . . . . .	18
Assignment command (PL/I). . . . .	2	ENABLE command. . . . .	18
AT ALLOCATE (PL/I) . . . . .	2	ENABLE prefix (full-screen mode). . . . .	19
AT APPEARANCE . . . . .	2	EVALUATE command (COBOL) . . . . .	19
AT CALL . . . . .	2	Expression command (C and C++) . . . . .	19
AT CHANGE . . . . .	3	FIND command . . . . .	19
AT CHANGE (remote) . . . . .	3	FINDBP command . . . . .	20
AT CURSOR (full-screen mode) . . . . .	3	for command (C and C++) . . . . .	20
AT DATE (COBOL) . . . . .	3	FREE command . . . . .	20
AT DELETE . . . . .	4	GO command . . . . .	20
AT ENTRY . . . . .	4	GOTO command . . . . .	20
AT ENTRY (remote) . . . . .	4	GOTO LABEL command . . . . .	21
AT EXIT . . . . .	4	%IF command (programming language neutral) . . . . .	21
AT GLOBAL . . . . .	4	IF command (assembler, disassembly, and LangX COBOL) . . . . .	21
AT GLOBAL LABEL (remote). . . . .	5	if command (C and C++). . . . .	21
AT LABEL . . . . .	5	IF command (COBOL). . . . .	21
AT LABEL (remote) . . . . .	5	IF command (PL/I). . . . .	21
AT LINE. . . . .	5	IMMEDIATE command (full-screen mode) . . . . .	22
AT LOAD . . . . .	6	INPUT command (C and C++ and COBOL) . . . . .	22
AT LOAD (remote) . . . . .	6	JUMPTO command. . . . .	22
AT OCCURRENCE . . . . .	6	JUMPTO LABEL command . . . . .	22
AT OFFSET (disassembly) . . . . .	6	LIST (blank) . . . . .	22
AT PATH . . . . .	6	LIST AT . . . . .	22
AT Prefix (full-screen mode) . . . . .	7	LIST AT (remote) . . . . .	23
AT STATEMENT . . . . .	7	LIST CALLS . . . . .	23
AT STATEMENT (remote) . . . . .	7	LIST CC command . . . . .	24
AT TERMINATION . . . . .	7	LIST CONTAINER . . . . .	24
BEGIN command (programming language neutral) . . . . .	7	LIST CURSOR (full-screen mode) . . . . .	24
block command (C and C++). . . . .	8	LIST DTCN or CADP . . . . .	24
break command (C and C++). . . . .	8	LIST expression . . . . .	24
CALL %CEBR . . . . .	8	L expression prefix . . . . .	25
CALL %CECI . . . . .	8	LIST FREQUENCY . . . . .	25
CALL %DUMP . . . . .	8	LIST LAST . . . . .	25
CALL %FA . . . . .	8	LIST LINE NUMBERS. . . . .	25
CALL %FM. . . . .	9		
CALL %HOGAN. . . . .	9		
CALL %VER . . . . .	9		

LIST LINES . . . . .	26	SET ECHO . . . . .	42
LIST MONITOR. . . . .	26	SET EQUATE. . . . .	42
LIST NAMES. . . . .	26	SET EXECUTE . . . . .	42
LIST NAMES LABELS (remote) . . . . .	26	SET EXPLICITDEBUG. . . . .	43
LIST ON (PL/I) . . . . .	26	SET FIND BOUNDS . . . . .	43
LIST PROCEDURES . . . . .	27	SET FREQUENCY . . . . .	43
LIST REGISTERS . . . . .	27	SET HISTORY . . . . .	43
LIST STATEMENT NUMBERS . . . . .	27	SET IGNORELINK . . . . .	43
LIST STATEMENTS . . . . .	27	SET INTERCEPT (C and C++) . . . . .	44
LIST STORAGE . . . . .	27	SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode). . . . .	44
LIST TRACE LOAD command . . . . .	28	SET INTERCEPT (COBOL, remote debug mode) . . . . .	44
LOAD command . . . . .	28	SET KEYS (full-screen and line mode) . . . . .	44
LOADDEBUGDATA (LDD) . . . . .	28	SET LDD . . . . .	44
MEMORY . . . . .	29	SET LIST BY SUBSCRIPT command (COBOL). . . . .	44
M prefix command . . . . .	29	SET LIST BY SUBSCRIPT command (Enterprise PL/I, full-screen mode only). . . . .	45
MONITOR command . . . . .	29	SET LIST TABULAR . . . . .	45
MOVE command (COBOL) . . . . .	29	SET LOG . . . . .	45
NAMES DISPLAY command . . . . .	30	SET LOG NUMBERS (full-screen and line mode). . . . .	45
NAMES EXCLUDE command . . . . .	30	SET LONGCUNAME (C, C++, and PL/I) . . . . .	45
NAMES INCLUDE command . . . . .	30	SET MDBG (C, C++) . . . . .	46
Null command . . . . .	30	SET MONITOR (full-screen and line mode). . . . .	46
ON command (PL/I) . . . . .	31	SET MSGID . . . . .	46
PANEL command (full-screen mode) . . . . .	31	SET NATIONAL LANGUAGE . . . . .	46
PERFORM command (COBOL). . . . .	31	SET PACE . . . . .	46
PLAYBACK BACKWARD command . . . . .	32	SET PFKEY . . . . .	46
PLAYBACK DISABLE command . . . . .	32	SET POPUP . . . . .	47
PLAYBACK ENABLE command . . . . .	32	SET PROGRAMMING LANGUAGE . . . . .	47
PLAYBACK FORWARD command . . . . .	33	SET PROMPT (full-screen and line mode) . . . . .	47
PLAYBACK START command . . . . .	33	SET QUALIFY . . . . .	47
PLAYBACK STOP command . . . . .	33	SET REFRESH (full-screen mode) . . . . .	47
POPUP command . . . . .	33	SET RESTORE . . . . .	48
POSITION command . . . . .	33	SET REWRITE (full-screen mode) . . . . .	48
Prefix commands (full-screen mode) . . . . .	33	SET REWRITE (remote debug mode). . . . .	48
PROCEDURE command . . . . .	34	SET SAVE . . . . .	48
QUALIFY RESET . . . . .	34	SET SCREEN (full-screen and line mode) . . . . .	48
QUERY command . . . . .	34	SET SCROLL DISPLAY (full-screen mode) . . . . .	49
QUERY prefix (full-screen mode) . . . . .	36	SET SEQUENCE (PL/I) . . . . .	49
QUIT command . . . . .	37	SET SOURCE. . . . .	49
QQUIT command . . . . .	37	SET SUFFIX (full-screen mode). . . . .	49
RESTORE command . . . . .	37	SET TEST . . . . .	49
RETRIEVE command (full-screen mode). . . . .	37	SET WARNING (C, C++, and PL/I) . . . . .	49
RUN command . . . . .	37	SET command (COBOL) . . . . .	50
RUNTO command . . . . .	37	SHOW prefix command (full-screen mode). . . . .	50
RUNTO prefix command (full-screen mode) . . . . .	38	STEP command . . . . .	50
SCROLL command (full-screen mode) . . . . .	38	STORAGE command . . . . .	50
SELECT command (PL/I) . . . . .	38	switch command (C and C++) . . . . .	50
SET ASSEMBLER ON/OFF . . . . .	38	SYSTEM command . . . . .	51
SET ASSEMBLER STEPOVER . . . . .	38	TRACE command . . . . .	51
SET AUTOMONITOR . . . . .	39	TRIGGER command . . . . .	51
SET CHANGE . . . . .	39	TSO command (z/OS). . . . .	52
SET COLOR (full-screen and line mode). . . . .	39	USE command . . . . .	53
SET COUNTRY . . . . .	40	while command (C and C++) . . . . .	53
SET DBCS . . . . .	40	WINDOW CLOSE . . . . .	53
SET DEFAULT DBG . . . . .	40	WINDOW OPEN . . . . .	53
SET DEFAULT LISTINGS. . . . .	41	WINDOW SIZE . . . . .	53
SET DEFAULT MDBG. . . . .	41	WINDOW SWAP . . . . .	54
SET DEFAULT SCROLL (full-screen mode). . . . .	41	WINDOW ZOOM . . . . .	54
SET DEFAULT VIEW . . . . .	41		
SET DEFAULT WINDOW (full-screen mode) . . . . .	42		
SET DISASSEMBLY. . . . .	42		
SET DYNDEBUG . . . . .	42		

**Chapter 2. z/OS Debugger built-in functions. . . . . 55**

%DEC (assembler, disassembly, and LangX COBOL) . . . . . 55  
%GENERATION (PL/I) . . . . . 55  
%HEX . . . . . 55  
%INSTANCES (C, C++, and PL/I). . . . . 55  
%RECURSION (C, C++, and PL/I) . . . . . 55  
%WHERE (assembler, disassembly, and LangX COBOL) . . . . . 55

**Chapter 3. EQAOPTS commands (optional). . . . . 57**

BROWSE . . . . . 59  
CACHENUM. . . . . 59  
CODEPAGE . . . . . 60  
COMMANDSDSN . . . . . 60  
DEFAULTVIEW . . . . . 60  
DLAYDBG. . . . . 60  
    DLAYDBGEND . . . . . 60  
    DLAYDBGDSN . . . . . 60  
    DLAYDBGTRC . . . . . 60  
    DLAYDBGXRF . . . . . 60  
DOPTACBDSN . . . . . 60  
DTCNDELETEDEADPROF . . . . . 60  
DTCNFORCExxxx . . . . . 61  
DYNDEBUG . . . . . 61  
EQAQPP . . . . . 61  
EXPLICITDEBUG . . . . . 61  
GPFDSN . . . . . 61  
HOSTPORTS . . . . . 61  
IGNOREODOLIMIT . . . . . 61  
LOGDSN . . . . . 61  
LOGDSNALLOC . . . . . 62

MAXTRANUSER . . . . . 62  
MDBG . . . . . 62  
MULTIPROCESS . . . . . 62  
NAMES . . . . . 62  
NODISPLAY . . . . . 62  
PREFERENCESDSN . . . . . 62  
SAVEBPDSN and SAVESETDSN . . . . . 62  
SAVEBPDSNALLOC and SAVESETDSNALLOC . . . . . 62  
SESSIONTIMEOUT. . . . . 63  
STARTSTOPMSG . . . . . 63  
    STARTSTOPMSGDSN . . . . . 63  
SUBSYS . . . . . 63  
SVCSCREEN . . . . . 63  
TCPIPDATAADSN . . . . . 63  
THREADTERMCOND. . . . . 63  
TIMACB . . . . . 63  
END. . . . . 63

**Chapter 4. z/OS Debugger variables . . . 65**

**Chapter 5. Reference card: Frequently used z/OS Debugger commands. . . . 67**

**Chapter 6. Reference card: Frequently used z/OS Debugger commands while debugging assembler programs. . . . 71**

**Notices . . . . . 75**

Copyright license . . . . . 76  
Programming interface information . . . . . 76  
Trademarks and service marks . . . . . 76





---

## About this document

z/OS Debugger combines the richness of the z/OS<sup>®</sup> environment with the power of Language Environment<sup>®</sup> to provide a debugger for programmers to isolate and fix their program bugs and test their applications. z/OS Debugger gives you the capability of testing programs in batch, using a nonprogrammable terminal in full-screen mode, or using a workstation interface to remotely debug your programs.

This document contains a summary of commands, built-in functions, and EQAOPTS commands provided by z/OS Debugger. Each topic contains the name of the command, built-in function, or EQAOPTS command and then a syntax diagram. For more information about each command or built-in function, refer to the *IBM z/OS Debugger Reference and Messages*. For more information about each EQAOPTS command, see the *IBM z/OS Debugger Customization Guide* or *IBM z/OS Debugger Reference and Messages*.

---

## Who might use this document

This document is intended for programmers using z/OS Debugger to debug high-level languages (HLLs) with Language Environment and assembler programs either with or without Language Environment. Throughout this document, the HLLs are referred to as C, C++, COBOL, and PL/I.

z/OS Debugger runs on the z/OS operating system and supports the following subsystems:

- CICS<sup>®</sup>
- DB2<sup>®</sup>
- IMS
- JES batch
- TSO
- UNIX System Services in remote debug mode or full-screen mode using the Terminal Interface Manager only

To use this document and debug a program written in one of the supported languages, you need to know how to write, compile, and run such a program.

---

## Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM<sup>®</sup> Resource Link<sup>®</sup> Web site at:

<http://www.ibm.com/servers/resourceLink>

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-8928), that includes this key code.

To obtain your IBM Resource Link user ID and password, log on to:

<http://www.ibm.com/servers/resourceLink>

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

**Note:** You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

---

## How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.

Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

### Symbols

The following symbols may be displayed in syntax diagrams:

Symbol	Definition
--------	------------

- |     |  |
|-----|--|
| ▶—  | Indicates the beginning of the syntax diagram.                   |
| —▶  | Indicates that the syntax diagram is continued to the next line. |
| ▶—  | Indicates that the syntax is continued from the previous line.   |
| —▶▶ | Indicates the end of the syntax diagram.                         |

### Syntax items

Syntax diagrams contain many different items. Syntax items include:

- Keywords - a command name or any other literal information.
- Variables - variables are italicized, appear in lowercase and represent the name of values you can supply.
- Delimiters - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- Operators - operators include add (+), subtract (-), multiply (\*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- Fragment references - a part of a syntax diagram, separated from the diagram to show greater detail.
- Separators - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

## Item type

### Definition

#### Required

Required items are displayed on the main path of the horizontal line.

#### Optional

Optional items are displayed below the main path of the horizontal line.

#### Default

Default items are displayed above the main path of the horizontal line.

## Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

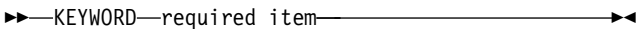
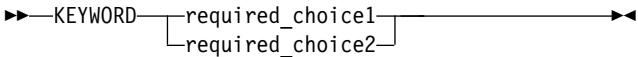
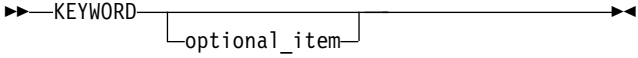
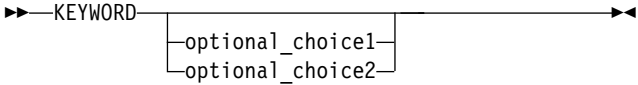
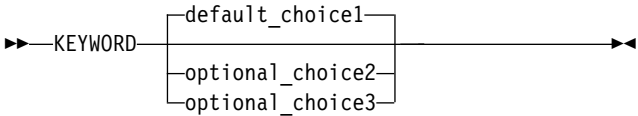

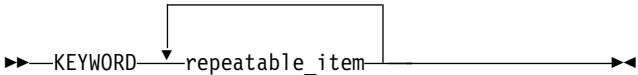
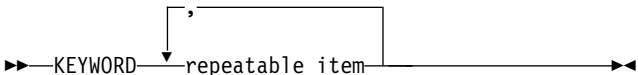
Item	Syntax example
Required item.	
Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice.	
A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
Optional item.	
Optional items appear below the main path of the horizontal line.	
Optional choice.	
An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.	
Default.	
Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.	
Variable.	
Variables appear in lowercase italics. They represent names or values.	
Repeatable item.	
An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.	
A character within the arrow means you must separate repeated items with that character.	
An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.	

Table 1. Syntax examples (continued)

Item	Syntax example
<p>Fragment.</p> <p>The <code>-  fragment  </code> symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.</p>	

## How to provide your comments

Your feedback is important in helping us to provide accurate, high-quality information. If you have comments about this document or any other z/OS Debugger documentation, you can leave a comment in IBM Knowledge Center:

- IBM Developer for z/OS, IBM Debug for z/OS, and IBM Developer for z/OS Enterprise Edition: <https://www.ibm.com/support/knowledgecenter/SSQ2R2>
- IBM Z Open Development: <https://www.ibm.com/support/knowledgecenter/SSUFAU>
- IBM Z Open Unit Test: <https://www.ibm.com/support/knowledgecenter/SSZHNR>

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.

---

## Summary of changes

### Version 14.2.0

The following changes were added for IBM z/OS Debugger Version 14.2.0.

- **New support**

- z/OS Version 2 Release 4
- Enterprise COBOL for z/OS Version 6 Release 3
- Enterprise PL/I for z/OS Version 5 Release 3

- **Product renaming**

- IBM Application Delivery Foundation for z Systems® is renamed as Application Delivery Foundation for z/OS.
- IBM Developer for z Systems Enterprise Edition is renamed as IBM Developer for z/OS Enterprise Edition.
- IBM Developer for z Systems is renamed as IBM Developer for z/OS.
- IBM Debug for Systems is renamed as IBM Debug for z/OS.

- **IBM Debug for z/OS client installation**

- In addition to using IBM Installation Manager, you can now install the IBM Debug for z/OS client with Eclipse p2. For more information, see the "Installing the IBM Debug for z/OS client" topic in IBM Knowledge Center.

- **Debug Profile Service**

- Debug Profile Service is a REST API that uses the HTTP protocol to provide RESTful access to a set of resources related to debug profiles. You can use this service to create, retrieve, update and delete debug profiles.

For more information about host configuration, see the "Adding support for Debug Profile Service and APIs" section in *IBM z/OS Debugger Customization Guide*.

For more details about the API, see the "z/OS Debug Profile Service API" documentation in IBM Knowledge Center.

- **Section breakpoint support**

- When you edit COBOL with the COBOL Editor or the z Systems LPEX Editor, a Toggle Section Breakpoint action is available in the left ruler context menu of the editor and the context menu for sections in the outline view. For more information, see the "Source entry breakpoints" topic in IBM Knowledge Center.

- **IMS Transaction Isolation facility**

- IMS Transaction Isolation facility no longer accesses IMS ACB libraries. It has a simplified setup and is compatible with ACB management by IMS Catalog. The setting of IMSISOORIGPSB in EQAOPTS no longer has any effect and the original PSB is always preserved. For more information, see the "Scenario F: Enabling the Transaction Isolation Facility" topic in *IBM z/OS Debugger Customization Guide*.
- Instructions for using the IMS PSTOP command are added to end the wait state of the IMS region when PWFI is used, so that you can continue with the program. For more information, see the "Using IMS pseudo wait-for-input (PWFI) with IMS Transaction Isolation" topic in *IBM z/OS Debugger User's Guide*.



---

## IBM z/OS Debugger as a component

IBM z/OS Debugger is the next iteration of IBM debug technology on z Systems and consolidates the IBM Integrated Debugger and IBM Debug Tool engines into one unified technology. IBM z/OS Debugger is progressing towards one remote debug mode based on Debug Tool compatibility mode. In support of this direction, Debug Tool compatibility mode, when available in the user interface, is selected by default for V14.1.2 or later.

IBM z/OS Debugger is a component of the following products:

### **IBM Developer for z/OS Enterprise Edition**

This product is included in IBM Application Delivery Foundation for z/OS. IBM Developer for z/OS Enterprise Edition provides all the debug features.

### **IBM Developer for z/OS**

IBM Developer for z/OS is a subset of IBM Developer for z/OS Enterprise Edition. IBM Developer for z/OS, previously known as IBM Developer for z Systems or IBM Rational® Developer for z Systems, is an Eclipse-based integrated development environment for creating and maintaining z/OS applications efficiently.

IBM Developer for z/OS includes all enhancements in IBM Developer for z/OS Enterprise Edition except for the debug features noted in Table 2 on page xiv.

### **IBM Debug for z/OS**

IBM Debug for z/OS is a subset of IBM Developer for z/OS Enterprise Edition. IBM Debug for z/OS focuses on debugging solutions for z/OS application developers. See Table 2 on page xiv for the debug features supported.

IBM Debug for z/OS does not provide advanced developer features that are available in IBM Developer for z/OS Enterprise Edition.

For information about how to install the IBM Debug for z/OS client, see Installation of IBM Developer for z Systems and IBM Debug for z Systems (<https://developer.ibm.com/mainframe/2016/12/02/installation-of-ibm-developer-for-z-systems-and-ibm-debug-for-z-systems/>).

### **IBM Z Open Development**

IBM Z Open Development offers an entry level toolset with the core capabilities needed to link z/OS development seamlessly with an established, open DevOps toolchain. IBM Z Open Development provides remote debug support for high level compiled languages. See Table 2 on page xiv for the debug features supported.

### **IBM Z Open Unit Test, extending IBM Z Open Development**

IBM Z Open Unit Test is an automated unit testing tool for batch and CICS programs. It helps minimize the time taken to unit test traditional z/OS applications, and helps businesses respond and deliver with speed.

IBM Z Open Unit Test provides code coverage support for high level compiled languages via IBM z/OS Debugger. It can only be used with the IBM Z Open Development offering and extends IBM Z Open Development

to help ensure that no untested code is delivered. See Table 2 for the debug features supported for the combination.

Table 2 maps out the features that differ in products. Not all the available features are listed. In this table, the letter X indicates the features that each product supports.

