

High Level Assembler for z/OS & z/VM & z/VSE



# IDF Reference Summary

*Version 1 Release 6*

**Note**

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

This edition applies to IBM High Level Assembler for z/OS & z/VM & z/VSE Toolkit Feature, Release 6, Program Number 5696-234 and to any subsequent releases until otherwise indicated in new editions. Make sure that you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality.

IBM welcomes your comments. For information on how to send comments, see “How to send your comments to IBM” on page xi.

© **Copyright IBM Corporation 1992, 2013.**

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

---

# Contents

## About this document . . . . . vii

Syntax notation . . . . . vii

## How to send your comments to IBM . . . . . xi

If you have a technical problem. . . . . xi

## Chapter 1. IDF basics . . . . . 1

Windows . . . . . 1

IDF address expressions . . . . . 2

Addresses displayed by IDF . . . . . 3

Cursor addressing . . . . . 3

PF keys . . . . . 3

Typeover storage modification . . . . . 3

## Chapter 2. IDF commands . . . . . 5

ABEND (CMS and z/OS) . . . . . 5

ADSTOP (CMS only) . . . . . 5

ADSTOPS (CMS only) . . . . . 5

AFPR. . . . . 5

ALARM . . . . . 6

ALET. . . . . 6

APROGMSG (CMS only) . . . . . 6

AREGS . . . . . 6

ARRAY . . . . . 6

AUDIT . . . . . 7

BACK . . . . . 7

BASE . . . . . 7

BINARY . . . . . 7

BIT . . . . . 7

BOTTOM . . . . . 8

BREAK . . . . . 8

BRIEF . . . . . 8

CALLERS . . . . . 8

CHARACTER . . . . . 9

CHECK . . . . . 9

CLOSE . . . . . 9

COLORS . . . . . 10

COMMAND . . . . . 10

COMPACT . . . . . 10

CREGS (CMS only). . . . . 10

CURSOR . . . . . 11

DBREAK . . . . . 11

DETAIL . . . . . 11

DISASM . . . . . 11

DOWN . . . . . 12

DROP GLOBAL . . . . . 12

DROP MODULE . . . . . 12

DROP SYMBOLS . . . . . 12

DUMP . . . . . 12

DUMPMODE. . . . . 13

EPNAMES. . . . . 13

EPOFFSET. . . . . 13

EXITEXEC. . . . . 13

EXLIMIT . . . . . 13

FIND . . . . . 14

FIRST . . . . . 14

FIXED . . . . . 14

FLOAT . . . . . 15

FMT. . . . . 15

FOLLOW . . . . . 15

FORMAT . . . . . 15

FPC . . . . . 16

FPR . . . . . 16

GLOBALS. . . . . 16

GOTO . . . . . 16

GPACK. . . . . 17

GPR. . . . . 17

GPRG (z/OS only) . . . . . 17

GPRH (z/OS only) . . . . . 17

GSTATUS . . . . . 17

HIDE . . . . . 18

HISTORY . . . . . 18

ICOUNT . . . . . 19

KWDSYN . . . . . 19

LANGUAGE + . . . . . 19

LANGUAGE COLOR . . . . . 19

LANGUAGE COMMENTS . . . . . 20

LANGUAGE DEBUG . . . . . 20

LANGUAGE DECLARES. . . . . 20

LANGUAGE DROP . . . . . 20

LANGUAGE LOAD . . . . . 21

LANGUAGE MACROS . . . . . 21

LANGUAGE OPTIONS . . . . . 21

LANGUAGE SCROLL. . . . . 21

LANGUAGE STATUS . . . . . 22

LANGUAGE STEM . . . . . 22

LANGUAGE VERSION . . . . . 22

LANGUAGE XPATH (CMS and z/OS) . . . . . 23

LAST . . . . . 23

LASTMSG. . . . . 23

LEFT . . . . . 23

LIBE (CMS and z/OS). . . . . 24

LOAD . . . . . 24

LOCATE . . . . . 24

LOCATION . . . . . 25

LOCATION ALET . . . . . 25

MACRO . . . . . 25

MAJOR. . . . . 25

MAP . . . . . 25

MAXIMIZE . . . . . 26

MINIMIZE . . . . . 26

MODE (CMS only) . . . . . 26

MODULE . . . . . 26

MODULE . . . . . 27

MODULE BASE. . . . . 27

MODULE SIZE . . . . . 27

MOVE . . . . . 28

MPACK . . . . . 28

MRUN . . . . . 28

MSG. . . . . 29

MSGID (CMS and z/OS). . . . . 29

MSGMODE . . . . .	29
MSTATUS . . . . .	29
MSTEP . . . . .	29
NAMES . . . . .	30
NEXT . . . . .	30
OFFSET . . . . .	30
OPEN . . . . .	31
OPTIONS . . . . .	31
ORDER. . . . .	31
OREGS . . . . .	31
PACKED . . . . .	32
PARMS. . . . .	32
PAUSE . . . . .	32
PER (CMS only). . . . .	32
PFK . . . . .	32
PFKDISP . . . . .	33
PLOCATES . . . . .	33
PRESERVE . . . . .	33
PREVIOUS . . . . .	33
PROGCK (CMS only) . . . . .	34
PSW. . . . .	34
PSWSTEAL (CMS only) . . . . .	34
QUALIFY . . . . .	34
QUIET . . . . .	34
QUIETLY . . . . .	34
QUIT . . . . .	35
RCQUIT . . . . .	35
REFRESH . . . . .	35
REGS . . . . .	35
REGS64 (z/OS only) . . . . .	35
REGSTOPS (CMS only) . . . . .	36
RESTORE . . . . .	36
RETRIEVE. . . . .	36
RIGHT . . . . .	36
RLOG . . . . .	36
RUN . . . . .	37
RUNEXIT . . . . .	37
R0-R15 . . . . .	37
SALIMIT . . . . .	37
SAREGS . . . . .	37
SAVE . . . . .	37
SEARCH . . . . .	38
SELFNUCX (CMS only) . . . . .	38
SET ADSTOP (CMS only). . . . .	38
SET AREG. . . . .	38
SET BREAK . . . . .	38
SET COMMAND . . . . .	39
SET EXITEXEC . . . . .	39
SET GLOBAL STEM . . . . .	39
SET GLOBAL TEXT . . . . .	39
SET ICOUNT. . . . .	39
SET OFFSET . . . . .	40
SET OPTION. . . . .	40
SET PSW . . . . .	40
SET REGSTOP (CMS only) . . . . .	40
SET SIZE . . . . .	41
SHOW . . . . .	41
SIZE. . . . .	42
SKIPSTEP . . . . .	42
SPACE . . . . .	43
STATUS . . . . .	43

STEP . . . . .	43
STMTSTEP . . . . .	43
STOKEY . . . . .	43
STOREMAP . . . . .	44
STRUCTURE . . . . .	44
SUBSET (CMS only) . . . . .	44
SVC (CMS only). . . . .	44
SWAP . . . . .	44
SYMBOL . . . . .	45
TASKS (TSO only) . . . . .	45
TITLE . . . . .	45
TOP . . . . .	45
TRIGGER LOAD . . . . .	45
TYPE . . . . .	46
UNION. . . . .	46
UNTIL . . . . .	46
UP . . . . .	46
VALUE. . . . .	46
VARIABLE . . . . .	46
VCHANGE . . . . .	47
VERSION . . . . .	47
VS . . . . .	47
VSEP . . . . .	47
WATCH . . . . .	47
WHERE . . . . .	48
XEDEXIT (CMS only) . . . . .	48
ZONED . . . . .	48

### Chapter 3. ASMIDF EXTRACT

<b>Command . . . . .</b>	<b>49</b>
ADSTOPS (CMS only). . . . .	49
ALET . . . . .	49
AREGS . . . . .	49
ARGUMENT . . . . .	49
ARRAY. . . . .	50
BREAK. . . . .	50
CALLERS . . . . .	50
CMDMSG . . . . .	50
COLORS . . . . .	51
CURSOR . . . . .	51
DISASM . . . . .	51
EVENT. . . . .	51
EXITEXEC. . . . .	52
GLOBAL . . . . .	52
GLOBAL STEM . . . . .	52
GLOBAL STEMS . . . . .	52
GSTATUS . . . . .	53
ICOUNT . . . . .	53
LANGUAGE ARGUMENTS. . . . .	53
LANGUAGE COMMANDS . . . . .	53
LANGUAGE OPTIONS . . . . .	53
LANGUAGE STATUS . . . . .	54
LANGUAGE STEM . . . . .	54
LANGUAGE VERSION . . . . .	54
LASTMSG. . . . .	54
LOAD . . . . .	54
LOCATION . . . . .	55
LOCATION ALET . . . . .	55
MAP . . . . .	55
MODE (CMS only) . . . . .	55
MODULES . . . . .	56

MSTATUS . . . . .	56
NAMES . . . . .	56
OPTIONS . . . . .	56
PER (CMS only) . . . . .	56
PFK . . . . .	57
PLIST . . . . .	57
PLOCATES . . . . .	57
QUALIFY . . . . .	57
QUERY SETTING . . . . .	58
REGS . . . . .	58
REGSTOPS (CMS only) . . . . .	58
SCOPE . . . . .	58
SCRVAR . . . . .	58
SELFNUCX . . . . .	59
SKIPSTEP . . . . .	59
SOURCE . . . . .	59
STOREMAP . . . . .	59
STRUCTURE . . . . .	60
SVC (CMS only) . . . . .	60
SYMBOLS . . . . .	60
TASKS . . . . .	60
TYPE . . . . .	61
VALUE . . . . .	61
VARIABLE . . . . .	61
VDECLARE . . . . .	61
VERSION . . . . .	62
VLOC . . . . .	62
VVALUE . . . . .	62
WINDOWS . . . . .	62

**Chapter 4. ASMIDF Options . . . . . 63**

**Chapter 5. ASMIDF Language Support 67**

Introduction . . . . .	67
A word about variables . . . . .	67
Invocation . . . . .	67
Options. . . . .	68
Displaying source . . . . .	68
Displaying variables . . . . .	68
Displaying structures . . . . .	68
Displaying array elements . . . . .	68
Altering variables . . . . .	69
Displaying type attributes . . . . .	69
LANGUAGE command aliases . . . . .	69
Hints and tips . . . . .	70

**Chapter 6. Using ASMLANGX . . . . . 71**

Invocation . . . . .	71
Options. . . . .	72
Examples . . . . .	73

**Notices . . . . . 75**

Trademarks . . . . .	76
----------------------	----

**Bibliography . . . . . 77**

**Glossary . . . . . 79**



---

## About this document

This document is intended to be used as a quick reference for the High Level Assembler Toolkit Feature Interactive Debug Facility (ASMIDF) User's Guide.

The Interactive Debug Facility, a feature of the IBM® *High Level Assembler Toolkit Feature*, is referred to as "ASMIDF" throughout this publication.

This book is divided into the following sections:

- ASMIDF basics
- ASMIDF commands
- ASMIDF SET command
- ASMIDF EXTRACT command
- ASMIDF options
- ASMIDF language support
- Using ASMLANGX

This document uses format conventions and syntax diagram conventions in describing language and statement elements.

CMS is used in this manual to refer to Conversational Monitor System on z/VM®.

---

## Syntax notation

Throughout this book, syntax descriptions use this structure:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ►— symbol indicates the beginning of a statement.

The —> symbol indicates that the statement syntax is continued on the next line.

The ►— symbol indicates that a statement is continued from the previous line.

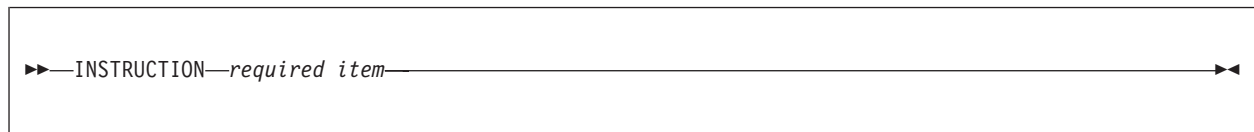
The —>◄ indicates the end of a statement.

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

- **Keywords** appear in uppercase letters (for example, `ASPACE`) or uppercase and lowercase (for example, `PATHFile`). They must be spelled exactly as shown. Lowercase letters are optional (for example, you could enter the `PATHFile` keyword as `PATHF`, `PATHFI`, `PATHFIL`, or `PATHFILE`).

**Variables** appear in all lowercase letters in a special typeface (for example, *integer*). They represent user-supplied names or values.

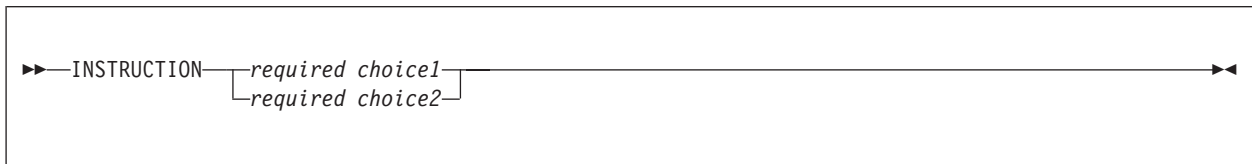
- If punctuation marks, parentheses, or such symbols are shown, they must be entered as part of the syntax.
- Required items appear on the horizontal line (the main path).



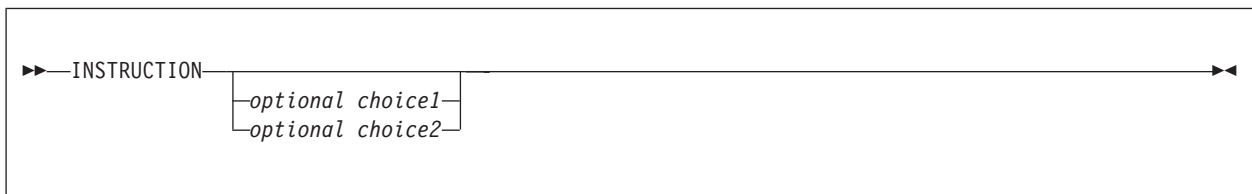
- Optional items appear below the main path. If the item is optional and is the default, the item appears above the main path.



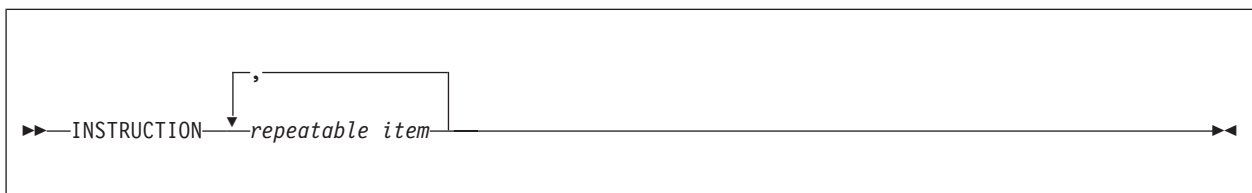
- When you can choose from two or more items, they appear vertically in a stack. If you **must** choose one of the items, one item of the stack appears on the main path.



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



- An arrow returning to the left above the main line indicates an item that can be repeated. When the repeat arrow contains a separator character, such as a comma, you must separate items with the separator character.

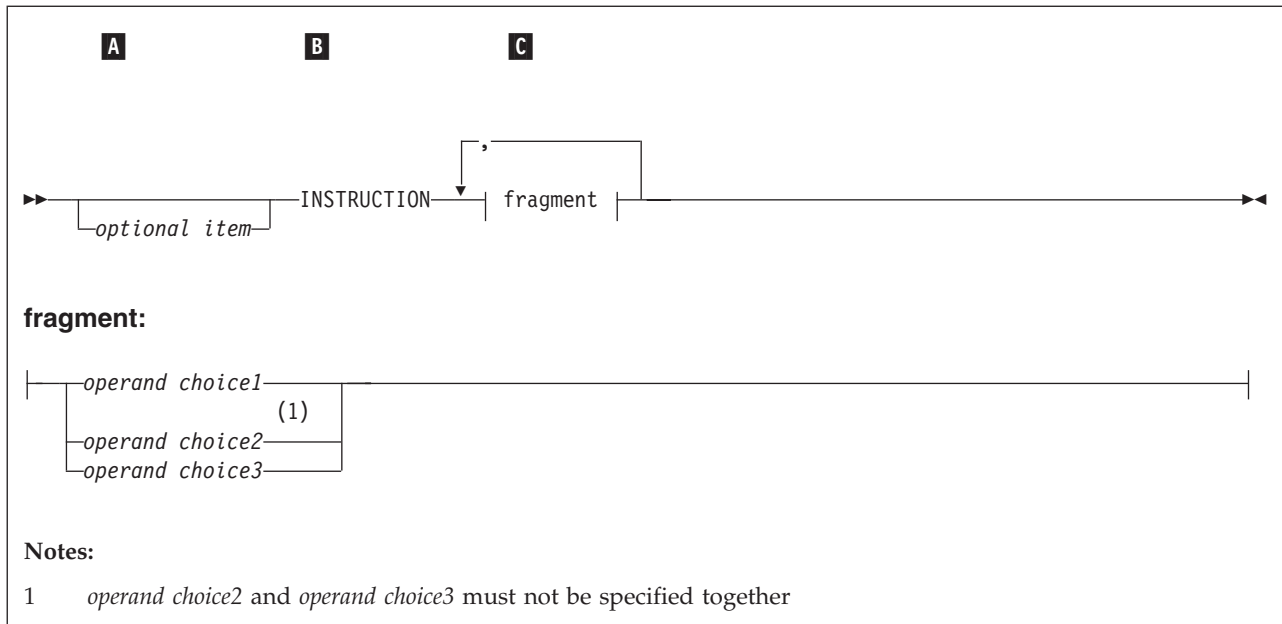


A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

## Format

The following example shows how the syntax is used.





- A**    The item is optional, and can be coded or not.
- B**    The INSTRUCTION key word must be specified and coded as shown.
- C**    The item referred to by “fragment” is a required operand. Allowable choices for this operand are given in the fragment of the syntax diagram shown below “fragment” at the bottom of the diagram. The operand can also be repeated. That is, more than one choice can be specified, with each choice separated by a comma.



---

## How to send your comments to IBM

If you especially like or dislike anything about this book, feel free to send us your comments.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information that is in this book and to the way in which the information is presented. Speak to your IBM representative if you have suggestions about the product itself.

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

You can get your comments to us quickly by sending an e-mail to [idrcf@hursley.ibm.com](mailto:idrcf@hursley.ibm.com). Alternatively, you can mail your comments to:

User Technologies,  
IBM United Kingdom Laboratories,  
Mail Point 095, Hursley Park,  
Winchester, Hampshire,  
SO21 2JN, United Kingdom

Please ensure that you include the book title, order number, and edition date.

---

## If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM support web page



---

## Chapter 1. IDF basics

On CMS and TSO you can activate IDF with the following command:

```
ASMIDF module_name (idf_options/Module_parameters
```

On z/OS<sup>®</sup>, you can activate IDF:

- In TSO batch, with the following command supplied on the DD card SYSTSIN:

```
ASMIDF module_name ( LU vtam_luid idf_options/module_parameters
```

- In batch, with the following JCL:

```
//stepname EXEC PGM=ASMIDFB,  
//          PARM='module_name ( NOSVC97 LU vtam_luid idf_options /  
//          module_parameters'
```

On z/VSE<sup>®</sup> you can activate IDF with the following JCL:

```
// EXEC ASMIDF,PARM='module_name (idf_options/module_parameters'
```

Where:

### **module\_name**

The name of the module to be debugged.

### **idf\_options**

Options directed to ASMIDF.

### **module\_parameters**

The parameters directed to the module to be debugged.

