Debug Tool for z/OS

Reference Summary

Version 13.1

SC14-7602-04

Debug Tool for z/OS

Reference Summary

Version 13.1

SC14-7602-04

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

Fifth Edition (October 2015)

This edition applies to Debug Tool for z/OS, Version 13.1 (Program Number 5655-Q10 with the PTF for APAR PI37275), 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 Release 1 (Program Number 5655-W32)
- 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 4.4 and earlier (Program Number 5655-W67)

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

You can find out more about Debug Tool by visiting the IBM Web site for Debug Tool at: www.ibm.com/software/products/us/en/debugtool

Contents

About this document

Debug Tool combines the richness of the z/OS® environment with the power of Language Environment® to provide a debugger for programmers to isolate and fix their program bugs and test their applications. Debug Tool 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 Debug Tool. 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 *Debug Tool Reference and Messages*. For more information about each EQAOPTS command, see the *Debug Tool Customization Guide* or *Debug Tool Reference and Messages*.

Who might use this document

This document is intended for programmers using Debug Tool 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.

Debug Tool runs on the z/OS operating system and supports the following subsystems:
- CICS®
- DB2®
- 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® Resource Link® 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

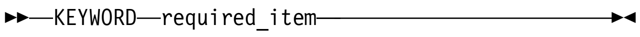
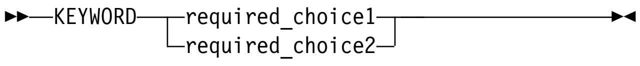
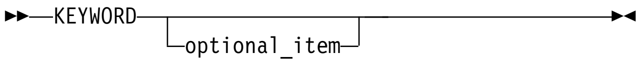
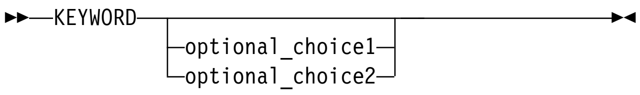
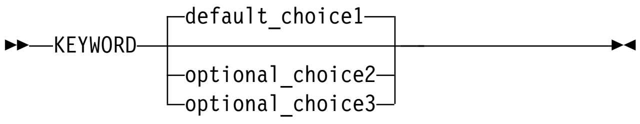
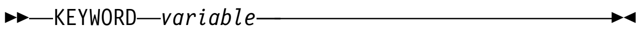
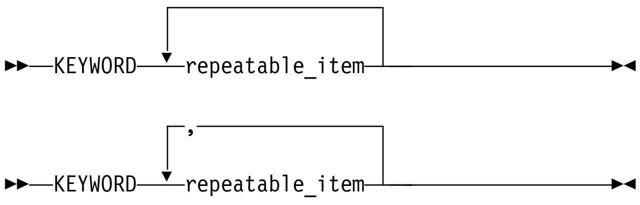
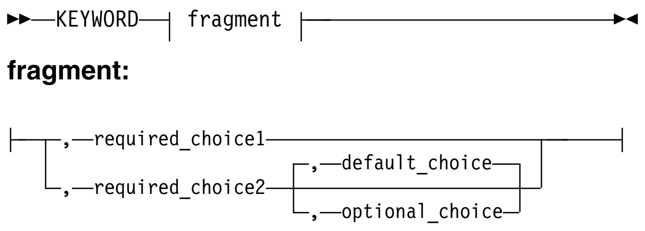
Item	**Syntax example**
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice. A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
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.	 default_choice1 KEYWORD optional_choice2 optional_choice3
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.	 KEYWORD repeatable_item , KEYWORD repeatable_item

Table 1. Syntax examples (continued)

Item	Syntax example
Fragment. The -| fragment |- 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.	 KEYWORD fragment fragment: ,—required_choice1 ,—required_choice2 ,—default_choice ,—optional_choice

How to send 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 Debug Tool documentation, contact us in one of these ways:

- Use the Online Readers' Comment Form at www.ibm.com/software/awdtools/rcf/. Be sure to include the name of the document, the publication number of the document, the version of Debug Tool, and, if applicable, the specific location (for example, page number) of the text that you are commenting on.
- Send your comments by email to comments@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of Debug Tool, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

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

Changes introduced with the PTF for APAR PI37275

- The IMS Transaction Isolation Facility is added to enhance the isolation of Debug Tool users in IMS message regions. Debug Tool users can register to debug certain IMS transactions in a given IMS system. Users can also start a private IMS region, which is cloned from the currently-running environment for the selected transaction.
- Support is added for deferred breakpoints for COBOL and PL/I programs. The new Debug Tool Deferred Breakpoints option in Debug Tool Utilities allows users to create and view a list of breakpoints prior to starting the debug session. It reduces the time spent on the debugging session and also the system resource usage.
- Support is added for the Debug Tool JCL Wizard. By using this new ISPF edit macro, **EQAJCL**, you can modify a JCL or procedure member and create statements to invoke Debug Tool in various environments. You can also create statements to invoke Debug Tool for the Terminal Interface Manager, Dedicated Terminal, or Remote GUI available with RD/z and PD Tools Studio.
- Support is added to allow STEP and GO in the RD/z Debug Configuration Startup commands.
- Support is added for the code coverage and load module analyzer plug-ins.
- New EQAOPTS commands DOPTACBDSN, IGNOREODOLIMIT, and MAXTRANUSER.

  For more information , see “DOPTACBDSN” on page 60, “IGNOREODOLIMIT” on page 61, and “MAXTRANUSER” on page 62.

Changes introduced with the PTF for APAR PI29800

- Support is added for debugging subtasks initiated by the ATTACH assembler macro. Multiple subtasks might be debugged concurrently, as well as the parent task.
- Delay debug mode is enhanced to support non-LE programs in certain circumstances.

Changes introduced with the PTF for APAR PI24559

- The environment specific Language Environment user exits EQADBCXT, EQADDCXT, and EQADICXT are now in sunset mode and will be removed in the next Debug Tool version. Users should move to the consolidated environment specific Language Environment user exit EQAD3CXT.

  Debug Tool Utilities option 'B Delay Debug Profile' and the DTSP Profile Manager plug-in now create an enhanced debug profile that contains a new <NM2> tag (rather than the old <PGM> tag) and will only work with the consolidated user exit.
- Delay debug mode is enhanced to include C functions. In addition, a load module name can now be paired with a program name or C function in the delay debug data set.
- Terminal Interface Manager now allows the same user to log on multiple times on separate terminals. This allows multiple tasks to be debugged by the same user ID simultaneously.
- Support is added for Automonitor for C/C++.

- M/L prefix commands are enhanced to support these prefix commands for C/C++.
- The CICS DTST transaction that is used to display, scan and modify CICS storage is enhanced to support 64 bit addresses.
- Code Coverage support is added for Enterprise COBOL for z/OS Version 5.
- CALL %FA command is enhanced to support remote debug mode.
- AT GLOBAL LABEL, AT LABEL, CLEAR AT GLOBAL LABEL, CLEAR AT LABEL, LIST AT GLOBAL LABEL, LIST AT LABEL and LIST NAMES LABEL commands are enhanced to support remote debug mode.

Changes introduced with the PTF for APAR PI16543

- Support is added for CICS TS 5.2.
- The DTCN Remote Plug-in is enhanced to support the use of SYSID (the SYSIDNT of a region in a CICSPLEX) to select CICS tasks to debug.
- The DTCN Remote Plug-in is enhanced to support additional CICS "Application" resource types that are introduced in CICS TS 5.2.

  You can use these new resource types to select a program to debug:
  - Platform
  - Application
  - Operation
  - Version
- Code Coverage support for interactive remote.
- Code Coverage support for z/OS XL C.
- SET LIST BY SUBSCRIPT ON support for COBOL for MFI. This includes QUERY LIST BY SUBSCRIPT support.
- The CLIENTID option is added to the EQAOPTS DLAYDBGXRF command. This option allows a remote Debug Tool user using the enhanced DTSP Plug-in to identify a specific DB/2 client ID, and to only trap DB/2 stored procedures executed using that client ID.

Changes introduced with the PTF for APAR PI06312

- The REPOSITORY option is added to the EQAOPTS DLAYDBGXRF command. This option instructs Debug Tool to communicate with the Terminal Interface Manager to determine whether a user requests to debug an IMS transaction or DB/2 stored procedure associated with a generic user ID. This is an alternative to using the cross reference table data set.
- The Swap IMS Transaction Class and Run Transaction utility is enhanced to allow the user to manipulate the TEST option that is used for the debug message region. This allows the user to supply commands or preference files, and to direct the Debug Tool session to the remote user interface or to an alternate Terminal Interface Manager user ID.
- For CICS, provide protection of storage that was GETMAINed in the current task by a program that is not the active program. This support is enabled via INITPARM=(EQA0CPLT='STG') in the CICS startup parameters and is only available during a remote debug session.
- <PROGRAMDSCOMPILEDATE> and <PROGRAMDSCOMPILETIME> tags are added to the XML tags for code coverage. These two tags specify the compile data and time of the program source that is contained in the program data set.
- Support is added for generating a client certificate to the remote debugger if it does not exist.

- A note is added to the Coverage Utility User's Guide and Messages about the accuracy of execution counts (frequency counts) for single statement Program Areas (PAs).

Changes introduced with Debug Tool for z/OS Version 13.1

- A method for gathering code coverage is added for the generation, viewing, and reporting of code coverage by using the Debug Tool mainframe interface (MFI) as the engine. This support is provided for applications written in Enterprise COBOL and Enterprise PL/I that are compiled with the TEST compiler option and its suboption SEPARATE. This is enabled via the new EQAOPTS CCPROGSELECTDSN, CCOUTPUTDSN, and CCOUTPUTDSNALLOC commands.
- Debug Tool is enhanced to provide the automatic start of IMS message processing program (MPP) regions and dynamic routing of transactions. This allows a developer to dynamically start an MPP region, route a transaction to that MPP region, and at the end of the transaction shutdown the MPP region created for the developer thus reducing system resources.
- To help with the ease of use of the MFI mode of Debug Tool for some users, an option is added that enables breakpoints, the current line, and the line with found text to be identified by a character indicator. This feature is enabled via a new EQAOPTS ALTDISP command.
- Debug Tool is enhanced to support the following languages and platforms:
  - Enterprise COBOL for z/OS V5.1
  - Enterprise PL/I for z/OS V4.4
  - CICS TS V5.1
  - DB2 V11
  - IMS V13
  - z/OS, V2.1
  - C/C++ V2.1
- Debug Tool is enhanced to support JCL for Batch Debugging in the DTSP plug-in. This facility is used to instrument JCL to initiate a debug session from the DTSP plug-in.
- Support is added for an IMS transaction that is associated with a generic ID. A new feature is added to the consolidated Language Environment user exit (EQAD3CXT) to search a new cross-reference table for the user ID of a user who wants to debug a IMS transaction that is started from the web and is associated with a generic ID. This enables Debug Tool to debug these transactions that use a generic ID. The user ID from the cross-reference table is used to find the user's Debug Tool user exit data set (userid.DBGTOOL.EQAUOPTS), which specifies the TEST runtime parameters and the display device address. An option is added to the Debug Tool Utilities ISPF panel, "C IMS Transaction and User ID Cross Reference Table", to allow each user to update the new cross reference table.
- Support is added for tracing load modules loaded by an application. Commands TRACE LOAD and LIST TRACE LOAD are added for Debug Tool's MFI mode. This set of commands allows the user to get a trace of load modules loaded by the application. Start the trace by issuing TRACE LOAD START. Use LIST TRACE LOAD to display the trace. The trace includes load modules known to Debug Tool at the time the TRACE LOAD START command is entered and all that are loaded while the trace is active. End the trace by issuing TRACE LOAD END. Note that when the trace is ended, all trace information is deleted.

- Support is added for terminating an idle Debug Tool session that uses the Terminal Interface Manager. Debug Tool supports a timeout option via the EQAOPTS SESSIONTIMEOUT command. This command allows the system programmer to establish a maximum wait time for debug sessions that use a dedicated terminal or the Terminal Interface Manager. If the debug session exceeds the specified time limit without any user interaction, the session will be terminated with either a QUIT or QUIT DEBUG.
- Debug Tool Coverage Utility "Create HTML Targeted Coverage Report" is enhanced to allow the user to select from a list of COBOL Program-IDs, ignore changes to non-executable code, and produce a summary of the targeted lines with selectable HTML links.
- Adds IMS information to start and stop messages generated by the EQAOPTS STARTSTOPMSG command.
- Adds EQAOPTS STARTSTOPMSGDSN command and a Debug Tool Utilities option "Non-CICS Debug Session Start and Stop Message Viewer" to collect and view Debug Tool debugger session start and stop information.
- Delay debug mode is enhanced with an EQAOPTS DLAYDBGCND command to control CONDITION trapping. In addition, an EQAOPTS DLAYDBGXRF command is added so that delay debug mode can use the "IMS Transaction and User ID Cross Reference Table". Further, NOTEST is now handled in delay debug mode.
- A confirmation message is added to Debug Tool Utilities Option 6 "Debug Tool User Exit Data Set" to indicate that the updates have been saved into the EQAUOPTS data set.
- The ON and AT OCCURRENCE commands are enhanced for Enterprise PL/I to support qualifying data.
- LIST LDD and CLEAR LDD commands are added to display and remove LDD commands known to Debug Tool. LIST CC, CC START, and CC STOP commands are added to gather and display code coverage data.
- Two EQAOPTS commands are added for remote debug mode. The EQAOPTS HOSTPORTS command specifies the specific host port number or range of host port numbers on the host for a TCP/IP connection from the host to a workstation. The EQAOPTS TCPIPDATADSN command provides the data set name for TCPIP.DATA via the SYSTCPD DD NAME when no GLOBALTCPIPDATA statement is configured.
- A timestamp is added to the EQAY999* messages that the Terminal Interface Manager issues if the +T trace flag is on.
- Debug Tool is enhanced to allow for using GOTO or JUMPTO command for programs that are compiled with OPT and NOEJPD suboptions of the Enterprise COBOL TEST compile option when SET WARNING setting is OFF.
- An updated DTST transaction is included to write messages to the operator log when a user changes storage. These messages are intended to provide an audit trail of DTST storage changes.
- Support is added for remote Playback through the Playback Toolbar in the Debug View.
- The EQALANGP and EQALANGX modules are moved from Debug Tool's EQAW.SEQAMOD library to Common Component's IPV.SIPVMODA library. They are now aliases of IPVLANGP and IPVLANGX respectively. This removes duplication between the two tools.
- Appendix "Quick start guide for compiling and assembling programs for use with IBM Problem Determination Tools products" in the *Debug Tool User's Guide*

is removed because this has been placed in the *IBM Problem Determination Tools for z/OS Common Component: Customization Guide and User Guide* instead.