Table 2. Debug feature comparison

	IBM Z Open Development	IBM Z Open Unit Test, extending IBM Z Open Development	IBM Debug for z/OS	IBM Developer for z/OS	IBM Developer for z/OS Enterprise Edition
<b>Main features</b>					
z/OS Debugger 3270 interface, including z/OS Debugger Utilities			X		X
z/OS Debugger remote debug	X	X	X <sup>1</sup>	X <sup>2</sup>	X
Debug Tool compatibility mode	X	X	X	X	X
Standard mode <sup>3</sup>			X	X	X
Debug Tool Plugins			X	X <sup>4</sup>	X
<b>Code Coverage features</b>					
Compiled Code Coverage		X	X <sup>1</sup>	X <sup>2</sup>	X
Headless Code Coverage		X		X	X
Java™ Code Coverage				X	X
z/OS Debugger Code Coverage (3270 and remote interfaces) <sup>5</sup>			X		X
<b>3270 features</b>					
z/OS Debugger full screen, batch or line mode			X		X



Table 2. Debug feature comparison (continued)

	IBM Z Open Development	IBM Z Open Unit Test, extending IBM Z Open Development	IBM Debug for z/OS	IBM Developer for z/OS	IBM Developer for z/OS Enterprise Edition
IMS Isolation support			X		X
<b>Remote debug features</b>					
Integration with Language Editors: • COBOL Editor • PLI Editor	X	X		X	X
Integration with Language Editors: • System z <sup>®</sup> LPEX Editor • Remote C/C++ Editor				X	X
Visual Debug	X	X		X	X
IMS Isolation GUI interface					X
Integration with CICS Explorer <sup>®</sup> views				X	X
Integration with Property groups	X	X		X	X
Source Entry Breakpoints	X	X		X	X
Team Debug support				X	X
<b>Compiler support features</b>					
Assembler support: Create EQALANGX files			X	X	X
Assembler support: Debugging <sup>6</sup>	X	X	X	X	X
LANGX COBOL support <sup>7</sup>			X	X	X

Table 2. Debug feature comparison (continued)

	IBM Z Open Development	IBM Z Open Unit Test, extending IBM Z Open Development	IBM Debug for z/OS	IBM Developer for z/OS	IBM Developer for z/OS Enterprise Edition
Support for Automatic Binary Optimizer (ABO)			X	X	X
IBM COBOL and CICS Command Level Conversion Aid for OS/390® & MVS & VM			X	X	X
Load Module Analyzer			X		X

**Notes:**

1. IBM Debug for z/OS includes z/OS Debugger remote debug and compiled code coverage GUI interface, but does not include Headless Code Coverage and Java Code Coverage.
2. IBM Developer for z/OS includes z/OS Debugger remote debug and compiled code coverage GUI interface, but does not include z/OS Debugger Code Coverage.
3. The following features are only supported in standard mode:
  - Support for 64 bit Enterprise PL/I for z/OS Version 5
  - Support for 64 bit C/C++ feature of z/OS
  - Source view for COBOL V6.2 and later
4. IBM Developer for z/OS includes Debug Tool plugins, but does not include Load Module Analyzer and z/OS Debugger Code Coverage.
5. z/OS Debugger Code Coverage can only be enabled in the 3270 interface.
6. Debugging assembler requires that you have EQALANGX files that have been created via ADFz Common Components or a product that ships the ADFz Common Components.
7. LANGX COBOL refers to any of the following programs:
  - A program compiled with the IBM OS/VS COBOL compiler.
  - A program compiled with the IBM VS COBOL II compiler with the NOTEST compiler option.
  - A program compiled with the IBM Enterprise COBOL for z/OS Version 3 or Version 4 compiler with the NOTEST compiler option.

---

## Chapter 1. z/OS Debugger commands

z/OS Debugger provides the following commands:

---

### ? command

Displays a list of all commands or, if used in combination with a command, displays a list of options that you can specify for that command.

►► ? ;

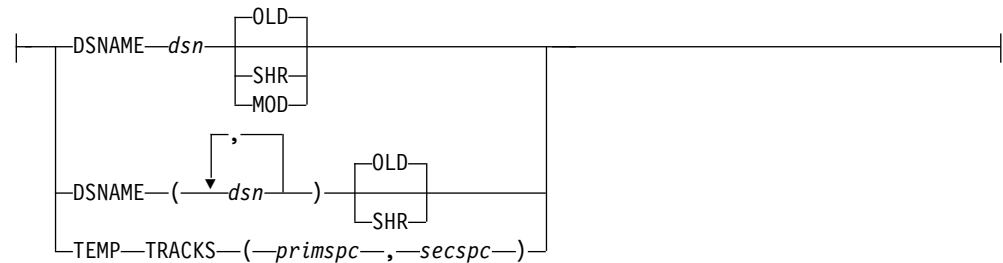
---

### ALLOCATE command

The ALLOCATE command allocates a file (*ddname*) to an existing data set, a concatenation of existing data sets, or a temporary data set.

►► ALLOCATE FILE *ddname* | attributes ;

#### attributes:



---

### ANALYZE command (PL/I)

The ANALYZE command displays the process of evaluating an expression and the data attributes of any intermediate results.

►► ANALYZE EXPRESSION ( *expression* ) ;

---

### Assignment command (assembler and disassembly)

The Assignment command copies the value of an expression to a specified memory location or register.

►► *receiver* [ < *receiverlen* > ] = *sourceexpr* ;

---

## Assignment command (LangX COBOL)

The Assignment command assigns the value of an expression to a specified reference. It is the equivalent of the COBOL COMPUTE statement.

►► '—*receiver*—' == '—*sourceexpr*—' ;

---

## Assignment command (PL/I)

The Assignment command copies the value of an expression to a specified reference.

►► *reference* == *expression* ;

---

## AT ALLOCATE (PL/I)

AT ALLOCATE gives z/OS Debugger control when storage for a named controlled variable or aggregate is dynamically allocated by PL/I.

►► AT —*every\_clause*— ALLOCATE —*identifier*—  
↓  
,  
↓  
—(*identifier*)—  
↓  
\* *command* ;

---

## AT APPEARANCE

Gives z/OS Debugger control when the specified compile unit is found in storage.

►► AT —*every\_clause*— APPEARANCE —*cu\_spec*—  
↓  
,  
↓  
—(*cu\_spec*)—  
↓  
\* *command* ;

---

## AT CALL

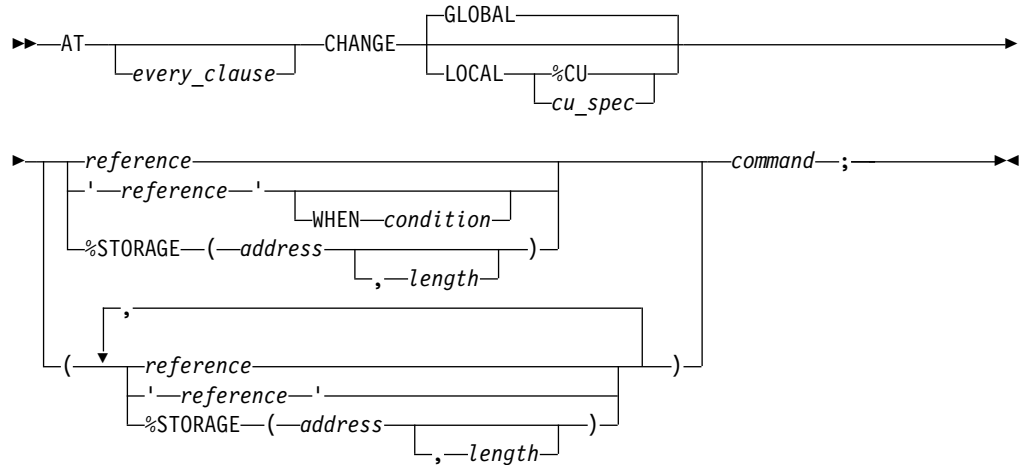
Gives z/OS Debugger control when the application code attempts to call the specified entry point.

►► AT —*every\_clause*— CALL —*entry\_name*—  
↓  
,  
↓  
—(*entry\_name*)—  
↓  
\* *command* ;

---

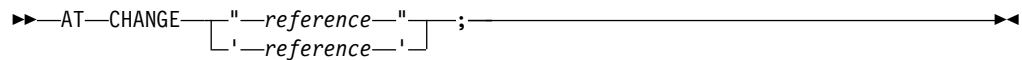
## AT CHANGE

Gives z/OS Debugger control when either the program or z/OS Debugger command changes the specified variable value or storage location, and, optionally, when the specified condition is met.



## AT CHANGE (remote)

Gives z/OS Debugger control when either the program or z/OS Debugger command changes the specified variable value.



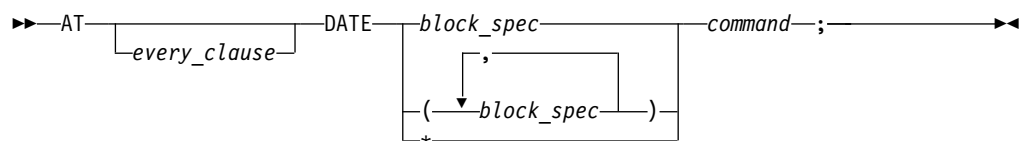
## AT CURSOR (full-screen mode)

Provides a cursor controlled method for setting a statement breakpoint.



## AT DATE (COBOL)

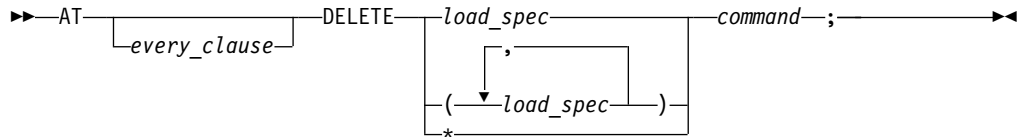
Gives z/OS Debugger control for each date processing statement within the specified block.



---

## AT DELETE

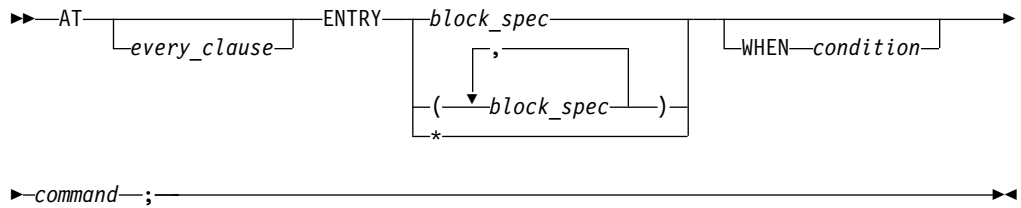
Gives z/OS Debugger control when a load module is removed from storage by a Language Environment delete service, such as on completion of a successful C release(), COBOL CANCEL, or PL/I RELEASE.



---

## AT ENTRY

Defines a breakpoint at the specified entry point in the specified block.



---

## AT ENTRY (remote)

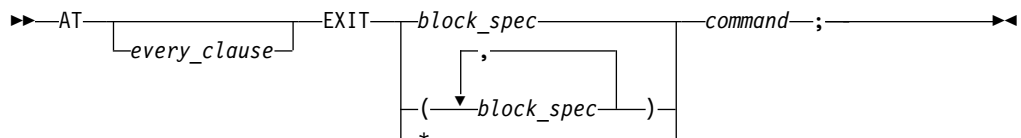
Defines a breakpoint at the specified entry point in the specified block.



---

## AT EXIT

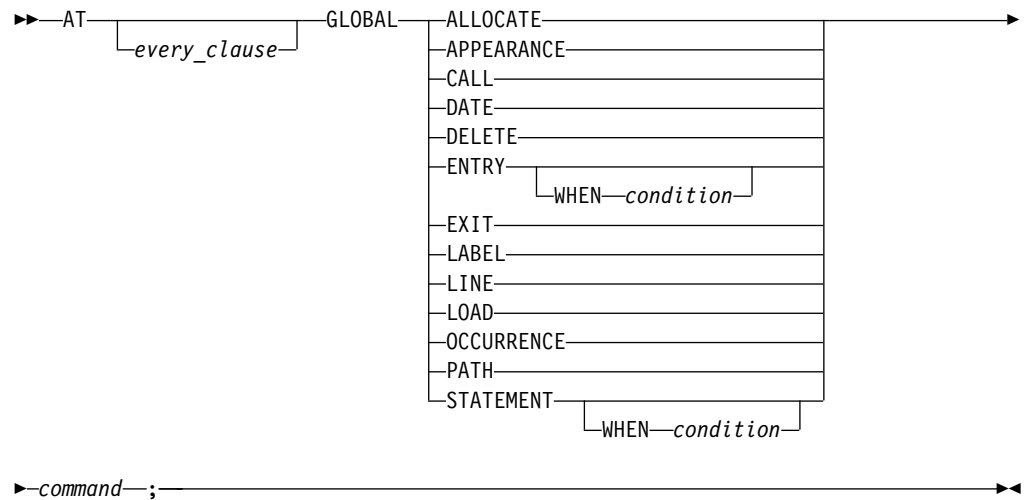
Defines a breakpoint at the specified exit point in the specified block.



---

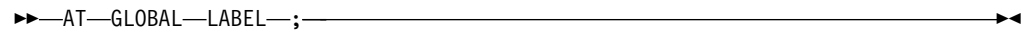
## AT GLOBAL

Gives z/OS Debugger control for every instance of the specified AT-condition.



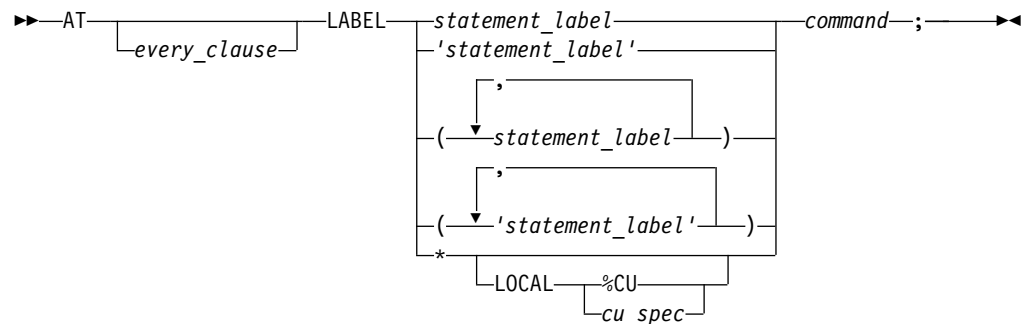
## AT GLOBAL LABEL (remote)

Gives z/OS Debugger control for every instance of the specified AT-Label condition.



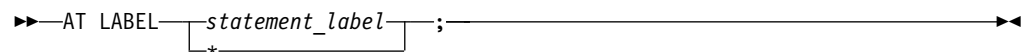
## AT LABEL

Gives z/OS Debugger control when execution has reached the specified statement label or group of labels.



## AT LABEL (remote)

Gives z/OS Debugger control when execution reaches the statement label that you specify.



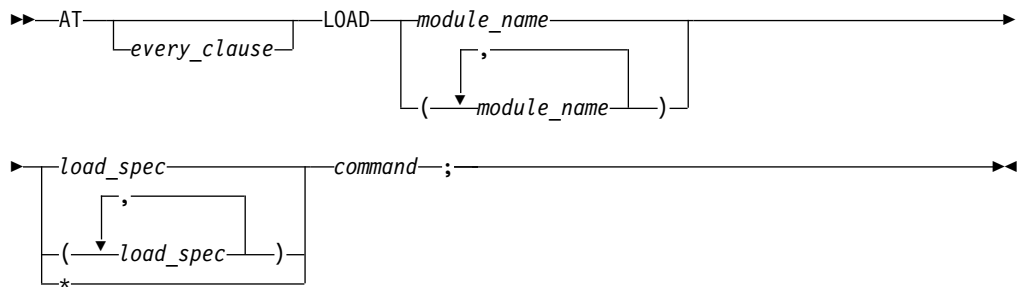
## AT LINE

Gives z/OS Debugger control at the specified line. See "AT STATEMENT" on page 7.

---

## AT LOAD

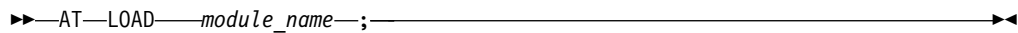
Gives z/OS Debugger control when the specified load module is brought into storage.



---

## AT LOAD (remote)

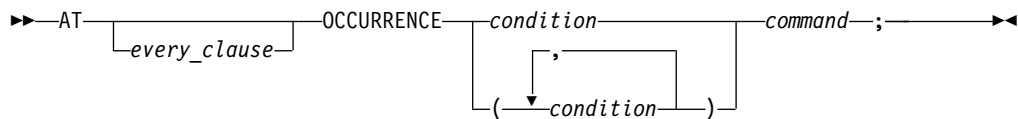
Gives z/OS Debugger control when the specified load module is brought into storage.



---

## AT OCCURRENCE

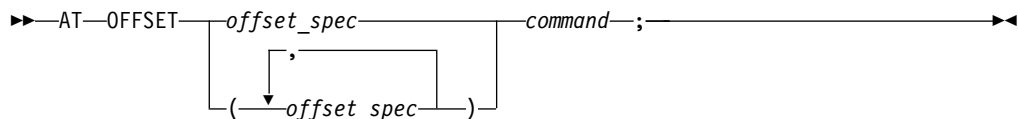
Gives z/OS Debugger control on a language or Language Environment condition or exception.



---

## AT OFFSET (disassembly)

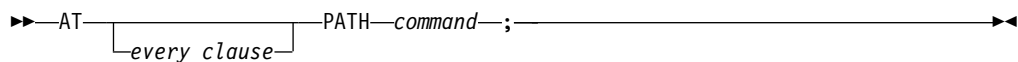
Gives z/OS Debugger control at the specified offset in the disassembly view.



---

## AT PATH

Gives z/OS Debugger control when the flow of control changes (at a path point).  
AT PATH is identical to AT GLOBAL PATH.



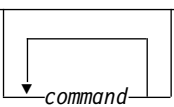




---

## block command (C and C++)

The block command allows you to group any number of z/OS Debugger commands into one command.

```
▶▶ {  } ; ▶▶
```

---

## break command (C and C++)

The break command allows you to terminate and exit a loop (that is, do, for, and while) or switch command from any point other than the logical end.

```
▶▶ break ; ▶▶
```

---

## CALL %CEBR

Starts the CICS Temporary Storage Browser Program.

```
▶▶ CALL %CEBR ; ▶▶
```

---

## CALL %CECI

Starts the CICS Command Level Interpreter Program.

```
▶▶ CALL %CECI ; ▶▶
```

---

## CALL %DUMP

Calls the Language Environment dump service to obtain a formatted dump.

```
▶▶ CALL %DUMP [ (-options_string [ , -title ] ) ] ; ▶▶
```

---

## CALL %FA

Starts and instructs IBM Fault Analyzer to provide a formatted dump of the current machine state.

```
▶▶ CALL %FA ; ▶▶
```

---

## CALL %FM

Starts IBM File Manager for z/OS.

▶▶ CALL %FM *userID* *BACKGROUND* ;

---

## CALL %HOGAN

Starts Computer Sciences Corporation's KORE-HOGAN application, also known as SMART (System Memory Access Retrieval Tool).

▶▶ CALL %HOGAN ;

---

## CALL %VER

Adds a line to the log describing the maintenance level of z/OS Debugger that you have installed on your system.

▶▶ CALL %VER ;

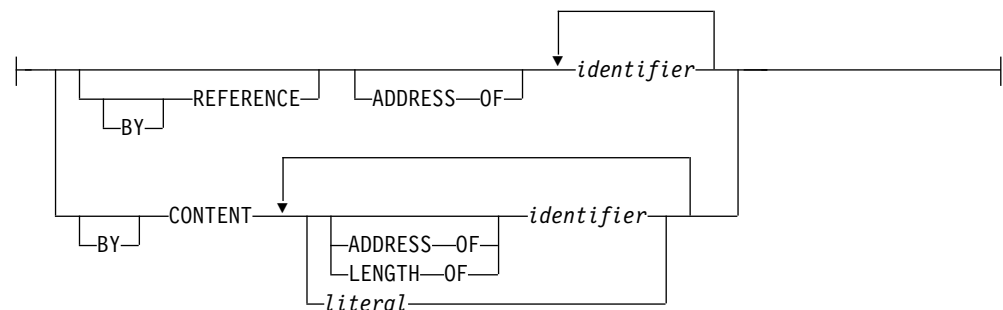
---

## CALL entry\_name (COBOL)

Calls an entry name in the application program.

▶▶ CALL *identifier* *literal* USING *identifier\_clause* ;

**identifier\_clause:**



---

## CALL procedure

Calls a procedure that has been defined with the PROCEDURE command.

▶▶ CALL *procedure\_name* ;

---

## CC command

Controls whether code coverage data is collected.