Some command examples are:

```
ASMIDF module_name (COLORS RWGY / in out (abcd  
ASMIDF module_name (PATH / in out (abcd
```

---

## Windows

The types of windows available within IDF are:

- AdStops window (CMS only)
  - Opened by the ADSTOPS and REGSTOPS commands.
  - Closed by the ADSTOPS, REGSTOPS, or CLOSE command.
  - Displays the current PER AdStops and Register Stops.
- Additional Floating-Point Registers window
  - Opened by the AFPR command.
  - Closed by the AFPR or CLOSE command.
  - Displays the current Additional Floating-Point Registers and the Floating-Point Control Register.
- Break window
  - Opened by the BREAK command.
  - Closed by the BREAK or CLOSE command.
  - Displays the active breakpoints and watchpoints.
- Current Registers window
  - Opened by the REGS, REGS64, AREGS, or CREGS command.
  - Closed by the REGS or CLOSE command.
  - Displays the current PSW, GPRs, and FPRs or ARs or CRs.
- Disassembly window
  - Opened by the DISASM or OPEN command.

- Closed by the DISASM or CLOSE command.
  - Displays disassembled instructions.
- Dump window
  - Opened by the DUMP or OPEN command.
  - Closed by the DUMP or CLOSE command.
  - Displays storage in "dump format".
- Entry Point Names window
  - Opened by the EPNAMES or OPEN command.
  - Closed by the EPNAMES or CLOSE command.
  - Displays information about the entry points in the module.
- Language Support Module information window
  - Opened by the VARIABLE, LANGUAGE, STRUCT, or OPEN command.
  - Closed by the VARIABLE or CLOSE command.
  - Displays information from IDF Language Support commands.
- Minimized Windows Viewer
  - Opened by the MINIMIZE command.
  - Closed by the MAXIMIZE command.
  - Lists the minimized windows.
- Old Registers window
  - Opened by the OREGS command.
  - Closed by the OREGS or CLOSE command.
  - Displays the old ARs, CRs or PSW, GPRs, FPRs, and instruction at PSW.
- Options window
  - Opened by the OPTIONS command.
  - Closed by the OPTIONS or CLOSE command.
  - Displays settings of various options.
- Skipped Subroutines window
  - Opened by the SKIPSTEP command.
  - Closed by the SKIPSTEP or CLOSE command.
  - Displays subroutines "skipped" during single stepping, statement stepping, or the PATH or FASTPATH processing.
- Target Status window
  - Opened by the STATUS command.
  - Closed by the STATUS or CLOSE command.
  - Displays information about the target program.

---

## IDF address expressions

IDF Address Expressions are made up of terms separated by plus or minus signs. A term can consist of a program symbol, a hex constant (X'5'), a decimal constant (f'4'), a character constant that is one character in length (c'A'), or an implicit numeric constant (247).

Program symbols are of the form "(module.csect) symbol". If supported by the active LSM, they may also be of the form "(module.csect) STMT#nnnnn". The *csect* in "(module.csect)" is only needed if the symbol occurs in multiple CSECTs. The *module*. in "(module.csect)" is only needed if the symbol is not in the currently qualified module. To select the module to be the currently qualified module, use the SET QUALIFY command.

Terms may be followed with a register designator. A register designator consists of the string R0 through R15 or AR0 through AR15, enclosed in parentheses. Using AR0 through AR15 directs the DUMP, SET ALET, and EXTRACT LOCATION commands to use the ALET in the specified AR.

Terms and register designators may be followed by indirection operators (% , >, ?, =>, ->, &, +>). If an indirection operator follows a term, IDF uses the contents of the word pointed to by the expression

evaluated thus far. Similarly, if an indirection operator follows a register designator, IDF is being told how to interpret the contents of the register. The word or register is treated as:

- A 24-bit address if the % or := operators are used.
- A 31-bit address if the ? or => operators are used.
- A 64-bit address if the & or +> operators are used.
- The appropriate size (24-bit, 31-bit, or 64-bit) depending on the AMODE of the PSW if the -> operator is used.

---

## Addresses displayed by IDF

Whenever appropriate, IDF displays addresses in symbolic form. It is normally of the form "(module.csect) symbol+offset". If the address corresponds to IDF Language extract data, it is of the form "(module.csect) STMT#nnnnn+offset". The module name is omitted if it is the currently qualified module, unless the FULLQUAL option is used.

---

## Cursor addressing

IDF allows you to specify addresses by placing the cursor in a field on the screen. IDF determines an address in the following ways:

- If the cursor is in a GPR, the contents of that displayed register are used. If the cursor is in an AR, the DUMP and SET ALET commands use the ALET in that AR and the DUMP command uses the address in the associated GPR.
- If the cursor is in the PSW, the address part of the PSW is used.
- If the cursor is in the hex part of a disassembled instruction, then:
  - All commands except DISASM and OPEN DISASM use the address of the halfword containing the cursor.
  - If the field containing the cursor is both the first field disassembled and a branch instruction, the DISASM and OPEN DISASM commands use the effective address of the branch instruction. Otherwise they behave like other commands.
- If the cursor is in a dump field, then:
  - All commands except DUMP and OPEN DUMP use the address of the beginning of the hexadecimal field containing the cursor, or the exact address of the character on which the cursor is positioned if it is in the character portion of the display.
  - If the field is both a fullword field and the first field in the dump display, the DUMP and OPEN DUMP commands use the *contents* of the field. Otherwise they behave like other commands.
- If the cursor is in the protected portion of a disassemble or dump line the starting address of the line is used.

---

## PF keys

The ENTER key and PF keys 1 through 24 can be set to any IDF command or to any IDF macro by the SET PFK command. When this is done, instead of typing the command, you can press the PF key.

The PF key settings are displayed at the bottom of the screen, unless turned off by the PFKDISP command.

---

## Typeover storage modification

- In a Dump window, or a Disassembly window containing storage being dumped, storage may be changed by overtyping the hex or character display of that storage.
- In the current registers window, the PSW, general purpose or access registers, and floating point registers may be changed by overtyping the displayed value.

- In a Disassembly window, the hex values of the instructions may be changed by overtyping them.
- In the Additional Floating-Point Registers window, the floating point registers may be changed by overtyping the displayed values.
- In the Entry Point Names window, the short entry point name may be changed by overtyping the displayed values.

The changes are immediately reflected on the screen as different instruction mnemonics, addresses, and so on.



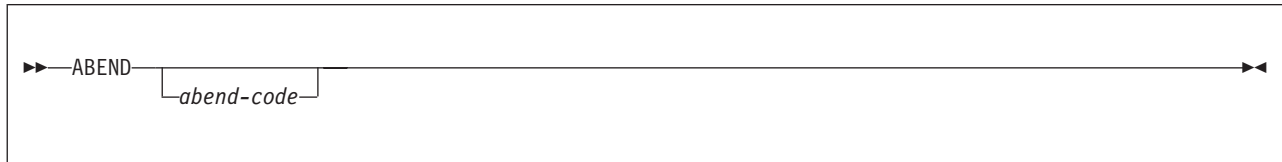
---

## Chapter 2. IDF commands

---

### ABEND (CMS and z/OS)

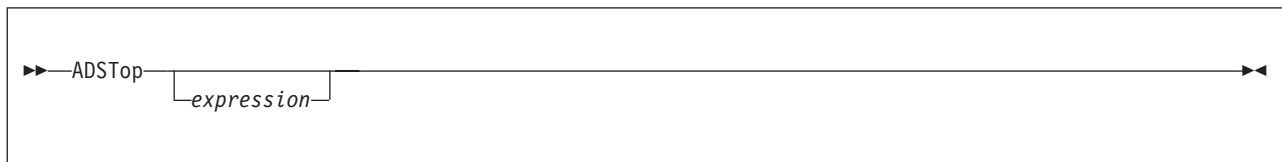
Performs IDF cleanup, then issues OS ABEND.



---

### ADSTOP (CMS only)

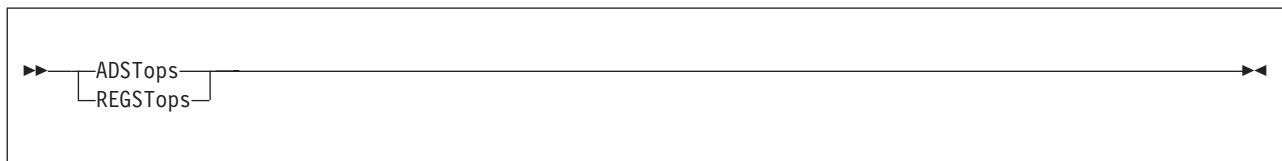
Sets one end of a PER ADSTOP range.



---

### ADSTOPS (CMS only)

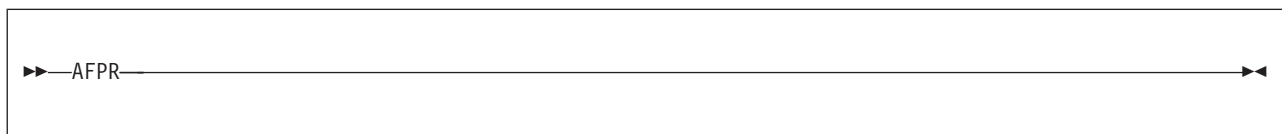
Displays the current Address Stops and Register Alteration Stops.



---

### AFPR

Displays the Additional Floating-Point Registers and the Floating-Point Control Register.



---

## ALARM

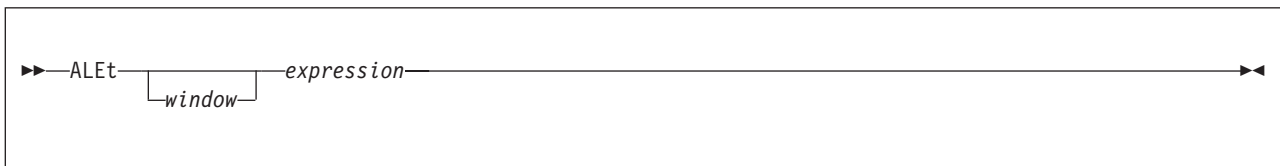
Enables or disables the terminal alarm.



---

## ALET

Sets the ALET for a dump window.



---

## APROGMSG (CMS only)

Enables or disables the trapping of asynchronous program-checks which occur while IDF displays the user interface.



---

## AREGS

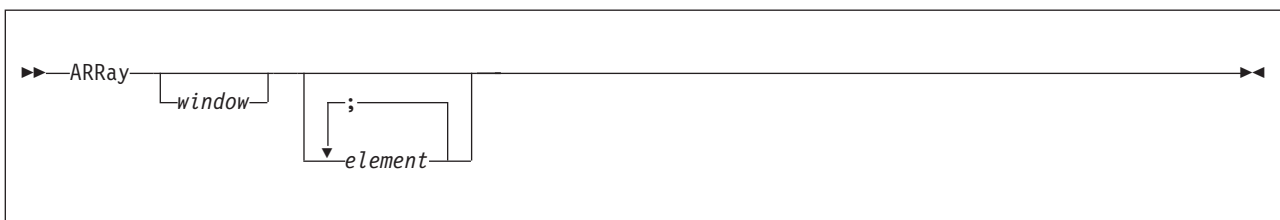
Rotates the register display between GPRs and ARs.



---

## ARRAY

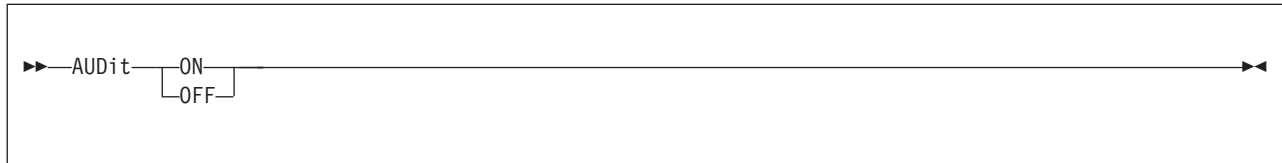
Enables variable display in the array format.



---

## AUDIT

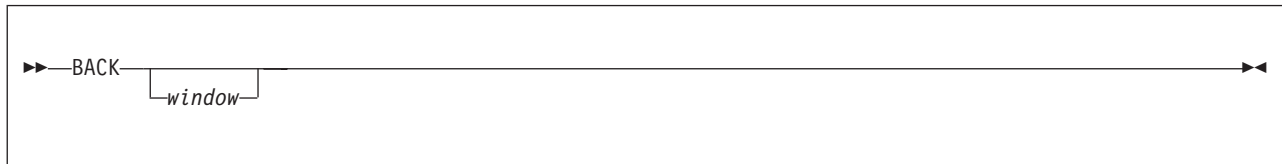
Enables or disables the VAR basing "audit trail".



---

## BACK

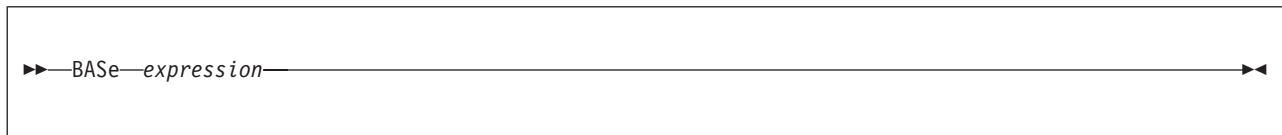
Displays previously dumped storage (the last 10 dumps can be displayed).



---

## BASE

Sets the base of a target.



---

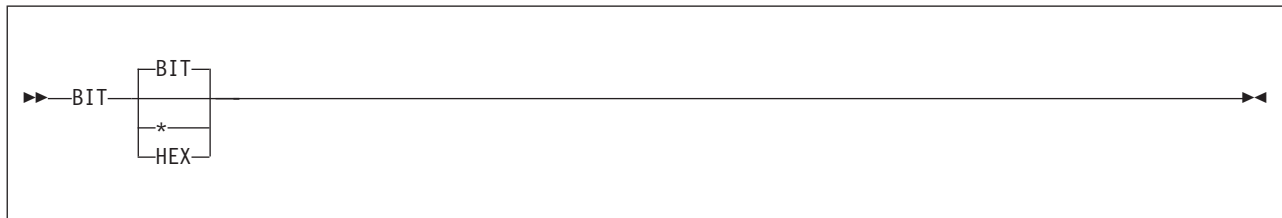
## BINARY

A synonym of the FIXED command.

---

## BIT

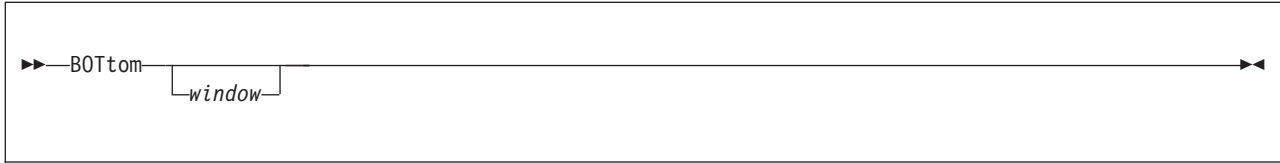
Sets or queries the VAR display format for BIT variables.



---

## BOTTOM

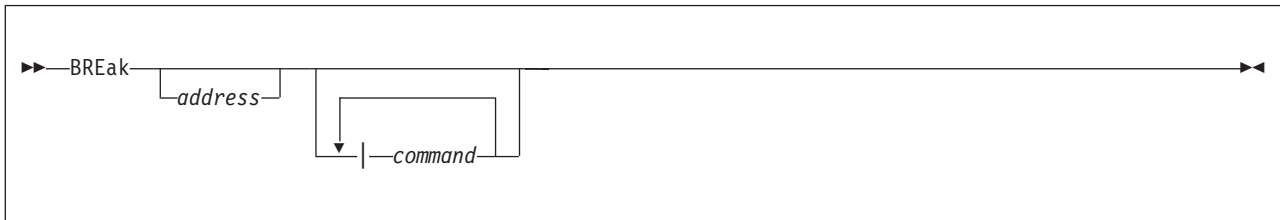
Displays source code at the highest available address within the current code section.



---

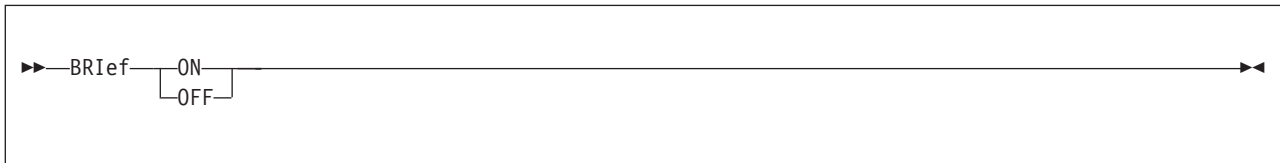
## BREAK

Sets an instruction breakpoint.



---

## BRIEF

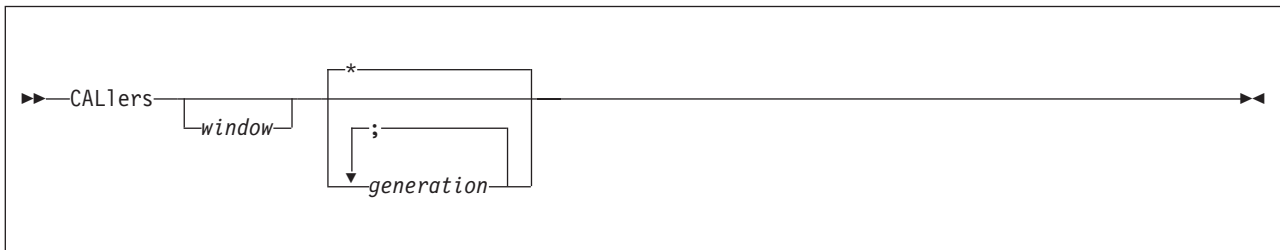


Disables or enables the display of VAR declaration information.

---

## CALLERS

Displays information for each generation in the program caller hierarchy.



- The information includes:
  - Location as:
    - (mod.sect)stmt#nnnnn+offset
    - program\_block\_name+offset (if known)
  - Save Area header
  - Save Area register values
- Caller generations are numbered:
  - 0 current program

- 1 parent (caller)
- 2 grand parent (caller of caller)
- ... and so on
- If particular caller generations are specified, only the corresponding information is shown.
- The default is "\*", to show *all* caller generations.

Also see the SAREGS and SALIMIT commands.

---

## CHARACTER

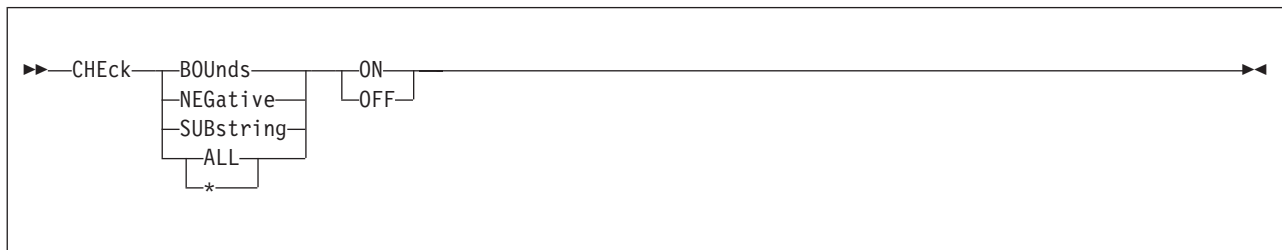
Sets or queries display format for CHARACTER variables.




---

## CHECK

Enables or disables the checking of types of input values.



### BOUNDS

Array index bounds

### NEGATIVE

Unsigned variable values

### SUBSTRING

Character or bit string substring limits

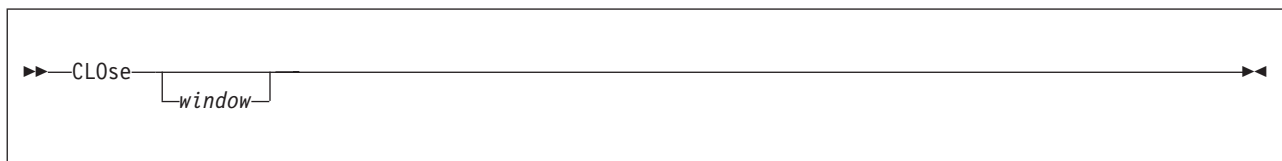
ALL | \*

All of the above

---

## CLOSE

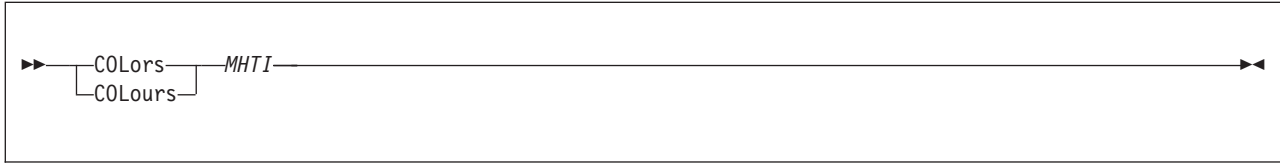
Closes a window.