Chapter 1. Debug Tool commands

Debug Tool 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:

```
              ┌─OLD─┐
├──┬─DSNAME──dsn──┼─────┼────────────────┬───────────────────────────────┤
   │              ├─SHR─┤                │
   │              └─MOD─┘                │
   │            ┌─,───┐                  │
   │            ▼     │      ┌─OLD─┐     │
   ├─DSNAME──(────dsn──┴──)──┼─────┼─────┤
   │                         └─SHR─┘     │
   └─TEMP──TRACKS──(──primspc──,──secspc──)─┘
```

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──┬────────────────────────┬──=──sourceexpr──;────────────►◄
              └─<──┬─────────────┬──>──┘
                   └─receiverlen─┘
```

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 Debug Tool control when storage for a named controlled variable or aggregate is dynamically allocated by PL/I.

```
►►─AT─┬──────────────┬─ALLOCATE─┬─identifier──────────┬─command─;───►◄
      └─every_clause─┘          │   ┌─,──────────┐    │
                                ├─(─▼─identifier─┴─)──┤
                                └─*───────────────────┘
```

AT APPEARANCE

Gives Debug Tool control when the specified compile unit is found in storage.

```
►►─AT─┬──────────────┬─APPEARANCE─┬─cu_spec──────────┬─command─;────►◄
      └─every_clause─┘            │   ┌─,───────┐    │
                                  ├─(─▼─cu_spec─┴─)──┤
                                  └─*────────────────┘
```

AT CALL

Gives Debug Tool control when the application code attempts to call the specified entry point.

```
►►─AT─┬──────────────┬─CALL─┬─entry_name──────────┬─command─;───────►◄
      └─every_clause─┘      │   ┌─,──────────┐    │
                            ├─(─▼─entry_name─┴─)──┤
                            └─*───────────────────┘
```

AT CHANGE

Gives Debug Tool control when either the program or Debug Tool command changes the specified variable value or storage location, and, optionally, when the specified condition is met.

```
                                      ┌─GLOBAL──────────────┐
►►──AT──┬──────────────┬──CHANGE──┼─────────────────────┼───────────────────►
        └─every_clause─┘          └─LOCAL──┬─%CU─────┬──┘
                                           └─cu_spec─┘

►──┬─┬─reference──────────────────────────────┬─────────────┬─command──;──────►◄
   │ ├─'──reference──'──┬─────────────────┬───┤             │
   │ │                  └─WHEN──condition─┘   │             │
   │ └─%STORAGE──(──address──┬───────────┬──)─┘             │
   │                         └─,──length─┘                  │
   │      ┌─,──────────────────────────────────────┐        │
   │      ▼                                        │        │
   └─(────┬─reference──────────────────────────────┬┴──)────┘
          ├─'──reference──'────────────────────────┤
          └─%STORAGE──(──address──┬───────────┬──)─┘
                                  └─,──length─┘
```

AT CHANGE (remote)

Gives Debug Tool control when either the program or Debug Tool command changes the specified variable value.

```
►►──AT──CHANGE──┬─"──reference──"─┬──;──────────────────────────────────────►◄
                └─'──reference──'─┘
```

AT CURSOR (full-screen mode)

Provides a cursor controlled method for setting a statement breakpoint.

```
                        ┌─CURSOR─┐
►►──AT──┬────────┬──┴────────┴──;───────────────────────────────────────────►◄
        └─TOGGLE─┘
```

AT DATE (COBOL)

Gives Debug Tool control for each date processing statement within the specified block.

```
►►──AT──┬──────────────┬──DATE──┬─block_spec──────────────┬─command──;───────►◄
        └─every_clause─┘        │      ┌─,──────────┐     │
                                │      ▼            │     │
                                ├─(────block_spec───┴──)─┤
                                └─*──────────────────────┘
```

AT DELETE

Gives Debug Tool 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──┬──────────────┬──DELETE──┬─load_spec────────────┬──command──;────────►◄
        └─every_clause─┘          │    ┌─,───────────┐   │
                                  │    ▼             │   │
                                  ├─(────load_spec───┴─)─┤
                                  └─*────────────────────┘
```

AT ENTRY

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

```
►►──AT──┬──────────────┬──ENTRY──┬─block_spec────────────┬──┬──────────────────┬──►
        └─every_clause─┘         │    ┌─,────────────┐   │  └─WHEN──condition─┘
                                 │    ▼              │   │
                                 ├─(────block_spec───┴─)─┤
                                 └─*─────────────────────┘

►─command──;──────────────────────────────────────────────────────────────────►◄
```

AT ENTRY (remote)

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

```
►►──AT──ENTRY──block_spec──;──────────────────────────────────────────────────►◄
```

AT EXIT

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

```
►►──AT──┬──────────────┬──EXIT──┬─block_spec────────────┬──command──;─────────►◄
        └─every_clause─┘        │    ┌─,────────────┐   │
                                │    ▼              │   │
                                ├─(────block_spec───┴─)─┤
                                └─*─────────────────────┘
```

AT GLOBAL

Gives Debug Tool control for every instance of the specified `AT`-condition.

```
►►──AT──┬──────────────┬──GLOBAL──┬─ALLOCATE───────────────────────┬──────────────►
        └─every_clause─┘          ├─APPEARANCE─────────────────────┤
                                  ├─CALL───────────────────────────┤
                                  ├─DATE───────────────────────────┤
                                  ├─DELETE─────────────────────────┤
                                  ├─ENTRY──┬────────────────────┬──┤
                                  │        └─WHEN──condition────┘  │
                                  ├─EXIT───────────────────────────┤
                                  ├─LABEL──────────────────────────┤
                                  ├─LINE───────────────────────────┤
                                  ├─LOAD───────────────────────────┤
                                  ├─OCCURRENCE─────────────────────┤
                                  ├─PATH───────────────────────────┤
                                  └─STATEMENT──┬────────────────────┬┘
                                               └─WHEN──condition────┘

►─command──;──────────────────────────────────────────────────────────────────────►◄
```

AT GLOBAL LABEL (remote)

Gives Debug Tool control for every instance of the specified `AT-Label` condition.

```
►►──AT──GLOBAL──LABEL──;──────────────────────────────────────────────────────────►◄
```

AT LABEL

Gives Debug Tool control when execution has reached the specified statement label or group of labels.

```
►►──AT──┬──────────────┬──LABEL──┬─statement_label──────────────┬─command──;──────►◄
        └─every_clause─┘         ├─'statement_label'────────────┤
                                 │    ┌─,───────────────┐       │
                                 │    ▼                 │       │
                                 ├─(───statement_label──┴──)────┤
                                 │    ┌─,─────────────────┐     │
                                 │    ▼                   │     │
                                 ├─(───'statement_label'──┴──)──┤
                                 └─*──┬──────────────────────┬──┘
                                      └─LOCAL──┬─%CU─────┬───┘
                                               └─cu_spec─┘
```

AT LABEL (remote)

Gives Debug Tool control when execution reaches the statement label that you specify.

```
►►──AT LABEL──┬─statement_label─┬──;───────────────────────────────────────────────►◄
              └─*───────────────┘
```

AT LINE

Gives Debug Tool control at the specified line. See "AT STATEMENT" on page 7.

AT LOAD

Gives Debug Tool control when the specified load module is brought into storage.

```
►►──AT──┬──────────────┬──LOAD──┬─module_name────────────────┬──────────────────►
        └─every_clause─┘        │    ┌─,───────────┐          │
                                └─(──▼─module_name─┴──)───────┘

►──┬─load_spec─────────────┬──command──;───────────────────────────────────►◄
   │    ┌─,─────────┐      │
   ├─(──▼─load_spec─┴──)───┤
   └─*─────────────────────┘
```

AT LOAD (remote)

Gives Debug Tool control when the specified load module is brought into storage.

```
►►──AT──LOAD──module_name──;──────────────────────────────────────────────►◄
```

AT OCCURRENCE

Gives Debug Tool control on a language or Language Environment condition or exception.

```
►►──AT──┬──────────────┬──OCCURRENCE──┬─condition──────────────┬──command──;──────►◄
        └─every_clause─┘              │    ┌─,─────────┐        │
                                      └─(──▼─condition─┴──)─────┘
```

AT OFFSET (disassembly)

Gives Debug Tool control at the specified offset in the disassembly view.

```
►►──AT──OFFSET──┬─offset_spec──────────────┬──command──;───────────────────►◄
                │    ┌─,───────────┐        │
                └─(──▼─offset_spec─┴──)─────┘
```

AT PATH

Gives Debug Tool control when the flow of control changes (at a path point). `AT PATH` is identical to `AT GLOBAL PATH`.

```
►►──AT──┬──────────────┬──PATH──command──;─────────────────────────────────►◄
        └─every_clause─┘
```

AT Prefix (full-screen mode)

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

```
►►──AT──┬─────────┬──;──────────────────────────────────────────────►◄
        └─integer─┘
```

AT STATEMENT

Gives Debug Tool control at each specified statement or line within the given set of ranges, and, optionally, when the specified condition is met.

```
►►──AT──┬──────────────┬──┬───────────┬──┬─statement_id_range────────────┬──────►
        └─every_clause─┘  ├─LINE──────┤  │   ┌─,──────────────────┐      │
                          └─STATEMENT─┘  ├─(─▼─statement_id_range─┴──)──┤
                                         └─*─────────────────────────────┘

►──┬──────────────────┬──command──;──────────────────────────────────►◄
   └─WHEN──condition──┘
```

AT STATEMENT (remote)

Gives Debug Tool control at each specified statement or line.

```
►►──AT──┬───────────┬──statement_id──;──────────────────────────────►◄
        ├─LINE──────┤
        └─STATEMENT─┘
```

AT TERMINATION

Gives Debug Tool control when the application program is terminated.

```
►►──AT──TERMINATION──command──;─────────────────────────────────────►◄
```

BEGIN command (programming language neutral)

BEGIN and END delimit a sequence of one or more commands to form one longer command.

```
                ┌─────────┐
►►──BEGIN──;──▼─command─┴──END──;───────────────────────────────────►◄
```

block command (C and C++)

The `block` command allows you to group any number of Debug Tool commands into one command.

```
►►──{──┬─────────────┬──}──;────────────────────────────────────────►◄
       │ ┌─────────┐ │
       └─▼─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 Debug Tool 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:

```
                                                 ┌────────────┐
                                                 ▼            │
├──┬──┬──────────────────┬──┬──────────────┬──identifier──┴──┬──────────────┤
   │  └─┬──────┬─REFERENCE┘  └─ADDRESS──OF─┘                 │
   │    └─BY───┘                                             │
   │                        ┌───────────────────────────┐    │
   │                        ▼                           │    │
   └─┬──────┬─CONTENT──────┬─┬──────────────┬─identifier─┬┴────┘
     └─BY───┘              │ ├─ADDRESS──OF──┤            │
                           │ └─LENGTH───OF──┘            │
                           └─literal─────────────────────┘
```

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 Debug Tool commands.