▶▶—CC—

START
STOP

—;————▶▶

---

## CHKSTGV

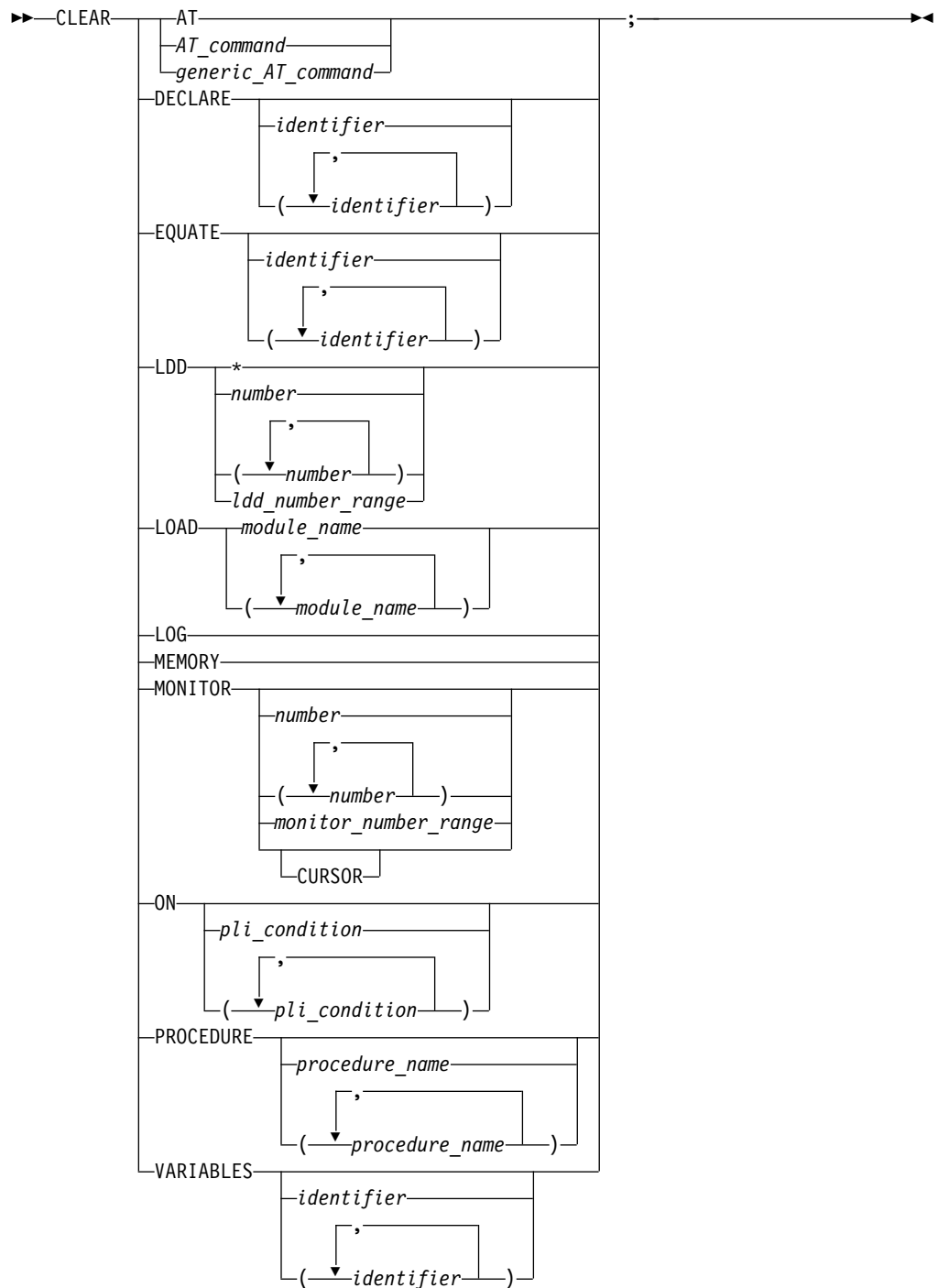
Check for a specific type of storage violation in the task you are currently debugging.

▶▶—CHKSTGV—;————▶▶

---

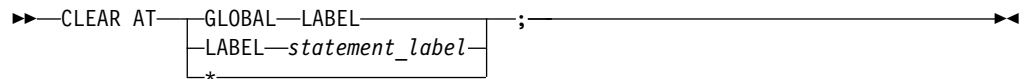
## CLEAR command

The CLEAR command removes the actions of previously issued z/OS Debugger commands.



## CLEAR AT (remote)

You can use the CLEAR AT command to remove actions that were completed by using the AT GLOBAL LABEL or the AT LABEL commands.



---

## CLEAR prefix (full-screen mode)

Clears a breakpoint when you issue this command through the Source window prefix area.

```
▶▶ CLEAR integer ;
```

---

## COMMENT command

The COMMENT command can be used to insert commentary in to the session log.

```
▶▶ COMMENT commentary ;
```

---

## COMPUTE command (COBOL)

The COMPUTE command assigns the value of an arithmetic expression to a specified reference.

```
▶▶ COMPUTE reference = expression ;
```

---

## CURSOR command (full-screen mode)

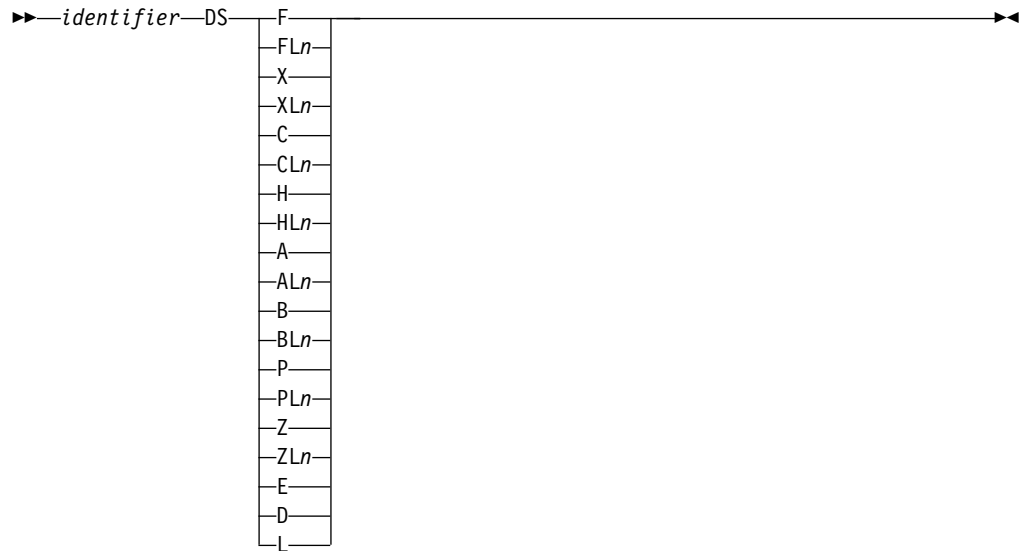
The CURSOR command moves the cursor between the last saved position on the z/OS Debugger session panel (excluding the header fields) and the command line.

```
▶▶ CURSOR ;
```

---

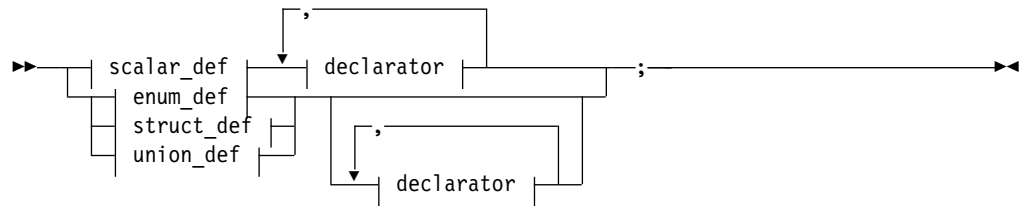
## Declarations (assembler, disassembly, and LangX COBOL)

Use declarations to create session variables and tags effective during a z/OS Debugger session.

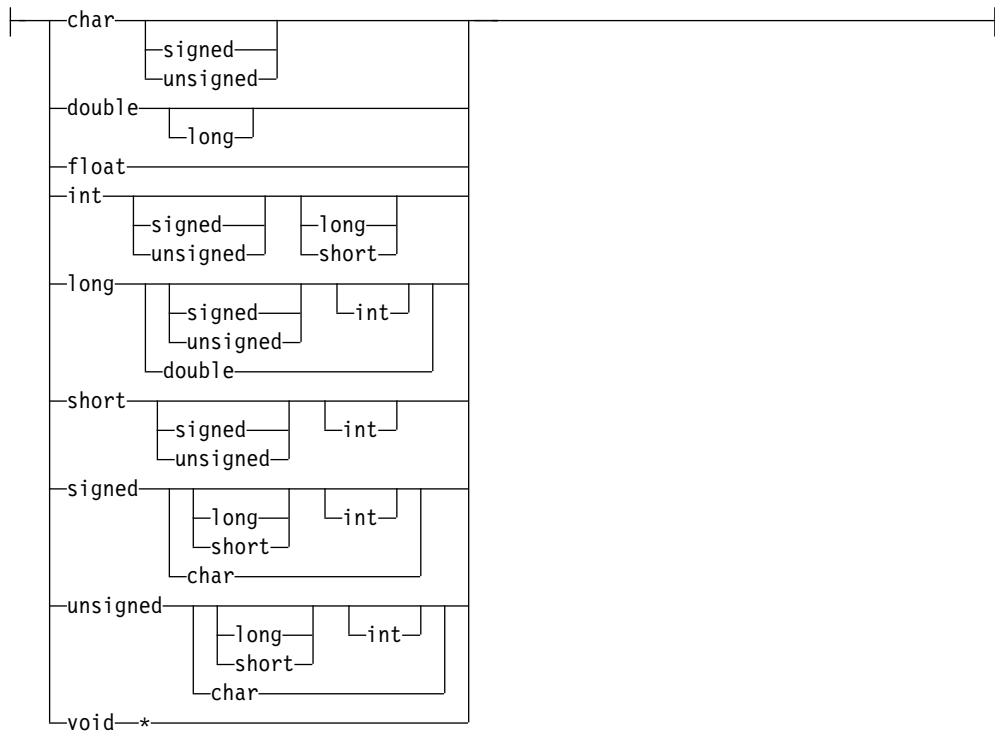


## Declarations (C and C++)

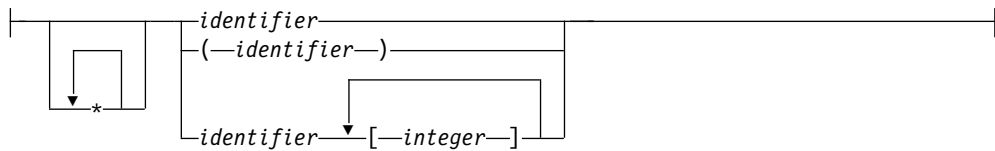
Use declarations to create session variables and tags effective during a z/OS Debugger session.



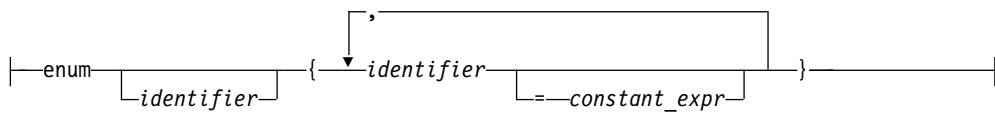
**scalar\_def:**



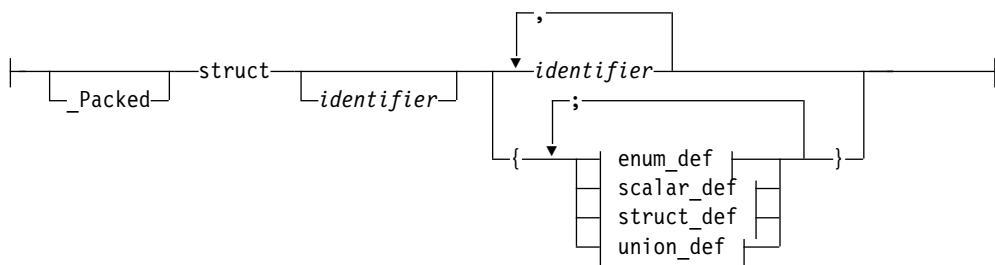
**declarator:**



**enum\_def:**

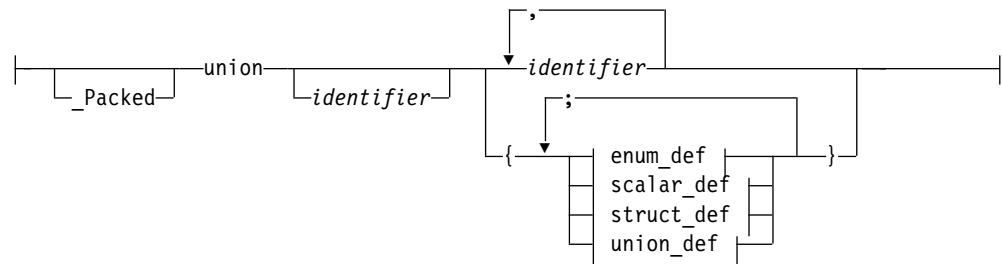


**struct\_def:**





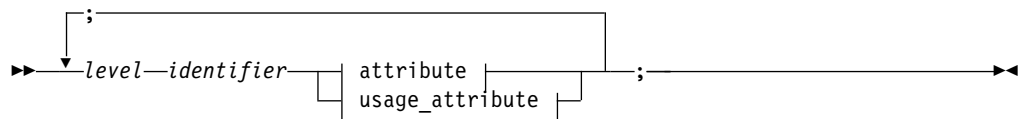
### union\_def:



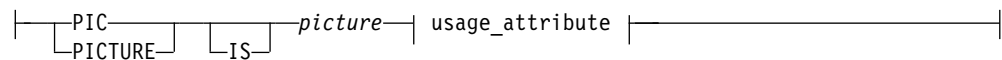
---

## Declarations (COBOL)

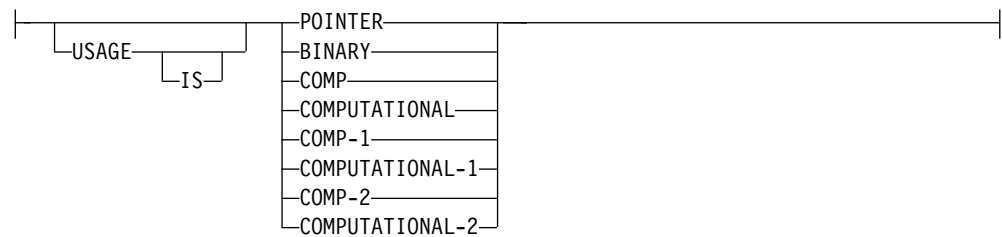
Use declarations to create session variables effective during a z/OS Debugger session.



### attribute:



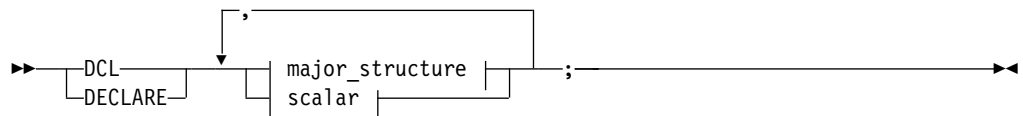
### usage\_attribute:



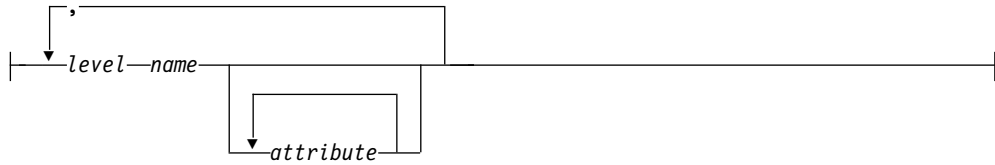
---

## DECLARE command (PL/I)

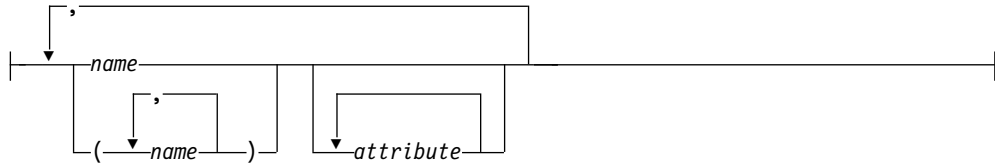
The DECLARE command creates session variables effective during a z/OS Debugger session.



### major\_structure:

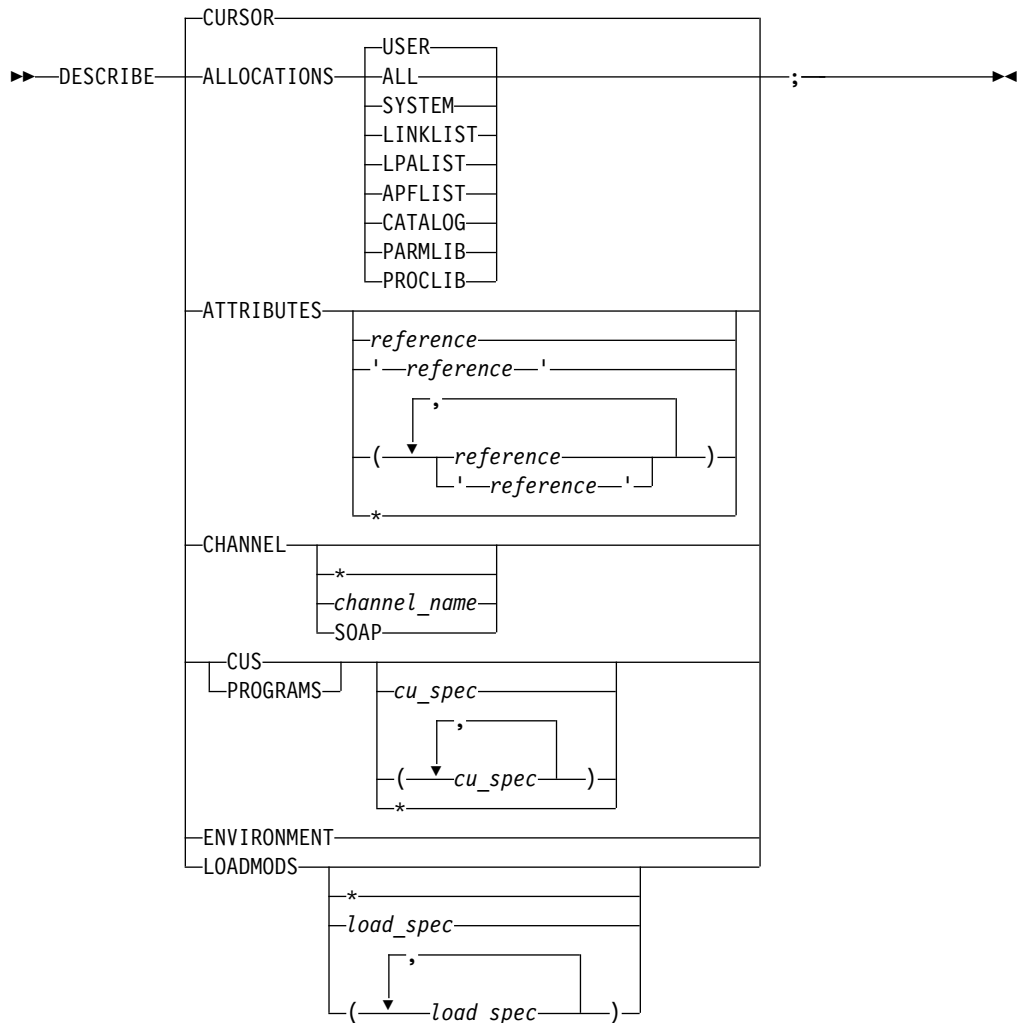


**scalar:**



## DESCRIBE command

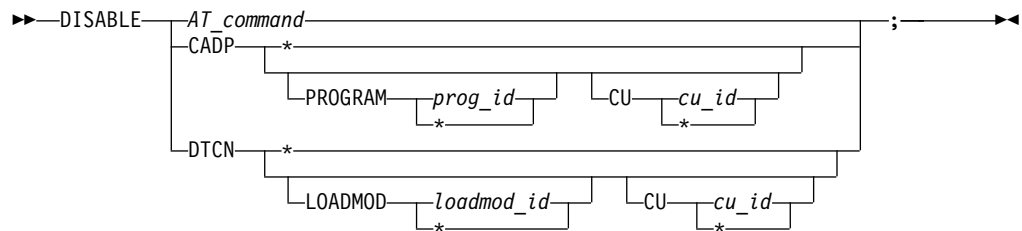
The DESCRIBE command displays the file allocations or attributes of references, compile units, known load modules, and the runtime environment.



---

## DISABLE command

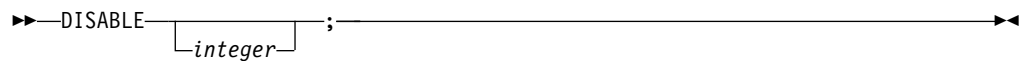
The DISABLE command makes an AT breakpoint inoperative or prevents z/OS Debugger from being started by CADP or DTCN. However, the breakpoint is not cleared. Later, you can make the breakpoint operative or allow z/OS Debugger to be started by CADP or DTCN by using the ENABLE command.



---

## DISABLE prefix (full-screen mode)

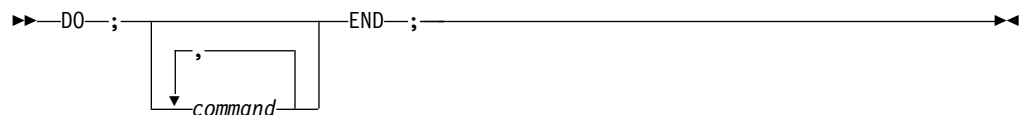
Disables a statement breakpoint or offset breakpoint when you issue this command through the Source window prefix area.



---

## DO command (assembler, disassembly, and LangX COBOL)

The DO command performs one or more commands that are collected into a group.