---

## COLORS

Sets display colors.



Each value is the first letter of one of the following colors:

Blue, Green, Pink, Red, Turquoise, Yellow, White

For example:

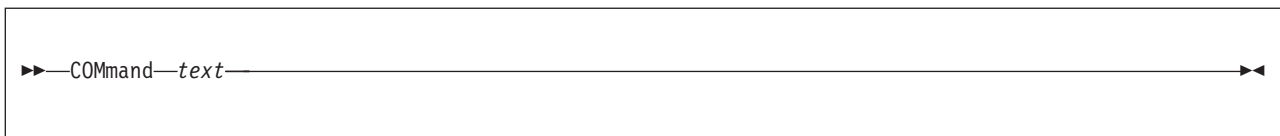
`COLORS BGRY`

gives blue messages, green headings, red text, and yellow input.

---

## COMMAND

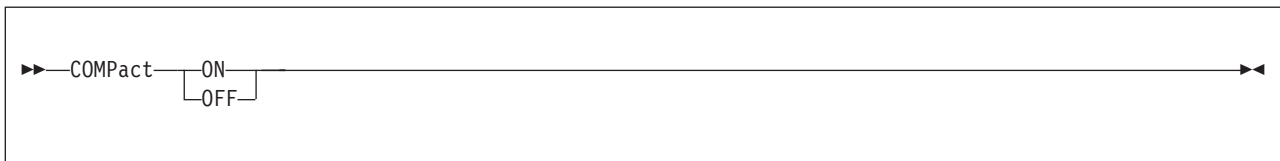
Performs an IDF command.



---

## COMPACT

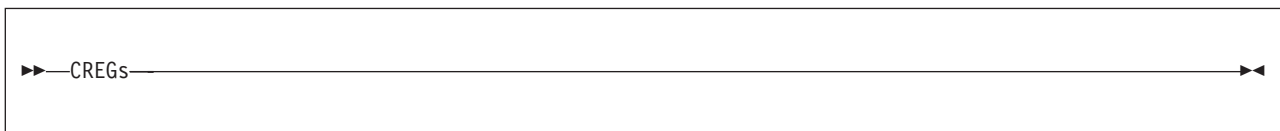
Enables or disables the compact variable display mode.



---

## CREGS (CMS only)

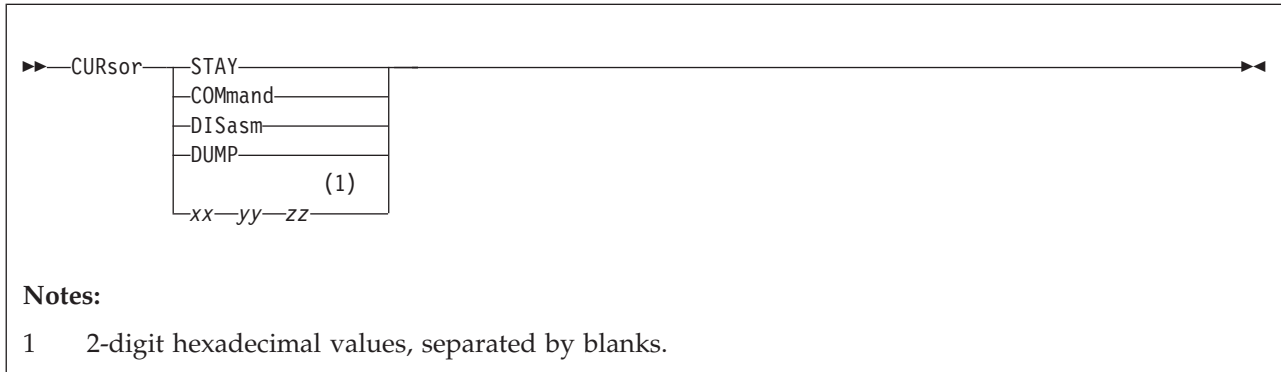
Rotates register display between GPRs and CRs.



---

## CURSOR

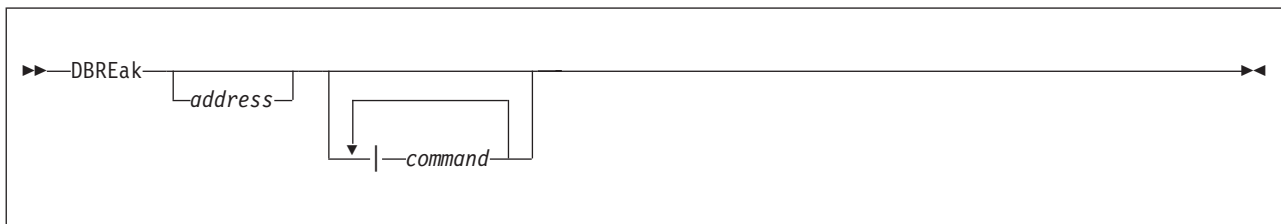
Positions the cursor within a window.



---

## DBREAK

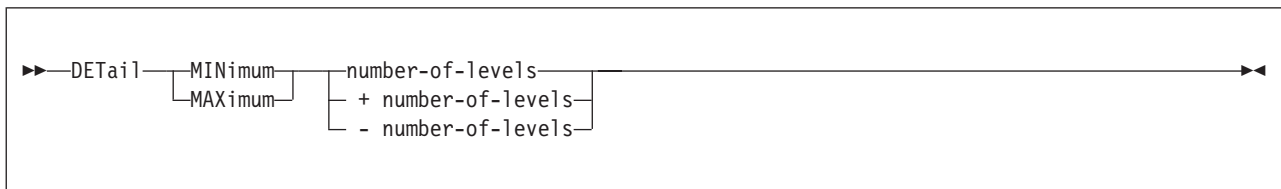
Sets a deferred instruction breakpoint in a module.



---

## DETAIL

Controls the display of data for Structure or Union components of intermediate depth.



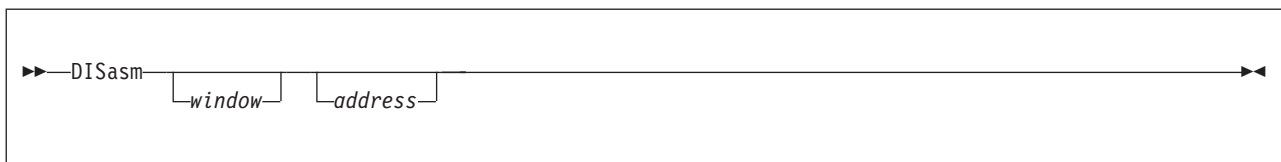
*number-of-levels*

The number of levels of Structure or Union components to be shown.

---

## DISASM

Displays a disassembly listing.



---

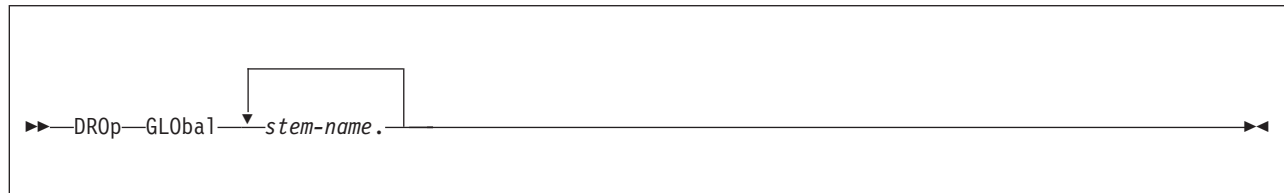
## DOWN

A synonym of the NEXT command.

---

## DROP GLOBAL

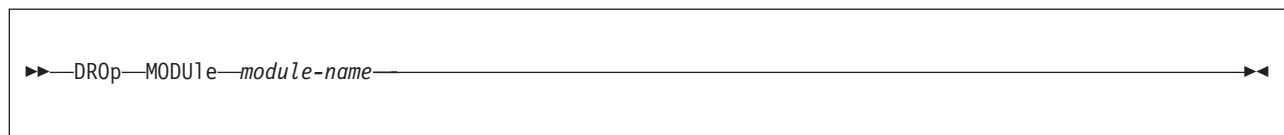
Discards information for stems from storage.



---

## DROP MODULE

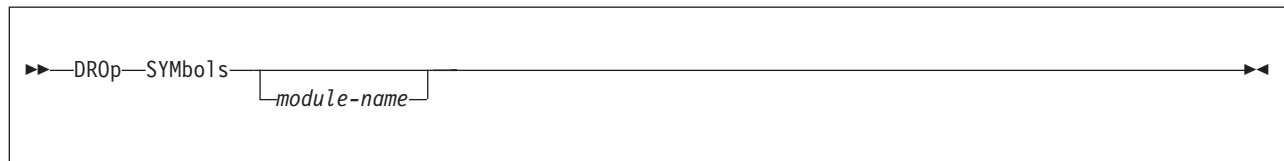
Discards information about module *module-name*.



---

## DROP SYMBOLS

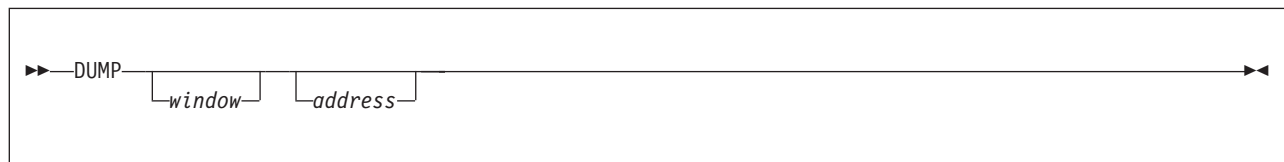
Discards IDF symbols.



---

## DUMP

Provides Storage Dump in the format HEX ... HEX \*char\*





---

## DUMPMODE

Toggles the Dump Format between symbolic and unformatted.

▶▶—DUMPMODE—▶▶

---

## EPNAMES

Toggles the Entry Point Names display.

▶▶—EPNAMES—▶▶

└─*section-name*─┘

---

## EPOFFSET

Specifies the entry-point-offset.

▶▶—EPOffset—*entry-point-offset*—▶▶

---

## EXITEXEC

Toggles the Exit Routine.

▶▶—EXITexec—▶▶

---

## EXLIMIT

Sets the maximum LSM stemmed array index during EXTRACT LANGUAGE commands execution.

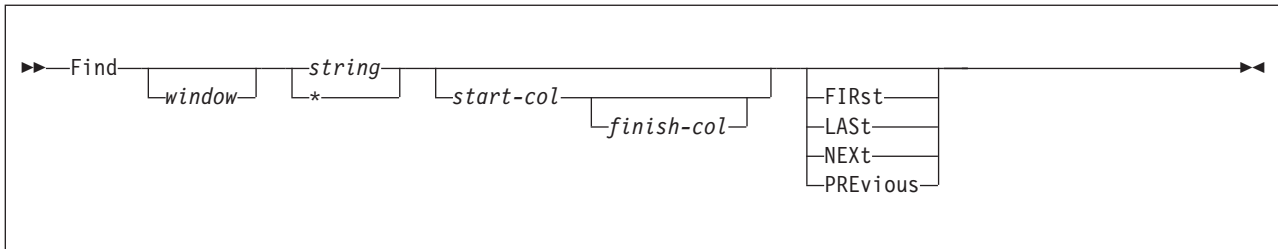
▶▶—EXLimit—*max-stemmed-array-index*—▶▶

- Prevents "run-away" if data being extracted is unbounded (for example: LANGUAGE EXTRACT VValue for Char(\*) or Bit(\*) variable).
- The default EXLimit is 20000.

---

## FIND

An ISPF-style source text search facility, which locates the string and displays the section of code where it occurs.



\* Use current search string

*start-col*

An integer; the column at which searching starts.

*finish-col*

An integer; the column at which searching finishes.

### FIRST

Begin the search at the lowest address, and look for the search string in a forward direction.

### LAST

Begin the search at the highest address, and look for the search string in a reverse direction.

### NEXT

Begin the search at the current address, and look for the search string in a forward direction.

### PREVIOUS

Begin the search at the current address, and look for the search string in a reverse direction.

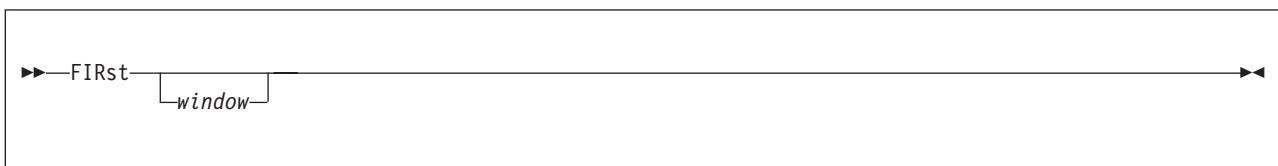
A search string that is numeric or contains embedded blanks must be enclosed in quotes. Both "... " and '...' forms are accepted.

Unless otherwise qualified, the search is performed from the current address, in the direction last specified.

---

## FIRST

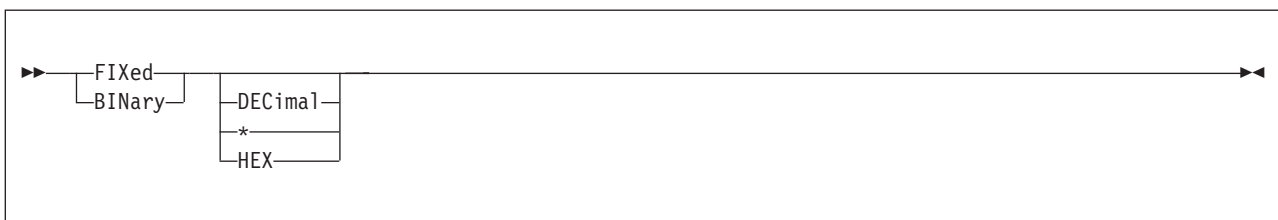
Displays the source code which corresponds to the lowest address.



---

## FIXED

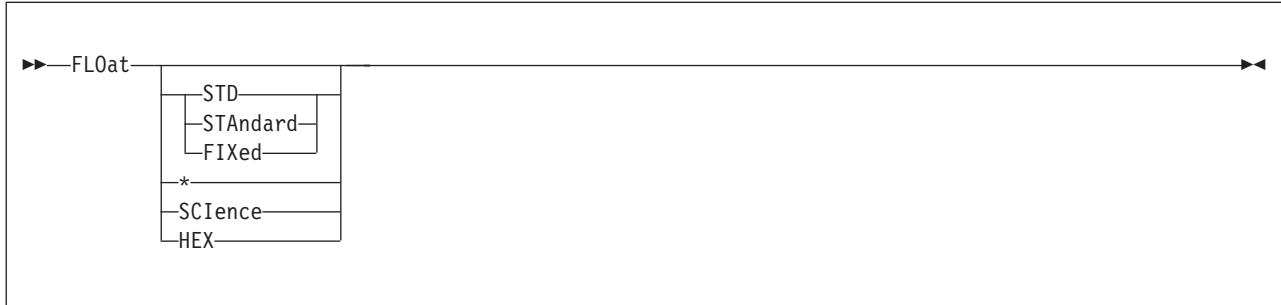
Sets or queries the VAR display format for FIXED variables.



---

## FLOAT

Sets or queries the VAR display format for FLOAT variables.



---

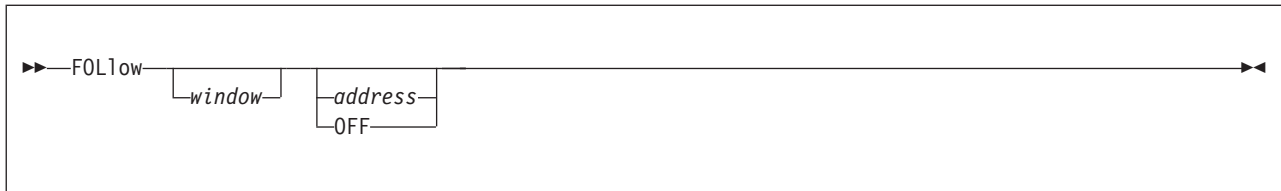
## FMT

A synonym of the FORMAT command.

---

## FOLLOW

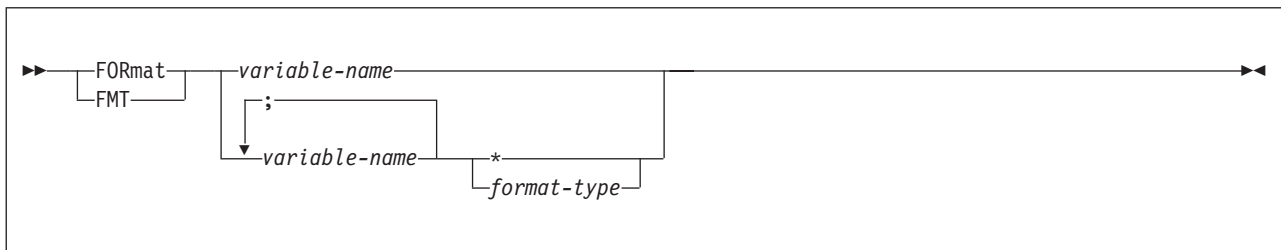
Directs the DUMP window to “follow” the contents of a register.



---

## FORMAT

Controls the display format for *individual* variables.



*format-type* must be appropriate for variable data type:

\* Reset display format to default for variable class.

### Bit

Refer to BIT

### Char

Refer to CHAR

### Fixed

Refer to FIXED

### Float

Refer to FLOAT

**Packed**

Refer to PACKED

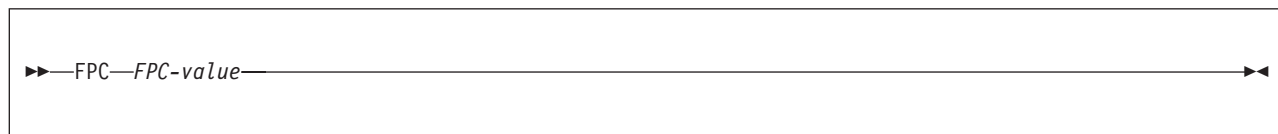
**Zoned**

Refer to ZONED

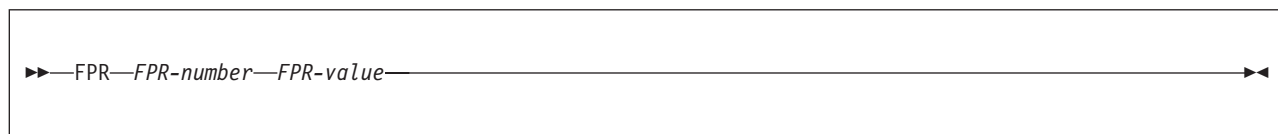
If the *format-type* is absent, displays the current display format for the variable.

**FPC**

Sets the Floating Point Control register.

**FPR**

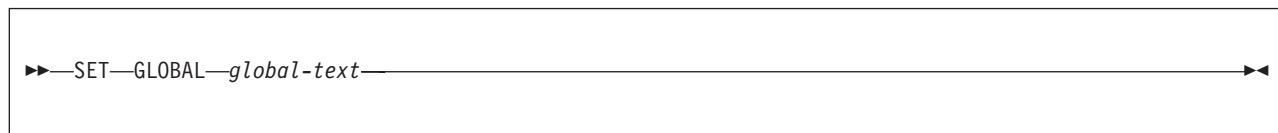
Sets a floating point register.