```
►►──CLEAR──┬─┬─AT───────────────────┬──────────┬──;──────────────────────────►◄
           │ ├─AT_command───────────┤          │
           │ └─generic_AT_command───┘          │
           ├─DECLARE──┬─────────────────────┬──┤
           │          ├─identifier──────────┤  │
           │          │   ┌─,──────────┐    │  │
           │          └─(─▼─identifier─┴─)──┘  │
           ├─EQUATE──┬─────────────────────┬───┤
           │         ├─identifier──────────┤   │
           │         │   ┌─,──────────┐    │   │
           │         └─(─▼─identifier─┴─)──┘   │
           ├─LDD──┬─*──────────────────┬───────┤
           │      ├─number─────────────┤       │
           │      │   ┌─,──────┐       │       │
           │      ├─(─▼─number─┴─)─────┤       │
           │      └─ldd_number_range───┘       │
           ├─LOAD──┬─module_name────────────┬──┤
           │       │   ┌─,───────────┐      │  │
           │       └─(─▼─module_name─┴─)────┘  │
           ├─LOG───────────────────────────────┤
           ├─MEMORY────────────────────────────┤
           ├─MONITOR──┬──────────────────────┬─┤
           │          ├─number───────────────┤ │
           │          │   ┌─,──────┐         │ │
           │          ├─(─▼─number─┴─)───────┤ │
           │          ├─monitor_number_range─┤ │
           │          └─┬────────┬───────────┘ │
           │            └─CURSOR─┘             │
           ├─ON──┬───────────────────────┬─────┤
           │     ├─pli_condition─────────┤     │
           │     │   ┌─,─────────────┐   │     │
           │     └─(─▼─pli_condition─┴─)─┘     │
           ├─PROCEDURE──┬──────────────────────────┬┤
           │            ├─procedure_name───────────┤│
           │            │   ┌─,──────────────┐     ││
           │            └─(─▼─procedure_name─┴─)───┘│
           └─VARIABLES──┬───────────────────────┬───┘
                        ├─identifier────────────┤
                        │   ┌─,──────────┐      │
                        └─(─▼─identifier─┴─)────┘
```

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 AT──┬─GLOBAL──LABEL──────────┬──;──────────────────────────────►◄
              ├─LABEL──statement_label─┤
              └─*──────────────────────┘
```

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──;─────────────────────────────►◄
```

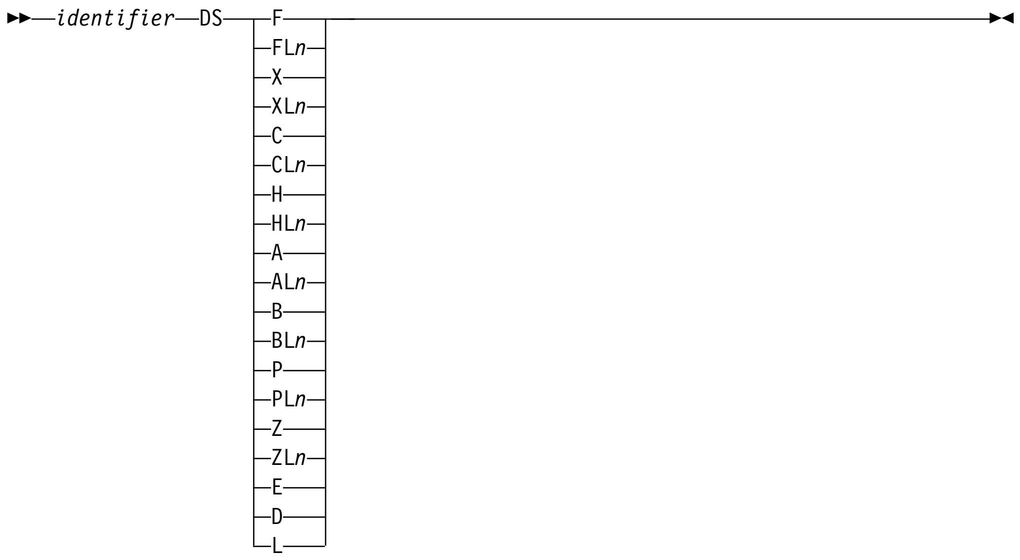
CURSOR command (full-screen mode)

The `CURSOR` command moves the cursor between the last saved position on the Debug Tool session panel (excluding the header fields) and the command line.

```
►►──CURSOR──;────────────────────────────────────────────────────────►◄
```

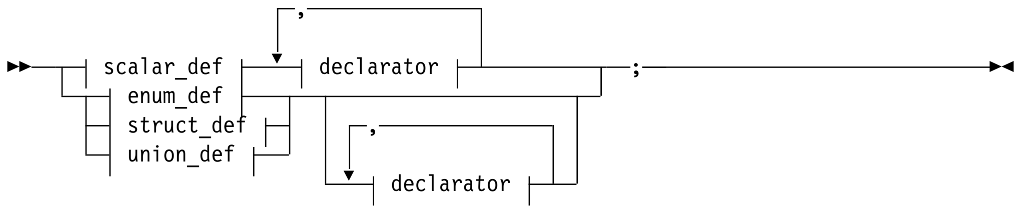
Declarations (assembler, disassembly, and LangX COBOL)

Use `declarations` to create session variables and tags effective during a Debug Tool session.

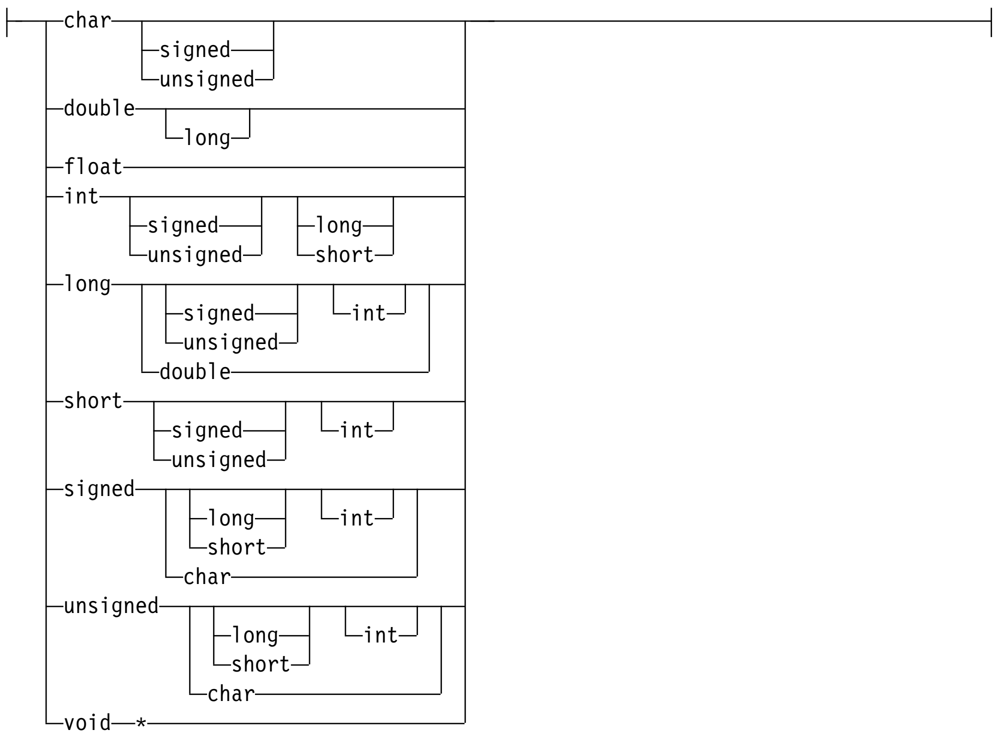


Declarations (C and C++)

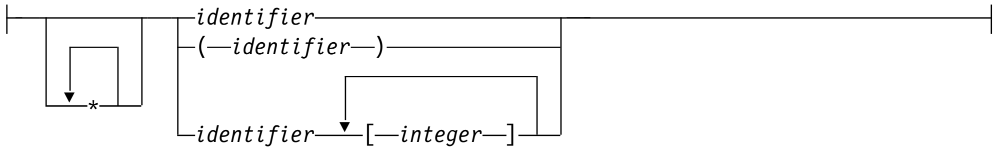
Use `declarations` to create session variables and tags effective during a Debug Tool session.



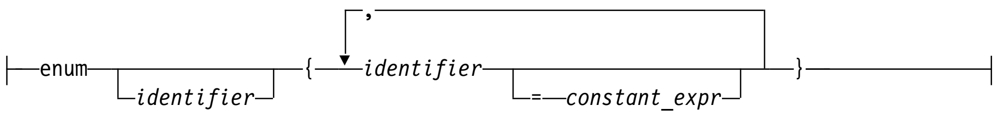
scalar_def:



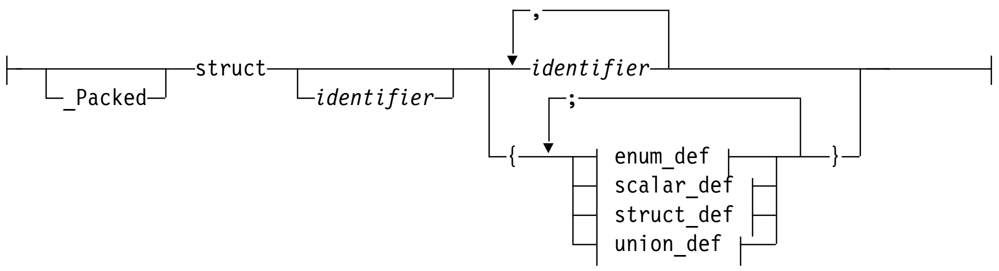
declarator:



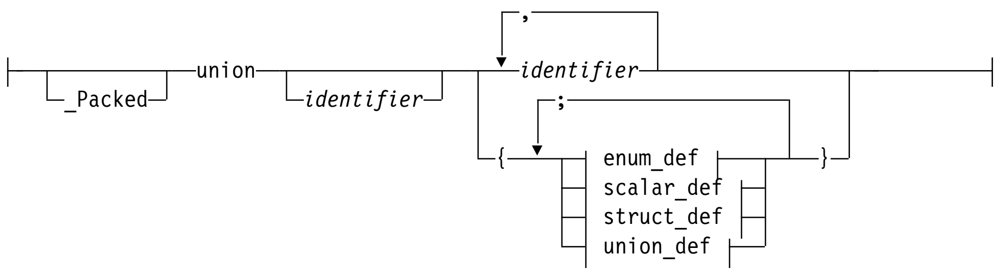
enum_def:



struct_def:

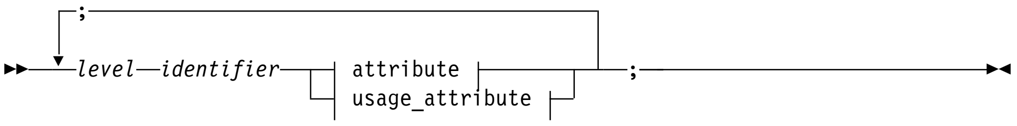


union_def:

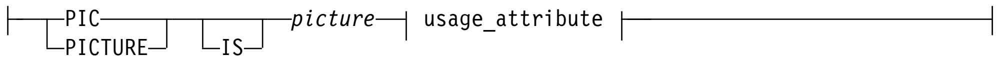


Declarations (COBOL)

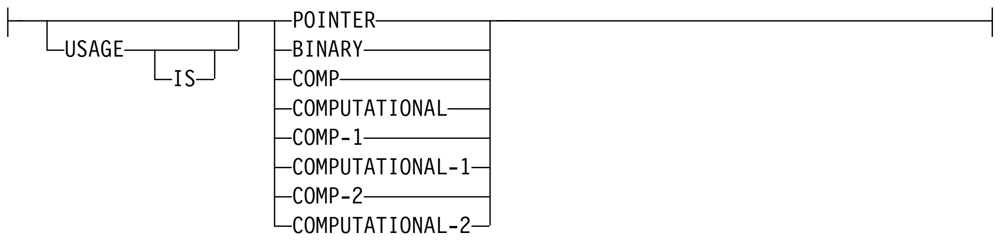
Use declarations to create session variables effective during a Debug Tool session.



attribute:

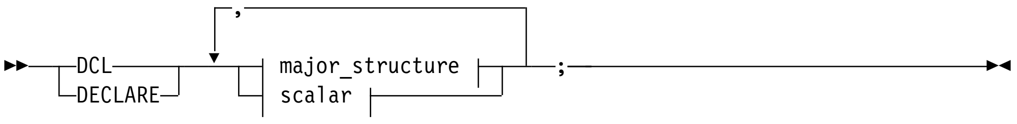


usage_attribute:

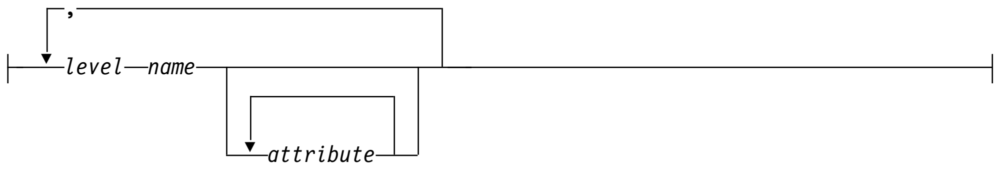


DECLARE command (PL/I)

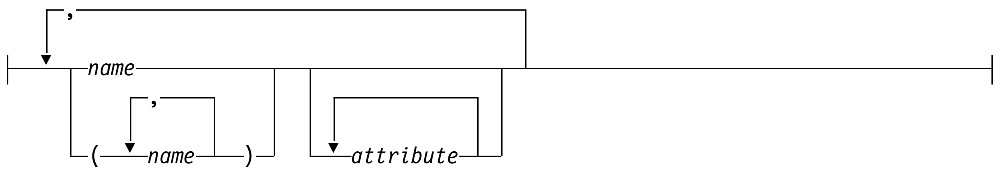
The DECLARE command creates session variables effective during a Debug Tool session.



major_structure:

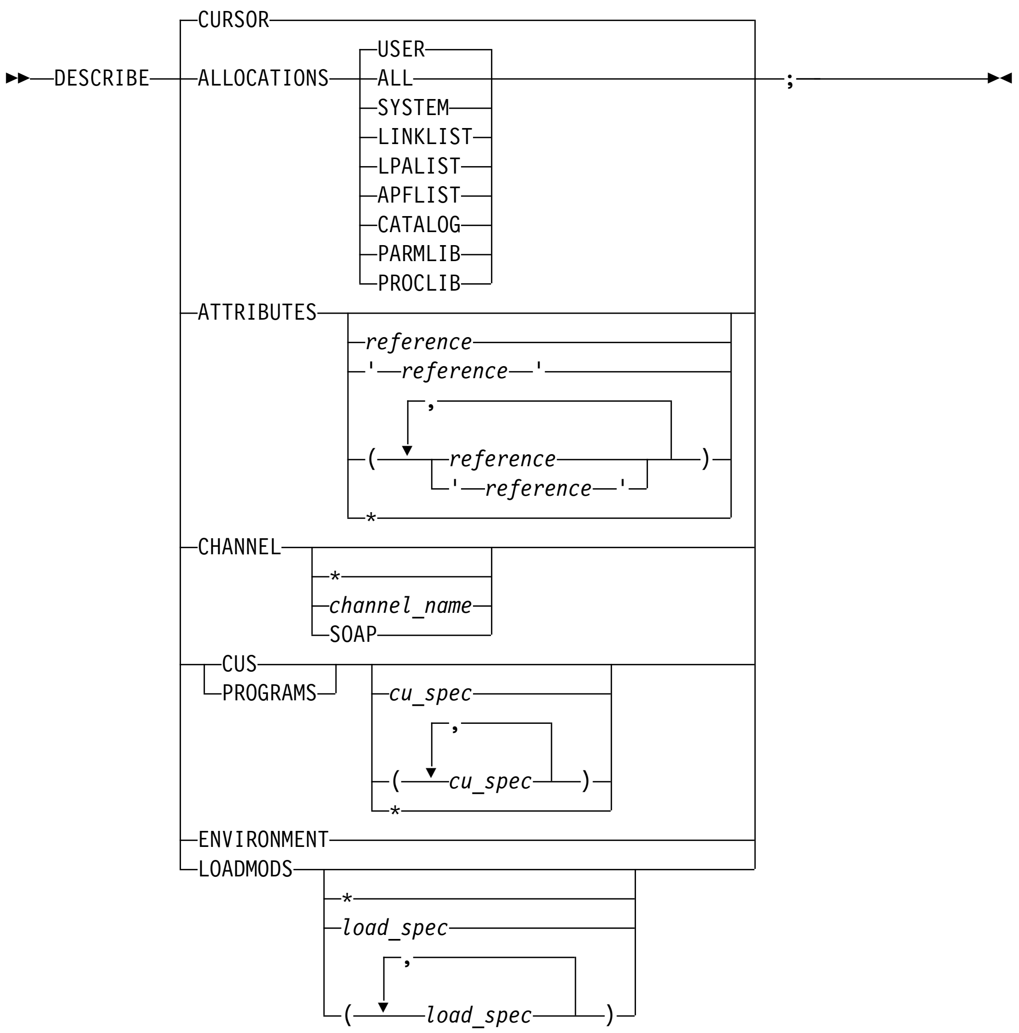


scalar:



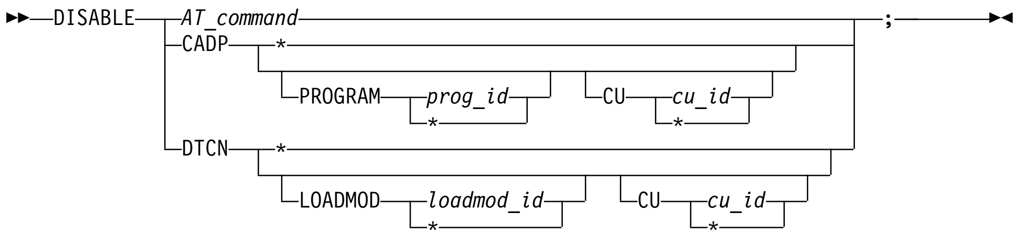
DESCRIBE command

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



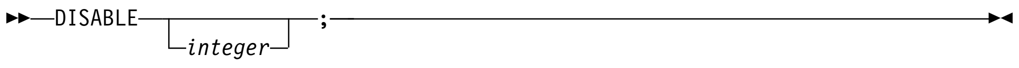
DISABLE command

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



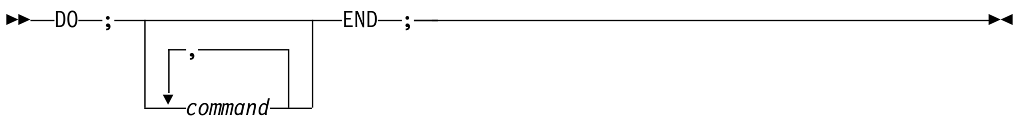
DISABLE prefix (full-screen mode)

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)

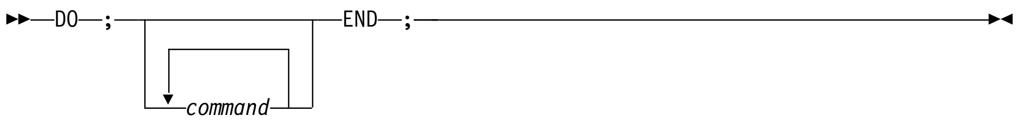
The DO command performs one or more commands that are collected into a group.



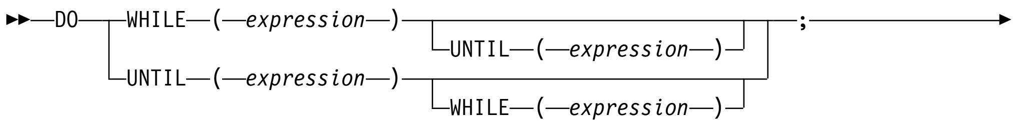
DO command (PL/I)

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

Simple



Repeating