---

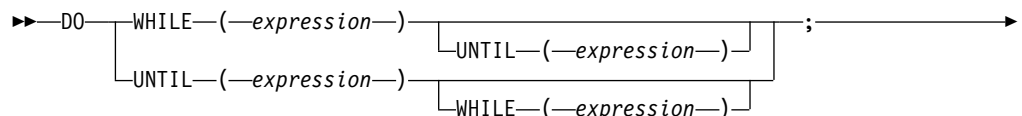
## DO command (PL/I)

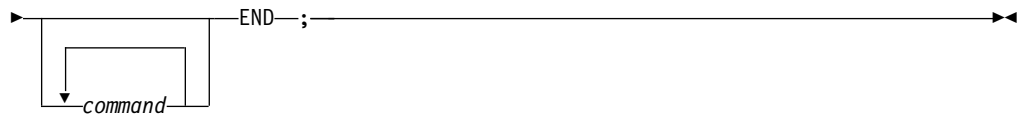
The DO command allows one or more commands to be collected into a group that can (optionally) be repeatedly executed.

### Simple

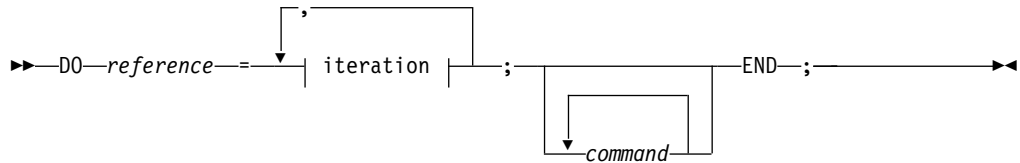


### Repeating

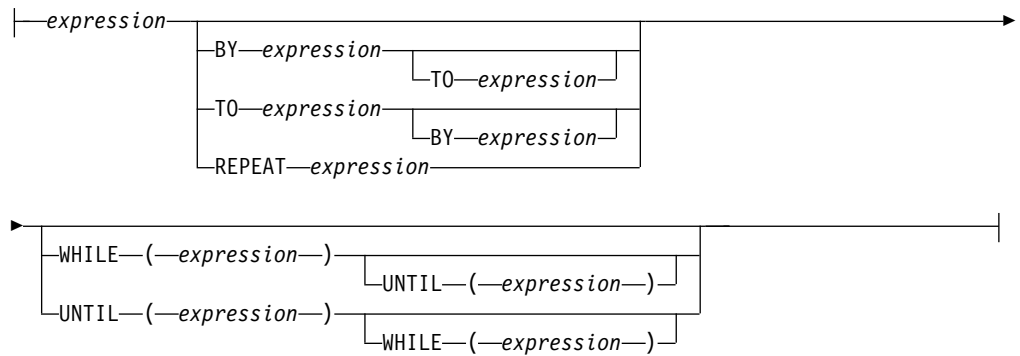




**Iterative**



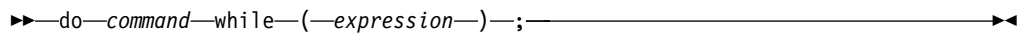
**iteration:**




---

**do/while command (C and C++)**

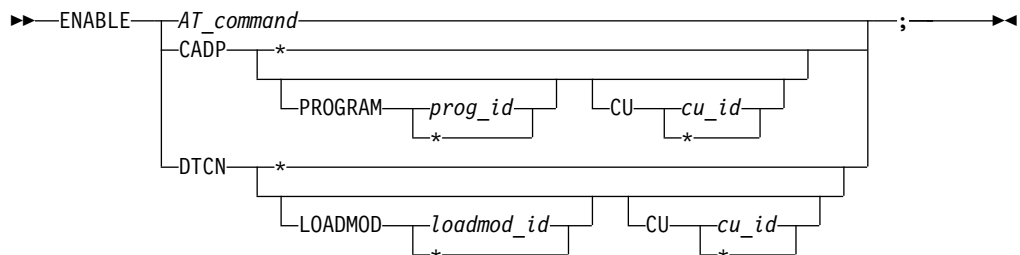
The do/while command performs a command before evaluating the test expression.




---

**ENABLE command**

The ENABLE command activates an AT or pattern match breakpoint after it was disabled.



---

## ENABLE prefix (full-screen mode)

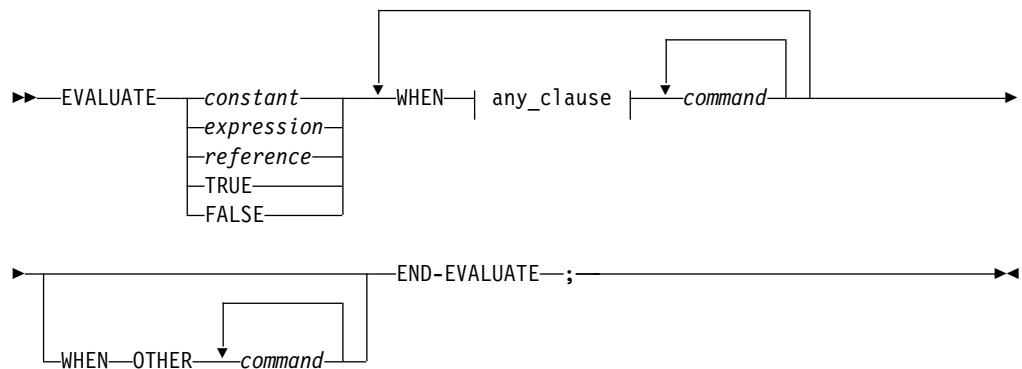
Enables a disabled statement breakpoint or a disabled offset breakpoint when you issue this command through the Source window prefix area.

►►—ENABLE—integer—;

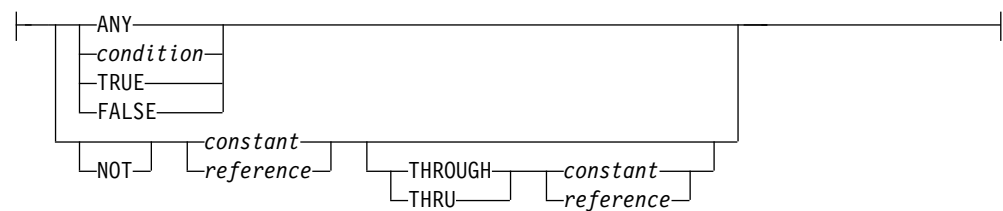
---

## EVALUATE command (COBOL)

The EVALUATE command provides a shorthand notation for a series of nested IF statements.



### any\_clause:



---

## Expression command (C and C++)

The Expression command evaluates the given expression.

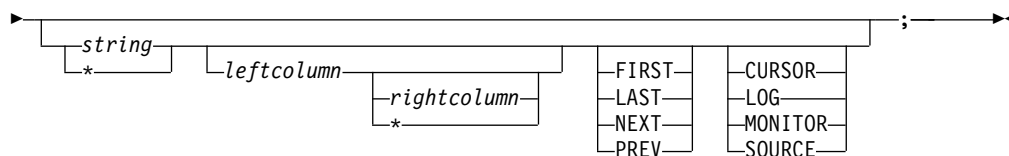
►►—expression—;

---

## FIND command

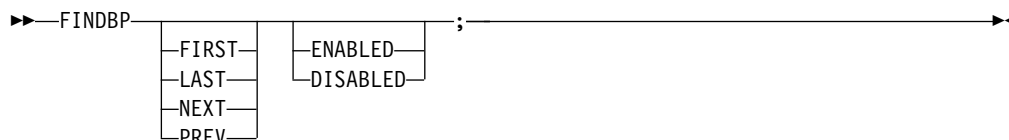
The FIND command helps you search through the Source window in full-screen and batch mode, and through the Log and Monitor windows in full-screen mode.

►►—FIND—



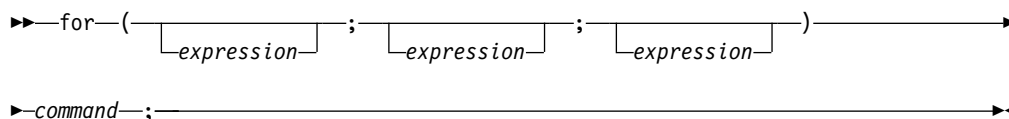
## FINDBP command

The FINDBP command provides full-screen search capability for line, statement and offset breakpoints in the source object.



## for command (C and C++)

The for command provides iterative looping similar to the C and C++ for statement.



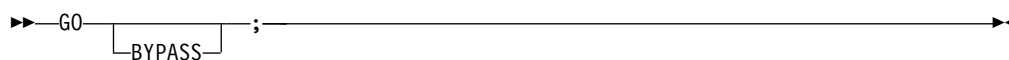
## FREE command

The FREE command releases a file that is currently allocated.



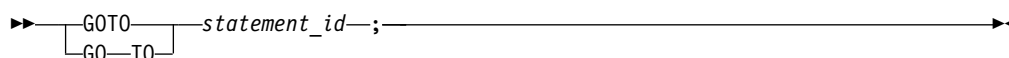
## GO command

The GO command causes z/OS Debugger to start or resume running your program.



## GOTO command

The GOTO command causes z/OS Debugger to resume program execution at the specified statement id.



---

## GOTO LABEL command

The GOTO LABEL command causes z/OS Debugger to resume program execution at the specified statement label. The specified label must be in the same block. If you want z/OS Debugger to return control to you at the target location, make sure there is a breakpoint at that location.

```
▶▶ GOTO LABEL 'statement_label';
```

---

## %IF command (programming language neutral)

The %IF command enables you write conditional statements that can be run in any supported programming language.

```
▶▶ %IF condition THEN command ELSE command;
```

---

## IF command (assembler, disassembly, and LangX COBOL)

The IF command lets you conditionally perform a command.

```
▶▶ IF condition THEN command ELSE command;
```

---

## if command (C and C++)

The if command lets you conditionally perform a command.

```
▶▶ if (expression) command else command;
```

---

## IF command (COBOL)

The IF command lets you conditionally perform a command.

```
▶▶ IF condition THEN command ELSE command END-IF;
```

---

## IF command (PL/I)

The IF command lets you conditionally perform a command.

```
▶▶ IF expression THEN command ELSE command;
```

---

## IMMEDIATE command (full-screen mode)

The IMMEDIATE command causes a command within a command list to be performed immediately.

▶▶ IMMEDIATE *command* ;

---

## INPUT command (C and C++ and COBOL)

The INPUT command provides data for an intercepted read and is valid only when there is a read pending for an intercepted file.

▶▶ INPUT *text* ;

---

## JUMPTO command

The JUMPTO command moves the resume point to the specified statement but does not resume the program.

▶▶ 

JUMPTO	<i>statement_id</i>	;
JUMP TO		

---

## JUMPTO LABEL command

The JUMPTO command moves the resume point to the specified label but does not resume the program.

▶▶ 

JUMPTO		<i>statement_label</i>	;
JUMP TO	LABEL	' <i>statement_label</i> '	

---

## LIST (blank)

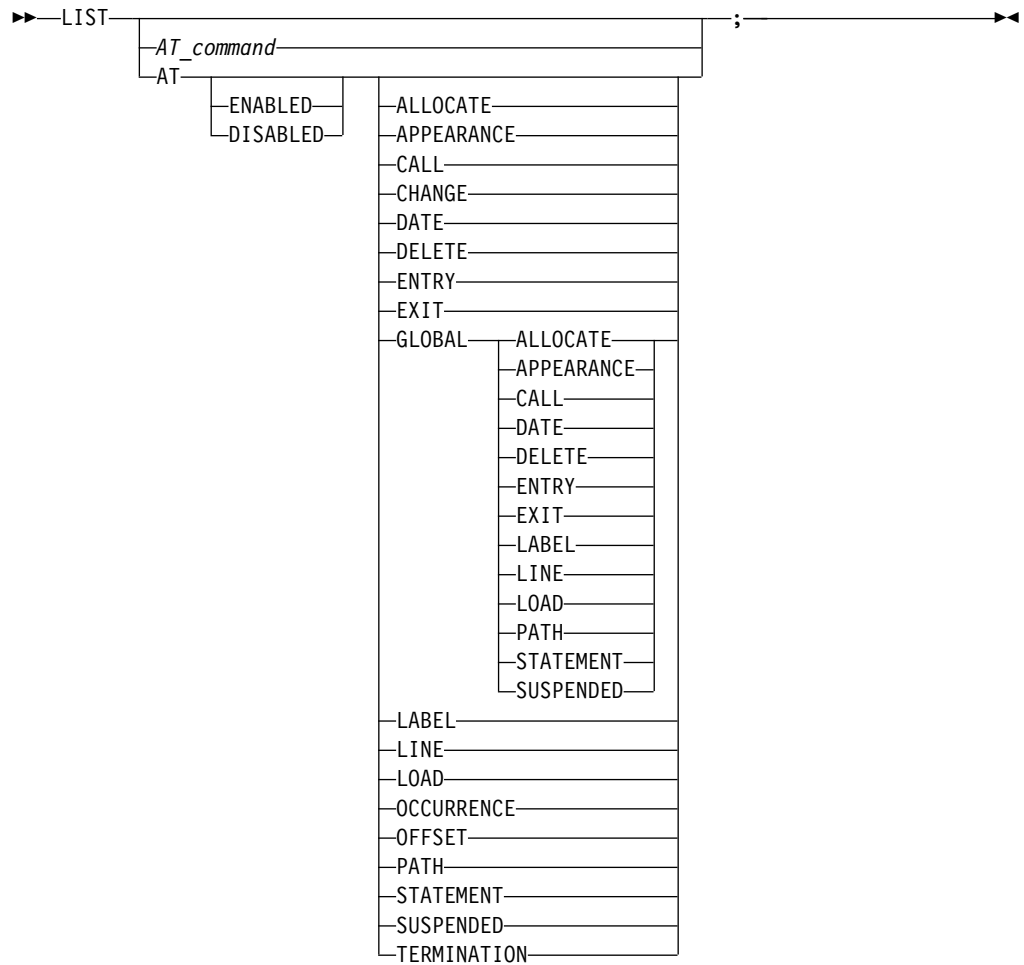
Displays the Source Identification panel, where associations are made between source listings or source files shown in the source window and their program units.

---

## LIST AT

Lists the currently defined breakpoints, including the action taken when the specified breakpoint is activated.





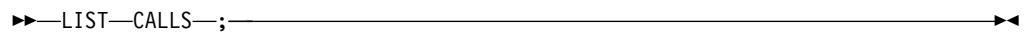
## LIST AT (remote)

Lists the currently defined AT GLOBAL LABEL or AT LABEL breakpoints.

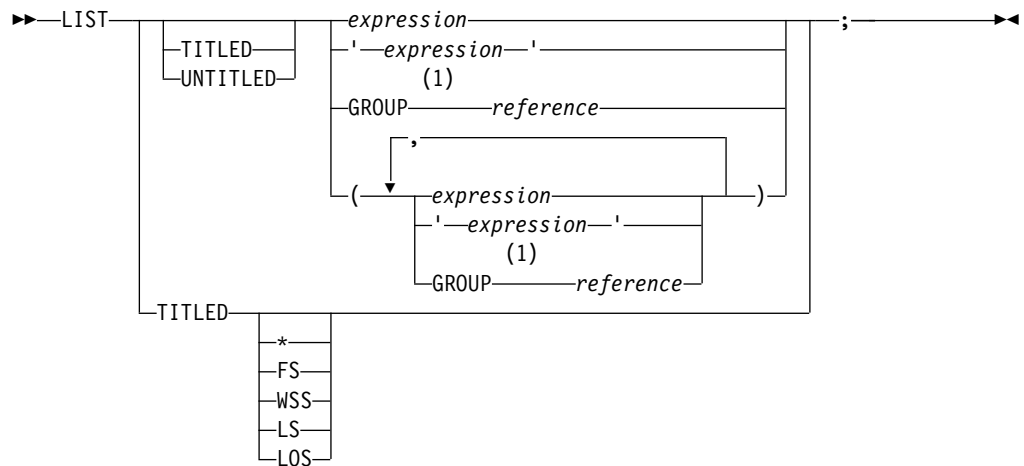


## LIST CALLS

Displays the dynamic chain of active blocks.







**Notes:**

- 1 Only for COBOL.

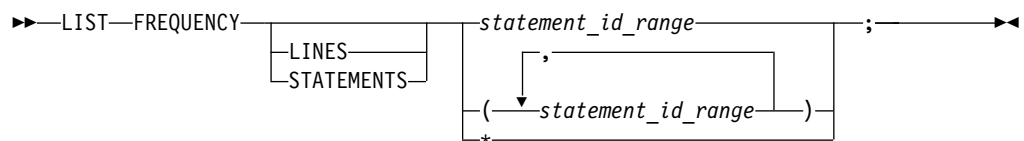
## L expression prefix

Entered through the prefix area of the Source window, displays the value of a variable or variables on that line.



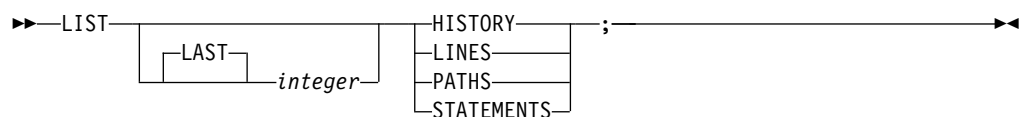
## LIST FREQUENCY

Lists statement execution counts.



## LIST LAST

Displays a list of recent entries in the history table.



## LIST LINE NUMBERS

Equivalent to LIST STATEMENT NUMBERS.

---

## LIST LINES

Equivalent to LIST STATEMENTS.

---

## LIST MONITOR

Lists all or selected members of the current set of MONITOR commands.

▶▶—LIST—MONITOR—integer—- —integer—;—▶▶

---

## LIST NAMES

Lists the names of variables, programs, or z/OS Debugger procedures.

▶▶—LIST—NAMES—pattern—

BLOCK	<u>block_spec</u>		
	<u>cu_spec</u>		
	<u>,</u>		
	<u>(</u>	<u>block_spec</u>	<u>)</u>
		<u>cu_spec</u>	
CUS			
LABELS			
PROCEDURES			
PROGRAMS			
TEST			

—;—▶▶

---

## LIST NAMES LABELS (remote)

Displays the names of all section and paragraph names in a COBOL program, and the names of all instruction labels in an assembler program.

▶▶—LIST—NAMES—LABELS—;—▶▶

---

## LIST ON (PL/I)

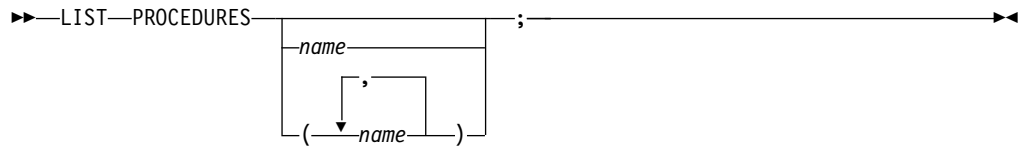
Lists the action (if any) currently defined for the specified PL/I conditions.

▶▶—LIST—ON—pli\_condition—;—▶▶

---

## LIST PROCEDURES

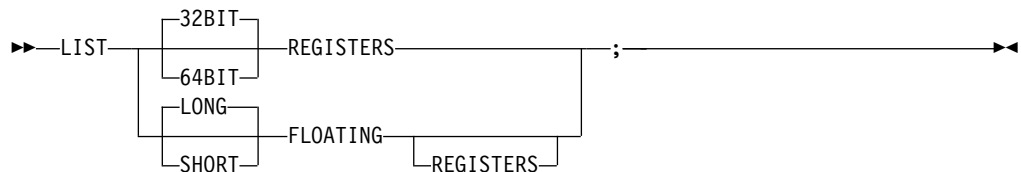
Lists the commands contained in the specified z/OS Debugger PROCEDURE definitions.



---

## LIST REGISTERS

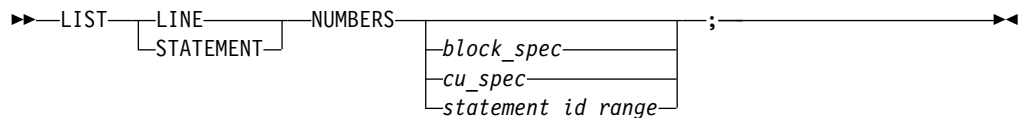
Displays the current register contents.



---

## LIST STATEMENT NUMBERS

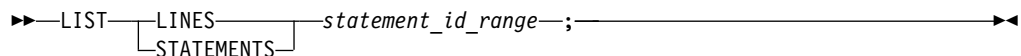
Lists all statement or line numbers that are valid locations for an AT LINE or AT STATEMENT breakpoint.



---

## LIST STATEMENTS

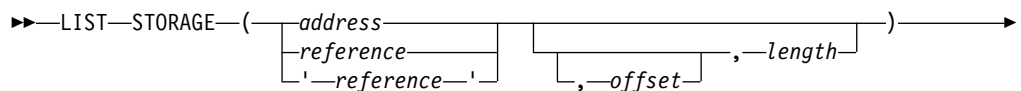
Lists one or more statements or lines from a file.

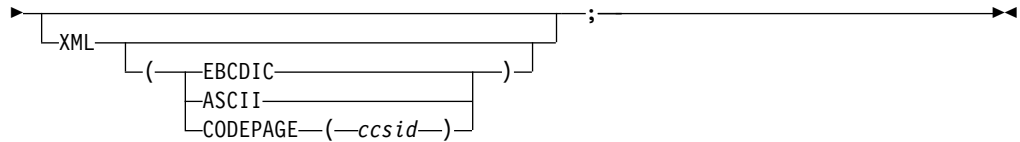


---

## LIST STORAGE

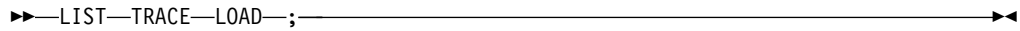
Displays the contents of storage at a particular address in hex format.