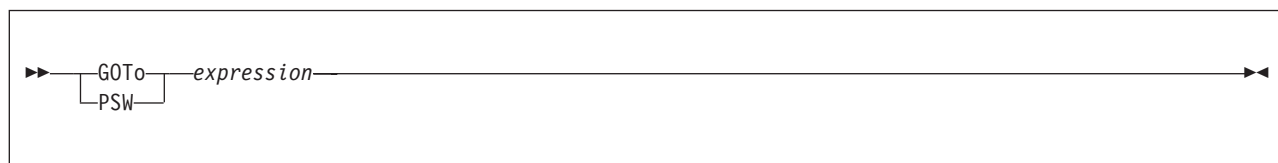
The FPR number is 0, 2, 4, or 6, except for OS/390® systems with binary floating point support, where registers 0 to 15 may be specified.

**GLOBALS**

Displays information about the Global Storage stems.

**GOTO**

Places an evaluated expression in the address portion of the PSW.



---

## GPACK

Returns the Global Storage data storage areas which no longer contain stem data.

▶▶ GPAck ◀◀

---

## GPR

Sets a general register.

▶▶ GPR *register-number* *expression* ◀◀

---

## GPRG (z/OS only)

Sets a 64-bit general register.

▶▶ GPRG *register-number* *expression* ◀◀

---

## GPRH (z/OS only)

Sets the upper 32-bits of a 64-bit general register.

▶▶ GPRH *register-number* *expression* ◀◀

---

## GSTATUS

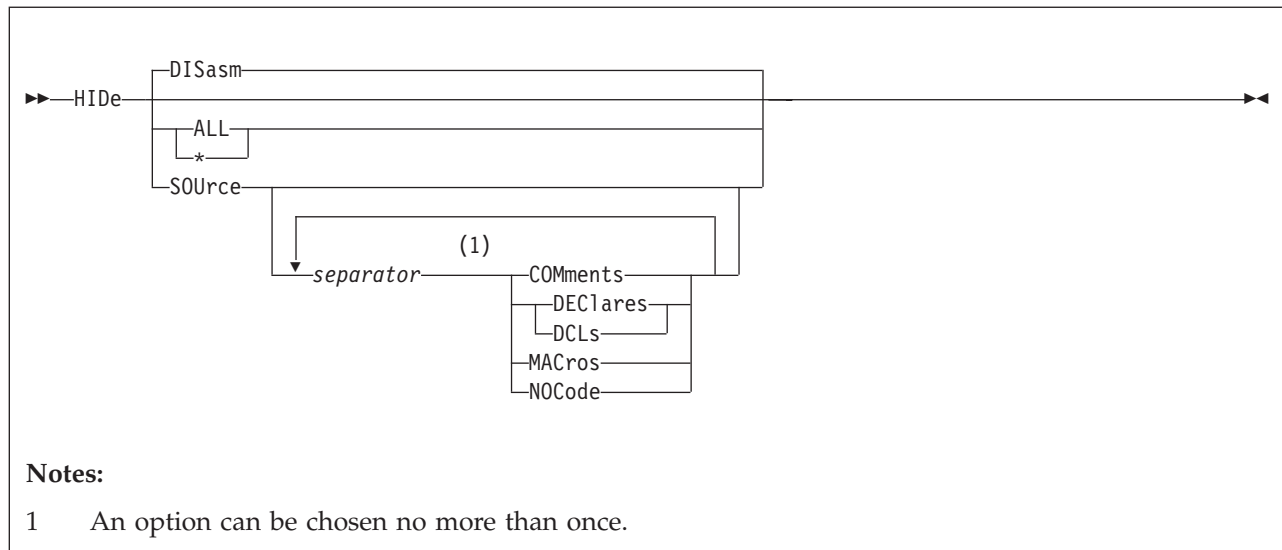
Displays information about the storage used to contain the Global Storage stem data loaded with SET GLOBAL STEM commands.

▶▶ GStatus ◀◀

---

## HIDE

Controls the display of source code and disassembly, by hiding information. The SHOW command controls the display by showing information.



### DISASM

Show source code only

### ALL | \*

Show source code only, excluding comments, declarations, macro expansions, and source lines with no corresponding object code

### SOURCE

Show disassembly only

### COMMENTS

Exclude block comment source code

### DECLARES | DCL

Exclude declaration source code

### MACROS

Exclude macro expansion source code

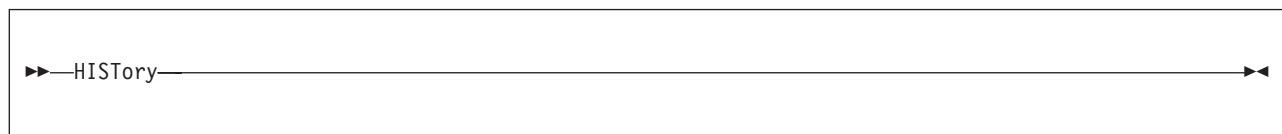
### NOCODE

Exclude source lines with no corresponding object code

---

## HISTORY

Reviews instruction history (PATH).



---

## ICOUNT

Displays the number of instructions executed since the last ICOUNT command.

```
▶▶ ICOUNT ▶▶
```

---

## KWDSYN

Defines a synonym of an IDF keyword.

```
▶▶ KWDSYN oldkwd newkwd ▶▶
```

---

## LANGUAGE +

Scrolls the LSM window.

```
▶▶ LANGUAGE window + - scroll-number-of-lines ▶▶
```

---

## LANGUAGE COLOR

Selects the color used to display source code.

```
▶▶ LANGUAGE COLOr COLOur BLUe RED PINK GREen TURquoise YELlow WHITe ▶▶
```

The default is the color used by IDF for text display.

---

## LANGUAGE COMMENTS

Enables or disables the block comment display.

▶▶—LANguage—COMments— ON OFF	—————▶▶
------------------------------------	---------

---

## LANGUAGE DEBUG

Enables or disables the display of IDF LSM interface debug information.

▶▶—LANguage—DEBug— <i>qualifiers</i> —	—————▶▶
--	---------

Should only be used as directed by IBM support.

---

## LANGUAGE DECLARES

Enables or disables the declare display.

▶▶—LANguage— DECLares DCL	ON OFF	—————▶▶
---------------------------------	-----------	---------

---

## LANGUAGE DROP

Removes one or more language extract files from memory.

▶▶—LANguage—DRop— *	————— <i>extract-file-name</i>	—————▶▶
------------------------	-----------------------------------	---------

\* All currently loaded extract files are removed.

*extract-file-name*

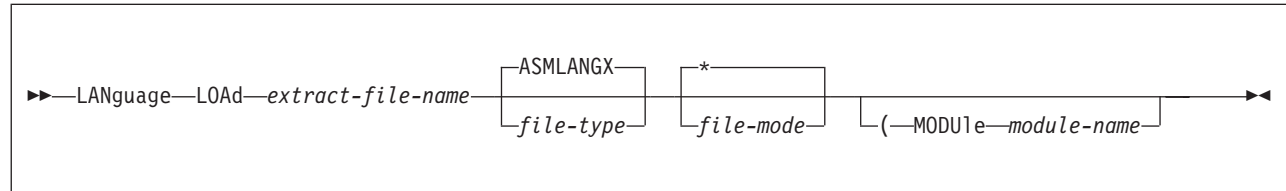
This extract file is removed.



---

## LANGUAGE LOAD

Loads an extract file, optionally associating it with a specific MODULE.



*extract-file-name*

z/OS PDS member name

*file-type*

(CMS only) z/OS DD name.

Specifying this option eliminates the search using the XPATH file types. The default XPATH is "ASMLANGX".

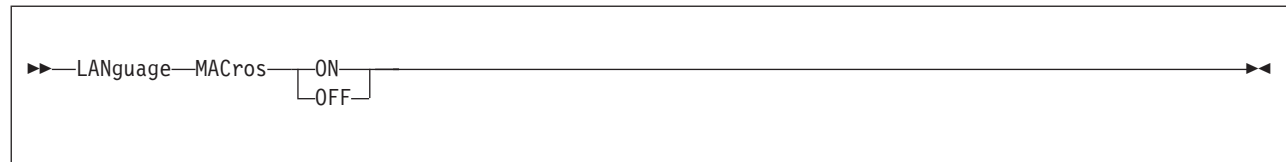
**MODULE** *module-name*

Associates an extract file with a module. See section "Options" on page 68.

---

## LANGUAGE MACROS

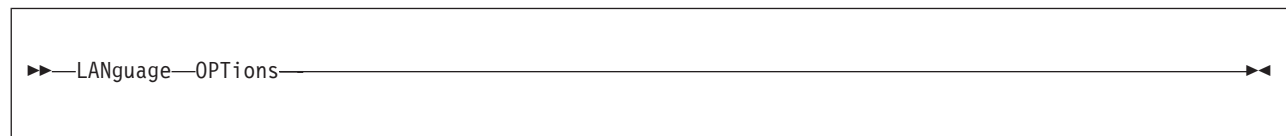
Enables or disables the display of assembler source generated by macros.



---

## LANGUAGE OPTIONS

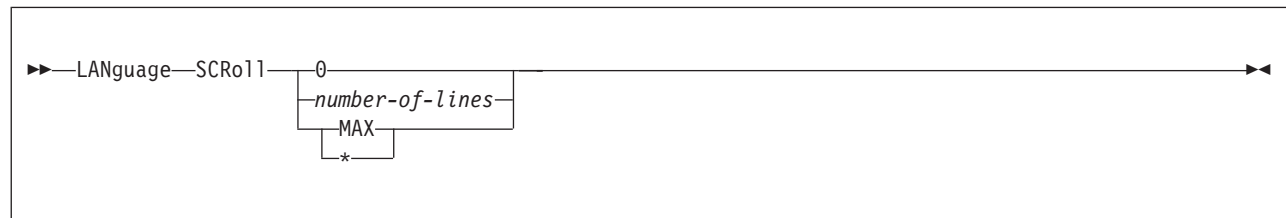
Displays the current value of ASMLANG settings and the Options save stack nesting level.



---

## LANGUAGE SCROLL

Sets the default scroll amount.



**0** Disable scrolling

**1-254**

Scroll by this number of lines

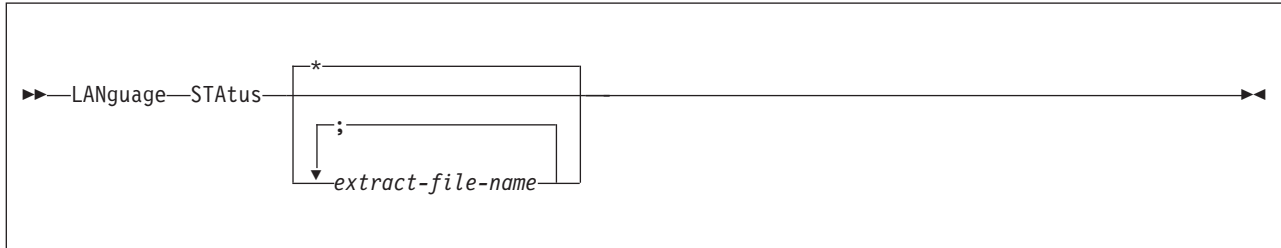
MAX | \*

Scroll by maximum amount (current size of LSM window)

---

## LANGUAGE STATUS

Displays information about extract files currently loaded.



\* Show all extract files.

extract-file-name

The name of the extract file to display information about.

---

## LANGUAGE STEM

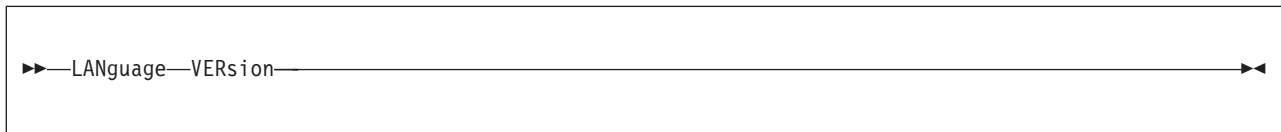
Alters the name of the REXX stemmed array variable for the EXTRACT LANGUAGE commands, and other EXTRACT commands.



---

## LANGUAGE VERSION

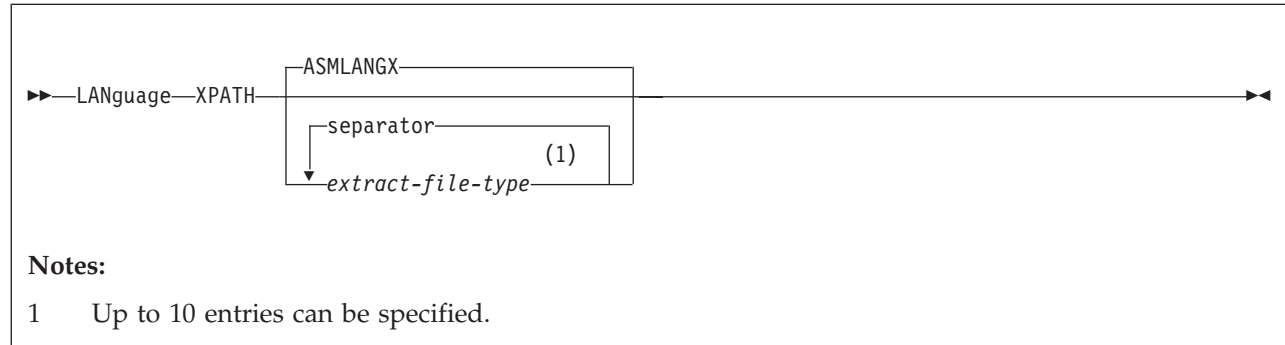
Displays the ASMLANG version identifier.



---

## LANGUAGE XPATH (CMS and z/OS)

Defines the extract file search path file type (z/OS DD name) information.



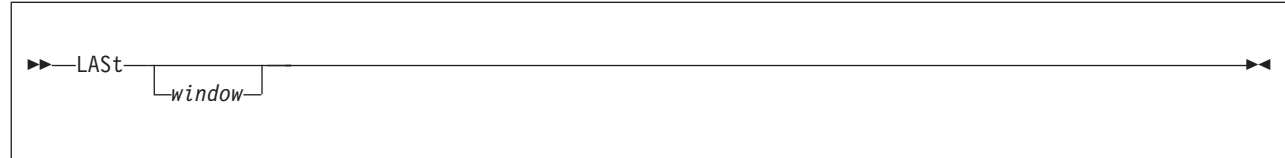
- Used to locate extract files for which the extract file type (z/OS DD name) has not been explicitly specified.
- XPATH entries are searched in the order specified.
- If parameters are specified, then sets XPATH as specified, with up to 10 entries.
- If parameters are *not* specified, then resets XPATH to the default of "ASMLANGX".

LANGUAGE STATUS displays the current XPATH.

---

## LAST

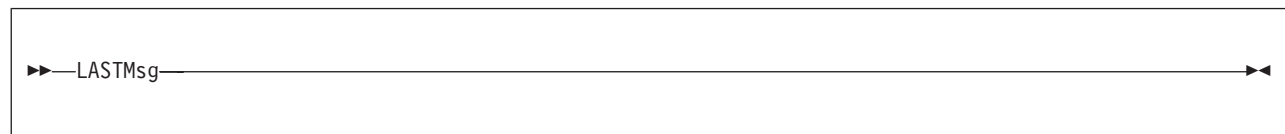
Displays source code at the highest address.



---

## LASTMSG

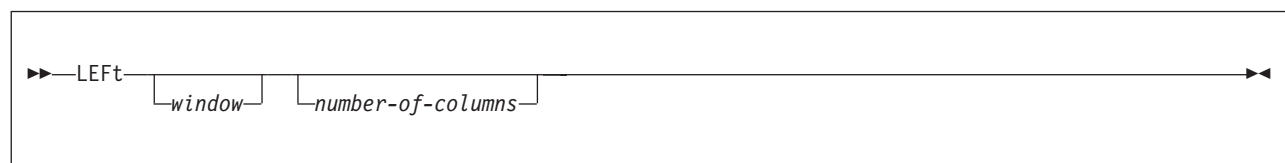
Displays last two messages.



---

## LEFT

Scrolls a window left.



---

## LIBE (CMS and z/OS)

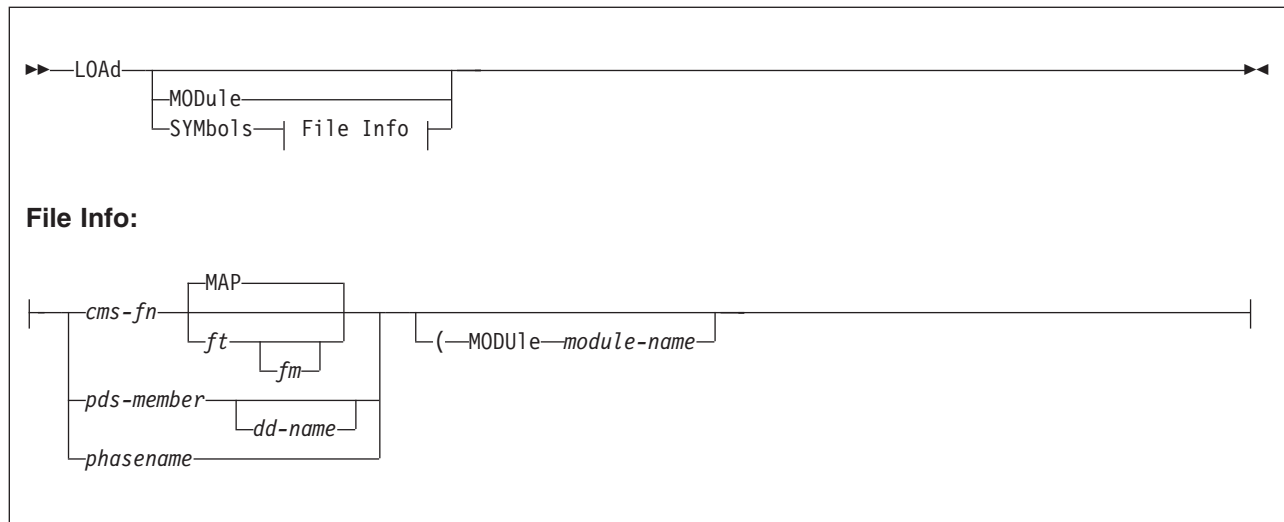
Nominates the source of the target program which IDF is to load.



---

## LOAD

Loads a target module and associated symbols.

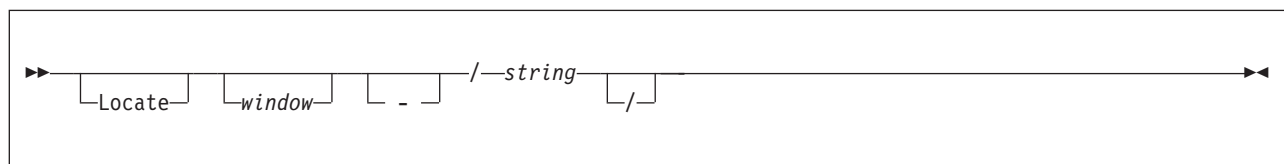


- LOAD loads a target module and symbols.
- LOAD MODULE loads a module.
- LOAD SYMBOLS loads symbols for a module.

---

## LOCATE

XEDIT-style source text search facility which locates the string and displays the section of code where it occurs. The search begins at first source line on screen.



If "-" is specified, the search is performed towards the beginning of the source information.

The trailing delimiter is only required if the string contains trailing blanks.

---

## LOCATION

Sets the main storage to MEMAREA (MEMAREA is a REXX variable).

▶▶—LOCation—*storage-start-address*—▶▶

---

## LOCATION ALET

Sets storage in a dataspace to MEMAREA (MEMAREA is a REXX variable).

▶▶—LOCation—ALEt—*access-link-entry-token*—*storage-start-address*—▶▶

---

## MACRO

Issues an IDF macro.

▶▶—MACro—*macro-name*—*macro-parameters*—▶▶

---

## MAJOR

Disables or enables display of data for Structure or Union major component.

▶▶—MAJor—*ON*/  
*OFF*—▶▶

---

## MAP

Displays information about modules.

▶▶—MAP—*window*—*\**  
*;*  
*module-name*—▶▶

\* Information shown for all modules.

*module-name*

Information shown for this module.