```
►─┬───────────────┬─END─;─────────────────────────────────────────────►◄
  │ ┌───────────┐ │
  │ ▼           │ │
  └───command───┴─┘
```

Iterative

```
                          ┌─,───────────────┐
                          ▼                 │
►►─DO─reference─=─────┤ iteration ├──┴─;─┬───────────────┬─END─;─────────►◄
                                          │ ┌───────────┐ │
                                          │ ▼           │ │
                                          └───command───┴─┘
```

iteration:

```
├──expression─┬─────────────────────────────────────┬──────────────────►
              ├─BY─expression─┬───────────────────┬─┤
              │               └─TO─expression─────┘ │
              ├─TO─expression─┬───────────────────┬─┤
              │               └─BY─expression─────┘ │
              └─REPEAT─expression───────────────────┘

►─┬──────────────────────────────────────────────────────────┬───────────┤
  ├─WHILE─(─expression─)─┬─────────────────────────────────┬─┤
  │                      └─UNTIL─(─expression─)────────────┘ │
  └─UNTIL─(─expression─)─┬─────────────────────────────────┬─┘
                         └─WHILE─(─expression─)────────────┘
```

do/while command (C and C++)

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

```
►►─do─command─while─(─expression─)─;──────────────────────────────────►◄
```

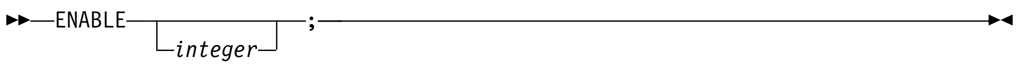
ENABLE command

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

```
►►─ENABLE─┬─AT_command────────────────────────────────────────────────┬─;────►◄
          ├─CADP──┬─*───────────────────────────────────────────┬─────┤
          │       └─┬──────────────────────┬─┬────────────────┬─┘     │
          │         └─PROGRAM─┬─prog_id─┬──┘ └─CU─┬─cu_id─┬───┘       │
          │                   └─*───────┘         └─*─────┘           │
          └─DTCN──┬─*───────────────────────────────────────────────┬─┘
                  └─┬─────────────────────────┬─┬─────────────────┬─┘
                    └─LOADMOD─┬─loadmod_id─┬──┘ └─CU─┬─cu_id─┬────┘
                              └─*──────────┘         └─*─────┘
```

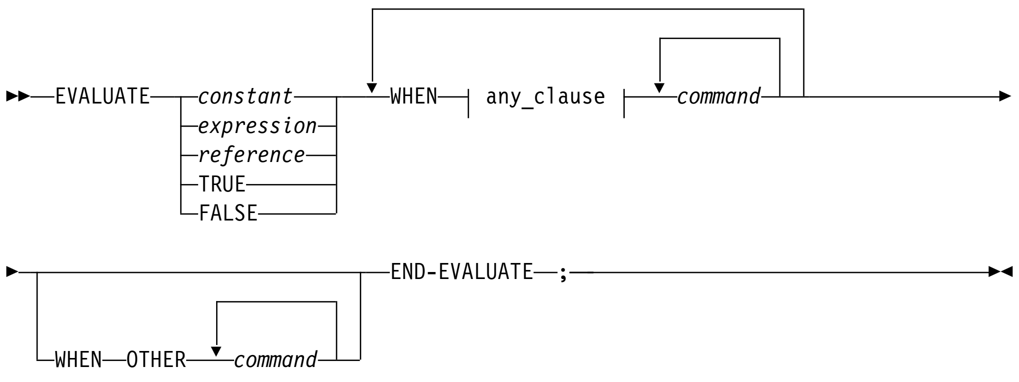
ENABLE prefix (full-screen mode)

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

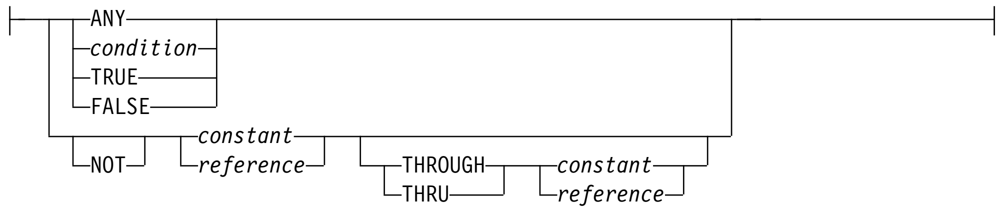


EVALUATE command (COBOL)

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

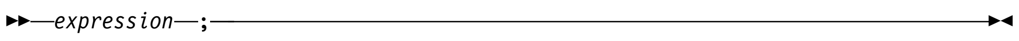


any_clause:



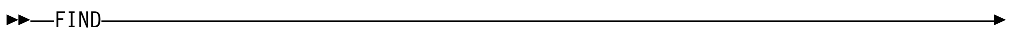
Expression command (C and C++)

The `Expression` command evaluates the given expression.



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.



```
►─┬──────────────────────────────────────────────────────────────┬─;───────►◄
  └─┬─string─┬──┬─────────────────────────────┬──┬───────┬──┬─────────┬─┘
    └─*──────┘  └─leftcolumn─┬───────────────┬─┘  ├─FIRST─┤  ├─CURSOR──┤
                             ├─rightcolumn─┤      ├─LAST──┤  ├─LOG─────┤
                             └─*───────────┘      ├─NEXT──┤  ├─MONITOR─┤
                                                  └─PREV──┘  └─SOURCE──┘
```

FINDBP command

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

```
►►──FINDBP──┬───────┬──┬──────────┬─;──────────────────────────────────►◄
            ├─FIRST─┤  ├─ENABLED──┤
            ├─LAST──┤  └─DISABLED─┘
            ├─NEXT──┤
            └─PREV──┘
```

for command (C and C++)

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

```
►►──for──(──┬────────────┬──;──┬────────────┬──;──┬────────────┬──)──────►
            └─expression─┘     └─expression─┘     └─expression─┘

►─command──;─────────────────────────────────────────────────────────────►◄
```

FREE command

The FREE command releases a file that is currently allocated.

```
►►──FREE──FILE──ddname──;────────────────────────────────────────────────►◄
```

GO command

The GO command causes Debug Tool to start or resume running your program.

```
►►──GO──┬────────┬──;────────────────────────────────────────────────────►◄
        └─BYPASS─┘
```

GOTO command

The GOTO command causes Debug Tool to resume program execution at the specified statement id.

```
►►──┬─GOTO───┬──statement_id──;──────────────────────────────────────────►◄
    └─GO──TO─┘
```

GOTO LABEL command

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

```
►►──┬─GOTO──┬──┬───────┬──┬─statement_label─────┬──;───────────────────►◄
    └─GO──TO─┘  └─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──┬──────────────┬──;──────►◄
        └─'──condition──'─┘                 └─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──┬──────┬───command─┴──┬────────────────────┬──END-IF──;──►◄
                   └─THEN─┘              │         ┌─────────┐ │
                                         │         ▼         │ │
                                         └─ELSE────command───┴─┘
```

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──┬──statement_id──;───────────────────────────────────►◄
    └─JUMP TO─┘
```

JUMPTO LABEL command

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

```
►►──┬─JUMPTO──┬──┬───────┬──┬─statement_label─────────┬──;─────────►◄
    └─JUMP TO─┘  └─LABEL─┘  └─'──statement_label──'───┘
```

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_command───────────────────────────────┤
          └─AT──┬──────────┬──┬─────────────────────┬─┘
                ├─ENABLED──┤  ├─ALLOCATE────────────┤
                └─DISABLED─┘  ├─APPEARANCE──────────┤
                              ├─CALL────────────────┤
                              ├─CHANGE──────────────┤
                              ├─DATE────────────────┤
                              ├─DELETE──────────────┤
                              ├─ENTRY───────────────┤
                              ├─EXIT────────────────┤
                              ├─GLOBAL──┬─ALLOCATE───┬─┤
                              │         ├─APPEARANCE─┤ │
                              │         ├─CALL───────┤ │
                              │         ├─DATE───────┤ │
                              │         ├─DELETE─────┤ │
                              │         ├─ENTRY──────┤ │
                              │         ├─EXIT───────┤ │
                              │         ├─LABEL──────┤ │
                              │         ├─LINE───────┤ │
                              │         ├─LOAD───────┤ │
                              │         ├─PATH───────┤ │
                              │         ├─STATEMENT──┤ │
                              │         └─SUSPENDED──┘ │
                              ├─LABEL───────────────┤
                              ├─LINE────────────────┤
                              ├─LOAD────────────────┤
                              ├─OCCURRENCE──────────┤
                              ├─OFFSET──────────────┤
                              ├─PATH────────────────┤
                              ├─STATEMENT───────────┤
                              ├─SUSPENDED───────────┤
                              └─TERMINATION─────────┘
```

LIST AT (remote)

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

```
►►──LIST──AT──┬─GLOBAL──LABEL──┬──;──────────────────────────►◄
              └─LABEL──┬───┬───┘
                       └─*─┘
```

LIST CALLS

Displays the dynamic chain of active blocks.

```
►►──LIST──CALLS──;───────────────────────────────────────────►◄
```

LIST CC command

Lists code coverage data.