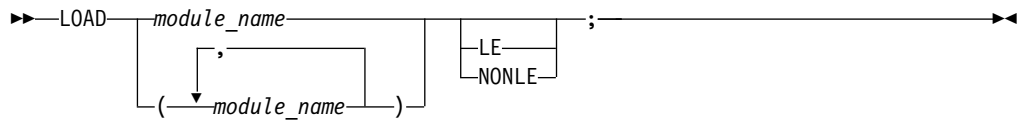
## LIST TRACE LOAD command

Displays the entries in the TRACE LOAD table that is created since the TRACE LOAD START command was issued.



## LOAD command

Specifies to load the named module using MVS™ LOAD services, EXEC CICS LOAD, or the Language Environment enclave-level load service for debugging purposes.

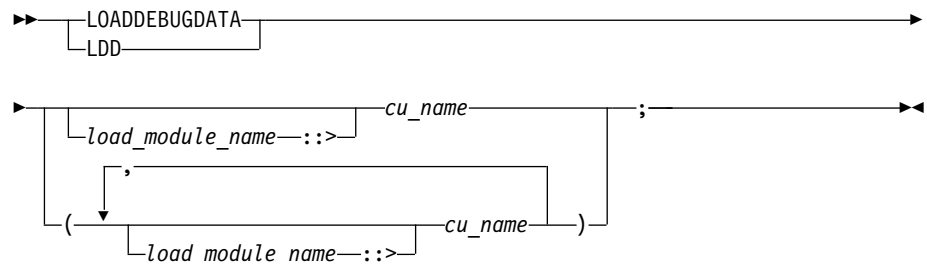


## LOADDEBUGDATA (LDD)

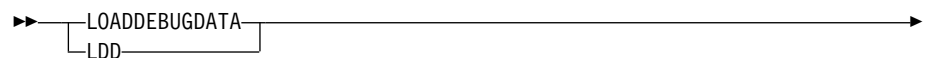
The LOADDEBUGDATA (LDD) command requests that z/OS Debugger load the debug data for a compile unit in either of the following situations:

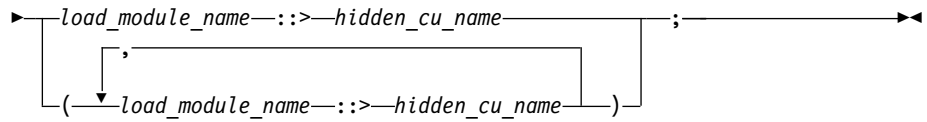
- The CU was written in assembler or LangX COBOL.
- Explicit debug mode is active and the CU was written in a Language Environment enabled, high-level language and compiled with the TEST or DEBUG compiler option.

### LDD for assembler or LangX COBOL CU



### LDD for high-level language CU in explicit debug mode





## MEMORY

Identifies an address in memory and then display the contents of memory at that location in the Memory window. The Memory window displays memory in dump format.



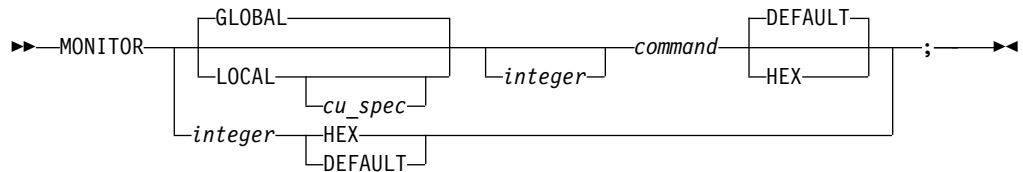
## M prefix command

The M prefix command, which you enter through the prefix area of the Source window, adds variables on that line to the Monitor window.



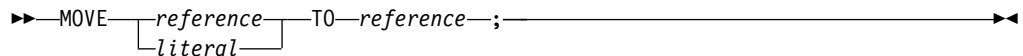
## MONITOR command

The MONITOR command defines or redefines a command whose output is displayed in the monitor window (full-screen mode), terminal output (line mode), or log file (batch mode).



## MOVE command (COBOL)

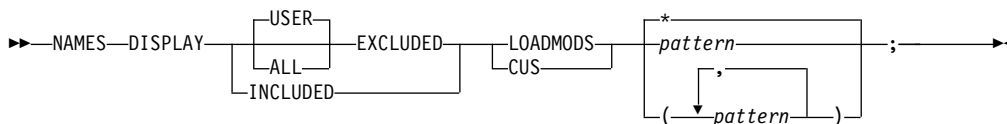
The MOVE command transfers data from one area of storage to another.



---

## NAMES DISPLAY command

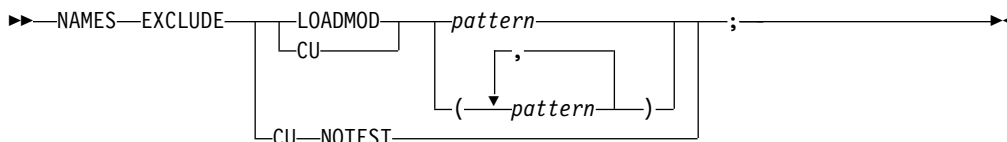
Use the NAMES DISPLAY command to indicate that you want a list of all the load modules or compile units that are currently excluded or included. If you do not specify the ALL parameter, only the names excluded by user commands appear in the list that is displayed. Names that z/OS Debugger excludes by default are not included in the list that is displayed.



---

## NAMES EXCLUDE command

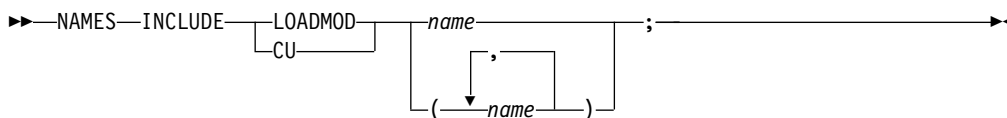
The NAMES EXCLUDE command enables you to indicate to z/OS Debugger the names of load modules or compile units that you do not need to debug. If these are data-only modules, z/OS Debugger does not process them. If they contain executable code, z/OS Debugger might process them in some cases. See "Optimizing the debugging of large applications" in the IBM z/OS Debugger User's Guide for more information about these situations.



---

## NAMES INCLUDE command

Use the NAMES INCLUDE command to indicate to z/OS Debugger that your program is a user load module or compile unit, not a system program. See "Debugging user programs that use system prefix names" in the IBM z/OS Debugger User's Guide for more information.



---

## Null command

The Null command is a semicolon written where a command is expected.

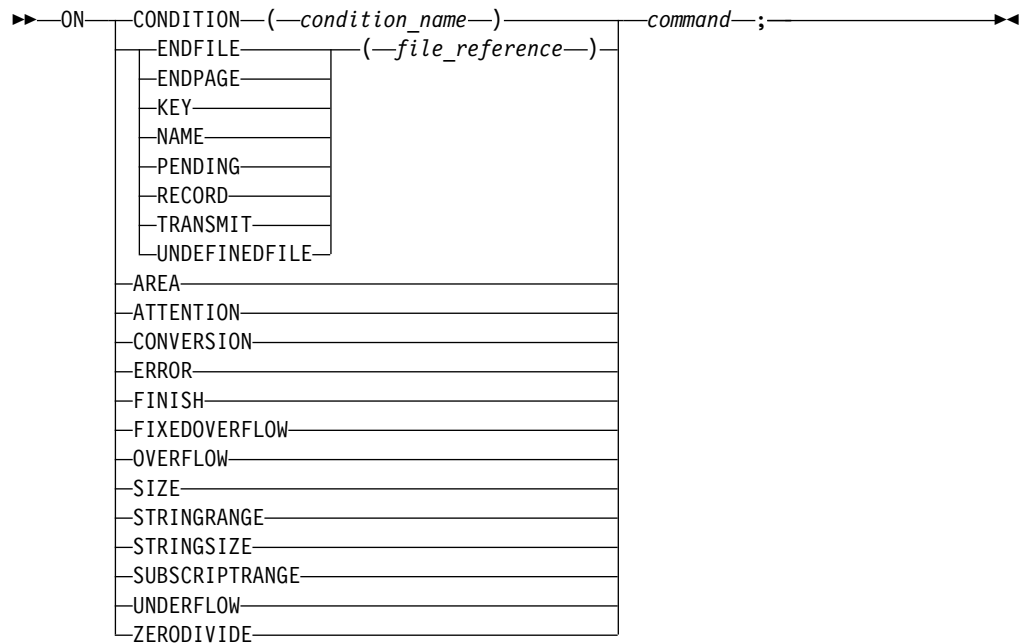




---

## ON command (PL/I)

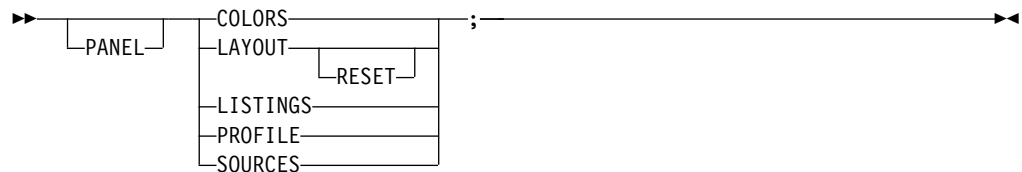
The ON command establishes the actions to be executed when the specified PL/I condition is raised.



---

## PANEL command (full-screen mode)

The PANEL command displays special panels.

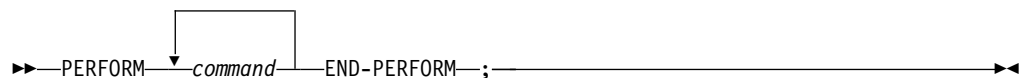


---

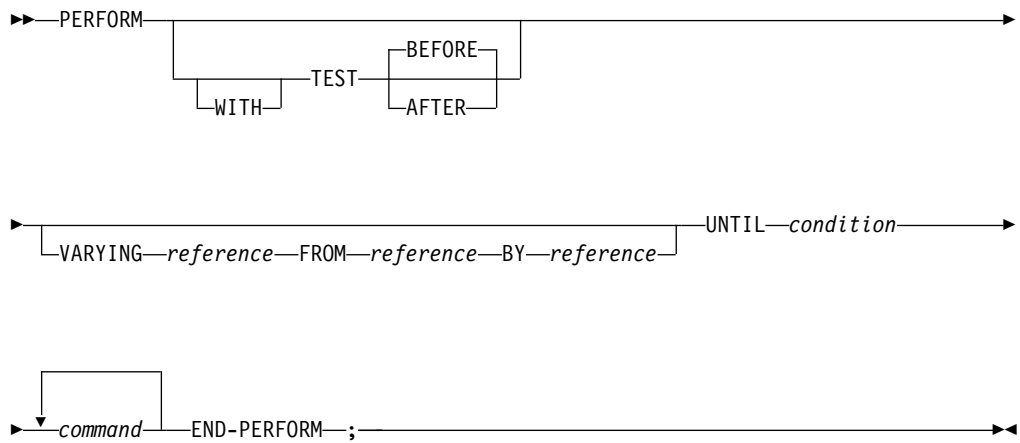
## PERFORM command (COBOL)

The PERFORM command transfers control explicitly to one or more statements and implicitly returns control to the next executable statement after execution of the specified statements is completed.

**Simple:**



**Repeating:**



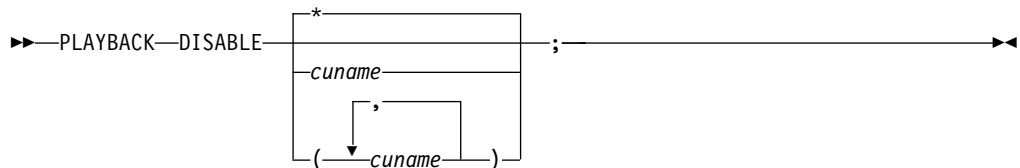
## PLAYBACK BACKWARD command

The **PLAYBACK BACKWARD** command indicates to z/OS Debugger to perform **STEP** and **RUNTO** commands backward, starting from the current point and going to previous points.



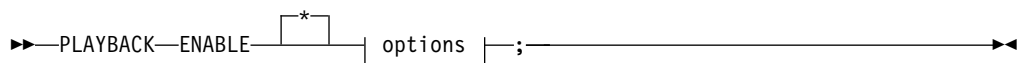
## PLAYBACK DISABLE command

The **PLAYBACK DISABLE** command informs z/OS Debugger to stop recording runtime environment information and discard any information recorded thus far.

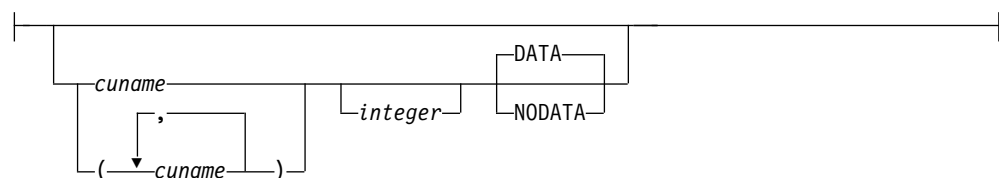


## PLAYBACK ENABLE command

The **PLAYBACK ENABLE** command informs z/OS Debugger to begin recording the application runtime environment information (steps history and data history).



### options:



---

## PLAYBACK FORWARD command

The PLAYBACK FORWARD command indicates to z/OS Debugger to perform STEP and RUNTO commands forward, starting from the current point and going to the next point.

▶▶—PLAYBACK—FORWARD—;—————▶▶

---

## PLAYBACK START command

The PLAYBACK START command suspends normal debugging and informs z/OS Debugger to replay the steps it recorded.

▶▶—PLAYBACK—START—;—————▶▶

---

## PLAYBACK STOP command

The PLAYBACK STOP command stops replaying recorded statements and resumes normal debugging at the point where the PLAYBACK START command was entered.

▶▶—PLAYBACK—STOP—;—————▶▶

---

## POPUP command

Displays the Command pop-up window, where you can type in multiline commands.

▶▶—POPUP—integer—;—————▶▶

---

## POSITION command

Positions the cursor to a specific line in the specified window.

▶▶—POSITION—*integer*—CURSOR  
LOG  
MONITOR  
SOURCE—;—————▶▶

---

## Prefix commands (full-screen mode)

The Prefix commands apply only to source listing lines and are typed into the prefix area in the source window. For details, see the section corresponding to the command name.

The following table summarizes the forms of the Prefix commands.

"AT Prefix (full-screen mode)" on page 7	Defines a statement breakpoint through the Source window prefix area.
--	---

"CLEAR prefix (full-screen mode)" on page 12	Clears a breakpoint through the Source window prefix area.
"DISABLE prefix (full-screen mode)" on page 17	Disables a breakpoint through the Source window prefix area.
"ENABLE prefix (full-screen mode)" on page 19	Enables a disabled breakpoint through the Source window prefix area.
"LIST expression" on page 24	Displays the value of a variable or variables on that line.
"QUERY prefix (full-screen mode)" on page 36	Queries what statements have breakpoints through the Source window prefix area.
"RUNTO prefix command (full-screen mode)" on page 38	Runs the program to the location that the cursor or statement identifier indicate in the Source window prefix area.
"SHOW prefix command (full-screen mode)" on page 50	Specifies what relative statement or verb within the line is to have its frequency count shown in the suffix area.

---

## PROCEDURE command

The PROCEDURE command allows the definition of a group of commands that can be accessed by using the CALL procedure command.

```

▶▶ name : PROCEDURE ; command END ; ▶▶

```

---

## QUALIFY RESET

This command is equivalent to SET QUALIFY RESET.

---

## QUERY command

The QUERY command displays the current value of the specified z/OS Debugger setting, the current setting of all the z/OS Debugger settings, or the current location in the suspended program.

```

▶▶ QUERY ▶▶
▶| Attributes A through I | Attributes J through P | Attributes Q through Z |; ▶▶

```

**Attributes A through I:**

ASSEMBLER
AUTOMONITOR
(1)
BROWSE MODE
CHANGE
COLORS
COUNTRY
(1)
CURRENT VIEW
DBCS
(1)
DEFAULT DBG
(1)
DEFAULT LISTINGS
(1)
DEFAULT MDBG
DEFAULT SCROLL
(1)
DEFAULT VIEW
DEFAULT WINDOW
DISASSEMBLY
DYNDEBUG
ECHO
(1)
EQAOPTS
EQUATES
EXECUTE
(1)
EXPLICITDEBUG
FIND BOUNDS
FREQUENCY
HISTORY
(1)
IGNORELINK
(1)
INTERCEPT

**Attributes J through P:**

KEYS
LDD
LIST—TABULAR
LOCATION
LOG
LOG—NUMBERS
LONGCUNAME
MDBG
MONITOR
COLUMN
DATATYPE
LIMIT
NUMBERS
WRAP
MSGID
LANGUAGE
NATIONAL
PACE
PFKEYS
PLAYBACK
PLAYBACK—LOCATION
POPUP
PROGRAMMING—LANGUAGE
PROMPT

#### Attributes Q through Z:

QUALIFY
REFRESH
RESTORE
(1)
REWRITE
SAVE
SCREEN
SCROLL—DISPLAY
(2)
SEQUENCE
SETS
SOURCE
SUFFIX
TEST
WARNING
WINDOW—SIZES

#### Notes:

- 1 You can use this command in remote debug mode.
- 2 Only for PL/I.

---

## QUERY prefix (full-screen mode)

Queries what statements on a particular line have statement breakpoints when you issue this command through the Source window prefix area.

►►—QUERY—;—◄◄

---

## QUIT command

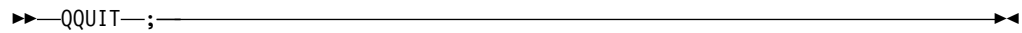
The QUIT command ends a z/OS Debugger session and if an expression is specified, sets the return code.



---

## QQUIT command

The QQUIT command ends a z/OS Debugger session without further prompting.



---

## RESTORE command

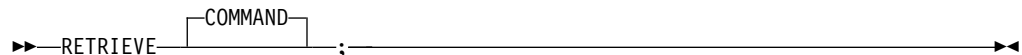
The RESTORE command enables you to explicitly restore the settings, breakpoints, and monitor specifications that were previously saved by the SET SAVE AUTO command when z/OS Debugger terminated.



---

## RETRIEVE command (full-screen mode)

The RETRIEVE command displays the last command entered on the command line.



---

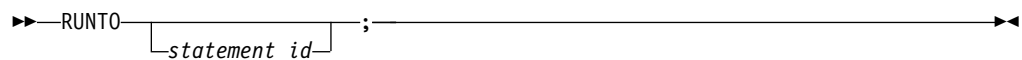
## RUN command

The RUN command is synonymous to the GO command.

---

## RUNTO command

The RUNTO command runs your program to a valid executable statement without setting a breakpoint.



---

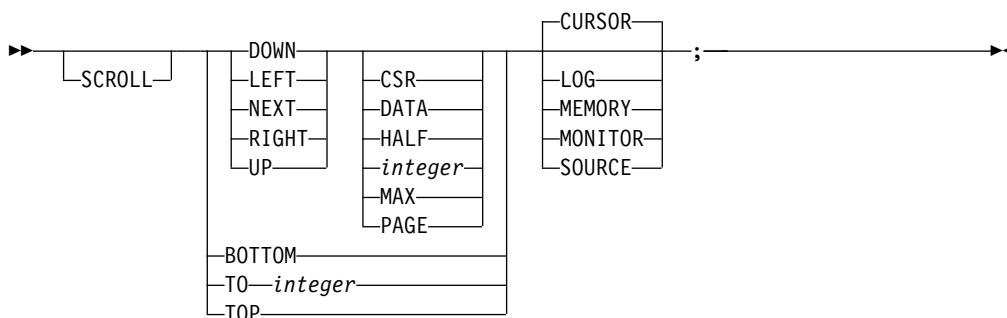
## RUNTO prefix command (full-screen mode)

Runs to the statement when you issue this command through the Source window prefix area.

---

## SCROLL command (full-screen mode)

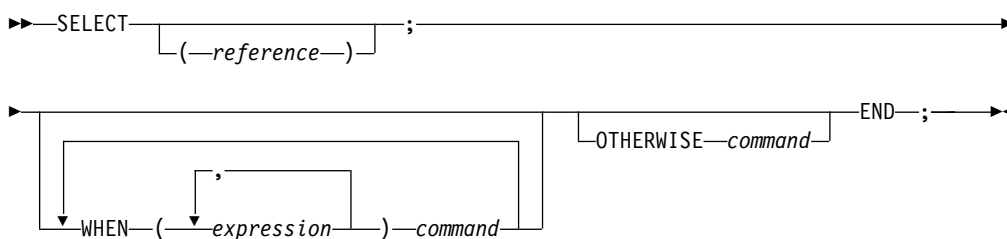
The SCROLL command provides horizontal and vertical scrolling in full-screen mode.



---

## SELECT command (PL/I)

The SELECT command chooses one of a set of alternate commands.



---

## SET ASSEMBLER ON/OFF

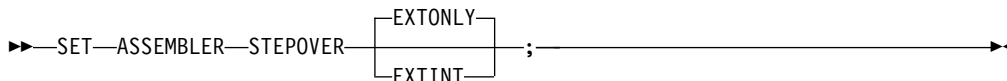
The SET ASSEMBLER ON/OFF command displays additional information that is useful when you debug an assembler program.