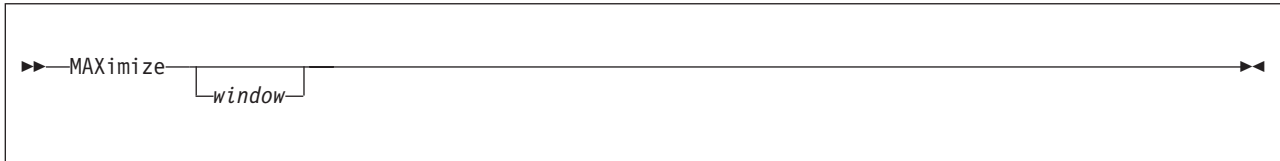
The information includes:

- Module location
- CSECT location
- Extract file associated with each CSECT

---

## MAXIMIZE

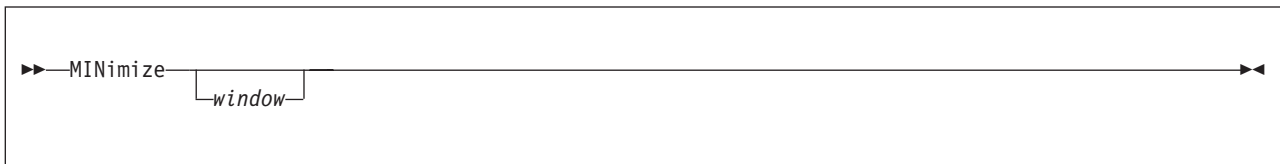
Maximizes a window.



---

## MINIMIZE

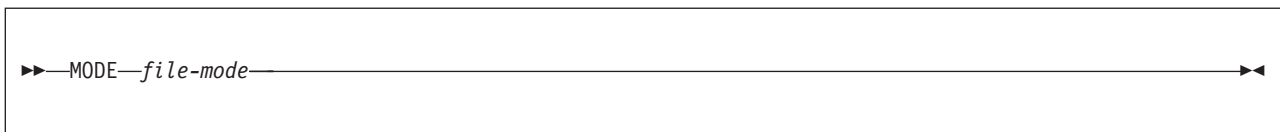
Minimizes a window.



---

## MODE (CMS only)

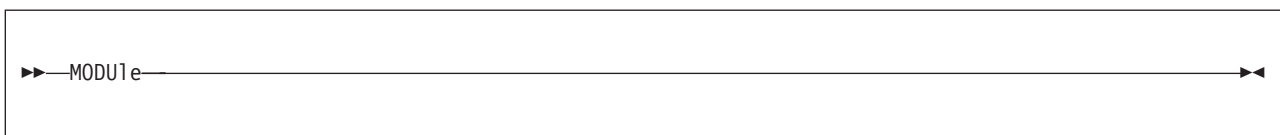
Sets the file mode for command and macro logging and play back.



---

## MODULE

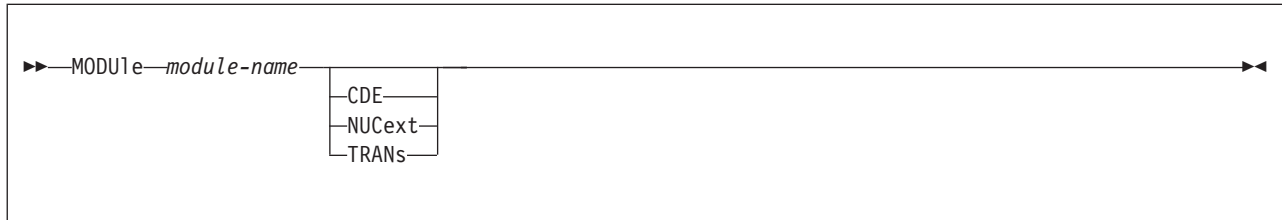
Prevents IDF from loading a target module. (Use only within a macro.)



---

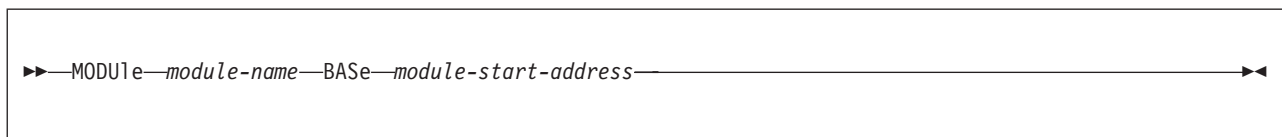
## MODULE

Sets the base and size of a module from system control blocks.



---

## MODULE BASE



Sets the base of module *modname*.

---

## MODULE SIZE

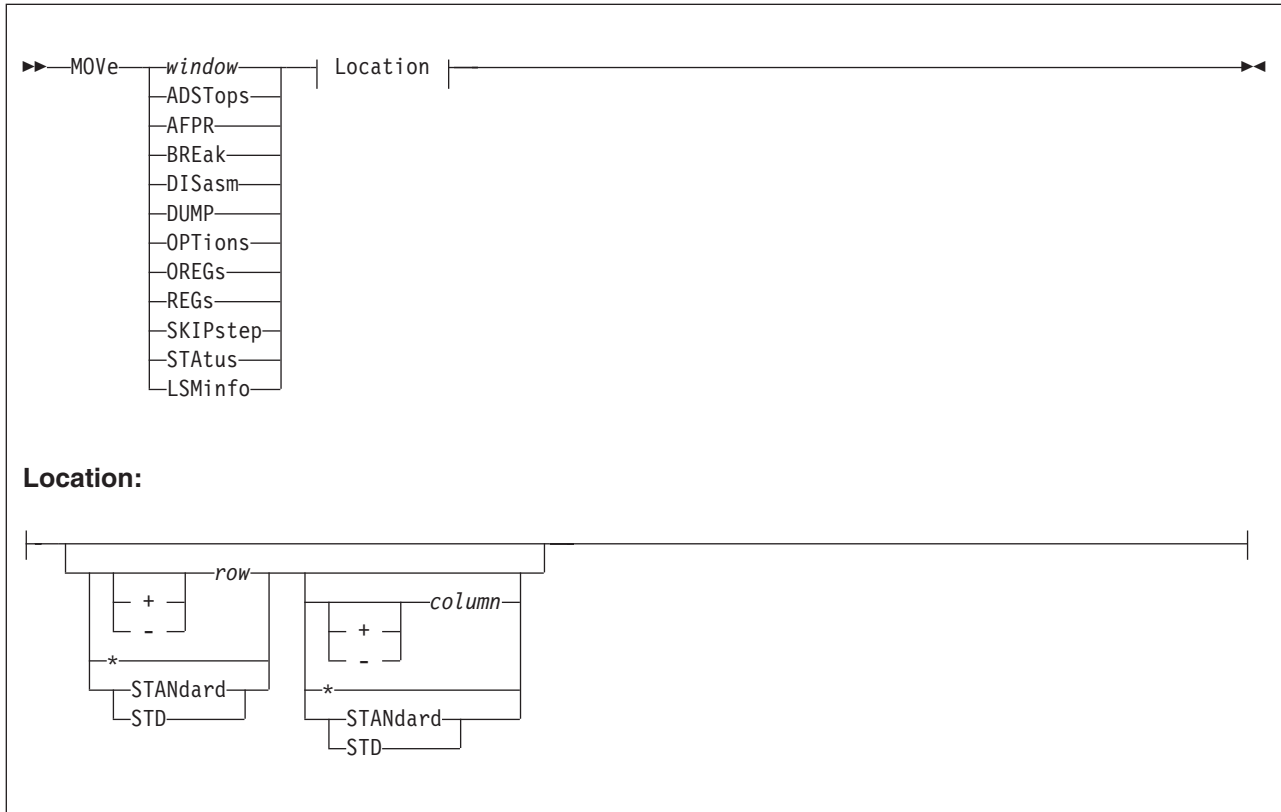
Sets the size of module *modname*.



---

## MOVE

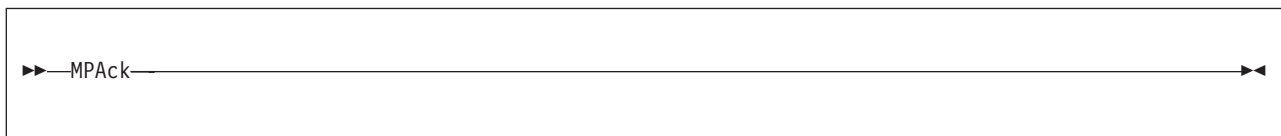
Moves a window around on the screen.



---

## MPACK

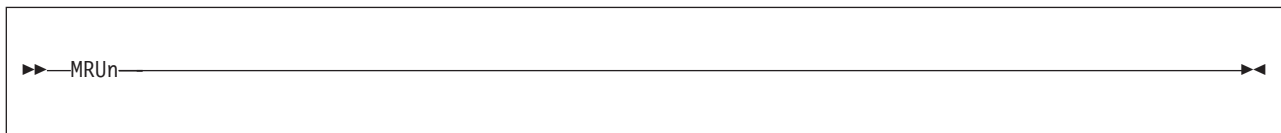
Returns unused areas in the extract data storage pool to allow use by other programs.



---

## MRUN

Execute program until next event. (Use only within a macro.)





---

## MSG

Sets the next message.

▶▶—MSG—*message-text*—◀◀

---

## MSGID (CMS and z/OS)

Toggles the display of the message identifier.

▶▶—MSGId— ON  
                   OFF—◀◀

---

## MSGMODE

Displays status and informational messages when various IDF commands have been issued via PF keys.

▶▶—MSGMode— ON  
                   OFF—◀◀

---

## MSTATUS

▶▶—MStatus—◀◀

---

Displays extract data memory status:

- number of compile areas
  - extract data storage consumption (total, direct, pooled)
  - extract data storage pool utilization, including number of AREAs in the pool which are unused
- 

## MSTEP

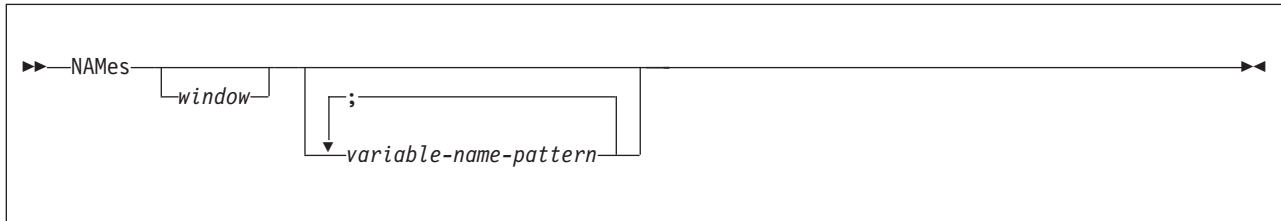
Executes the next program instruction. (Use only within a macro.)

▶▶—MStep—◀◀

---

---

## NAMES

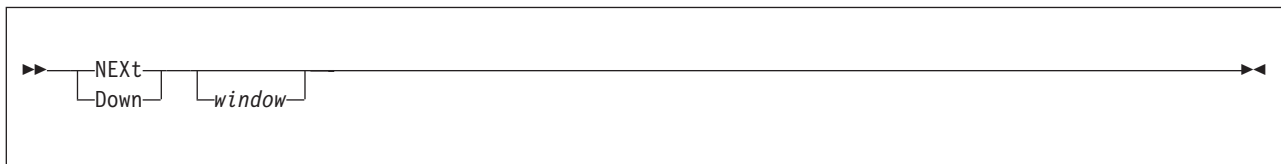


- If name patterns are specified, displays the symbol names associated with those patterns.
- Otherwise, displays all symbol names.
- All eligible symbols are shown:
  - within the current extract file with valid scoping
  - within the External Symbols List for other extract files which are loaded
- special pattern match meta-characters:
  - ? matches a single arbitrary character
  - % matches zero or more arbitrary characters
  - \ A backslash (\) followed by any character matches that character. Most useful when you need to match a real "?", "%", or "\".

---

## NEXT

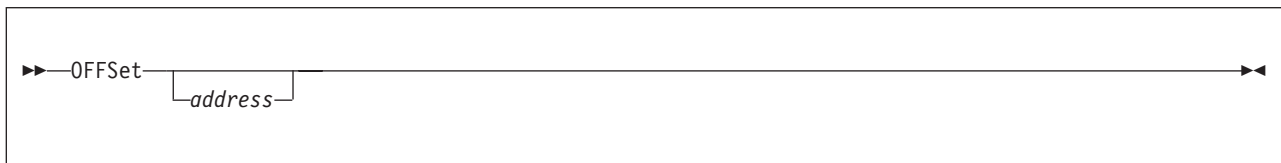
Scrolls a window forward.



---

## OFFSET

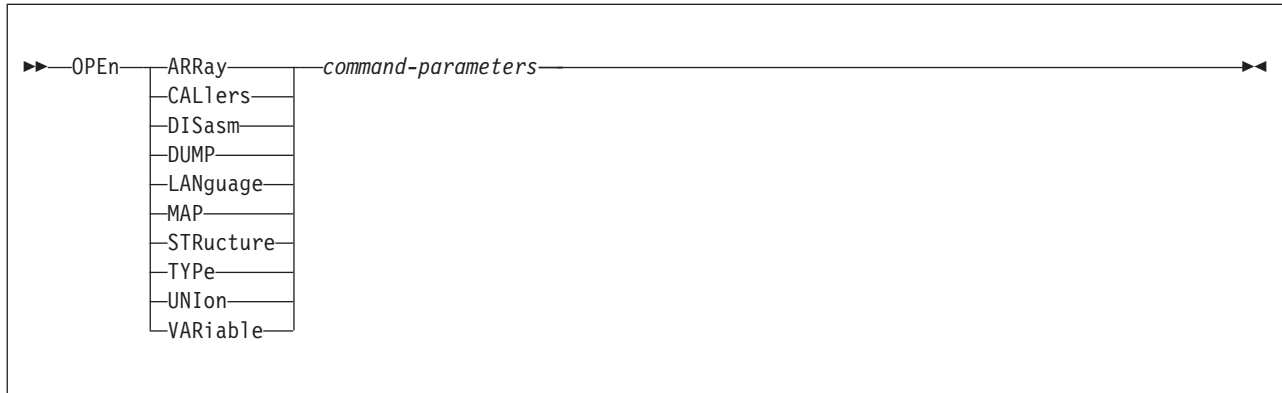
Sets or queries the current offset.



---

## OPEN

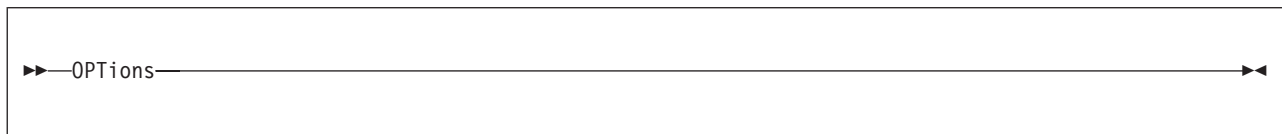
Opens a window.



---

## OPTIONS

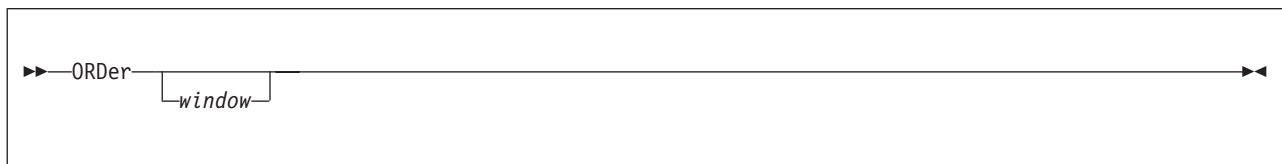
Toggles the options window.



---

## ORDER

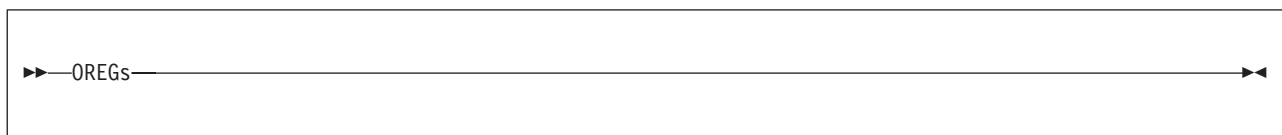
Makes a window the first displayed.



---

## OREGS

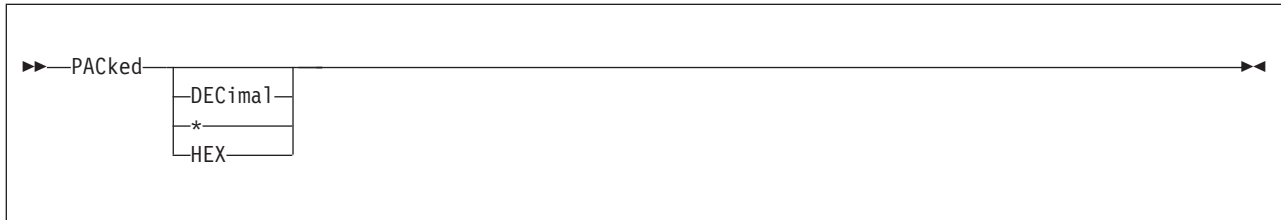
Toggles the old registers window.



---

## PACKED

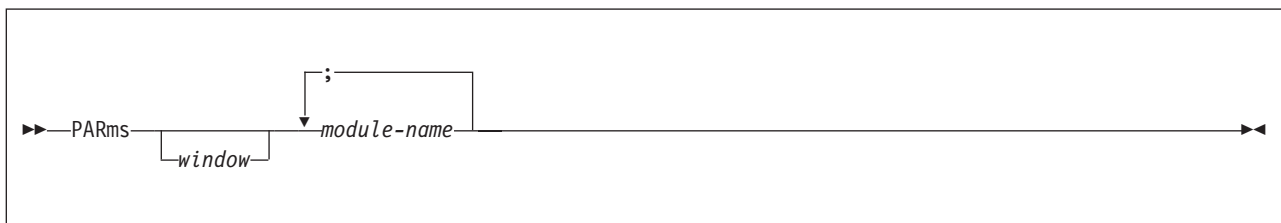
Selects the default VAR display format for Packed Decimal variables.



---

## PARMS

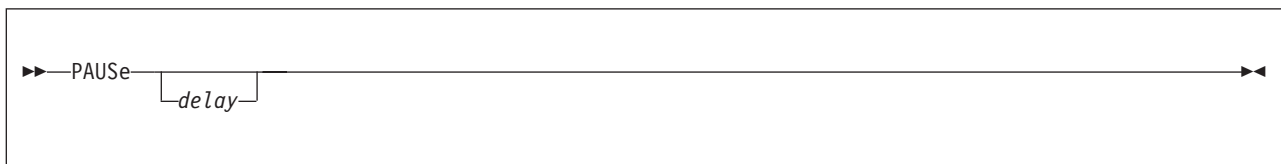
Displays the Parameter List for module names.



---

## PAUSE

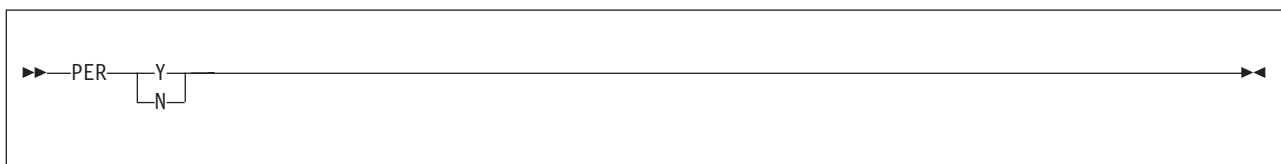
Delays the execution of IDF for a number of seconds.



---

## PER (CMS only)

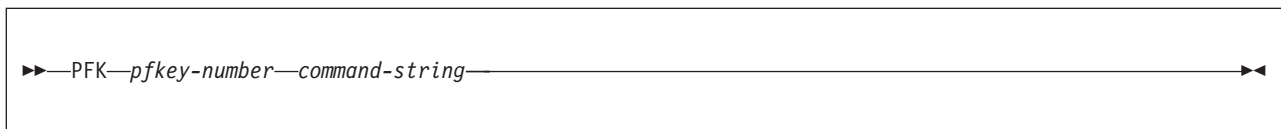
Enables or disables PER.