```
        ┌─NOTEXECUTED─┐  ┌─NOSOURCE─┐
►►─LIST─CC─┼─────────────┼──┼──────────┼─;──────────────────────►◄
        ├─EXECUTED────┤  └─SOURCE───┘
        └─ALL─────────┘
```

LIST CONTAINER

Displays the contents of a CICS container.

```
►►─LIST─CONTAINER─┬──────────────┬─container_name──────────────►
                  └─channel_name─┘

►─┬──────────────────────────────────────────────────┬─────────►
  ├─(─index─)────────────────────────────────────────┤
  ├─(─sub_string_start─:─sub_string_end─)────────────┤
  └─(─sub_string_start─::─sub_string_length─)────────┘

►─┬─────────────────────────────────────────┬─;─────────────►◄
  └─XML─┬──────────────────────────────────┬┘
        └─(─┬─EBCDIC──────────────┬─)──────┘
            ├─ASCII───────────────┤
            └─CODEPAGE─(─ccsid─)──┘
```

LIST CURSOR (full-screen mode)

Provides a cursor controlled method for displaying variables, structures, and arrays.

```
         ┌─CURSOR─┐
►►─LIST──┴────────┴─;────────────────────────────────────────►◄
```

LIST DTCN or CADP

List the programs and compile units that were disabled by the DISABLE command.

```
►►─LIST─┬─DTCN─┬─;───────────────────────────────────────────►◄
        └─CADP─┘
```

LIST expression

Displays values of expressions.

```
►►──LIST──┬─────────────┬──┬─expression──────────────────────────┬──┬──;────────►◄
          │ ├─TITLED───┤│  ├─'──expression──'──────────────────────┤  │
          │ └─UNTITLED─┘│  │       (1)                            │  │
          │             │  ├─GROUP──────reference──────────────────┤  │
          │             │  │       ┌─,──────────────────────┐      │  │
          │             │  │       ▼                        │      │  │
          │             │  └─(────┬─expression─────────────┬─┴──)─┘  │
          │             │         ├─'──expression──'─────────┤         │
          │             │         │       (1)              │         │
          │             │         └─GROUP──────reference───┘         │
          └─TITLED──┬─────┬──────────────────────────────────────────┘
                    ├─*───┤
                    ├─FS──┤
                    ├─WSS─┤
                    ├─LS──┤
                    └─LOS─┘
```

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.

```
►►──L──┬──────────────────────────┬──────────────────────────────────►◄
       │  ┌─,──────┐              │
       │  ▼        │              │
       ├───integer─┴──────────────┤
       └─integer── -──integer─────┘
```

LIST FREQUENCY

Lists statement execution counts.

```
►►──LIST──FREQUENCY──┬──────────────┬──┬─statement_id_range───────────┬──;─────►◄
                     ├─LINES────────┤  │     ┌─,──────────────────┐   │
                     └─STATEMENTS───┘  │     ▼                    │   │
                                       ├─(───statement_id_range───┴─)─┤
                                       └─*────────────────────────────┘
```

LIST LAST

Displays a list of recent entries in the history table.

```
►►──LIST──┬─────────────────────┬──┬─HISTORY────┬──;──────────────────►◄
          │ ┌─LAST─┐            │  ├─LINES──────┤
          └─┴──────┴──integer───┘  ├─PATHS──────┤
                                   └─STATEMENTS─┘
```

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 Debug Tool procedures.

```
►►──LIST──NAMES──┬─────────┬──┬──────────────────────────────────────────┬──;─────►◄
                 └─pattern─┘  ├─BLOCK──┬─block_spec──────────────────────┤
                              │        ├─cu_spec─────────────────────────┤
                              │        │   ┌─,─────────────────┐         │
                              │        │   ▼                   │         │
                              │        └─(───┬─block_spec─┬──┴──)──────┤
                              │              └─cu_spec────┘              │
                              ├─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 Debug Tool `PROCEDURE` definitions.

```
►►──LIST──PROCEDURES──┬───────────────────┬──;────────────────────────────────►◄
                      ├─name──────────────┤
                      │    ┌─,────┐       │
                      │    ▼      │       │
                      └─(────name─┴──)────┘
```

LIST REGISTERS

Displays the current register contents.

```
            ┌─32BIT─┐
►►──LIST──┬─┤       ├─REGISTERS───────────┬──;─────────────────────────►◄
          │ └─64BIT─┘                     │
          │ ┌─LONG──┐                     │
          └─┤       ├─FLOATING──┬───────────┬┘
            └─SHORT─┘           └─REGISTERS─┘
```

LIST STATEMENT NUMBERS

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

```
►►──LIST──┬─LINE──────┬─NUMBERS──┬──────────────────────┬──;────────────►◄
          └─STATEMENT─┘          ├─block_spec───────────┤
                                 ├─cu_spec──────────────┤
                                 └─statement_id_range───┘
```

LIST STATEMENTS

Lists one or more statements or lines from a file.

```
►►──LIST──┬─LINES──────┬─statement_id_range──;──────────────────────────►◄
          └─STATEMENTS─┘
```

LIST STORAGE

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

```
►►──LIST──STORAGE──(──┬─address──────────┬──┬────────────────────────────┬──)──────►
                      ├─reference────────┤  └─┬──────────────┬─,──length─┘
                      └─'──reference──'──┘    └─,──offset────┘

►──┬────────────────────────────────────────────┬──;──────────────────────►◄
   └─XML──┬───────────────────────────────────┬─┘
          └─(──┬─EBCDIC──────────────────┬──)─┘
               ├─ASCII───────────────────┤
               └─CODEPAGE──(──ccsid──)───┘
```

LIST TRACE LOAD command

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

```
►►──LIST──TRACE──LOAD──;────────────────────────────────────────────────►◄
```

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.

```
►►──LOAD──┬─module_name──────────────┬──┬───────┬──;──────────────────────────►◄
          │    ┌─,──────────────┐    │  ├─LE────┤
          │    ▼                │    │  └─NONLE─┘
          └─(──┴──module_name───┴──)─┘
```

LOADDEBUGDATA (LDD)

The `LOADDEBUGDATA` (`LDD`) command requests that Debug Tool 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

```
►►──┬─LOADDEBUGDATA─┬─────────────────────────────────────────────────►
    └─LDD───────────┘

►──┬─┬───────────────────────┬─cu_name──────────────┬──;──────────────►◄
   │ └─load_module_name──::>─┘                      │
   │    ┌─,───────────────────────────────────┐     │
   │    ▼                                     │     │
   └─(──┴─┬───────────────────────┬─cu_name───┴──)──┘
          └─load_module_name──::>─┘
```

LDD for high-level language CU in explicit debug mode

```
►►──┬─LOADDEBUGDATA─┬─────────────────────────────────────────────────►
    └─LDD───────────┘

►──┬─load_module_name──::>──hidden_cu_name──────────────┬──;──────────►◄
   │    ┌─,───────────────────────────────────────┐     │
   │    ▼                                         │     │
   └─(──┴─load_module_name──::>──hidden_cu_name───┴──)──┘
```

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.

```
►►──MEMORY──┬─address─────────────┬──;────────────────────────────────►◄
            ├─reference───────────┤
            ├─'──reference──'─────┤
            └─simple_expression───┘
```

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.

```
►►──M──┬─────────────────────┬─────────────────────────────────►◄
       │ ┌─,──────────┐      │
       │ ▼            │      │
       ├───integer────┴──────┤
       └─integer── -──integer┘
```

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).

```
                 ┌─GLOBAL────────────────┐                        ┌─DEFAULT─┐
►►──MONITOR──┬───┼───────────────────────┼──┬─────────┬──command──┼─────────┼──┬──;──────►◄
             │   └─LOCAL──┬───────────┬──┘  └─integer─┘           └─HEX─────┘  │
             │            └─cu_spec───┘                                        │
             └─integer────┬─HEX──────┬─────────────────────────────────────────┘
                          └─DEFAULT──┘
```

MOVE command (COBOL)

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

```
►►──MOVE──┬─reference─┬──TO──reference──;───────────────────────►◄
          └─literal───┘
```

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 Debug Tool excludes by default are not included in the list that is displayed.

```
                         ┌─USER─┐                                  ┌─*──────────────────────┐
►►──NAMES──DISPLAY──┬────┼──────┼──EXCLUDED─┬──┬─LOADMODS─┬────────┼─pattern────────────────┼──;──────►◄
                    │    └─ALL──┘           │  └─CUS──────┘        │     ┌─,───────┐        │
                    └─INCLUDED──────────────┘                      │     ▼         │        │
                                                                   └─(────pattern──┴──)─────┘
```

NAMES EXCLUDE command

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

```
>>──NAMES──EXCLUDE──┬─┬─LOADMOD─┬──┬─pattern──────────┬─┬──;──────────────><
                    │ └─CU──────┘  │   ┌─,──────┐     │ │
                    │              │   ▼        │     │ │
                    │              └─(───pattern─┴──)─┘ │
                    └─CU──NOTEST────────────────────────┘
```

NAMES INCLUDE command

Use the NAMES INCLUDE command to indicate to Debug Tool 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 Debug Tool User's Guide for more information.

```
>>──NAMES──INCLUDE──┬─LOADMOD─┬──┬─name──────────┬──;─────────────────><
                    └─CU──────┘  │   ┌─,────┐    │
                                 │   ▼      │    │
                                 └─(───name─┴──)─┘
```

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.

```
>>──ON──┬─CONDITION──(──condition_name──)─────────┬─command──;───────────><
        ├─┬─ENDFILE──────┬──(──file_reference──)──┤
        │ ├─ENDPAGE──────┤                        │
        │ ├─KEY──────────┤                        │
        │ ├─NAME─────────┤                        │
        │ ├─PENDING──────┤                        │
        │ ├─RECORD───────┤                        │
        │ ├─TRANSMIT─────┤                        │
        │ └─UNDEFINEDFILE┘                        │
        ├─AREA────────────────────────────────────┤
        ├─ATTENTION───────────────────────────────┤
        ├─CONVERSION──────────────────────────────┤
        ├─ERROR───────────────────────────────────┤
        ├─FINISH──────────────────────────────────┤
        ├─FIXEDOVERFLOW───────────────────────────┤
        ├─OVERFLOW────────────────────────────────┤
        ├─SIZE────────────────────────────────────┤
        ├─STRINGRANGE─────────────────────────────┤
        ├─STRINGSIZE──────────────────────────────┤
        ├─SUBSCRIPTRANGE──────────────────────────┤
        ├─UNDERFLOW───────────────────────────────┤
        └─ZERODIVIDE──────────────────────────────┘
```

PANEL command (full-screen mode)

The PANEL command displays special panels.

```
>>─┬───────┬─┬─COLORS──────────┬─;──────────────────────────────><
   └─PANEL─┘ ├─LAYOUT─┬───────┬┤
             │        └─RESET─┘│
             ├─LISTINGS────────┤
             ├─PROFILE─────────┤
             └─SOURCES─────────┘
```

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:

```
                ┌──────────┐
                ▼          │
>>──PERFORM────command─────┴──END-PERFORM──;────────────────────><
```

Repeating:

```
>>──PERFORM──┬───────────────────────────────┬───────────────────>
             └─┬──────┬──TEST──┬─BEFORE─┬────┘
               └─WITH─┘        └─AFTER──┘

>─┬────────────────────────────────────────────────────┬─UNTIL──condition──────>
  └─VARYING──reference──FROM──reference──BY──reference─┘

   ┌──────────┐
   ▼          │
>──command────┴──END-PERFORM──;─────────────────────────────────><
```

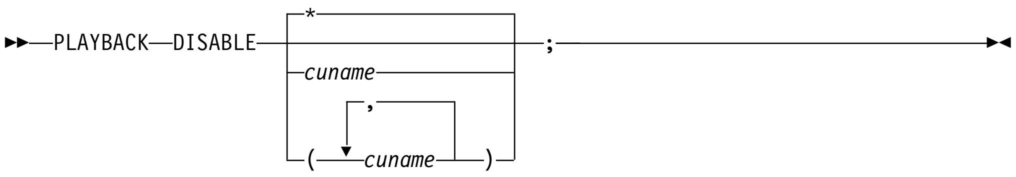
PLAYBACK BACKWARD command

The PLAYBACK BACKWARD command indicates to Debug Tool to perform STEP and RUNTO commands backward, starting from the current point and going to previous points.

```
>>──PLAYBACK──BACKWARD──;───────────────────────────────────────><
```

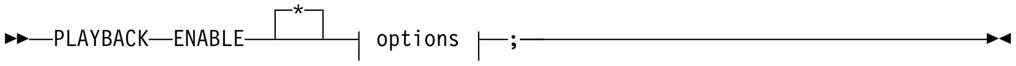
PLAYBACK DISABLE command

The PLAYBACK DISABLE command informs Debug Tool to stop recording runtime environment information and discard any information recorded thus far.

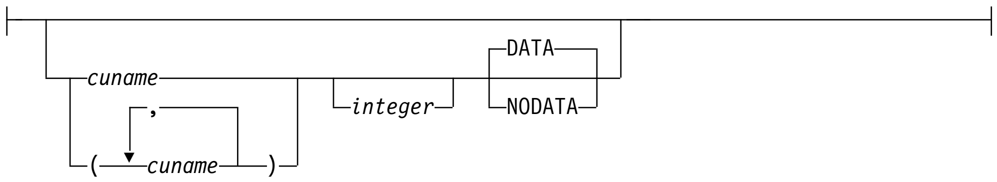


PLAYBACK ENABLE command

The PLAYBACK ENABLE command informs Debug Tool to begin recording the application runtime environment information (steps history and data history).