---

## SET ASSEMBLER STEPOVER

The SET ASSEMBLER STEPOVER command specifies how z/OS Debugger processes STEP OVER commands in assembler compile units.

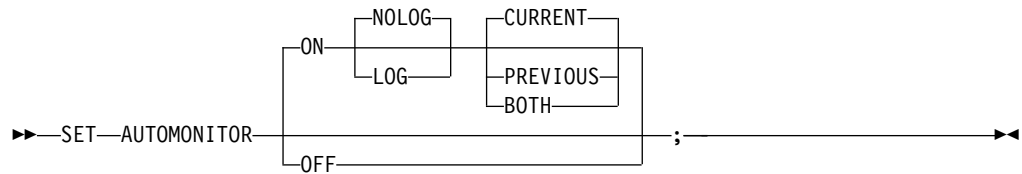




---

## SET AUTOMONITOR

Controls the monitoring of data items at the statement that z/OS Debugger runs next, ran most recently, or both.



---

## SET CHANGE

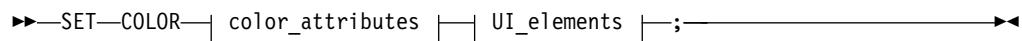
Controls the frequency of checking the AT CHANGE breakpoints.



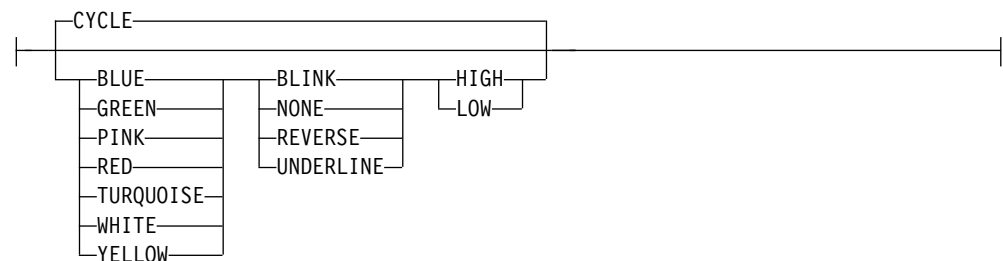
---

## SET COLOR (full-screen and line mode)

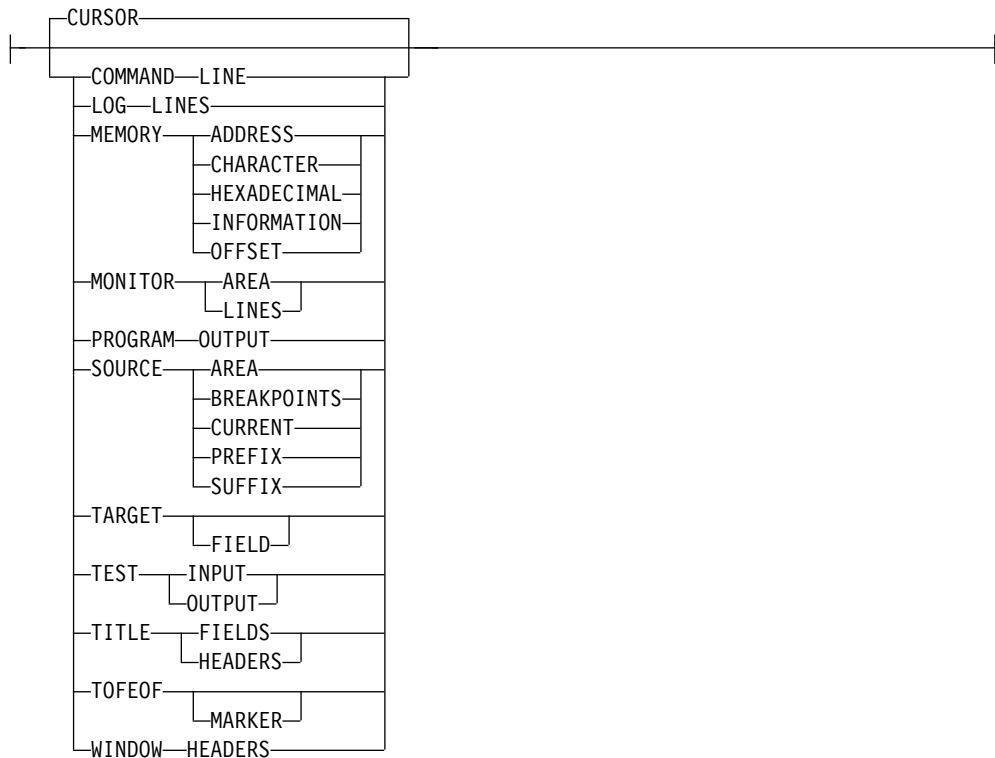
Provides control of the color, highlighting, and intensity attributes when the SCREEN setting is ON.



### color\_attributes:



### UI\_elements:



## SET COUNTRY

Changes the current national country setting for the application program.

►► SET COUNTRY *country\_code* ;

## SET DBCS

Controls whether shift-in and shift-out codes are interpreted on input and supplied on DBCS output.

►► SET DBCS  ON  OFF ;

## SET DEFAULT DBG

Defines a default partitioned data set DD name or DS name that z/OS Debugger searches through to locate the .dbg files.

►► SET DEFAULT DBG 

<i>ddname</i>
<i>dsn</i>

 ;

( *dsn* )



---

## SET DEFAULT WINDOW (full-screen mode)

Specifies what window is selected when a window referencing command (for example, FIND, SCROLL, or WINDOW) is issued without explicit window identification and the cursor is outside the window areas.

▶▶ SET DEFAULT WINDOW 

LOG
MEMORY
MONITOR
SOURCE

 ;

---

## SET DISASSEMBLY

Controls whether the disassembly view is displayed in the Source window.

▶▶ SET DISASSEMBLY 

ON
OFF

 ;

---

## SET DYNDEBUG

Controls whether to activate the Dynamic Debug facility.

▶▶ SET DYNDEBUG 

ON
OFF

 ;

---

## SET ECHO

Controls whether GO and STEP commands are recorded in the log window when they are not subcommands.

▶▶ SET ECHO 

ON
OFF

*
keyword

 ;

---

## SET EQUATE

Equates a symbol to a string of characters.

▶▶ SET EQUATE *identifier* = *string* ;

---

## SET EXECUTE

Controls whether commands from all input sources are performed or just syntax checked (primarily for checking USE files).

▶▶ SET EXECUTE 

ON
OFF

 ;

---

---

## SET EXPLICITDEBUG

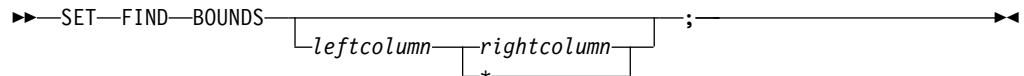
Controls whether explicit debug mode is active.



---

## SET FIND BOUNDS

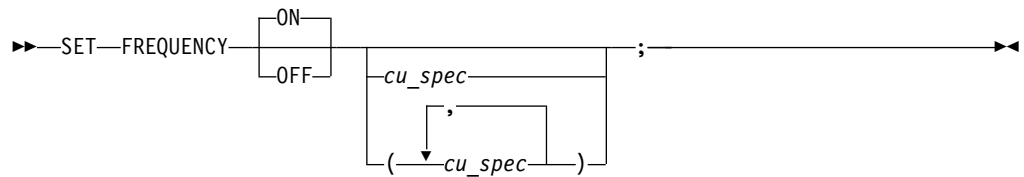
Specifies the default left and right columns for a FIND command in the Source window and in line mode that does not specify any columns information.



---

## SET FREQUENCY

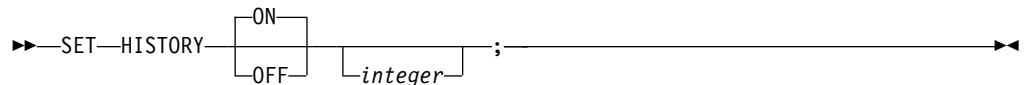
Controls whether statement executions are counted.



---

## SET HISTORY

Specifies whether entries to z/OS Debugger are recorded in the history table and optionally adjusts the size of the table.



---

## SET IGNORELINK

Specifies that any new LINK level (nested enclave) is ignored while the setting is ON.



---

## SET INTERCEPT (C and C++)

Intercepts input to and output from specified files.

▶▶ SET INTERCEPT  ON  OFF FILE *file\_spec* ; ▶▶

---

## SET INTERCEPT (COBOL, full-screen mode, line mode, batch mode)

Intercepts input to and output from the console.

▶▶ SET INTERCEPT  ON  OFF CONSOLE ; ▶▶

---

## SET INTERCEPT (COBOL, remote debug mode)

Intercepts output from COBOL DISPLAY statements.

▶▶ SET INTERCEPT  ON  OFF ; ▶▶

---

## SET KEYS (full-screen and line mode)

Controls whether PF key definitions are displayed when the SCREEN setting is ON.

▶▶ SET KEYS  ON  OFF  12  24 ; ▶▶

---

## SET LDD

Controls how debug data is loaded for assemblies containing multiple CSECTs.

▶▶ SET LDD  SINGLE  ALL ; ▶▶

---

## SET LIST BY SUBSCRIPT command (COBOL)

Controls whether z/OS Debugger displays elements in an array as they are stored in memory.

▶▶ SET LIST BY SUBSCRIPT  ON  OFF ; ▶▶

---

## SET LIST BY SUBSCRIPT command (Enterprise PL/I, full-screen mode only)

Controls whether z/OS Debugger displays elements in an array as they are stored in memory.

▶▶ SET LIST BY SUBSCRIPT  ON  OFF ;

---

## SET LIST TABULAR

Controls whether to format the output of the LIST command in a tabular format.

▶▶ SET LIST TABULAR  OFF  ON ;

---

## SET LOG

Controls whether each command that z/OS Debugger runs and the output of that command is stored in the log file. Defines (or redefines) the name and location of the file and whether the information is appended to an existing file or is written over existing information.

▶▶ SET LOG  ON  OFF ;  
ON FILE *fileid*  OLD  MOD  
KEEP *count*

---

## SET LOG NUMBERS (full-screen and line mode)

Controls whether line numbers are shown in the log window.

▶▶ SET LOG NUMBERS  ON  OFF ;

---

## SET LONGCUNAME (C, C++, and PL/I)

Controls whether the CU name is displayed in short or long format.

▶▶ SET LONGCUNAME  ON  OFF ;

---

## SET MDBG (C, C++)

Associates a .mdbg file to one load module or DLL.

▶▶ SET—MDBG—(*lm\_spec*)—*fileid*—; ▶▶

---

## SET MONITOR (full-screen and line mode)

Controls the format and layout of variable names and values displayed in the Monitor window.

▶▶ SET—MONITOR—

COLUMN	}	}
DATATYPE		
NUMBERS		
WRAP		
LIMIT— <i>integer</i>		ON
		OFF

—; ▶▶

---

## SET MSGID

Controls whether the z/OS Debugger messages are displayed with the message prefix identifiers.

▶▶ SET—MSGID—

ON
OFF

—; ▶▶

---

## SET NATIONAL LANGUAGE

Switches your application to a different runtime national language that determines what translation is used when a message is displayed.

▶▶ SET—

NATIONAL
----------

—LANGUAGE—*language\_code*—; ▶▶

---

## SET PACE

Specifies the maximum speed (in steps per second) of animated execution.

▶▶ SET—PACE—*number*—; ▶▶

---

## SET PFKEY

Associates a z/OS Debugger command with a Program Function key (PF key).

▶▶ SET—PF*n*—

<i>string</i>
---------------

—=

<i>command</i>
----------------

—; ▶▶

---



---

## SET POPUP

Controls the number of lines displayed in the Command pop-up window.

►► SET POPUP *integer* ;

---

## SET PROGRAMMING LANGUAGE

Sets the current programming language.

►► SET PROGRAMMING LANGUAGE

CYCLE
AUTOMATIC
HOLD
ASSEMBLER
C
COBOL
DISASSEMBLY
LANGXCOBOL
PLI
HOLD

;

---

## SET PROMPT (full-screen and line mode)

Controls whether the current program location is automatically shown as part of the prompt message in line mode.

►► SET PROMPT

LONG
SHORT

;

---

## SET QUALIFY

Simplifies the identification of references and statement numbers by resetting the point of view to a new block, compile unit, or load module.

►► SET QUALIFY

BLOCK	<i>block_spec</i>
CU	<i>cu_spec</i>
PROGRAM	<i>address</i>
LOAD	<i>load_spec</i>
RESET	
RETURN	
UP	

;

---

## SET REFRESH (full-screen mode)

Controls screen refreshing.

►► SET REFRESH

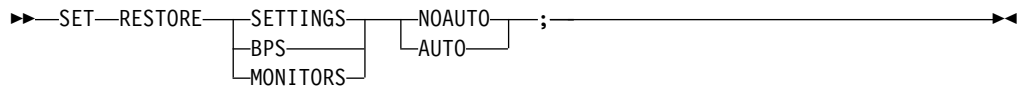
ON
OFF

;

---

## SET RESTORE

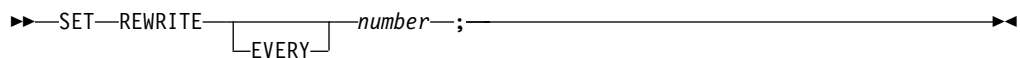
Controls the restoring of settings, breakpoints, and monitor specifications.



---

## SET REWRITE (full-screen mode)

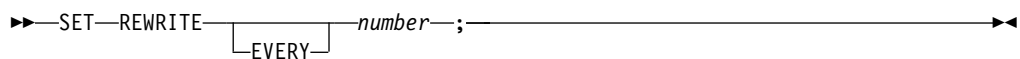
Forces a periodic screen rewrite during long sequences of output.



---

## SET REWRITE (remote debug mode)

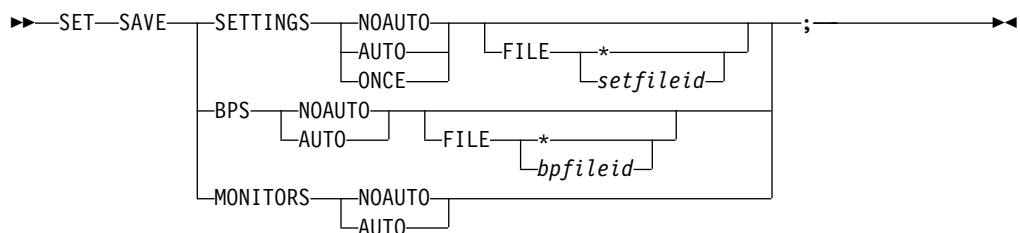
Sets the maximum number of COBOL DISPLAY statements that the remote debugger displays in the Debug Console.



---

## SET SAVE

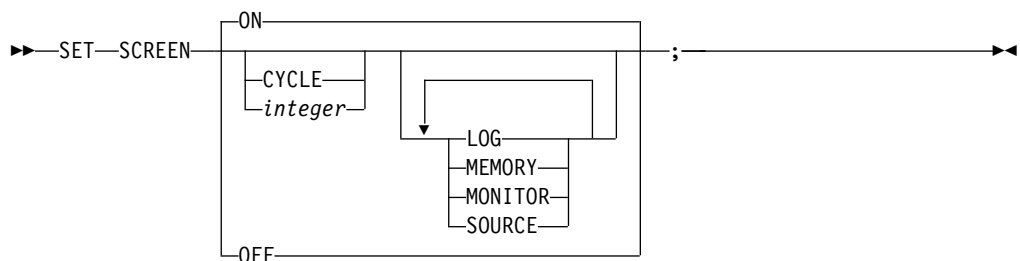
Controls the saving of settings, breakpoints, and monitor specifications.



---

## SET SCREEN (full-screen and line mode)

Controls how information is displayed on the screen.



---

## SET SCROLL DISPLAY (full-screen mode)

Controls whether the scroll field is displayed when operating in full-screen mode.

```
▶▶ SET SCROLL DISPLAY  ON  OFF ;
```

---

## SET SEQUENCE (PL/I)

Controls whether z/OS Debugger interprets data after column 72 in a commands or preference file as a sequence number.

```
▶▶ SET SEQUENCE  OFF  ON ;
```

---

## SET SOURCE

Associates a source file, compiler listing or separate debug file with one or more compile units.

```
▶▶ SET SOURCE  ON  OFF ( cu_spec ) fileid ;
```

---

## SET SUFFIX (full-screen mode)

Controls the display of frequency counts at the right edge of the Source window when in full-screen mode.

```
▶▶ SET SUFFIX  ON  OFF ;
```

---

## SET TEST

Overrides the initial TEST runtime options specified at invocation.

```
▶▶ SET TEST test_level  (-test_level-) ;
```

---

## SET WARNING (C, C++, and PL/I)

Controls display of the z/OS Debugger warning messages and whether exceptions are reflected to the application program.

```
▶▶ SET WARNING  ON  OFF ;
```

---

## SET command (COBOL)

The SET command assigns a value to a COBOL reference.

►► SET *reference* TO 

<i>reference</i>
<i>literal</i>
TRUE

 ;

---

## SHOW prefix command (full-screen mode)

The SHOW prefix command specifies what relative statement (for C) or relative verb (for COBOL) within the line is to have its frequency count temporarily shown in the suffix area.

►► SHOW 

<i>integer</i>
----------------

 ;

---

## STEP command

The STEP command causes z/OS Debugger to dynamically step through a program, executing one or more program statements. In full-screen mode, it provides animated execution.

►► STEP 

<i>integer</i>
*

INTO
OVER
RETURN

 ;

---

## STORAGE command

The STORAGE command enables you to alter up to eight bytes of storage.

►► STORAGE ( 

<i>address</i>
<i>reference</i>
' <i>reference</i> '

 , 

<i>offset</i>
---------------

 , 

<i>length</i>
---------------

 ) = *value* ;

---

## switch command (C and C++)

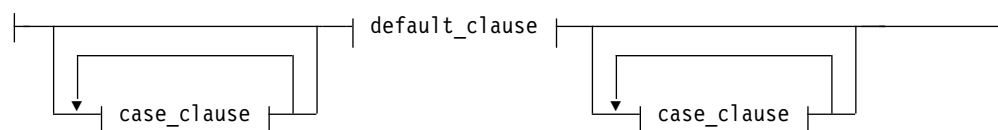
The switch command enables you to transfer control to different commands within the switch body, depending on the value of the switch expression.

►► switch ( *expression* ) { 

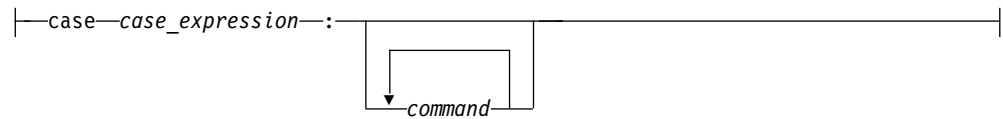
switch_body
-------------

 } ;

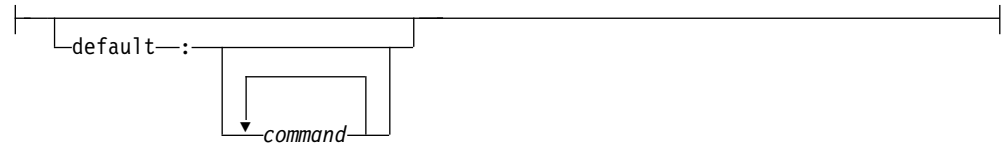
**switch\_body:**



### case\_clause:



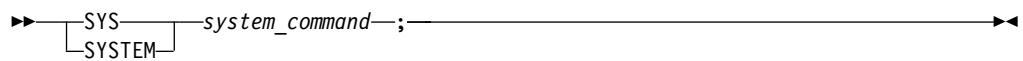
### default\_clause:



---

## SYSTEM command

The SYSTEM command lets you issue TSO commands during a z/OS Debugger session.