---

## PFK

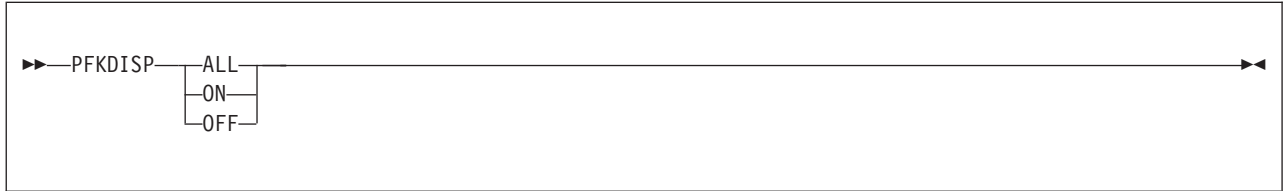
Assigns a command to a PF key.



---

## PFKDISP

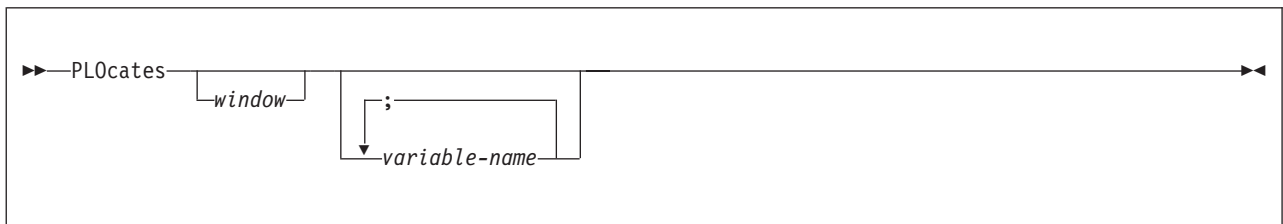
Toggles the display of the PF keys settings.



---

## PLOCATES

Displays Pointer Locates information for variables.

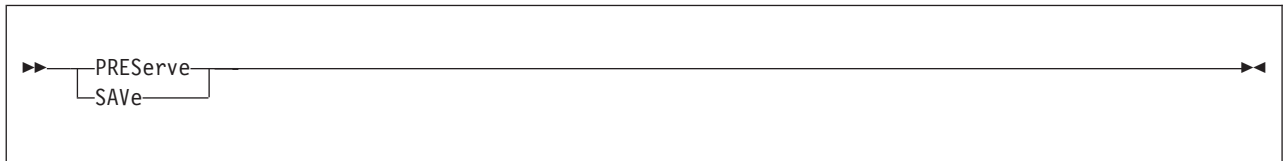


PLOCATES is a command alias which is active *after* ASMLANG has been activated.

---

## PRESERVE

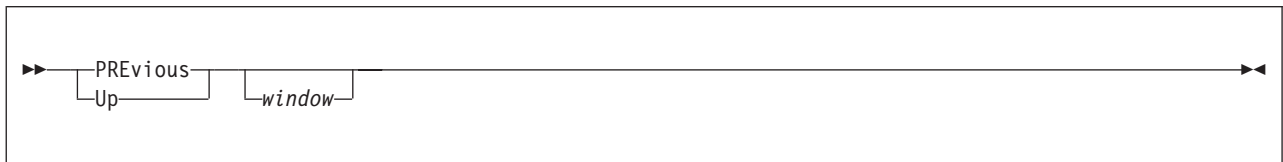
Saves LSM Options and Settings in a 32 element stack.



---

## PREVIOUS

Scrolls a window backward.



---

## PROGCK (CMS only)

Simulates program check "nn".

```
▶▶—PROGck—check-code—▶▶  
    └─PROGchk┘
```

---

## PSW

A synonym of the GOTO command.

---

## PSWSTEAL (CMS only)

Declares a PSW "stealing" location.

```
▶▶—PSWSTEAL—address—▶▶
```

---

## QUALIFY

Sets the currently qualified module.

```
▶▶—QUALify—default-module-name—▶▶
```

---

## QUIET

Disables or enables display of informational messages.

```
▶▶—QUIET—└─ON┘  
          └─OFF┘
```

---

## QUIETLY

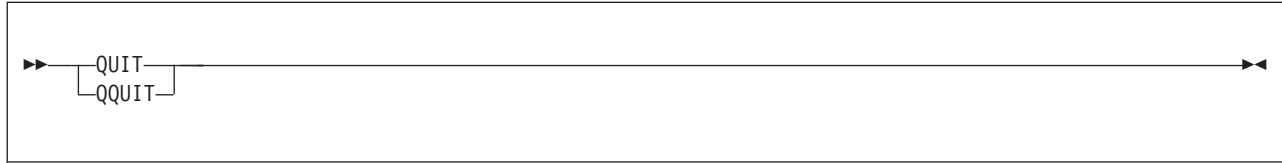
Temporarily suppresses the display of I, W, and E messages during execution of a command.

```
▶▶—QUIETLY—command-name—command-parameters—▶▶
```

---

## QUIT

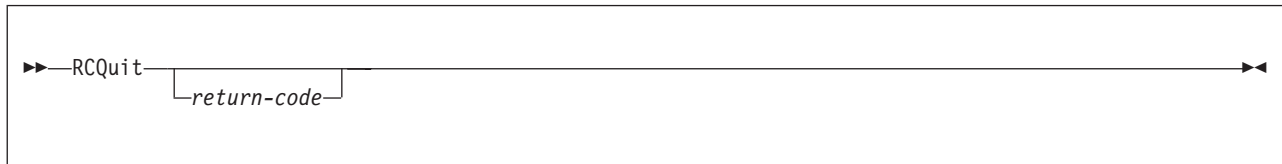
Returns to z/OS, TSO, CMS, or z/VSE.



---

## RCQUIT

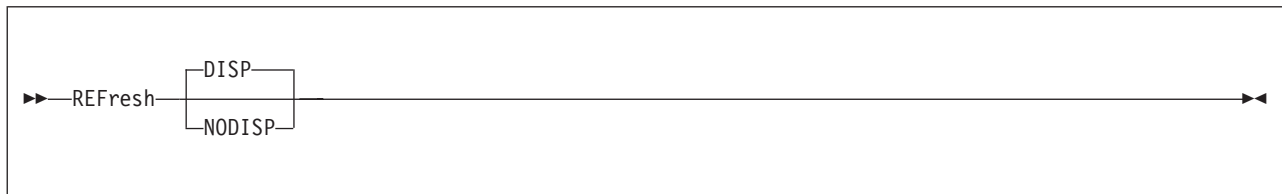
Returns to z/OS, TSO, CMS, or z/VSE with a return code.



---

## REFRESH

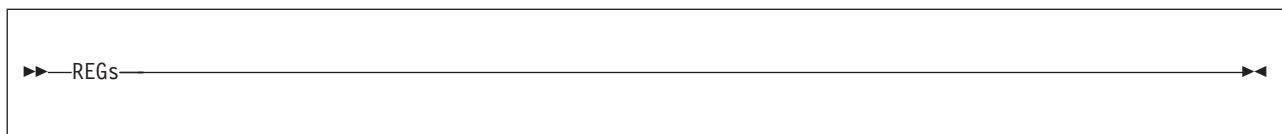
Refreshes windows.



---

## REGS

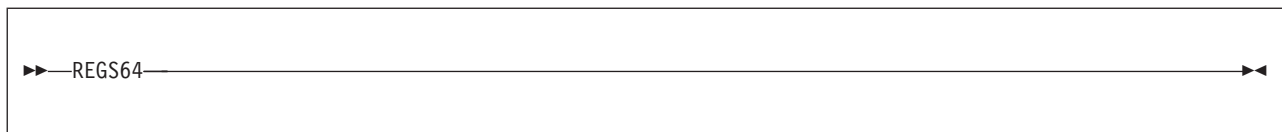
Toggles the Register display.



---

## REGS64 (z/OS only)

Toggles the Current Registers and Old Registers windows between displaying 31-bit and 64-bit registers.



---

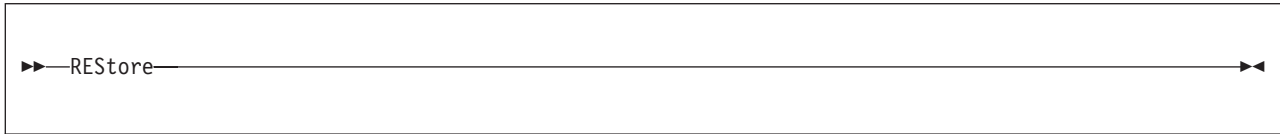
## REGSTOPS (CMS only)

A synonym of the ADSTOPS command.

---

## RESTORE

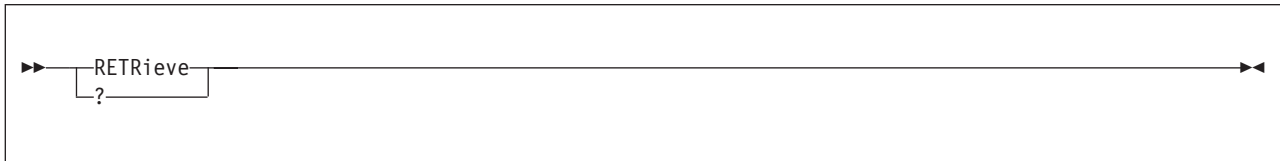
Restores LSM Options and Settings from a 32 element stack.



---

## RETRIEVE

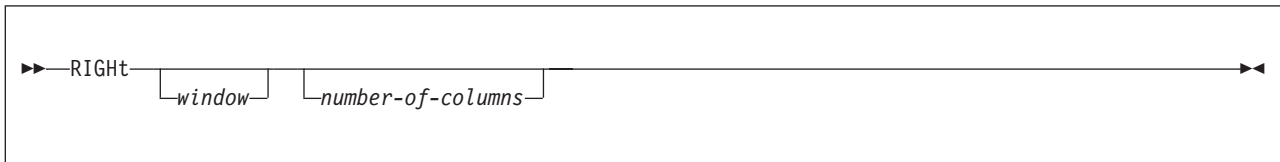
Puts the previous command in the command area.



---

## RIGHT

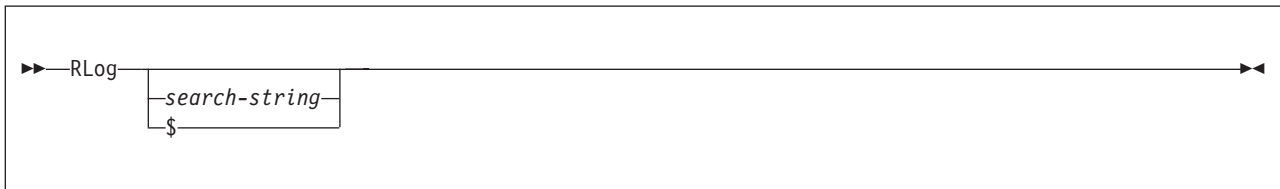
Scrolls a window to the right.



---

## RLOG

Executes commands stored in the command log.





---

## RUN

Runs the program until the next event.

▶▶—RUN—◀◀

---

## RUNEXIT

Executes the current exit routine (PFKey).

▶▶—RUNExit—◀◀

---

## R0-R15

Sets a General Purpose register.

▶▶—Rn—*expression*—◀◀

---

## SALIMIT

Sets the maximum Program Caller hierarchy depth for the CALLERS command.

▶▶—SALimit—*max-caller-display-depth*—◀◀

The default value is 100, the range is 1 to 999999.

---

## SAREGS

Enables or disables the display of Save Area header and registers for the CALLERS command.

▶▶—SAREgs— ON  
                   OFF—◀◀

---

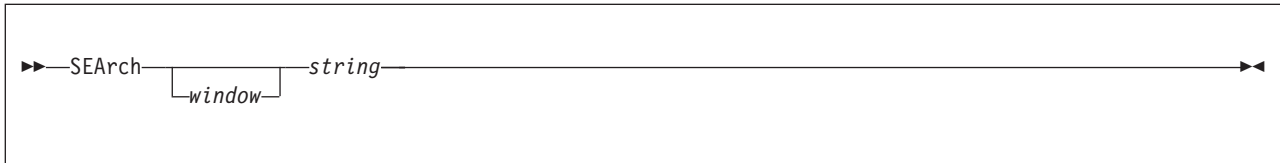
## SAVE

A synonym of the PRESERVE command.

---

## SEARCH

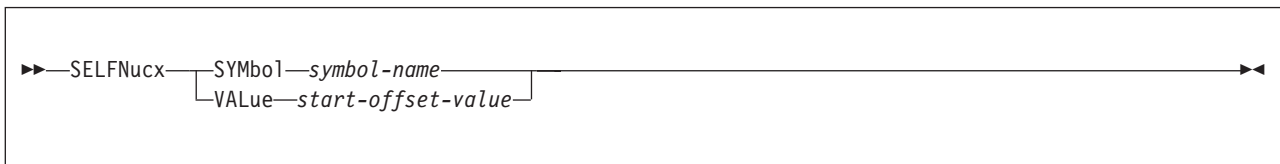
Searches for a string in storage.



---

## SELFNUCX (CMS only)

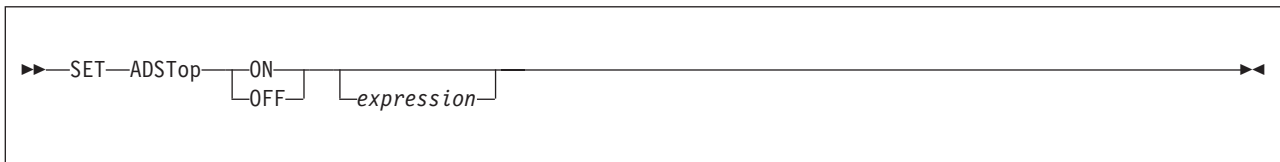
Sets the offset in module of nucleus extension.



---

## SET ADSTOP (CMS only)

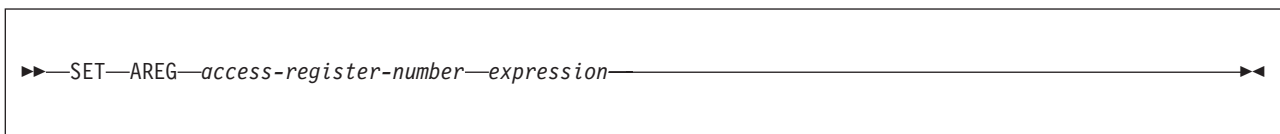
Sets or clears one end of a PER ADSTOP range.



---

## SET AREG

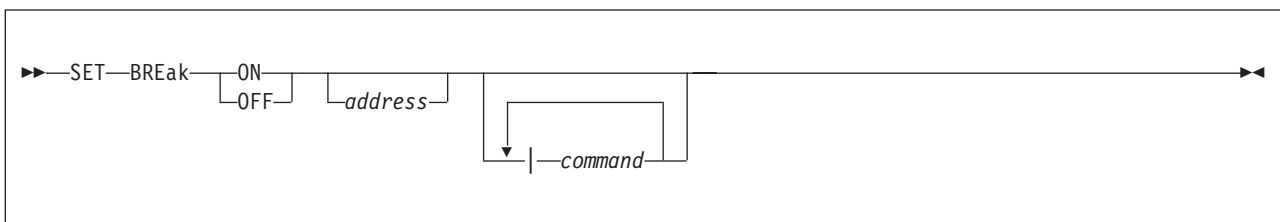
Sets ARn.



---

## SET BREAK

Sets or clears a breakpoint at an address.



---

## SET COMMAND

Places text on the command line.

▶▶ SET COMMAND *text* ▶▶

---

## SET EXITEXEC

IDF-exit-exec is the current exit.

▶▶ SET EXITEXEC 

EXIT
IDF-exit-exec

 ▶▶

---

## SET GLOBAL STEM

Writes data in a REXX stemmed array.

▶▶ SET GLOBAL 

<i>stem-name</i>
------------------

 . ▶▶

---

## SET GLOBAL TEXT

Sets the IDF global area to text.

▶▶ SET GLOBAL *global-text* ▶▶

---

## SET ICOUNT

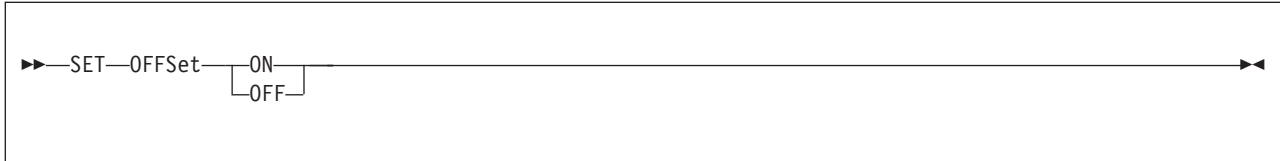
Sets the instructions counted.

▶▶ SET ICOUNT *instruction-count* ▶▶

---

## SET OFFSET

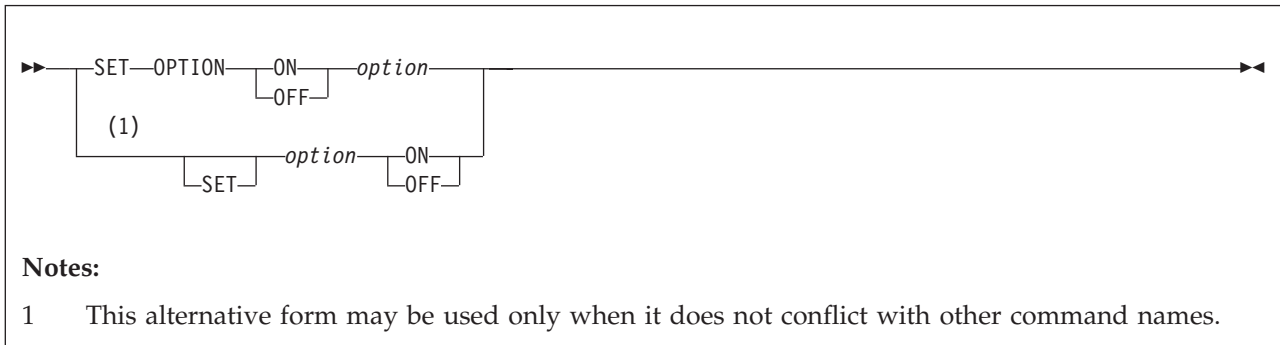
Toggles the display of addresses using offsets.



---

## SET OPTION

Enables or disables an IDF option.



---

## SET PSW

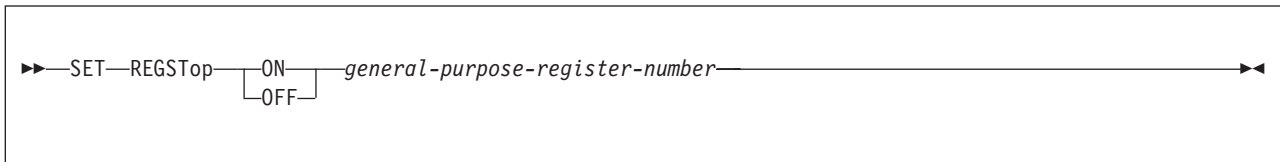
Sets the current PSW to a value.



---

## SET REGSTOP (CMS only)

Toggles PER monitoring of General Purpose Register (GPR) contents.