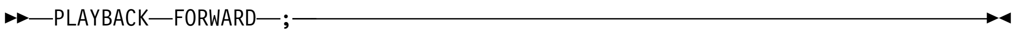


options:



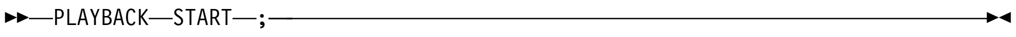
PLAYBACK FORWARD command

The PLAYBACK FORWARD command indicates to Debug Tool to perform STEP and RUNTO commands forward, starting from the current point and going to the next point.



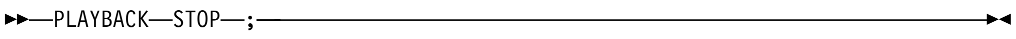
PLAYBACK START command

The PLAYBACK START command suspends normal debugging and informs Debug Tool to replay the steps it recorded.



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.



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.

```
                        ┌─CURSOR──┐
►►──POSITION──integer──┼─────────┼──;──────────────────────────────►◄
                        ├─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 Debug Tool setting, the current setting of all the Debug Tool 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 Debug Tool session and if an expression is specified, sets the return code.

```
►►──QUIT──┬─────────────────┬──;──────────────────────────────────►◄
          ├─(──expression──)─┤
          ├─ABEND────────────┤
          └─DEBUG──┬──────┬──┘
                   └─TASK─┘
```

QQUIT command

The QQUIT command ends a Debug Tool session without further prompting.

```
►►──QQUIT──;──────────────────────────────────────────────────────►◄
```

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 Debug Tool terminated.

```
►►──RESTORE──┬─SETTINGS─────────┬──;──────────────────────────────►◄
             ├─BPS──────────────┤
             ├─MONITORS─────────┤
             ├─BPS──MONITORS────┤
             └─MONITORS──BPS────┘
```

RETRIEVE command (full-screen mode)

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

```
              ┌─COMMAND─┐
►►──RETRIEVE──┴─────────┴──;──────────────────────────────────────►◄
```

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──┬──────────────┬──;────────────────────────────────────►◄
           └─statement_id─┘
```

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.

```
                                                  ┌─CURSOR──┐
►►──┬────────┬──┬──┬─DOWN──┬──┬─────────┬─┬──┼─────────┼─;──────────────────────►◄
    └─SCROLL─┘  │  ├─LEFT──┤  ├─CSR─────┤ │  ├─LOG─────┤
                │  ├─NEXT──┤  ├─DATA────┤ │  ├─MEMORY──┤
                │  ├─RIGHT─┤  ├─HALF────┤ │  ├─MONITOR─┤
                │  └─UP────┘  ├─integer─┤ │  └─SOURCE──┘
                │             ├─MAX─────┤ │
                │             └─PAGE────┘ │
                ├─BOTTOM──────────────────┤
                ├─TO──integer─────────────┤
                └─TOP─────────────────────┘
```

SELECT command (PL/I)

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

```
►►──SELECT──┬─────────────────┬─;─────────────────────────────────────────────►
            └─(──reference──)─┘

►──┬──────────────────────────────────────────┬──┬──────────────────────┬─END──;───►◄
   │ ┌────────────────────────────────────┐   │  └─OTHERWISE──command───┘
   │ │                 ┌─,──────┐          │   │
   │ ▼                 ▼        │          │   │
   └───WHEN──(───expression───┴──)──command─┴───┘
```

SET ASSEMBLER ON/OFF

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

```
                    ┌─ON──┐
►►──SET──ASSEMBLER──┼─────┼─;──────────────────────────────────────────►◄
                    └─OFF─┘
```

SET ASSEMBLER STEPOVER

The `SET ASSEMBLER STEPOVER` command specifies how Debug Tool processes STEP OVER commands in assembler compile units.

```
                              ┌─EXTONLY─┐
►►──SET──ASSEMBLER──STEPOVER──┼─────────┼──;──────────────────────────►◄
                              └─EXTINT──┘
```

SET AUTOMONITOR

Controls the monitoring of data items at the statement that Debug Tool runs next, ran most recently, or both.

```
                         ┌─NOLOG─┐  ┌─CURRENT──┐
                    ┌─ON─┼───────┼──┼──────────┼─┐
                    │    └─LOG───┘  ├─PREVIOUS─┤ │
                    │               └─BOTH─────┘ │
►►──SET──AUTOMONITOR─┼───────────────────────────┼─;─────────────►◄
                    └─OFF────────────────────────┘
```

SET CHANGE

Controls the frequency of checking the `AT CHANGE` breakpoints.

```
               ┌─STATEMENT─┐
►►──SET──CHANGE─┼───────────┼─;───────────────────────────────────►◄
               ├─ALL───────┤
               ├─BLOCK─────┤
               ├─LINE──────┤
               └─PATH──────┘
```

SET COLOR (full-screen and line mode)

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

```
►►──SET──COLOR──┤ color_attributes ├──┤ UI_elements ├──;──────────►◄
```

color_attributes:

```
  ┌─CYCLE──────────────────────────────────────┐
├─┼────────────────────────────────────────────┼──────────────────┤
  └─┬─BLUE──────┬──┬─BLINK─────┬──┬─HIGH─┬─────┘
    ├─GREEN─────┤  ├─NONE──────┤  └─LOW──┘
    ├─PINK──────┤  ├─REVERSE───┤
    ├─RED───────┤  └─UNDERLINE─┘
    ├─TURQUOISE─┤
    ├─WHITE─────┤
    └─YELLOW────┘
```

UI_elements:

```
   ┌─CURSOR────────────────────────┐
├──┼───────────────────────────────┼─────────────────────────────────────────────────┤
   └─┬─COMMAND──LINE─────────────┬─┘
     ├─LOG──LINES────────────────┤
     ├─MEMORY───┬─ADDRESS──────┬─┤
     │          ├─CHARACTER────┤ │
     │          ├─HEXADECIMAL──┤ │
     │          ├─INFORMATION──┤ │
     │          └─OFFSET───────┘ │
     ├─MONITOR───┬─AREA──┬───────┤
     │           └─LINES─┘       │
     ├─PROGRAM──OUTPUT───────────┤
     ├─SOURCE───┬─AREA─────────┬─┤
     │          ├─BREAKPOINTS──┤ │
     │          ├─CURRENT──────┤ │
     │          ├─PREFIX───────┤ │
     │          └─SUFFIX───────┘ │
     ├─TARGET───┬───────┬────────┤
     │          └─FIELD─┘        │
     ├─TEST───┬─INPUT──┬─────────┤
     │        └─OUTPUT─┘         │
     ├─TITLE───┬─FIELDS──┬───────┤
     │         └─HEADERS─┘       │
     ├─TOFEOF───┬────────┬───────┤
     │          └─MARKER─┘       │
     └─WINDOW──HEADERS───────────┘
```

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.

```
                  ┌─ON──┐
▶▶──SET──DBCS─────┼─────┼──;───────────────────────────────────────────────▶◀
                  └─OFF─┘
```

SET DEFAULT DBG

Defines a default partitioned data set DD name or DS name that Debug Tool searches through to locate the .dbg files.

```
▶▶──SET──DEFAULT──DBG──┬──────────────────┬──;─────────────────────────────▶◀
                       ├─ddname───────────┤
                       ├─dsn──────────────┤
                       │     ┌─,──┐       │
                       │     ▼    │       │
                       └─(────dsn─┴──)────┘
```

SET DEFAULT LISTINGS

Defines a default partitioned data set DD name or DS name whose members are searched for program source, listings, or separate debug files.

```
►►──SET──DEFAULT──LISTINGS──┬─────────────────┬──;──────────────────────►◄
                            ├─ddname──────────┤
                            ├─dsn─────────────┤
                            │    ┌─,───┐      │
                            └─(──▼─dsn─┴──)───┘
```

SET DEFAULT MDBG

Defines a default partitioned data set DD name or DS name that Debug Tool searches through to locate the .mdbg files.

```
►►──SET──DEFAULT──MDBG──┬─────────────────┬──;──────────────────────────►◄
                        ├─ddname──────────┤
                        ├─dsn─────────────┤
                        │    ┌─,───┐      │
                        └─(──▼─dsn─┴──)───┘
```

SET DEFAULT SCROLL (full-screen mode)

Sets the default scroll amount that is used when a `SCROLL` command is issued without the amount specified.

```
►►──SET──DEFAULT──SCROLL──┬─CSR─────┬──;────────────────────────────────►◄
                          ├─DATA────┤
                          ├─HALF────┤
                          ├─integer─┤
                          ├─MAX─────┤
                          └─PAGE────┘
```

SET DEFAULT VIEW

Controls the default view for assembler compile units.

```
►►──SET──DEFAULT──VIEW──┬─STANDARD─┬──;─────────────────────────────────►◄
                        └─NOMACGEN─┘
```

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.

```
                    ┌─ON──┐
►►──SET──DISASSEMBLY──┴─OFF─┴──;─────────────────────────────────────►◄
```

SET DYNDEBUG

Controls whether to activate the Dynamic Debug facility.

```
                 ┌─ON──┐
►►──SET──DYNDEBUG──┤     ├──;──────────────────────────────────────────►◄
                 └─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).

```
                ┌─ON──┐
►►──SET──EXECUTE──┤     ├──;───────────────────────────────────────────►◄
                └─OFF─┘
```

SET EXPLICITDEBUG

Controls whether explicit debug mode is active.

```
                          ┌─ON──┐
►►──SET──EXPLICITDEBUG────┤     ├──;──────────────────────────────────►◄
                          └─OFF─┘
```

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──FIND──BOUNDS───┬───────────────────────────────────┬──;──────►◄
                        └─leftcolumn───┬─rightcolumn─┬──────┘
                                       └─*───────────┘
```

SET FREQUENCY

Controls whether statement executions are counted.

```
                     ┌─ON──┐
►►──SET──FREQUENCY───┤     ├──┬──────────────────────┬──;─────────────►◄
                     └─OFF─┘  ├─cu_spec──────────────┤
                              │    ┌─,───────┐       │
                              │    ▼         │       │
                              └─(──┴─cu_spec─┴──)────┘
```

SET HISTORY

Specifies whether entries to Debug Tool are recorded in the history table and optionally adjusts the size of the table.

```
                   ┌─ON──┐
►►──SET──HISTORY───┤     ├──┬───────────┬──;──────────────────────────►◄
                   └─OFF─┘  └─integer───┘
```

SET IGNORELINK

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

```
                      ┌─ON──┐
►►──SET──IGNORELINK───┤     ├──;──────────────────────────────────────►◄
                      └─OFF─┘
```

SET INTERCEPT (C and C++)

Intercepts input to and output from specified files.

```
                        ┌─ON──┐
►►──SET──INTERCEPT──────┼─────┼────FILE──file_spec────;──────────────────────►◄
                        └─OFF─┘
```

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

Intercepts input to and output from the console.

```
                        ┌─ON──┐
►►──SET──INTERCEPT──────┼─────┼──CONSOLE──;──────────────────────────────────►◄
                        └─OFF─┘
```

SET INTERCEPT (COBOL, remote debug mode)

Intercepts output from COBOL DISPLAY statements.

```
                        ┌─ON──┐
►►──SET──INTERCEPT──────┼─────┼──;───────────────────────────────────────────►◄
                        └─OFF─┘
```

SET KEYS (full-screen and line mode)

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

```
                   ┌─ON──┐   ┌─12─┐
►►──SET──KEYS──────┼─────┼───┼────┼──;───────────────────────────────────────►◄
                   └─OFF─┘   └─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 Debug Tool displays elements in an array as they are stored in memory.

```
                                   ┌─ON──┐
►►──SET──LIST──BY──SUBSCRIPT───────┼─────┼──;────────────────────────────────►◄
                                   └─OFF─┘
```

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

Controls whether Debug Tool displays elements in an array as they are stored in memory.

```
                           ┌─ON──┐
►►──SET──LIST──BY──SUBSCRIPT──┼─────┼──;──────────────────────────────────►◄
                           └─OFF─┘
```

SET LIST TABULAR

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

```
                        ┌─OFF─┐
►►──SET──LIST──TABULAR──┴─ON──┴──;────────────────────────────────────────►◄
```

SET LOG

Controls whether each command that Debug Tool 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.

```
            ┌─ON──────────────────────────┐
►►──SET──LOG──┼─────────────────────────────┼──;──────────────────────────►◄
            │                   ┌─OLD─┐   │
            ├─ON──FILE──fileid──┼─────┼───┤
            │                   └─MOD─┘   │
            ├─OFF─────────────────────────┤
            └─KEEP──count─────────────────┘
```

SET LOG NUMBERS (full-screen and line mode)

Controls whether line numbers are shown in the log window.

```
                         ┌─ON──┐
►►──SET──LOG──NUMBERS────┼─────┼──;───────────────────────────────────────►◄
                         └─OFF─┘
```

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

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

```
                       ┌─ON──┐
►►──SET──LONGCUNAME────┼─────┼──;─────────────────────────────────────────►◄
                       └─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───┬──┬─ON──┬──┬──;─────────────────────►◄
                  │  ├─DATATYPE─┤  └─OFF─┘  │
                  │  ├─NUMBERS──┤           │
                  │  └─WRAP─────┘           │
                  └─LIMIT──integer──────────┘
```

SET MSGID

Controls whether the Debug Tool messages are displayed with the message prefix identifiers.

```
                ┌─ON──┐
►►──SET──MSGID──┴─────┴──;───────────────────────────────────────────►◄
                └─OFF─┘
```

SET NATIONAL LANGUAGE

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

```
        ┌─NATIONAL─┐
►►──SET─┴──────────┴──LANGUAGE──language_code──;─────────────────────►◄
```

SET PACE

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

```
►►──SET──PACE──number──;─────────────────────────────────────────────►◄
```

SET PFKEY

Associates a Debug Tool command with a Program Function key (PF key).