---

## TRACE command

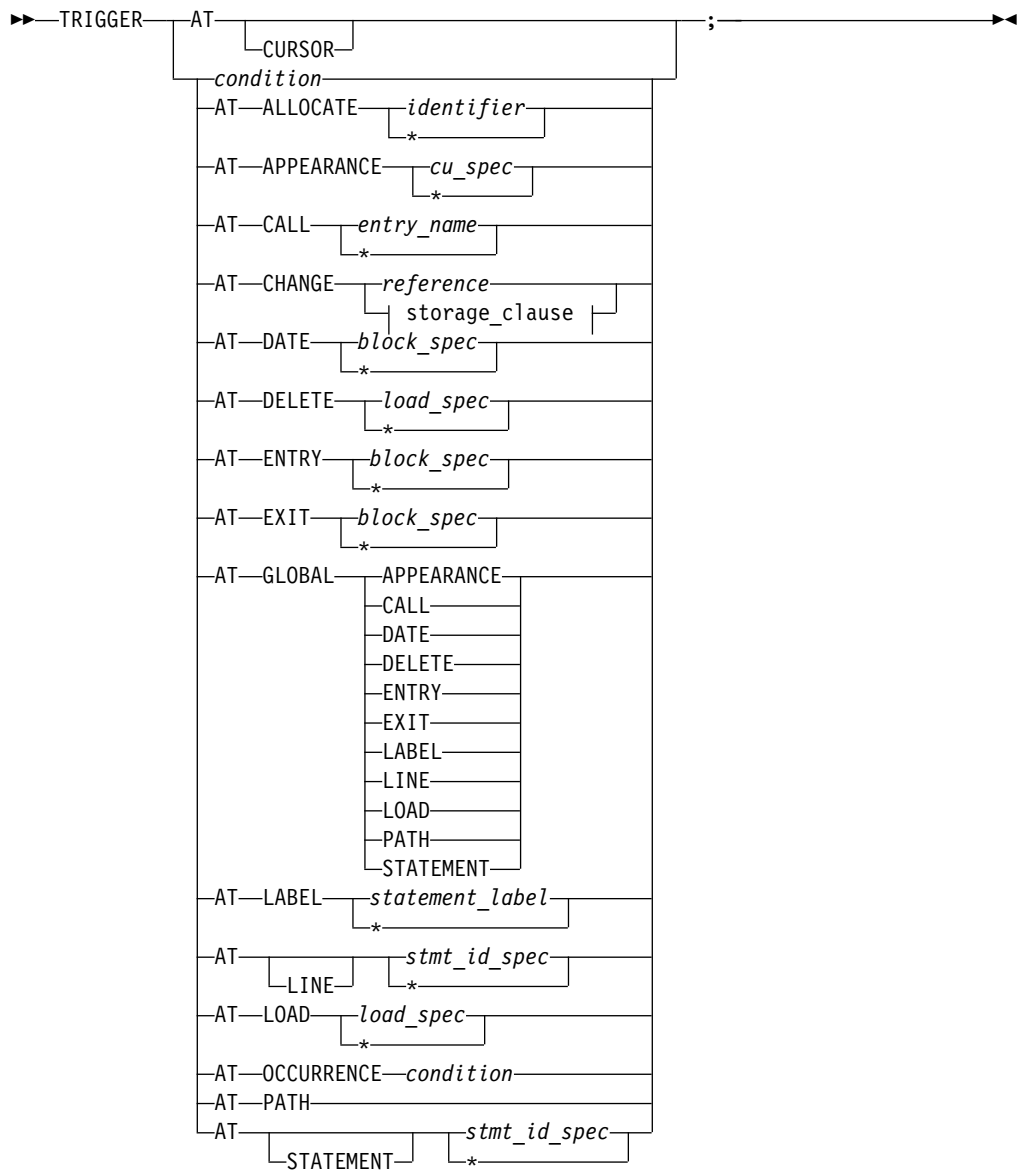
The TRACE command creates a trace of all load modules and DLLs loaded during a debug session. The trace can then be listed at any point during the debug session.



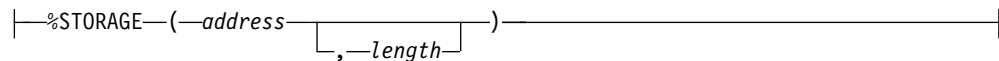
---

## TRIGGER command

The TRIGGER command raises the specified AT-condition in z/OS Debugger, or it raises the specified programming language condition in your program.

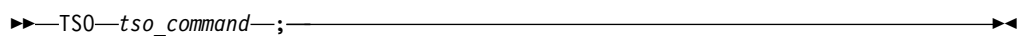


**storage\_clause:**



## TSO command (z/OS)

The TSO command lets you issue TSO commands during a z/OS Debugger session and is valid only in a TSO environment.



---

## USE command

The USE command causes the z/OS Debugger commands in the specified file or data set to be either performed or syntax checked.

```
▶▶ USE ddname ;  
      └──┬──  
        dsname
```

---

## while command (C and C++)

The while command enables you to repeatedly perform the body of a loop until the specified condition is no longer met or evaluates to false.

```
▶▶ while (expression) command ;
```

---

## WINDOW CLOSE

Closes the specified window in the z/OS Debugger full-screen session panel.

```
▶▶ WINDOW CLOSE CURSOR ;  
                  └──┬──  
                    LOG  
                    MEMORY  
                    MONITOR  
                    SOURCE
```

---

## WINDOW OPEN

Opens a previously-closed window in the z/OS Debugger full-screen session panel.

```
▶▶ WINDOW OPEN LOG ;  
                └──┬──  
                  MEMORY  
                  MONITOR  
                  SOURCE
```

---

## WINDOW SIZE

Controls the relative size of currently visible windows in the z/OS Debugger full-screen session panel.

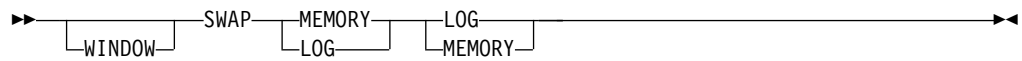
```
▶▶ WINDOW SIZE integer CURSOR ;  
                └──┬──  
                  LOG  
                  MEMORY  
                  MONITOR  
                  SOURCE
```

---

---

## WINDOW SWAP

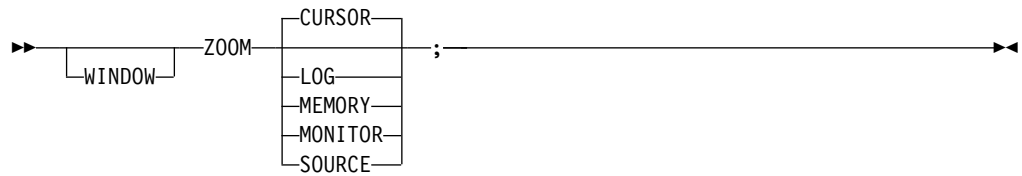
Replaces the logical window being displayed in a physical window with another logical window. The order of the operands is not important.



---

## WINDOW ZOOM

Expands the indicated window to fill the entire screen or restores the screen to the currently defined window configuration.





---

## Chapter 2. z/OS Debugger built-in functions

z/OS Debugger provides you with the following built-in functions:

---

### **%DEC (assembler, disassembly, and LangX COBOL)**

Returns the decimal value of an operand.

---

### **%GENERATION (PL/I)**

Returns a specific generation of a controlled variable in your program.

---

### **%HEX**

Returns the hexadecimal value of an operand.

---

### **%INSTANCES (C, C++, and PL/I)**

Returns the maximum value of %RECURSION (the most recent recursion number) for a given block.

---

### **%RECURSION (C, C++, and PL/I)**

Returns a specific instance of an automatic variable or a parameter in a recursive procedure.

---

### **%WHERE (assembler, disassembly, and LangX COBOL)**

Returns a string that is the address of the operand. %WHERE can be used *only* as the outermost expression in the LIST command.



---

## Chapter 3. EQAOPTS commands (optional)

The EQAOPTS commands (formerly known as options) alter certain z/OS Debugger behaviors. This topic summarizes the syntax of the EQAOPTS commands, followed by a list briefly describing each command. For a complete description of each command and to learn how to use these commands, see *IBM z/OS Debugger Customization Guide* and *IBM z/OS Debugger Reference and Messages*.

The follow diagram describes the syntax of the EQAOPTS commands:

EQAXOPT	BROWSE	RACF	
		ON	
		OFF	
	CACHENUM	cache_value	
	CCOUTPUTDSN	'file_name_pattern'	LOUD
	CCOUTPUTSNALLOC	'allocation_parms'	LOUD
	CCPROGSELECTDSN	'file_name_pattern'	LOUD
	CODEPAGE	nnnn	
	COMMANDSDSN	'file_name_pattern'	LOUD
	DEFAULTVIEW	STANDARD	
		NOMACGEN	
	DLAYDBG	No	
		Yes	
	DLAYBGCND	ALL	
		NONE	
	DLAYBGDSN	'file_name_pattern'	
	DLAYBGTRC	trace_level	
	DLAYBGXRF	DSN	'file_name'
	DOPTACBDSN	'data_set_name'	
	DTCNDELETEDEADPROF	No	
		Yes	
	DTCNFORCECID		YES
	DTCNFORCEIP		NO
	DTCNFORCELOADMODID		
	DTCNFORCENETNAME		
	DTCNFORCEPROGID		
	DTCNFORCETERMID		
	DTCNFORCETRANID		
	DTCNFORCEUSERID		
	DYNDEBUG	ON	
		OFF	
	EQAQPP	ON	
		OFF	
	EXPLICITDEBUG	ON	
		OFF	
	GPFDSN	'file_name'	
	HOSTPORTS	HOSTPORTS_options	
	IGNOREODOLIMIT	YES	
		NO	
	LOGDSN	'file_name_pattern'	LOUD
	LOGDSNALLOC	'allocation_parms'	LOUD
	MAXTRANUSER	max_trans	
	MDBG	YES	
		NO	
	MULTIPROCESS	PARENT	
		CHILD	
		PROMPT	
		EXEC=	NONE
			ANY
	NAMES	NAMES_options	
	NODISPLAY	DEFAULT	
		QUITDEBUG	
	PREFERENCESDSN	'file_name_pattern'	LOUD
	SAVEBPSDSN	'file_name_pattern'	
	SAVESETDSN		
	SAVEBPSDSNALLOC	'allocation_parms'	LOUD
	SAVESETDSNALLOC		LOUD
	SESSIONTIMEOUT	SESSIONTIMEOUT_options	
	STARTSTOPMSG	STARTSTOPMSG_options	WTO
	STARTSTOPMSGDSN	'file_name'	LOUD
	SUBSYS	four_character_name	
	SVCScreen	SVCScreen_options	
	TCPIPDATA	file_name	
	THREADTERMCOND	NOPROMPT	
		PROMPT	
	TIMACB	ACB_name	
	END		

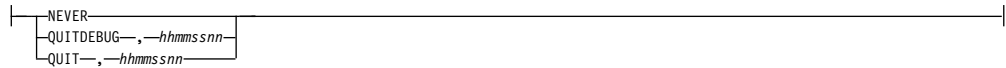
### HOSTPORTS\_options:



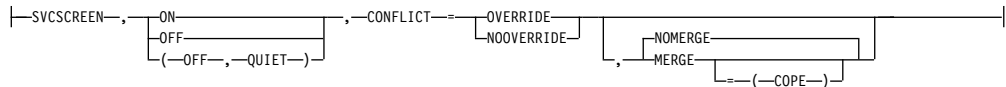
### NAMES\_options:



### SESSIONTIMEOUT\_options:



### SVCScreen\_options:



### STARTSTOPMSG\_options:



---

## BROWSE

Specifies whether a user with sufficient RACF<sup>®</sup> authority to the applicable browse mode RACF facility can start z/OS Debugger in browse mode.

---

## CACHENUM

Specifies the maximum number of program items to be held in an in-memory cache for a debug session. Increase this number to improve performance for applications which have many programs; however, a larger number also increases the storage usage by the debugged task.

---

## CODEPAGE

Indicates which code page to use so that NLS characters are properly communicated between z/OS Debugger and a remote debugger and properly displayed in full screen mode.

---

## COMMANDSDSN

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a commands file. z/OS Debugger reads a member from this commands file every time it starts.

---

## DEFAULTVIEW

Provides a method of setting the initial value for the SET DEFAULT VIEW command.

---

## DLAYDBG

Enables z/OS Debugger to delay the starting of a debug session until z/OS Debugger recognizes a certain program (CU).

### DLAYDBGEND

Indicates whether you want z/OS Debugger to monitor condition events in the delay debug mode.

### DLAYDBGDSN

Indicates that z/OS Debugger is to use the specified naming pattern when it constructs the delay debug profile data set name.

### DLAYDBGTRC

Indicates that z/OS Debugger is to generate a trace during the pattern match process in the delay debug mode. z/OS Debugger uses the WTO (write to operator) command to output the trace.

### DLAYDBGXRF

Indicates that you want z/OS Debugger to use the cross reference file to find the user ID when it constructs the delay debug profile data set name.

---

## DOPTACBDSN

Identifies the data set that will contain DOPT PSBs that are created by z/OS Debugger for the IMS Transaction Isolation Facility. This data set will be used to store ACBs generated for the EQATccn DOPT PSBs.

There is no default value for this command. If your site will use the IMS Transaction Isolation Facility, this command must be specified.

---

## DTCNDELETEDEADPROF

Controls the deletion of DTCN profiles. A dead profile is a profile whose owner has logged off or disconnected from the CICS region.

---

## DTCNFORCExxxxxx

Controls DTCN behavior for the conditions described in Table 3. When you set a DTCNFORCExxxxxx option to YES, DTCN forces users to specify the respective resource type. The default setting is NO.

*Table 3. List of EQAOPTS DTCNFORCExxxxxx commands and their corresponding DTCN fields*

EQAOPTS DTCNFORCExxxxxx command	DTCN field name
DTCNFORCECUID or DTCNFORCEPROGID	CU
DTCNFORCEIP	IP Name/ Address
DTCNFORCELOADMODID	LoadMod
DTCNFORCENETNAME	NetName
DTCNFORCETERMID	Terminal Id
DTCNFORCETRANID	Transaction Id
DTCNFORCEUSERID	User Id

### **Related tasks**

"Requiring users to specify resource types" in the *IBM z/OS Debugger Customization Guide*

---

## DYNDEBUG

Controls the initial default for the SET DYNDEBUG command.

---

## EQAQPP

Indicates the presence of Q++ programs.

---

## EXPLICITDEBUG

Controls whether explicit debug mode is active.

---

## GPFDSN

Specifies the data set name for the global preferences file.

---

## HOSTPORTS

Specifies a host port or range of ports to use for a TCP/IP connection from the host to a workstation when using remote debug mode.

---

## IGNOREODOLIMIT

Instructs z/OS Debugger to display COBOL table data items even when an ODO value is out of range.

---

## LOGDSN

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a log file.

---

## LOGDSNALLOC

Indicates the allocation parameters z/OS Debugger uses to create the data set named by the LOGDSN command.

---

## MAXTRANUSER

Defines the maximum number of IMS transactions that a single user can register using the IBM Transaction Isolation Facility.

---

## MDBG

For z/OS XL C/C++, Version 1 Release 11, if you have compiled with the `DEBUG(FORMAT(DWARF))` compiler option, specifies whether z/OS Debugger obtains debug data from `.mdbg` or `.dbg` files.

---

## MULTIPROCESS

Controls the behavior of z/OS Debugger when a new POSIX process is created by a `fork()` or `exec()` function in the application.

It instructs z/OS Debugger to perform any of the following tasks when a new POSIX process is created:

- Continue debugging the current process. The current process is also referred to as the PARENT process.
  - Stop debugging the current process and start debugging the newly created process. The newly created process is also referred to as the CHILD process.
  - Prompt you to decide whether to follow the PARENT or CHILD process.
- 

## NAMES

Provides a method of entering NAMES commands that apply before the first load module and any of the compile units contained in that load module are processed.

---

## NODISPLAY

Controls z/OS Debugger behavior when the terminal using full-screen mode using the Terminal Interface Manager or the remote debugger are not available.

---

## PREFERENCESDSN

Indicates the naming pattern z/OS Debugger uses to locate the data set containing a preferences file. z/OS Debugger reads this preferences file every time it starts.

---

## SAVEBPDSN and SAVESETDSN

Specifies the data set names to be used to save the breakpoints (SAVEBPDSN) and settings (SAVESETDSN). One qualifier in each of these data set names should be `&&USERID`, which represents the user ID of the current user.

---

## SAVEBPDSNALLOC and SAVESETDSNALLOC

Indicates the allocation parameters z/OS Debugger uses to create the data sets named by the SAVEBPDSN and SAVESETDSN commands.

---



---

## SESSIONTIMEOUT

Establishes a timeout for idle z/OS Debugger sessions

---

## STARTSTOPMSG

Controls whether to issue a message when each debugging session is initiated or terminated.

## STARTSTOPMSGDSN

Indicates that you want z/OS Debugger to write a message in the message file when the debug session is started or stopped.

---

## SUBSYS

Provides a 1 to 4 character subsystem name. If an Enterprise PL/I or C/C++ source file is found to have a DSORG of DA or VSAM and this parameter is supplied, then this parameter is passed to SVC 99 (dynamic allocation) through the SUBSYS text unit when z/OS Debugger allocates the source file.

---

## SVCSCREEN

Controls z/OS Debugger's enablement of SVC screening.

---

## TCPIPDATA DSN

Instructs z/OS Debugger to dynamically allocate the specified *file-name* to DDNAME SYSTCPD (if SYSTCPD is not already allocated).

---

## THREADTERMCOND

Specifies whether z/OS Debugger should suppress the prompt it displays when the thread termination condition, FINISH condition, or CEE067 is raised by Language Environment.

---

## TIMACB

Specifies an alternate Terminal Information Manager ACB name.

**Related tasks**

"Running the Terminal Interface Manager on more than one LPAR on the same VTAM® network" in the *IBM z/OS Debugger Customization Guide*

---

## END

Identifies the last EQAOPTS command. You must always specify this command and make it the last command in the input stream.



## Chapter 4. z/OS Debugger variables

The following table summarizes the z/OS Debugger variables. For more information about these variables, see *IBM z/OS Debugger Reference and Messages*.

z/OS Debugger variable	Value
%ADDRESS	Address of the location where your program was interrupted
%AMODE	Current <sup>®</sup> AMODE of the suspended program
%BLOCK	Name of the current block
%CAAADDRESS	Address of the CAA control block associated with the suspended program
%CC	(Assembler and disassembly only) Condition code from current PSW
%CONDITION	Name or number of the condition when z/OS Debugger is entered because of an AT OCCURRENCE
%COUNTRY	Current country code
%CU	Name of the primary entry point of the current compile unit
%EPA	Address of the primary entry point in the current compile unit
%EPAR $n$ or %EPRH $n$	(Assembler, disassembly, C and C++, and PL/I only) Extended-precision floating-point registers
%EPRB $n$	(Assembler and disassembly only) Extended-precision floating-point registers in binary format
%EPRD $n$	(Assembler and Disassembly only) Extended-precision floating-point registers in decimal format
%FPR $n$ or %FPRH $n$	Single-precision floating-point registers in hexadecimal format
%FPRB $n$	(Assembler and Disassembly only) Single-precision floating-point registers in binary format
%FPRD $n$	(Assembler and Disassembly only) Single-precision floating-point registers in decimal format
%GPR $n$	32-bit base General Purpose Registers at the point of interruption in a program
%GPRG $n$	64-bit General Purpose Registers at the point of interruption in a program
%GPRH $n$	32-bit high General Purpose Registers at the point of interruption in a program
%HARDWARE	Type of hardware where the application is running
%LINE or %STATEMENT	Current source line number
%LOAD	Name of the load module of the current program, or an asterisk (*)
%LPR $n$ or %LPRH $n$	Double-precision floating-point registers in hexadecimal format
%LPRB $n$	(Assembler and Disassembly only) Double-precision floating-point registers in binary format
%LPRD $n$	(Assembler and Disassembly only) Double-precision floating-point registers in decimal format
%NLANGUAGE	National language currently in use

<b>z/OS Debugger variable</b>	<b>Value</b>
%PATHCODE	Integer identifying the type of change occurring when the program flow reaches a point of discontinuity, and the path condition is raised
%PLANGUAGE	Current programming language
%PROGMASK	(Assembler and disassembly only) Program mask from current PSW
%PROGRAM	Equivalent to %CU
%PSW	(Assembler and disassembly only) Current Program Status Word
%RC	Return code from the most recent z/OS Debugger command
%RSTDSETS	A value of 1 if user settings have been restored and 0 otherwise
%RUNMODE	String identifying the presentation mode of z/OS Debugger
%Rn	32-bit base General Purpose Registers for the currently qualified assembler or disassembly CU
%STATEMENT	Equivalent to %LINE
%SUBSYSTEM	Name of the underlying subsystem, if any, where the program is running
%SYSTEM	Name of the operating system supporting the program

---

## Chapter 5. Reference card: Frequently used z/OS Debugger commands

The following reference card provides a list of frequently used z/OS Debugger commands.

The\ reference cards are designed to be printed from a PDF file. If you are viewing this page through IBM Knowledge Center, follow these instructions to view the reference card through a PDF file and print it:

1. Click on the "PDF version" topic underneath "IBM z/OS Debugger Reference Summary" in the navigation pane. The PDF file for *IBM z/OS Debugger Reference Summary* opens.
2. In the Bookmarks (Table of Contents) view of Adobe Reader, click on the appendix heading for the reference card you want to print.
3. Scroll to the first page of the reference card, identify the page number in the Adobe Reader toolbar, then make a note of this page number. Do not use the page number printed on the bottom of the page. Scroll through the reference card to determine the number of pages it spans.
4. Click on Print in the Adobe Reader toolbar or click on **File > Print**. In the "Print Range" box, select Pages and then enter the page number you noted in the previous step, followed by a dash and the last page of the reference card. For example, if the first page of the reference card is on page 75 and the reference card spans four pages, enter "75-79" in the Pages field. If you are capable of printing in duplex, enable duplex printing. Do not alter any other setting; print these pages as you would any portrait-oriented page.
5. Click on OK or Print to start printing.

**Setting breakpoints (stopping points), at statements in a program**

**A** A is the abbreviation for AT. Enter through the prefix area of the Source window. Sets a breakpoint on line where A is entered.

**PF6** Sets a breakpoint on line where cursor is located.

**AT 509** Sets a breakpoint on line 509.

**AT LABEL *label\_name***  
Sets a breakpoint on a label, paragraph, or section name.

**Clearing (removing) breakpoints set at statements in a program**

**C** C is the abbreviation for CLEAR AT. Type a C in the prefix area of the Source window. When you press Enter, z/OS Debugger removes the breakpoint on the line where C is in the prefix area.