---

## SET SIZE

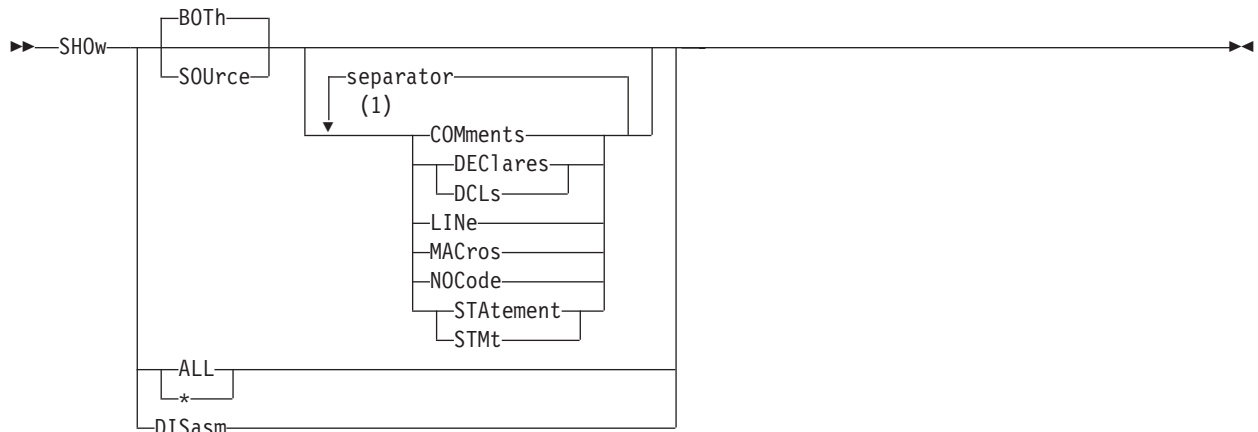
Sets the size of the program.

► SET SIZE *module-length* ◀

---

## SHOW

Controls source code and disassembly display, by showing information. The HIDE command controls the display by hiding information.



### Notes:

- 1 An option can be chosen no more than once.

### BOTH

Show both source code and disassembly

### SOURCE

Show source code only

### separator

A comma, blank, or semicolon

### COMMENTS

Show block comment source code

### DECLARES | DCL

Show declaration source code

### LINE

Show source line number with source text

### MACROS

Show macro expansion source code

### NOCODE

Show source lines with no corresponding object code

### STATEMENT | STMT

Show source statement number with source text

### ALL | \*

Show all source code and disassembly

## DISASM

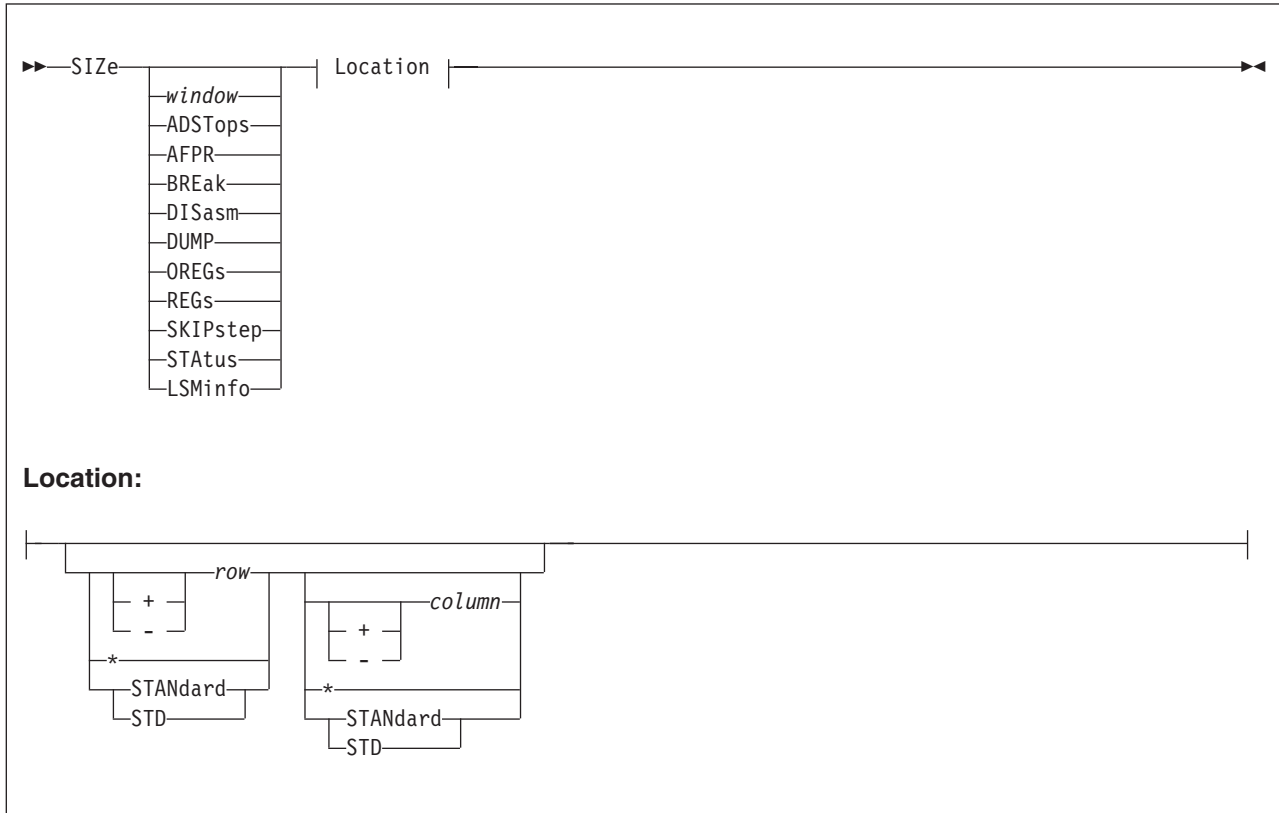
Show disassembly only

Initial settings: BOTH, COMMENTS, DCL, MACROS, NOCODE, STMT

---

## SIZE

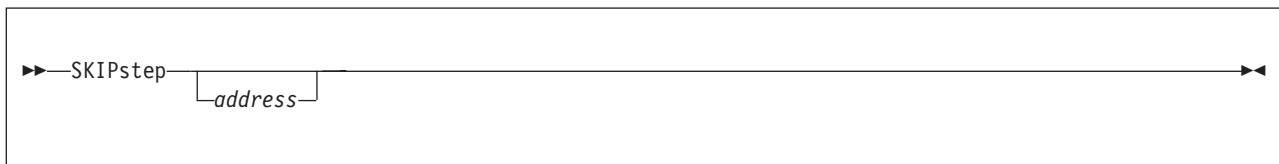
Resizes a window.



---

## SKIPSTEP

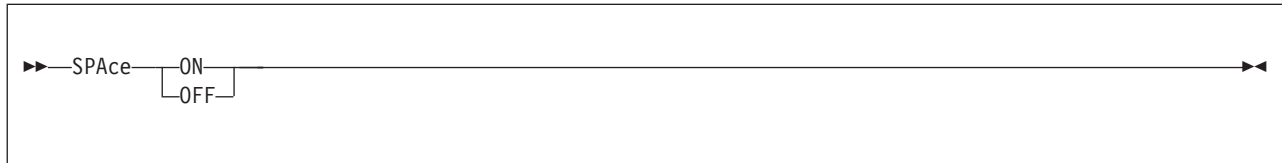
Sets a subroutine to be skipped.



---

## SPACE

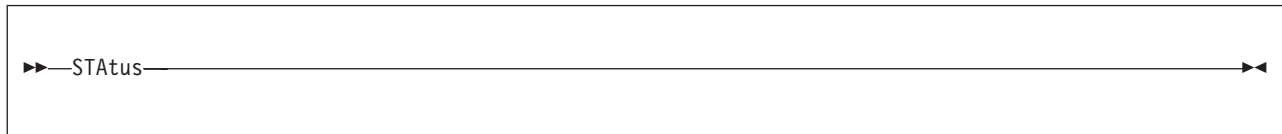
Toggles the insertion of a blank line between variables or sets of components.



---

## STATUS

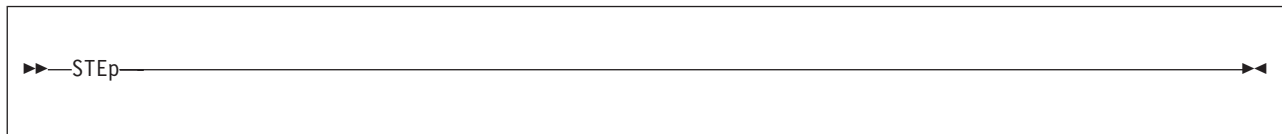
Toggles the program status window.



---

## STEP

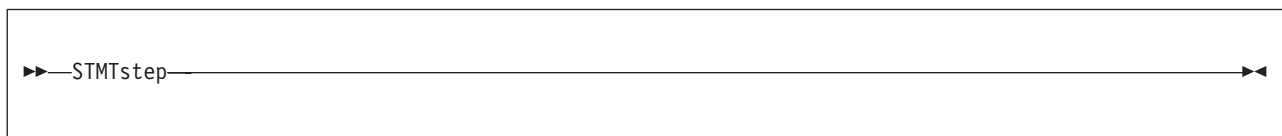
Steps to the next instruction in the program.



---

## STMTSTEP

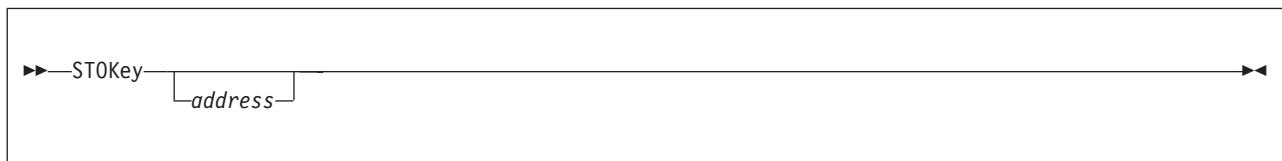
Steps to the next statement in the target program.



---

## STOKEY

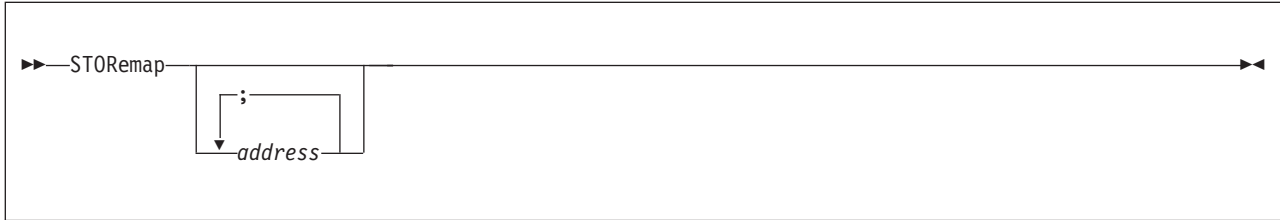
Displays the Storage Key.



---

## STOREMAP

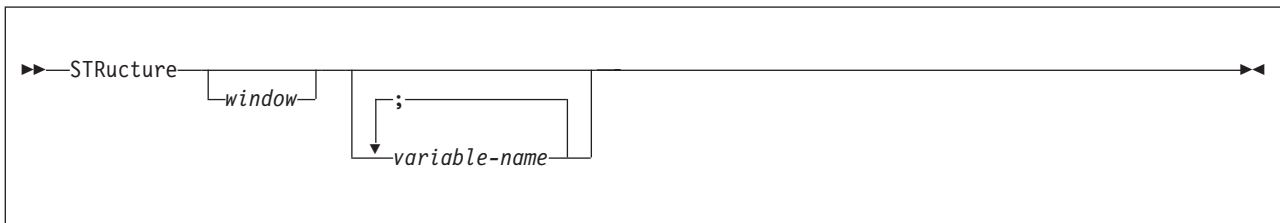
Displays information about storage allocation.



---

## STRUCTURE

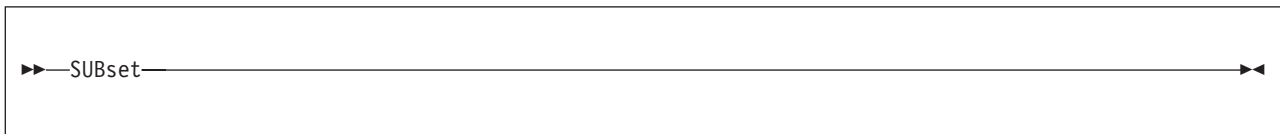
Enables variable display in a structure format.



---

## SUBSET (CMS only)

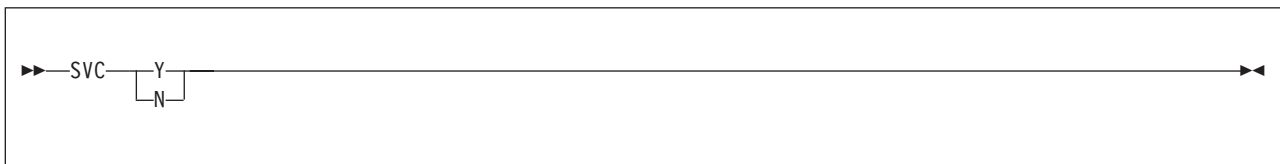
Enters a CMS Subset.



---

## SVC (CMS only)

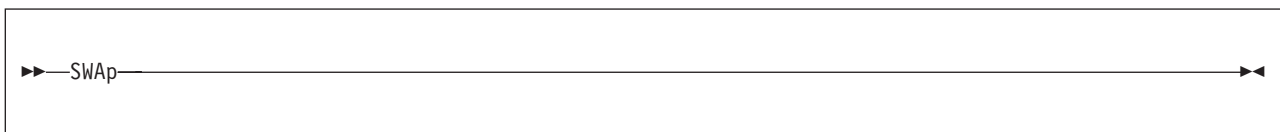
Monitors SVCs.



---

## SWAP

Displays the application screen.

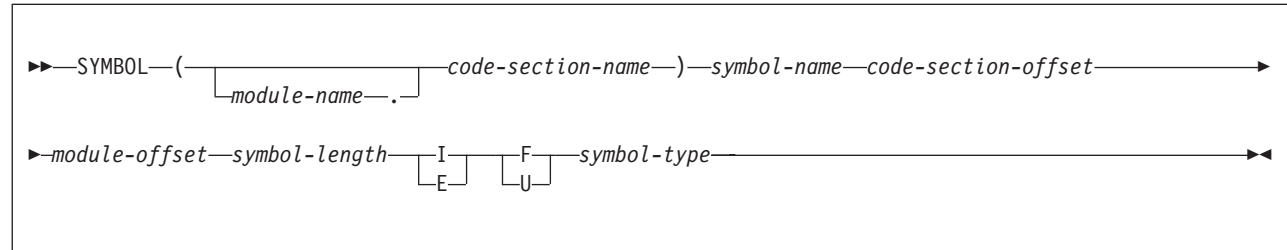




---

## SYMBOL

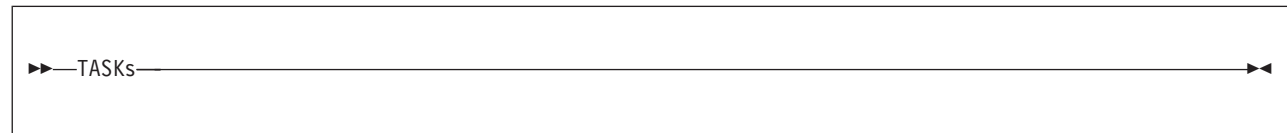
Adds a symbol to the symbol table.



---

## TASKS (TSO only)

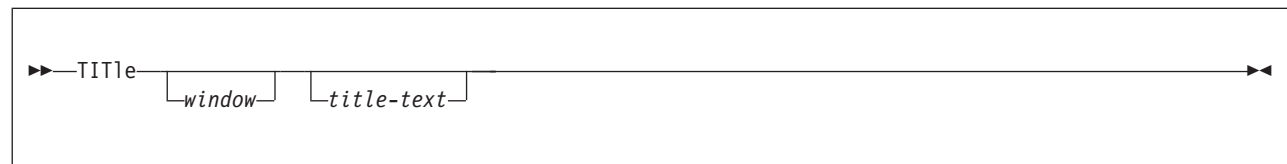
Display information about currently executing tasks



---

## TITLE

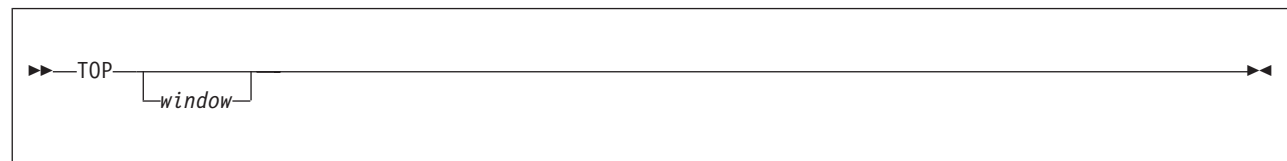
Sets the value of the title text.



---

## TOP

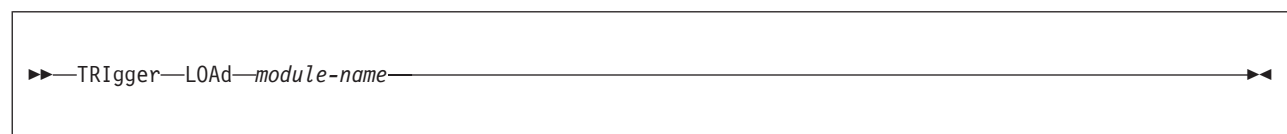
Displays source code at the lowest address within the current code section.



---

## TRIGGER LOAD

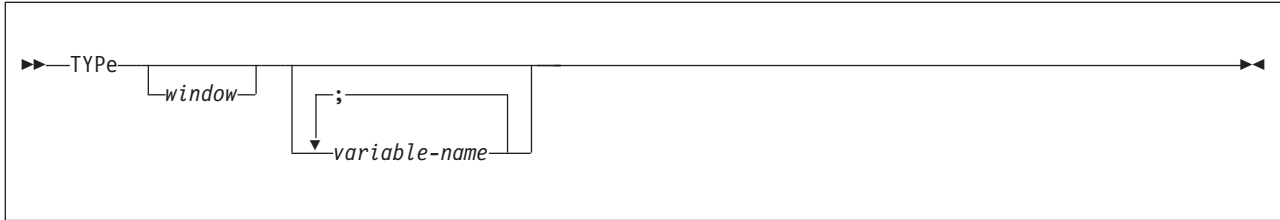
Installs deferred breakpoints in a loaded module.



---

## TYPE

Displays type attributes for variables.



---

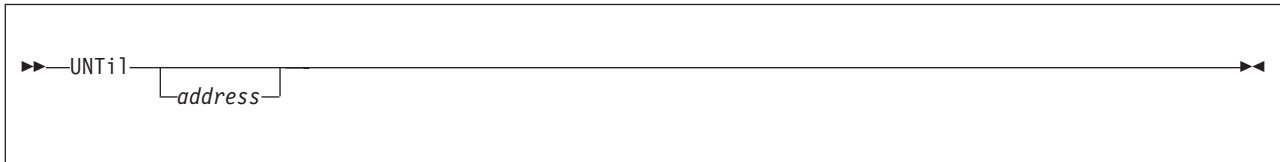
## UNION

A synonym of the STRUCTURE command.

---

## UNTIL

Executes a program until an address (not including the address).



---

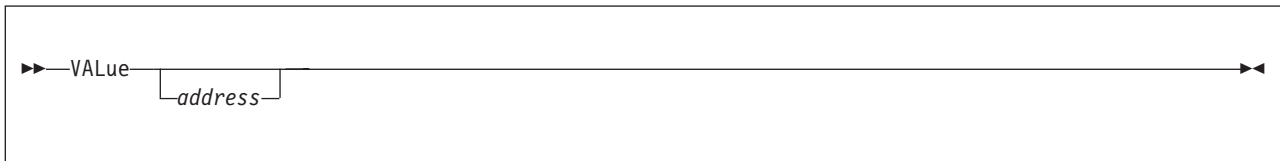
## UP

This is a synonym of the PREVIOUS command.

---

## VALUE

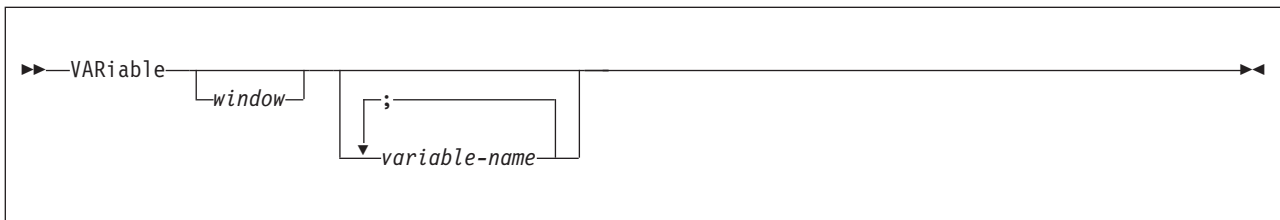
Evaluates an expression and displays it.