```
►►──SET──PFn──┬────────┬──=──┬─────────┬──;──────────────────────────►◄
              └─string─┘     └─command─┘
```

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.

```
                              ┌─CYCLE────────────────────────┐
►►──SET──PROGRAMMING──LANGUAGE─┼──────────────────────────────┼──;──────────►◄
                              ├─AUTOMATIC────────────────────┤
                              ├─HOLD─────────────────────────┤
                              └─┬─ASSEMBLER───┬──┬──────┬─┘
                                ├─C───────────┤  └─HOLD─┘
                                ├─COBOL───────┤
                                ├─DISASSEMBLY─┤
                                ├─LANGXCOBOL──┤
                                └─PLI─────────┘
```

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.

```
►►──┬─────┬──QUALIFY──┬─BLOCK──block_spec─────────────┬──;──────────────►◄
    └─SET─┘           ├─┬─CU──────┬──┬─cu_spec─┬──┤
                      │ └─PROGRAM─┘  └─address─┘  │
                      ├─LOAD──┬───────────┬───────┤
                      │       └─load_spec─┘       │
                      ├─RESET─────────────────────┤
                      ├─RETURN────────────────────┤
                      └─UP────────────────────────┘
```

SET REFRESH (full-screen mode)

Controls screen refreshing.

```
                  ┌─ON──┐
►►──SET──REFRESH──┼─────┼──;────────────────────────────────────────►◄
                  └─OFF─┘
```

SET RESTORE

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

```
►►──SET──RESTORE──┬─SETTINGS─┬──┬─NOAUTO─┬──;─────────────────────────►◄
                  ├─BPS──────┤  └─AUTO───┘
                  └─MONITORS─┘
```

SET REWRITE (full-screen mode)

Forces a periodic screen rewrite during long sequences of output.

```
►►──SET──REWRITE──┬───────┬──number──;────────────────────────────────►◄
                  └─EVERY─┘
```

SET REWRITE (remote debug mode)

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

```
►►──SET──REWRITE──┬───────┬──number──;────────────────────────────────►◄
                  └─EVERY─┘
```

SET SAVE

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

```
►►──SET──SAVE──┬─SETTINGS──┬─NOAUTO─┬──┬─────────────────────────┬─┬──;───────►◄
               │           ├─AUTO───┤  └─FILE──┬─*─────────┬─────┘ │
               │           └─ONCE───┘          └─setfileid─┘       │
               ├─BPS──┬─NOAUTO─┬──┬────────────────────────┬───────┤
               │      └─AUTO───┘  └─FILE──┬─*────────┬─────┘       │
               │                          └─bpfileid─┘             │
               └─MONITORS──┬─NOAUTO─┬──────────────────────────────┘
                           └─AUTO───┘
```

SET SCREEN (full-screen and line mode)

Controls how information is displayed on the screen.

```
                ┌─ON──────────────────────────────────────┐
►►──SET──SCREEN─┼──┬───────────┬──┬─────────────────────┬─┼──;────────────►◄
                │  ├─CYCLE─────┤  │  ┌───────────────┐  │ │
                │  └─integer───┘  │  ▼               │  │ │
                │                 └────┬─LOG─────┬───┴──┘ │
                │                      ├─MEMORY──┤        │
                │                      ├─MONITOR─┤        │
                │                      └─SOURCE──┘        │
                └─OFF─────────────────────────────────────┘
```

SET SCROLL DISPLAY (full-screen mode)

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

```
                             ┌─ON──┐
►►──SET──SCROLL──DISPLAY─────┼─────┼──;──────────────────────────────────────►◄
                             └─OFF─┘
```

SET SEQUENCE (PL/I)

Controls whether Debug Tool interprets data after column 72 in a commands or preference file as a sequence number.

```
                       ┌─OFF─┐
►►──SET──SEQUENCE──────┴─ON──┴───;───────────────────────────────────────────►◄
```

SET SOURCE

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

```
                                 ┌─,───────┐
                       ┌─ON──┐   ▼         │
►►──SET──SOURCE────────┼─────┼──(──cu_spec─┴──)──┬────────┬──;───────────────►◄
                       └─OFF─┘                   └─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.

```
                       ┌─ON──┐
►►──SET──SUFFIX────────┼─────┼──;────────────────────────────────────────────►◄
                       └─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 Debug Tool warning messages and whether exceptions are reflected to the application program.

```
                        ┌─ON──┐
►►──SET──WARNING────────┼─────┼──;───────────────────────────────────────────►◄
                        └─OFF─┘
```

SET command (COBOL)

The SET command assigns a value to a COBOL reference.

```
►►──SET──reference──TO──┬─reference─┬──;──────────────────────────────►◄
                        ├─literal───┤
                        └─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──┬─────────┬──;──────────────────────────────────────────────►◄
          └─integer─┘
```

STEP command

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

```
                         ┌─INTO───┐
►►──STEP──┬─────────┬────┼────────┼──;────────────────────────────────►◄
          ├─integer─┤    ├─OVER───┤
          └─*───────┘    └─RETURN─┘
```

STORAGE command

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

```
►►──STORAGE──(──┬─address─────────┬──┬────────────────────────┬──)──=──value──;──────►◄
                ├─reference───────┤  └─┬──────────┬──,──length─┘
                └─'──reference──'─┘    └─,──offset─┘
```

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:

```
├──┬───────────────────┬──┤ default_clause ├──┬───────────────────┬──┤
   │ ┌───────────────┐ │                      │ ┌───────────────┐ │
   │ ▼               │ │                      │ ▼               │ │
   └───┤ case_clause ├─┴─┘                    └───┤ case_clause ├─┴─┘
```

case_clause:

```
├──case──case_expression──:──┬──────────────┬──────────────────────────┤
                             │ ┌──────────┐ │
                             │ ▼          │ │
                             └──command───┴─┘
```

default_clause:

```
├──┬──────────────────────────────┬──────────────────────────────────────┤
   └─default──:──┬──────────────┬─┘
                 │ ┌──────────┐ │
                 │ ▼          │ │
                 └──command───┴─┘
```

SYSTEM command

The SYSTEM command lets you issue TSO commands during a Debug Tool session.

```
►►──┬─SYS────┬──system_command──;──────────────────────────────────►◄
    └─SYSTEM─┘
```

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.

```
►►──TRACE──LOAD──┬─START─┬──;──────────────────────────────────────►◄
                 └─STOP──┘
```

TRIGGER command

The TRIGGER command raises the specified AT-condition in Debug Tool, or it raises the specified programming language condition in your program.

```
►►──TRIGGER──┬─AT──┬────────┬──────────────────────────────┬─;───────────►◄
             │     └─CURSOR─┘                              │
             └┬─condition─────────────────────────────────┬┘
              ├─AT──ALLOCATE──┬─identifier─┬──────────────┤
              │               └─*──────────┘              │
              ├─AT──APPEARANCE──┬─cu_spec─┬───────────────┤
              │                 └─*───────┘               │
              ├─AT──CALL──┬─entry_name─┬──────────────────┤
              │           └─*──────────┘                  │
              ├─AT──CHANGE──┬─reference─────────────┬─────┤
              │             └─┤ storage_clause ├────┘     │
              ├─AT──DATE──┬─block_spec─┬──────────────────┤
              │           └─*──────────┘                  │
              ├─AT──DELETE──┬─load_spec─┬─────────────────┤
              │             └─*─────────┘                 │
              ├─AT──ENTRY──┬─block_spec─┬─────────────────┤
              │            └─*──────────┘                 │
              ├─AT──EXIT──┬─block_spec─┬──────────────────┤
              │           └─*──────────┘                  │
              ├─AT──GLOBAL──┬─APPEARANCE─┬────────────────┤
              │             ├─CALL───────┤                │
              │             ├─DATE───────┤                │
              │             ├─DELETE─────┤                │
              │             ├─ENTRY──────┤                │
              │             ├─EXIT───────┤                │
              │             ├─LABEL──────┤                │
              │             ├─LINE───────┤                │
              │             ├─LOAD───────┤                │
              │             ├─PATH───────┤                │
              │             └─STATEMENT──┘                │
              ├─AT──LABEL──┬─statement_label─┬────────────┤
              │            └─*───────────────┘            │
              ├─AT──┬──────┬──┬─stmt_id_spec─┬────────────┤
              │     └─LINE─┘  └─*────────────┘            │
              ├─AT──LOAD──┬─load_spec─┬───────────────────┤
              │           └─*─────────┘                   │
              ├─AT──OCCURRENCE──condition─────────────────┤
              ├─AT──PATH──────────────────────────────────┤
              └─AT──┬───────────┬──┬─stmt_id_spec─┬───────┘
                    └─STATEMENT─┘  └─*────────────┘
```

storage_clause:

```
├──%STORAGE──(──address──┬───────────┬──)──────────────────────────┤
                         └─,──length─┘
```

TSO command (z/OS)

The TSO command lets you issue TSO commands during a Debug Tool session and is valid only in a TSO environment.

```
►►──TSO──tso_command──;───────────────────────────────────────────►◄
```

USE command

The USE command causes the Debug Tool 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 Debug Tool full-screen session panel.

```
                         ┌─CURSOR──┐
►►──┬────────┬──CLOSE────┼─────────┼──;────────────────────────────────►◄
    └─WINDOW─┘           ├─LOG─────┤
                         ├─MEMORY──┤
                         ├─MONITOR─┤
                         └─SOURCE──┘
```

WINDOW OPEN

Opens a previously-closed window in the Debug Tool full-screen session panel.

```
►►──┬────────┬──OPEN──┬─────────┬──;───────────────────────────────────►◄
    └─WINDOW─┘        ├─LOG─────┤
                      ├─MEMORY──┤
                      ├─MONITOR─┤
                      └─SOURCE──┘
```

WINDOW SIZE

Controls the relative size of currently visible windows in the Debug Tool full-screen session panel.

```
                                           ┌─CURSOR──┐
►►──┬────────┬──SIZE──┬─────────┬──────────┼─────────┼──;──────────────►◄
    └─WINDOW─┘        └─integer─┘          ├─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.

```
►►──┬────────┬──SWAP──┬─MEMORY─┬──┬─LOG────┬───────────────────────────────►◄
    └─WINDOW─┘        └─LOG────┘  └─MEMORY─┘
```

WINDOW ZOOM

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

```
                      ┌─CURSOR──┐
►►──┬────────┬──ZOOM──┼─────────┼──;───────────────────────────────────►◄
    └─WINDOW─┘        ├─LOG─────┤
                      ├─MEMORY──┤
                      ├─MONITOR─┤
                      └─SOURCE──┘
```

Chapter 2. Debug Tool built-in functions

Debug Tool 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 Debug Tool 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 *Debug Tool Customization Guide* and *Debug Tool 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─┘  │
            ├─CCOUTPUTDSNALLOC─,─'─allocation_parms─'─,─┬──────┬┤
            │                                           └─LOUD─┘│
            ├─CCPROGSELECTDSN─,─'─file_name_pattern─'─,─┬──────┬┤
            │                                           └─LOUD─┘│
            ├─CODEPAGE─,─nnnn─────────────────────────────────┤
            ├─COMMANDSDSN─,─'─file_name_pattern─'─,─┬──────┬──┤
            │                                       └─LOUD─┘  │
            ├─DEFAULTVIEW─,─┬─STANDARD─┬──────────────────────┤
            │               └─NOMACGEN─┘                      │
            ├─DLAYDBG─,─┬─No──┬───────────────────────────────┤
            │           └─Yes─┘                               │
            │             ┌─ALL──┐                            │
            ├─DLAYDBGCND─,─┴─NONE─┴───────────────────────────┤
            ├─DLAYDBGDSN─,─'─file_name_pattern─'──────────────┤
            ├─DLAYDBGTRC─,─trace_level────────────────────────┤
            ├─DLAYDBGXRF─,─DSN─,─'file_name'──────────────────┤
            ├─DOPTACBDSN─,─'data_set_name'────────────────────┤
            │                      ┌─No──┐                    │
            ├─DTCNDELETEDEADPROF─,─┴─Yes─┴────────────────────┤
            ├─┬─DTCNFORCECUID─,───────┬─┬─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──┘                                  │
            │                 ┌─PARENT─┐                      │
            ├─MULTIPROCESS─,─┼─CHILD──┼─┬──────────────────────┬┤
            │                 └─PROMPT─┘ │          ┌─NONE─┐   ││
            │                            └─,─EXEC=─┴─ANY──┴───┘│
            ├─NAMES─,─┤ NAMES_options ├───────────────────────┤
            ├─NODISPLAY─,─┬─DEFAULT───┬───────────────────────┤
            │             └─QUITDEBUG─┘                       │
            ├─PREFERENCESDSN─,─'─file_name_pattern─'─,─┬──────┬┤
            │                                          └─LOUD─┘│
            ├─┬─SAVEBPDSN──┬─,─'─file_name_pattern─'──────────┤
            │ └─SAVESETDSN─┘                                  │
            ├─┬─SAVEBPDSNALLOC──┬─,─'─allocation_parms─'─,─┬──────┬┤
            │ └─SAVESETDSNALLOC─┘                          └─LOUD─┘│
            ├─SESSIONTIMEOUT─,─┤ SESSIONTIMEOUT_options ├─────┤
            ├─STARTSTOPMSG─,─┤ STARTSTOPMSG_options ├─,─WTO───┤
            ├─STARTSTOPMSGDSN─,─'file_name'─┬──────────┬──────┤
            │                               └─,─LOUD───┘      │
            ├─SUBSYS─┬───────────────────────────┬────────────┤
            │        └─,─four_character_name─────┘            │
            ├─SVCSCREEN─,─┤ SVCSCREEN_options ├───────────────┤
            ├─TCPIPDATADSN─┬─────────────────┬────────────────┤
            │              └─,─file_name─────┘                │
            ├─THREADTERMCOND─,─┬─NOPROMPT─┬───────────────────┤
            │                  └─PROMPT───┘                   │
            ├─TIMACB─,─ACB_name───────────────────────────────┤
            └─END─────────────────────────────────────────────┘
```

HOSTPORTS_options:

```
├──┬──────────────────────────────────┬──────────────────────────────────────────────────────────┤
   ├─,─port_number────────────────────┤
   ├─,─port_number_range──────────────┤
   │      ┌─,───────────────────┐     │
   │      ▼                     │     │
   └─,─(───┬─port_number───────┬┴──)─┘
           └─port_number_range─┘
```

NAMES_options:

```
├──┬─EXCLUDE─,──┬─LOADMOD─┬─,─pattern─┬──────────────────────────────────────────────────────────┤
   │            └─CU──────┘           │
   └─INCLUDE─,──┬─LOADMOD─┬─,─name────┘
                └─CU──────┘
```

SESSIONTIMEOUT_options:

```
├──┬─NEVER───────────────────┬──────────────────────────────────────────────────────────────────┤
   ├─QUITDEBUG─,─hhmmssnn────┤
   └─QUIT─,─hhmmssnn─────────┘
```

SVCSCREEN_options:

```
├─SVCSCREEN─,──┬─ON──────────────────┬─,─CONFLICT─=──┬─OVERRIDE───┬──┬───────────────────────────┬──────┤
               ├─OFF─────────────────┤               └─NOOVERRIDE─┘  │      ┌─NOMERGE──────────┐ │
               └─(─OFF─,─QUIET─)─────┘                               └─,────┴─MERGE─┬──────────┬┘─┘
                                                                                     └─=─(─COPE─)─┘
```

STARTSTOPMSG_options:

```
├──┬─ALL─────────────────────┬──────────────────────────────────────────────────────────────────┤
   ├─NONE────────────────────┤
   ├─CICS────────────────────┤
   ├─TSO─────────────────────┤
   ├─BATCHTSO────────────────┤
   ├─IMS─────────────────────┤
   ├─OTHER───────────────────┤
   │     ┌─,──────────────┐  │
   │     ▼                │  │
   └─(────┬─CICS──────┬───┴─)┘
          ├─TSO───────┤
          ├─BATCHTSO──┤
          ├─IMS───────┤
          └─OTHER─────┘
```

BROWSE

Specifies whether a user with sufficient RACF® authority to the applicable browse mode RACF facility can start Debug Tool 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 Debug Tool and a remote debugger and properly displayed in full screen mode.

COMMANDSDSN

Indicates the naming pattern Debug Tool uses to locate the data set containing a commands file. Debug Tool 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 Debug Tool to delay the starting of a debug session until Debug Tool recognizes a certain program (CU).

DLAYDBGCND

Indicates whether you want Debug Tool to monitor condition events in the delay debug mode.

DLAYDBGDSN

Indicates that Debug Tool is to use the specified naming pattern when it constructs the delay debug profile data set name.

DLAYDBGTRC

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

DLAYDBGXRF

Indicates that you want Debug Tool 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 Debug Tool for the IMS Transaction Isolation Facility. This data set will be used to store ACBs generated for the EQATcccn 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.

DTCNFORCE*xxxxxx*

Controls DTCN behavior for the conditions described in Table 2. When you set a DTCNFORCE*xxxxxx* option to YES, DTCN forces users to specify the respective resource type. The default setting is NO.

Table 2. List of EQAOPTS DTCNFORCExxxxxx commands and their corresponding DTCN fields

EQAOPTS DTCNFORCE*xxxxxx* 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 *Debug Tool 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 Debug Tool to display COBOL table data items even when an ODO value
| is out of range.

LOGDSN

Indicates the naming pattern Debug Tool uses to locate the data set containing a log file.

LOGDSNALLOC

Indicates the allocation parameters Debug Tool 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 Debug Tool obtains debug data from .mdbg or .dbg files.

MULTIPROCESS

Controls the behavior of Debug Tool when a new POSIX process is created by a fork() or exec() function in the application.

It instructs Debug Tool 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 Debug Tool 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 Debug Tool uses to locate the data set containing a preferences file. Debug Tool 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 Debug Tool uses to create the data sets named by the `SAVEBPDSN` and `SAVESETDSN` commands.

SESSIONTIMEOUT

Establishes a timeout for idle Debug Tool sessions

STARTSTOPMSG

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

STARTSTOPMSGDSN

Indicates that you want Debug Tool 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 Debug Tool allocates the source file.

SVCSCREEN

Controls Debug Tool's enablement of SVC screening.

TCPIPDATADSN

Instructs Debug Tool to dynamically allocate the specified *file-name* to DDNAME SYSTCPD (if SYSTCPD is not already allocated).

THREADTERMCOND

Specifies whether Debug Tool 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 *Debug Tool 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. Debug Tool variables

The following table summarizes the Debug Tool variables. For more information about these variables, see *Debug Tool Reference and Messages*.

Debug Tool variable	**Value**
`%ADDRESS`	Address of the location where your program was interrupted
`%AMODE`	Current® AMODE of the suspended program
`%BLOCK`	Name of the current block
`%CAAADRESS`	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 Debug Tool 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

Debug Tool variable	Value
%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 Debug Tool command
%RSTDSETS	A value of 1 if user settings have been restored and 0 otherwise
%RUNMODE	String identifying the presentation mode of Debug Tool
%R*n*	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 Debug Tool commands

The following reference card provides a list of frequently used Debug Tool commands.

The reference cards are designed to be printed from a PDF file. If you are viewing this page through an information center or from BookManager®, follow these instructions to view the reference card through a PDF file and print it:

1. Open the PDF file for *Debug Tool Reference Summary* by doing one of the following steps:
   a. If you are viewing this topic from IBM System z® Enterprise Development Tools and Compilers information center, click on "PDF version" underneath the "Reference Summary" topic in the navigation pane.
   b. If you are viewing this topic from the Rational® Developer for System z information center, go to the IBM System z Enterprise Development Tools and Compilers information center. Expand "Debug Tool for z/OS", then "Reference Summary". Click on "PDF version".
   c. If you are viewing this topic from BookManager, start your internet browser and go to the IBM System z Enterprise Development Tools and Compilers information center. Expand "Debug Tool for z/OS", then "Reference Summary". Click on "PDF version".
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, Debug Tool 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 Debug Tool 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 Debug Tool 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 Debug Tool 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 Debug Tool searches for source files and debug information files. For example, SYSDEBUG files, LANGX files, and compiler listings. Debug Tool 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 Debug Tool searches for source files and debug information files.

`LISTING` or **`LIST`**
Displays a list of programs known to Debug Tool. Then, you can specify the name of the source file or debug information file for each program.

`LDD` *`assember_CSECT`* or **`LDD`** *`LangX_COBOL_program`*
Load debug information about *assember_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, Debug Tool displays a scale to indicate that there is more to see.

`CLEAR  MONITOR`
Clears 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, Debug Tool 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`
Debug Tool does not display `STEP` and `GO` commands in the Log window. However, if a log file is open, Debug Tool writes them to the log file.

`SET LOG ON FILE` *`file_name`* **`OLD`**
Opens a log file. When Debug Tool 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 Debug Tool 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, Debug Tool runs the called program but does not display it. Debug Tool stops at the statement after the call.

`LOAD` *`program_name`*
Make *program_name* known to Debug Tool.

`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 Debug Tool 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, Debug Tool 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. Debug Tool 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
Debug Tool displays the PF key settings for PF keys 1-12 at the bottom of the screen.

SET KEYS ON 24
Debug Tool displays the PF key settings for PF keys 13-24 at the bottom of the screen.

SET KEYS OFF
Debug Tool 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 Debug Tool 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. Debug Tool prompts you to enter the rest of the command.

Abbreviating commands

(use partial keywords)
You can abbreviate keywords in Debug Tool 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.

QQUIT
Ends the debugging session without prompting you.

QUIT DEBUG
Ends the debugging session but program continues to run. Debug Tool will not be restarted.

QUIT DEBUG TASK
This command works only for CICS. Ends debugging session but the program continues to run. Debug Tool will not be restarted. To start Debug Tool, 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 Debug Tool commands while debugging assembler programs

The following reference card provides a list of frequently used Debug Tool 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 an information center or from BookManager, follow these instructions to view the reference card through a PDF file and print it:

1. Open the PDF file for *Debug Tool Reference Summary* by doing one of the following steps:
   a. If you are viewing this topic from IBM System z Enterprise Development Tools and Compilers information center, click on "PDF version" underneath the "Reference Summary" topic in the navigation pane.
   b. If you are viewing this topic from the Rational Developer for System z information center, go to the IBM System z Enterprise Development Tools and Compilers information center. Expand "Debug Tool for z/OS", then "Reference Summary". Click on "PDF version".
   c. If you are viewing this topic from BookManager, start your internet browser and go to the IBM System z Enterprise Development Tools and Compilers information center. Expand "Debug Tool for z/OS", then "Reference Summary". Click on "PDF version".
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. Debug Tool 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. Debug Tool sets a breakpoint on that line.

`AT 509` Debug Tool sets a breakpoint on line 509.

`AT EXIT *` Debug Tool sets a breakpoint at every program exit point.

`AT ENTRY *` Debug Tool sets a breakpoint at every program entry point.

`AT ENTRY` *`cuname`*
Debug Tool sets a breakpoint at entry point of *cuname*.

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. Debug Tool 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. Debug Tool removes the breakpoint on that line.

`CLEAR AT 509`
Debug Tool removes the breakpoint on line 509.

`CLEAR AT EXIT *`
Debug Tool removes breakpoints from every program exit point.

`CLEAR AT ENTRY *`
Debug Tool removes breakpoints from every program entry point.

`CLEAR AT` Debug Tool removes all breakpoints.

Set a deferred breakpoint

A deferred breakpoint is a breakpoint you set on a program that is not yet known to Debug Tool.

`AT ENTRY` *`cuname`*
If the block name is the same as *cuname*, you no longer have to specify *cuname* as *`cuname`*`:>`*`cuname`*.

Setting conditional breakpoints

`AT CHANGE ITEMNO`
After the value of ITEMNO changes, Debug Tool 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, Debug Tool 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; GO; ;
```

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`
Debug Tool 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`
Debug Tool adds the variable ITEMNO and its current value to the Monitor window, and displays everything on one line.

`LIST ITEMNO`
Debug Tool adds contents of ITEMNO to the log.

`SET AUTO ON`
Debug Tool automatically displays values in the Monitor window.

`SET AUTO ON LOG`
Debug Tool automatically displays values in the log.

Monitor window prefix commands

Type these letters into the prefix area of the Monitor window.

`C` Debug Tool 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'`
Debug Tool 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` Debug Tool steps into the program being called and continues stepping through that program.

`STEP OVER` Debug Tool 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 *progname* to Debug Tool.
2. Be able to set a breakpoint in *progname*.
3. Display the source of *progname* in the Source window.

```
LOAD progname;
QUALIFY progname
QUALIFY RESET;
```

Return to or find the next line to run

`QUALIFY RESET`
Debug Tool displays, in the Source window, the next line it will run.

Load assembler & OSVS COBOL debugging information

`LDD` *`asmcsect`*
Debug Tool loads debug information from EQALANGX into the Source window.

`LDD` *`oscob_prog`*
Debug Tool loads debug information from EQALANGX into the Source window.

Working with registers

`MON LI REG` Debug Tool adds all registers to the Monitor window.

`LIST STORAGE (%GPR6->+1,5);`
Debug Tool display 5 bytes of storage starting at the address pointed to by register 6 plus 1 byte.

`MON LI STORAGE (R6->+1,5);`
Debug Tool 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)`
Debug Tool 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` Debug Tool makes line 27 the current point of execution, but does not run line 27.

`RUNTO 27` Debug Tool runs the program to line 27, then stops.

Saving monitors, breakpoints and settings across debugging sessions

`SAVE MONITORS`
Debug Tool saves all variables being monitored in the Monitor window.

`SAVE BPS` Debug Tool saves all the breakpoints.

`SAVE SETTINGS`
Debug Tool saves all the settings set by the `SET` command.

`RESTORE MONITORS`
Debug Tool restores all the variables saved by the `SAVE MONITORS` command to the Monitor window.

`RESTORE BPS`
Debug Tool restores all the breakpoints saved by the `SAVE BPS` command.

`RESTORE SETTINGS`
Debug Tool restores all the settings saved by the `SAVE SETTINGS` command.

Memory window

`MEMORY ITEMNO`
Debug Tool displays raw storage beginning at the location of ITEMNO.

`MEMORY X'2503D008'`
Debug Tool displays memory beginning base address X'2503D008'..

`SWAP MEMORY LOG`
Debug Tool switches between the Memory window and the Log window.

`SET EQUATE SW='SWAP MEM LOG'`
Debug Tool 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`
Debug Tool does not stop for Language Environment class 1 exceptions: EXEC CICS RETURN and STOP RUN in batch.

`QUIT DEBUG TASK`
Stop Debug Tool 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` Debug Tool produces a DUMP that Fault Analyzer can use in its analysis.

End program debugging (testing)

`QUIT` Ends Debug Tool and program stops running.

`QQUIT` Ends Debug Tool without prompt.

`QUIT DEBUG` Ends Debug Tool but program continues running.

`QUIT DEBUG TASK`
Ends Debug Tool 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 CSR"=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

Copyright license

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 Debug Tool.

Trademarks and service marks

Product Number: 5655-Q10

Printed in USA

SC14-7602-04