**PF6** Removes a breakpoint on line where cursor is located.

**CLEAR AT 509**  
Removes the breakpoint on line 509.

**CLEAR AT LABEL *label\_name***  
Clears a breakpoint from a label, paragraph, or section name.

**Setting breakpoints (stopping points) triggered by a change in the value of a variable**

**AT CHANGE ITEMNO**  
Sets a breakpoint that stops the program when the value of ITEMNO changes.

**Clearing (removing) breakpoints triggered by a change in the value of a variable**

**CLEAR AT CHANGE ITEMNO**  
Removes the breakpoint that stops the program when the value of ITEMNO changes.

**Setting breakpoints (stopping points) at the entrance or exit of a program**

**AT ENTRY *cu\_name***  
Sets a breakpoint that stops the program when it enters *cu\_name*.

**AT ENTRY \*** Sets breakpoints that stop a program whenever z/OS Debugger enters a known program.

**AT EXIT *cu\_name***

Sets a breakpoint that stops the program when it exits *cu\_name*.

**AT EXIT \*** Sets breakpoints that stop a program whenever z/OS Debugger exits a known program.

**Clearing (removing) breakpoints set at the entrance or exit of a program**

**CLEAR AT ENTRY *cu\_name***

Clears the breakpoint that stops the program when it enters *cu\_name*.

**CLEAR AT ENTRY \***

Clears all breakpoints that stop a program whenever z/OS Debugger enters a known program.

**CLEAR AT EXIT *cu\_name***

Clears a breakpoint that stops the program when it exits *cu\_name*.

**CLEAR AT EXIT \***

Removes the breakpoints at every exit point in every program.

**Making breakpoints conditional**

Add a WHEN clause to make breakpoints conditional.

**AT CHANGE ITEMNO WHEN ITEMNO = '0805'**  
Stop after the value of ITEMNO changes, but only if ITEMNO is equal to the specified value.

**AT CHANGE CUSTID WHEN ACCT-BAL > 100**  
Stop after the value of CUSTID changes, but only if ACCT-BAL is greater than the specified value.

**AT 509 WHEN ITEMNO = '0805'**  
Stop at statement 509, but only if ITEMNO is equal to the specified value.

**Commands that work on all breakpoints**

**LIST AT** Displays all breakpoints in the Log window.

**CLEAR AT** Clears all breakpoints.

**DISABLE AT** Temporarily disables (deactivates) all breakpoints.

**ENABLE AT** Enables (activates) all disabled breakpoints.

**Identifying and loading a program's source and debug information****SET DEFAULT LISTINGS *source\_info\_library***

Identifies a source library (PDS or PDSE) where z/OS Debugger searches for source files and debug information files. For example, SYSDBUG files, LANGX files, and compiler listings. z/OS Debugger displays this information in the Source window.

**SET DEFAULT LISTINGS (*source\_info.lib1, source\_info.lib2, ...*)**

Identifies a concatenation of source libraries (PDS or PDSE) where z/OS Debugger searches for source files and debug information files.

**LISTING or LIST**

Displays a list of programs known to z/OS Debugger. Then, you can specify the name of the source file or debug information file for each program.

**LOD assembler\_CSECT or LOD LangX\_COBOL\_program**

Load debug information about *assembler\_CSECT* or *LangX\_COBOL\_program* from the EQALANGX file into the Source window.

### Displaying variables in the Monitor window

#### **SET AUTOMONITOR ON**

Automatically displays the values of variables referenced by the current statement in the Monitor window.

#### **SET AUTOMONITOR ON BOTH**

Automatically displays the values of variables referenced by both the current statement and the previously run statement in the Monitor window.

#### **MONITOR LIST ITEMNO**

Adds the ITEMNO variable and its value to the Monitor window.

#### **SET MONITOR DATATYPE ON**

Display the data types of variables.

#### **SET MON WRAP OFF**

Displays values on a single line. If the value is longer than the visible area, z/OS Debugger displays a scale to indicate that there is more to see.

#### **CLEAR MONITOR**

Cleares all items from the Monitor window.

**C** C is the abbreviation for the CLEAR MONITOR command. Type in the letter C in the prefix area of the monitor window. When you press Enter, z/OS Debugger removes the variable on the line where C is in the prefix area.

### Changing values of variables

#### **Type over value displayed in the Monitor window**

Move cursor to value displayed in Monitor window, type in new value, then press Enter.

#### **MOVE 24 to ACCUM-X**

For COBOL programs, replace the value of ACCUM-X with 24.

#### **ACCUMX = 24**

For some languages, replaces the value of ACCUMX with 24.

### Displaying variables in Log window and controlling Log window options

#### **LIST CUST-ID or LIST TITLED CUST-ID**

Displays the value of a variable. Only some programming languages require TITLED.

**PF4 or LIST** Displays the value of a variable identified by the location of the cursor.

#### **LIST TITLED WSS or LS or FS or LOS**

Display contents of specific SECTIONS for COBOL programs. WSS means Working-Storage Section, LS means Linkage Section, FS means File Section, and LOS means Local-Storage Section.

#### **LIST TITLED \***

Displays the values of all variables.

#### **SET ECHO OFF**

z/OS Debugger does not display STEP and GO commands in the Log window. However, if a log file is open, z/OS Debugger writes them to the log file.

#### **SET LOG ON FILE file\_name OLD**

Opens a log file. When z/OS Debugger opens the log file, all items it writes to the Log window are also written to the log file.

### Refresh the Source window to display the current statement

#### **QUALIFY RESET**

Repositions source in the Source window so that z/OS Debugger displays the current program and current statement.

### Working with called programs

#### **STEP or STEP INTO**

When the current statement is a CALL, steps into the called program.

#### **STEP OVER**

When the current statement is a CALL, z/OS Debugger runs the called program but does not display it. z/OS Debugger stops at the statement after the call.

#### **LOAD program\_name**

Make program\_name known to z/OS Debugger.

#### **QUALIFY program\_name**

Displays the program program\_name in the Source window. When the program is displayed in the Source window, you can set a breakpoint or work with variables in that program.

#### **QUALIFY RESET**

Reposition to the current program and the current line.

### Controlling program execution

#### **STEP or PF2**

Run one statement or line.

#### **GO or PF9**

Run the program until z/OS Debugger encounters a breakpoint, the program finishes, or an abend occurs.

#### **RUNTO 27**

Runs the program and then stops before it runs line 27.

#### **R**

R is the abbreviation of RUNTO. Type in the command in the prefix area of the Source window. When you press Enter, z/OS Debugger runs the program until it reaches the line with the R in the prefix area.

#### **GO BYPASS**

Resume running a program after encountering an abend. Enter this command immediately after an abend occurs. z/OS Debugger skips the statement that caused the abend and continues running the program from the next logical statement.

### Skipping (do not run) over program statements

#### **JUMPTO 27**

Moves the point at which the program resumes execution to line 27, does not run any statements between the current point and line 27, and then pauses at line 27. When you enter a GO or STEP command, the program resumes running at line 27.

#### **GOTO 27**

Moves the point at which the program resumes execution to line 27, does not run any statements between the current point and line 27, and then resumes running the program at line 27.

**Commands that work in the prefix area of the Source window**

- A** A is the abbreviation for AT. Sets a breakpoint on the line.
- C** C is the abbreviation for CLEAR AT. Clears the breakpoint from the line.
- D** D is the abbreviation for DISABLE AT. Disables the breakpoint on the line.
- E** E is the abbreviation for ENABLE AT. Enables the breakpoint on the line.
- L** L is the abbreviation for LIST. Displays all variables referenced by the statement in the log. This prefix command works only for programs compiled with specific compilers.
- L1, L2, L3, ...** L is the abbreviation for LIST. Displays the first, second, third, and so on variable referenced by the statement in the log. This prefix command works only for programs compiled with specific compilers.

**M** M is the abbreviation for MONITOR LIST. Displays all variables reference by the statement in the Monitor window. This prefix command works only for programs compiled with specific compilers.

**M1, M2, M3, ...**

M is the abbreviation for MONITOR LIST. Displays first, second, third, and so on variable referenced by the statement in the Monitor window. This prefix command works only for programs compiled with specific compilers.

**Commands that work in the prefix area of the Monitor window**

- C** C is the abbreviation for CLEAR MONITOR. Removes the variable from the Monitor window.
- D** D is the abbreviation for default. Displays the value of the variable in a format based on its declared data type.
- H** H is the abbreviation for hexadecimal. Displays the value of the variable in hexadecimal format.

**Working with PF keys**

**QUERY PFKEYS** Displays the PF key settings in the log.

**SET KEYS ON** z/OS Debugger displays the PF key settings for PF keys 1-12 at the bottom of the screen.

**SET KEYS ON 24** z/OS Debugger displays the PF key settings for PF keys 13-24 at the bottom of the screen.

**SET KEYS OFF** z/OS Debugger removes the PF key settings from the bottom of the screen.

**SET PF16 "MON" = MONITOR LIST** Example of assigning a command to a PF key. In this example, you assign the MONITOR LIST command to the PF16 key. When z/OS Debugger displays PF keys 13-24 at the bottom of the screen, it shows "PF16=MON".

**Default PF key settings**

- PF1 or PF13** ? (HELP)
- PF2 or PF14** STEP
- PF3 or PF15** END
- PF4 or PF16** LIST
- PF5 or PF17** FIND
- PF6 or PF18** AT/CLEAR
- PF7 or PF19** UP
- PF8 or PF20** DOWN
- PF9 or PF21** GO
- PF10 or PF22** ZOOM
- PF11 or PF23** ZOOM LOG
- PF12 or PF24** RETRIEVE

**Displaying help for commands**

- ?** Displays a list of commands
- AT ?** Example of displaying help for the AT command. Enter all or part of a command, followed by a question mark ("?") to display keywords that are valid at the location of the question mark.

**Continuing a long command**

**- (dash at the end of a line)** To continue a long command (for example, a command that exceeds the size of the command line), type a dash at the end of a partial command and then press Enter. z/OS Debugger prompts you to enter the rest of the command.

**Abbreviating commands**

**(use partial keywords)** You can abbreviate keywords in z/OS Debugger commands to the least number of letters that make the keyword unambiguous. For example, you can abbreviate the command MONITOR LIST VARX to MON LIST VARX or MO LIS VARX.

**Ending a debugging session**

- QUIT** Ends the debugging session and prompts you to verify that you want to end the debugging session.
- QUIT** Ends the debugging session without prompting you.
- QUIT DEBUG** Ends the debugging session but program continues to run. z/OS Debugger will not be restarted.
- QUIT DEBUG TASK** This command works only for CICS. Ends debugging session but the program continues to run. z/OS Debugger will not be restarted. To start z/OS Debugger, start another iteration of a pseudo-conversational task.

**QUIT ABEND** Ends the debugging session and terminates the program with an abend at the current location.



---

## Chapter 6. Reference card: Frequently used z/OS Debugger commands while debugging assembler programs

The following reference card provides a list of frequently used z/OS Debugger commands while debugging assembler programs. Enter all the commands on the command line, unless otherwise indicated.

The reference cards are designed to be printed from a PDF file. If you are viewing this page through IBM Knowledge Center, follow these instructions to view the reference card through a PDF file and print it:

1. Click on the "PDF version" topic underneath "IBM z/OS Debugger Reference Summary" in the navigation pane. The PDF file for *IBM z/OS Debugger Reference Summary* opens.
2. In the Bookmarks (Table of Contents) view of Adobe Reader, click on the appendix heading for the reference card you want to print.
3. Scroll to the first page of the reference card, identify the page number in the Adobe Reader toolbar, then make a note of this page number. Do not use the page number printed on the bottom of the page. Scroll through the reference card to determine the number of pages it spans.
4. Click on Print in the Adobe Reader toolbar or click on File->Print. In the "Print Range" box, select Pages and then enter the page number you noted in the previous step, followed by a dash and the last page of the reference card. For example, if the first page of the reference card is on page 75 and the reference card spans four pages, enter "75-79" in the Pages field. If you are capable of printing in duplex, enable duplex printing. Do not alter any other setting; print these pages as you would any portrait-oriented page.
5. Click on OK or Print to start printing.

Setting breakpoints (stopping points) in a program

- A** Type A in prefix area of the Source window on line where you want to set a breakpoint. Press Enter. z/OS Debugger sets a breakpoint on that line. (A is the abbreviation for AT.)
- PF6** Move cursor to line where you want to set the breakpoint. Press PF6. z/OS Debugger sets a breakpoint on that line.
- AT 509** z/OS Debugger sets a breakpoint on line 509.
- AT EXIT \*** z/OS Debugger sets a breakpoint at every program exit point.
- AT ENTRY \*** z/OS Debugger sets a breakpoint at every program entry point.
- AT ENTRY *curname***  
z/OS Debugger sets a breakpoint at entry point of *curname*.

Clearing (removing) breakpoints set in a program

- C** Type a C in the prefix area of the Source window on a line containing a breakpoint you want to remove. Press Enter. z/OS Debugger removes the breakpoint on that line. (C is the abbreviation for CLEAR AT.)
- PF6** Move cursor to line where you want to remove the breakpoint. Press PF6. z/OS Debugger removes the breakpoint on that line.
- CLEAR AT 509** z/OS Debugger removes the breakpoint on line 509.
- CLEAR AT EXIT \***  
z/OS Debugger removes breakpoints from every program exit point.
- CLEAR AT ENTRY \***  
z/OS Debugger removes breakpoints from every program entry point.
- CLEAR AT** z/OS Debugger removes all breakpoints.

Set a deferred breakpoint

- A deferred breakpoint is a breakpoint you set on a program that is not yet known to z/OS Debugger.
- AT ENTRY *curname***  
If the block name is the same as *curname*, you no longer have to specify *curname* as *curname*:>*curname*.

Setting conditional breakpoints

- AT CHANGE ITEMNO**  
After the value of ITEMNO changes, z/OS Debugger stops the program. ITEMNO must be known through DC, DS, or USING.
- AT CHANGE \_STORAGE (R6 + 1::5)**  
After the data at the address (R6+1) changes, z/OS Debugger stops the program.
- Example of how to stop when a variable reaches a specific value, display line number where program stopped, then display value of ITEMNO:  
AT CHANGE ITEMNO WHEN ITEMNO = '00006' DO QUERY LOC;  
LIST ('ITEMNO CHANGED HERE', ITEMNO)  
END; 60; ;
- You can substitute ITEMNO = '00006' with ITEMNO = 6 or \_STORAGE (R6 + 1::5). The AT CHANGE, WHEN, and DO clause must be on the same line.

Clearing (removing) conditional breakpoints

- CLEAR AT CHANGE ITEMNO**  
z/OS Debugger removes the breakpoint that stops the program when the value of ITEMNO changes.

Viewing data within a program

- MONITOR LIST ITEMNO SET MON WRAP OFF**  
z/OS Debugger adds the variable ITEMNO and its current value to the Monitor window, and displays everything on one line.
- LIST ITEMNO**  
z/OS Debugger adds contents of ITEMNO to the log.
- SET AUTO ON**  
z/OS Debugger automatically displays values in the Monitor window.
- SET AUTO ON LOG**  
z/OS Debugger automatically displays values in the log.

Monitor window prefix commands

- Type these letters into the prefix area of the Monitor window.
- C** z/OS Debugger runs the CLEAR MONITOR *n* command, where *n* is the monitor number assigned to the variable being monitored on that line.
- H** Display the value of the variables on this line in hexadecimal format.
- DEF** Display the value of the variables on this line in their declared format.

Changing values of variables

- ITEMNO = '00006'**  
z/OS Debugger replaces the value of ITEMNO with 00006.

Type over the displayed value

In the Monitor window, if the value is between black boxes, type the new value over the displayed value.

Working with called programs

- STEP INTO** z/OS Debugger steps into the program being called and continues stepping through that program.
- STEP OVER** z/OS Debugger runs the program being called without displaying the source for that program.
- Enter the following series of commands to do the following tasks:
1. Identify *program1* to z/OS Debugger.
  2. Be able to set a breakpoint in *program1*.
  3. Display the source of *program1* in the Source window.
- LOAD *program1*;  
QUALIFY *program1*  
QUALIFY RESET;

Return to or find the next line to run

- QUALIFY RESET**  
z/OS Debugger displays, in the Source window, the next line it will run.
- Load assembler & OSVS COBOL debugging information**
- LOD *asmsect***  
z/OS Debugger loads debug information from EQALANGX into the Source window.
- LOD *oscob\_prog***  
z/OS Debugger loads debug information from EQALANGX into the Source window.

### Working with registers

**MON LI REG** z/OS Debugger adds all registers to the Monitor window.

**LIST STORAGE (%PR6->+1,5);**  
z/OS Debugger display 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.

**MON LI STORAGE (R6->+1,5);**  
z/OS Debugger adds to the Monitor window the 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.

**MON LI \_STORAGE(R6 + 1;:5)**  
z/OS Debugger adds to the Monitor window the 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.

### Controlling program execution

**STEP** Run one instruction.

**PF2** PF key to run one instruction.

**GO** Run the program until it abends, encounters a breakpoint, or finishes.

**PF9** PF key to run the program until it abends, encounters a breakpoint, or finishes.

**JUMPTO 27** z/OS Debugger makes line 27 the current point of execution, but does not run line 27.

**RUNTO 27** z/OS Debugger runs the program to line 27, then stops.

### Saving monitors, breakpoints and settings across debugging sessions

**SAVE MONITORS** z/OS Debugger saves all variables being monitored in the Monitor window.

**SAVE BPS** z/OS Debugger saves all the breakpoints.

**SAVE SETTINGS** z/OS Debugger saves all the settings set by the SET command.

**RESTORE MONITORS** z/OS Debugger restores all the variables saved by the SAVE MONITORS command to the Monitor window.

**RESTORE BPS** z/OS Debugger restores all the breakpoints saved by the SAVE BPS command.

**RESTORE SETTINGS** z/OS Debugger restores all the settings saved by the SAVE SETTINGS command.

### Memory window

**MEMORY ITEMNO** z/OS Debugger displays raw storage beginning at the location of ITEMNO.

**MEMORY X'2503D008'**  
z/OS Debugger displays memory beginning base address X'2503D008'.

### SWAP MEMORY LOG

z/OS Debugger switches between the Memory window and the Log window.

### SET EQUATE SW- 'SWAP MEM LOG'

z/OS Debugger assigns the letters "SW" with the command SWAP MEMORY LOG. Afterwards, you enter SW to switch between the Memory window and the Log window.

### CICS pseudo-conversational programs

**SET TEST ERROR** z/OS Debugger does not stop for Language Environment class 1 exceptions: EXEC CICS RETURN and STOP RUN in batch.

### QUIT DEBUG TASK

Stop z/OS Debugger for this task.

### Technique to bypass an abend condition

**GO BYPASS** Skip the exception condition and stop at the next breakpoint.

### Technique to call Fault Analyzer for DUMP

**CALL %FA** z/OS Debugger produces a DUMP that Fault Analyzer can use in its analysis.

### End program debugging (testing)

**QUIT** Ends z/OS Debugger and program stops running.

**QQUIT** Ends z/OS Debugger without prompt.

**QUIT DEBUG** Ends z/OS Debugger but program continues running.

**QUIT DEBUG TASK** Ends z/OS Debugger but program continues running.

**QUIT ABEND** Causes a ROLLBACK for IMS and DB2 programs.

### Suggested PF key settings

**PF1** "PF13/24"=SET KEYS 24;  
**PF2** "KEEP"=MON LOCAL %CU LIST CURSOR;  
**PF3** "QUIT"=QUIT;  
**PF4** "CURRLINE"=QUALIFY RESET  
**PF5** "FIND"=FIND;  
**PF7** "UP"=SCROLL UP;  
**PF8** "DOWN"=SCROLL DOWN;  
**PF9** "GO 1"=STEP  
**PF10** "ZOOM WIN"=IMMEDIATE ZOOM;  
**PF11** "ZOOM LOG"=WINDOW ZOOM LOG;  
**PF12** "GO"=GO;  
**PF13** "PF1/12"=SET KEYS 12;  
**PF14** "CLR KEEP"=CLEAR MONITOR;  
**PF15** "QUIT"=QUIT;  
**PF16** "LIST CSK"=LIST CURSOR;  
**PF17** "FIND"=IMMEDIATE FIND;  
**PF18** "AT/CLEAR"=AT TOGGLE;  
**PF19** "TOP"=TOP;  
**PF20** "BOTTOM"=BOT;  
**PF21** "GO 1"=STEP;  
**PF22** "ZOOM WIN"=IMMEDIATE ZOOM;  
**PF23** "ZOOM LOG"=WINDOW ZOOM LOG;  
**PF24** "RETRIEVE"=IMMEDIATE RETRIEVE;



---

## Notices

This information was developed for products and services offered in the U.S.A. IBM might 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 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 Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
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:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with the 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 might 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.

---

## Copyright license

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or functions of these programs.

---

## Programming interface information

This book is intended to help you debug application programs. This publication documents intended Programming Interfaces that allow you to write programs to obtain the services of z/OS Debugger.

---

## Trademarks and service marks

IBM, the IBM logo, and [ibm.com](http://ibm.com)<sup>®</sup> are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Java and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

MasterCraft is a trademark of Tata Consultancy Services Ltd.





Product Number: 5724-T07

Printed in USA

SC27-9581-00