---

## VARIABLE

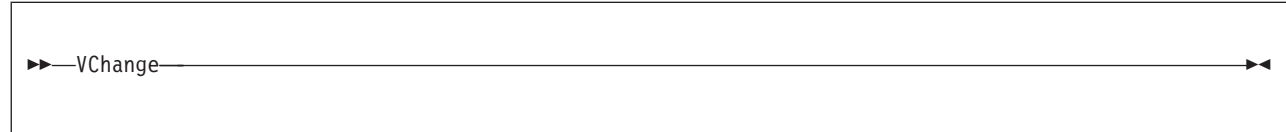
Enables variable display.



---

## VCHANGE

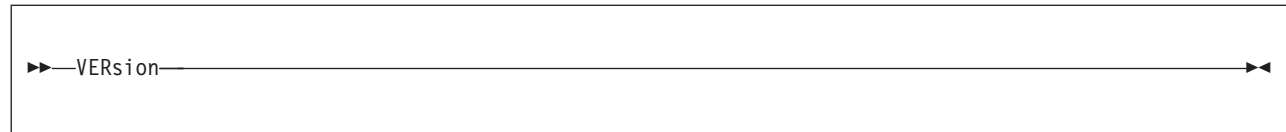
Logs commands (special purpose).



---

## VERSION

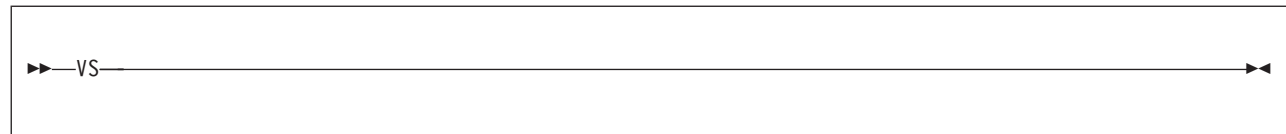
Displays the IDF Version.



---

## VS

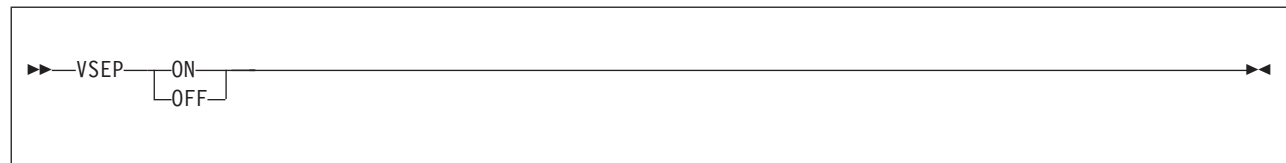
Special command logging.



---

## VSEP

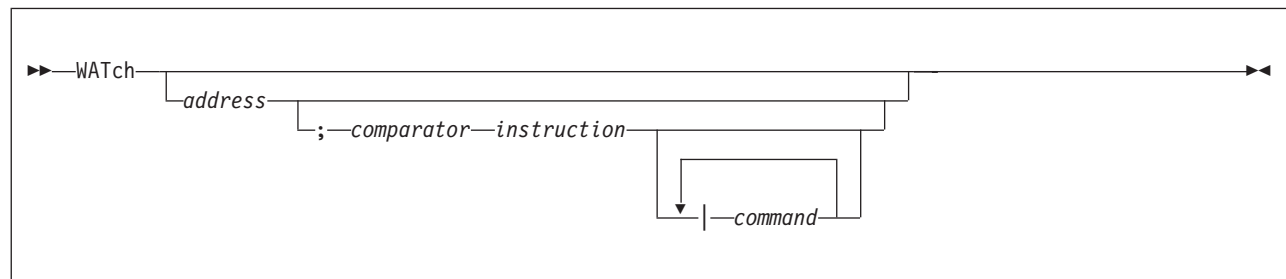
Enables or disables the blank line separating multiple variables.



---

## WATCH

Sets a "watchpoint" condition that must be true before a particular breakpoint takes effect.



*address*  
Address of the breakpoint.

*comparator*

The condition being checked, for example = or LT.

*instruction*

An S/370 comparison instruction.

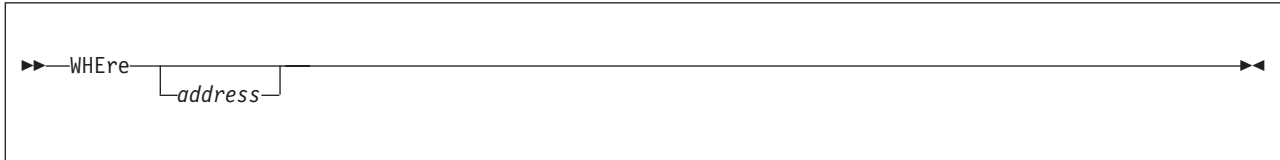
*command*

A command that is issued when the breakpoint is taken.

---

## WHERE

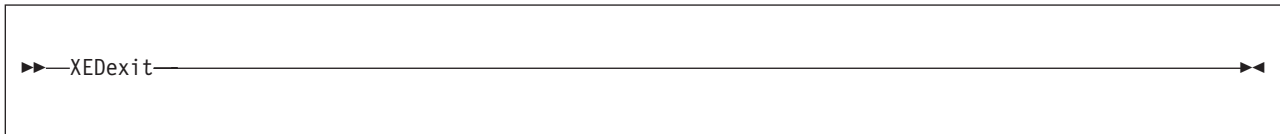
Displays the symbolic name for an address.



---

## XEDEXIT (CMS only)

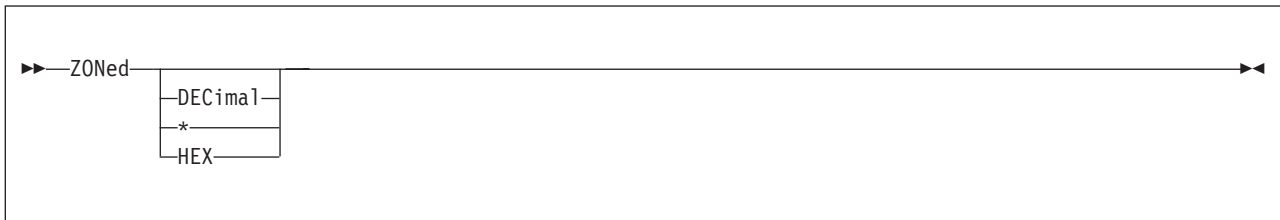
Xedit exit macro (EXIT ASM).



---

## ZONED

Selects the default VAR display format for Zoned Decimal variables.



---

## Chapter 3. ASMIDF EXTRACT Command

---

### ADSTOPS (CMS only)

All storage modification stops.

```
▶▶—EXTRact—ADSTops—▶▶
```

Sets ADSTOP.n

---

### ALET

The ALET used to qualify the dataspace to be displayed in a Dump window.

```
▶▶—EXTRact—LOCATIon—ALET—alet—number-of-bytes—start-address—▶▶
```

Sets ALET

---

### AREGS

The Access Registers

```
▶▶—EXTRact—AREGs—▶▶
```

Sets AR.n, OAR.n

---

### ARGUMENT

An address argument from the command line or cursor position.

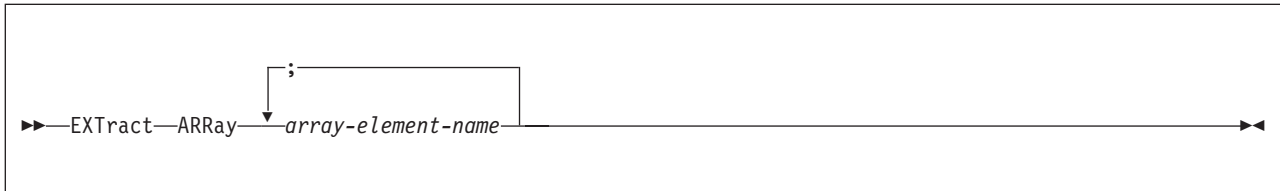
```
▶▶—EXTRact—ARGument—  
└─ARGs—┘ └─argument—┘▶▶
```

Sets SOURCE, FIELD, EXACT, INDIRECT

---

## ARRAY

Returns information about array elements.

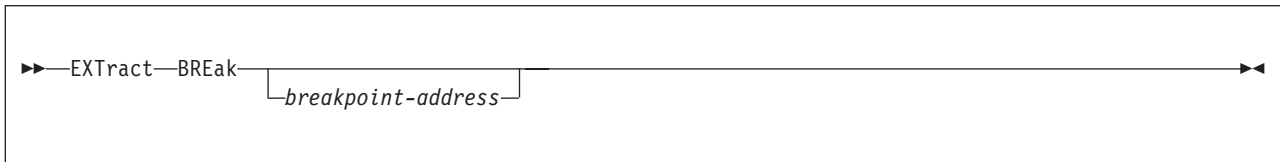


Sets stemname.0, stemname.n

---

## BREAK

One or all breakpoints.

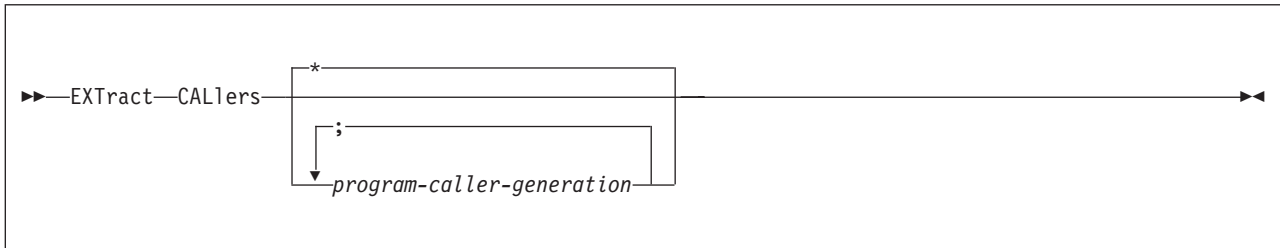


Sets BREAK.n, PBREAK.n

---

## CALLERS

Returns information for each generation in the program caller hierarchy.

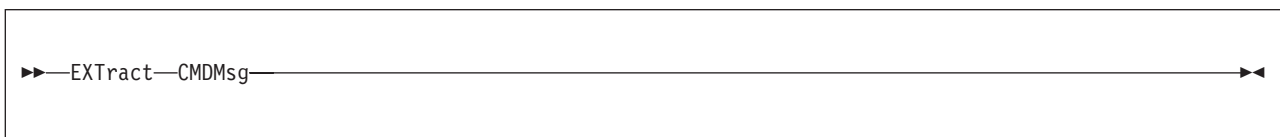


Sets stemname.0, stemname.n

---

## CMDMSG

Contents of the command line and message lines.

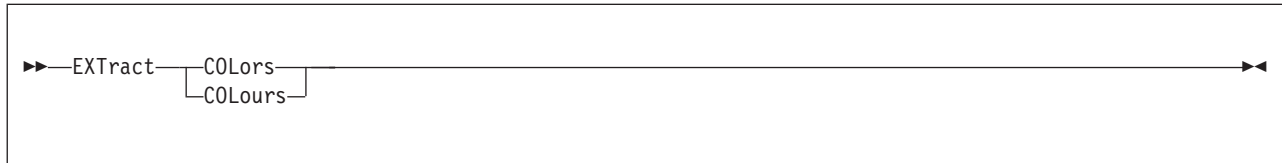


Sets COMMAND, MSG1, MSG2

---

## COLORS

Color settings.

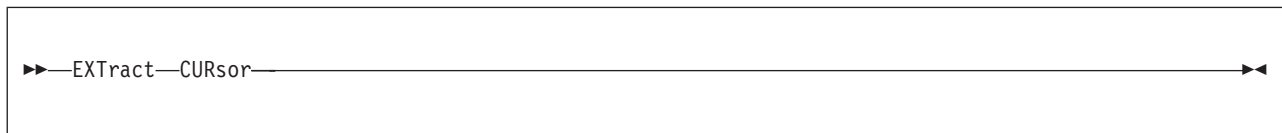


Sets COLORS

---

## CURSOR

Current position of the cursor.

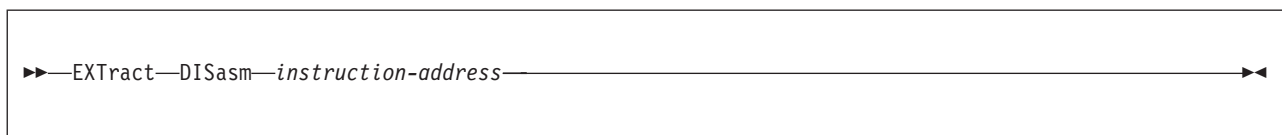


Sets DISPLAY, SOURCE, FIELD, EXACT, INDIRECT, HEXCURSR, CPDISASM, CPDUMP, NPDISASM, NPDUMP

---

## DISASM

Data about an instruction.

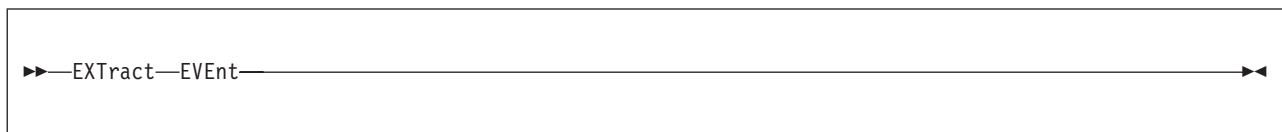


Sets INSTR, NINSTR, CSECT

---

## EVENT

Data about the last event



Sets EVENT, COMMAND

---

## EXITEXEC

The name of the currently assigned exit routine.

▶▶—EXTRACT—EXITEXEC—◀◀

Sets EXITEXEC

---

## GLOBAL

Return the current setting of the ASMIDF global variable.

▶▶—EXTRACT—GLOBAL—◀◀

Sets GLOBAL

---

## GLOBAL STEM

Return the data of the Global Storage stems.

▶▶—EXTRACT—GLOBAL—

stem-name.

---

## GLOBAL STEMS

Return the names of all currently defined Global Storage stems.

▶▶—EXTRACT—GLOBAL—STEMS—◀◀

Sets GLOBALS.0 , GLOBALS.n



---

## GSTATUS

Returns information about the storage used to contain the Global Storage data.

▶▶—EXTRACT—GSTATUS—▶▶

---

## ICOUNT

Number of instructions executed since the last ICOUNT command.

▶▶—EXTRACT—ICOUNT—▶▶

Sets ICOUNT

---

## LANGUAGE ARGUMENTS

Returns the current command arguments for each LSM information window

▶▶—EXTRACT—LANGUAGE—  
                          └─ARGs—  
                          └─ARGuments—▶▶

---

## LANGUAGE COMMANDS

Returns the current command for each LSM information window

▶▶—EXTRACT—LANGUAGE—  
                          └─COMmands—  
                          └─CMDs—▶▶

---

## LANGUAGE OPTIONS

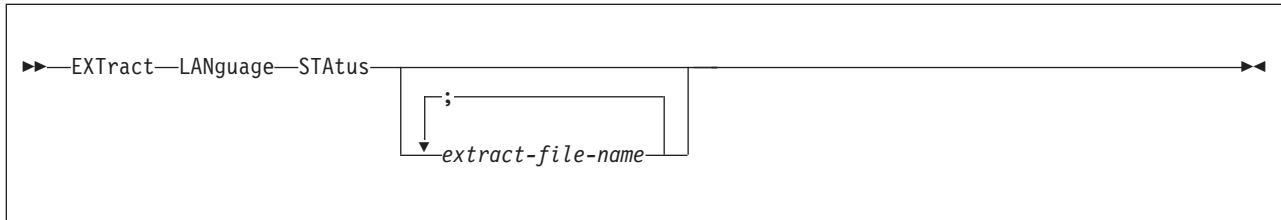
Returns information about the current value of the various ASMIDF Language Support settings.

▶▶—EXTRACT—LANGUAGE—OPTIONS—▶▶

---

## LANGUAGE STATUS

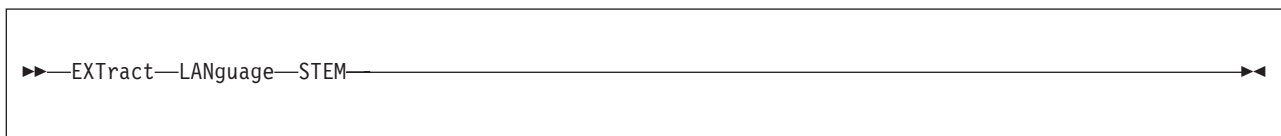
Returns information about the extract files that have been loaded



---

## LANGUAGE STEM

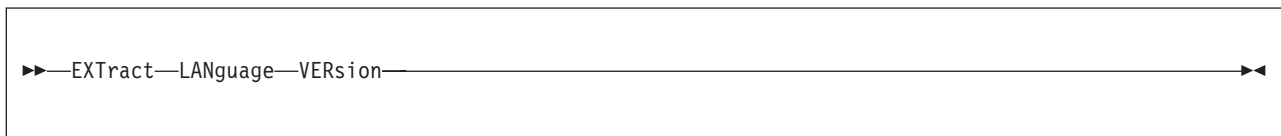
Returns the name of the REXX stemmed variable array



---

## LANGUAGE VERSION

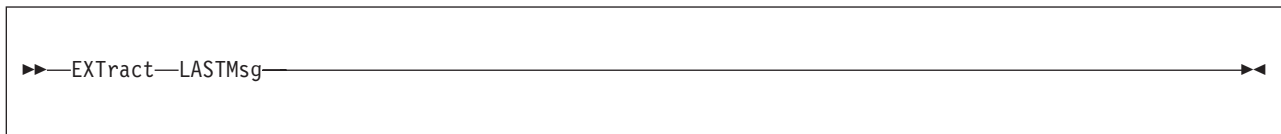
Returns the ASMIDF Language Support version



---

## LASTMSG

Returns the last ten messages issued by SET MSG

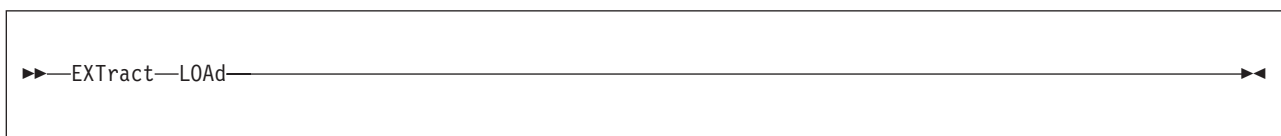


Returns LASTMSG.n and LASTMSGM.n

---

## LOAD

Obtain information about the target program



Sets NAME, AREA, SYMBOL, ORIGIN, EPOFFSET, OFFSET, SIZE, LSM, LOADLIB

---

## LOCATION

Extracts bytes of main memory

▶▶ EXTRACT LOCATION *number-of-bytes start-address* ◀◀

Sets MEMAREA

---

## LOCATION ALET

Storage from a dataspace

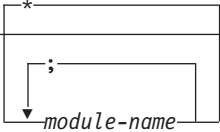
▶▶ EXTRACT LOCATION ALET *alet number-of-bytes start-address* ◀◀

Sets MEMAREA

---

## MAP

Returns information about the location of all modules and code sections known to ASMIDF.

▶▶ EXTRACT MAP  ◀◀

---

## MODE (CMS only)

Current file mode

▶▶ EXTRACT MODE ◀◀

Sets MODE

---

## MODULES

Information about defined modules

▶▶—EXTRACT—MODULES—▶▶

Sets MODULES.n

---

## MSTATUS

Returns information about the storage used to contain extract data information

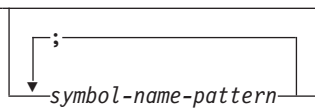
▶▶—EXTRACT—MSTATUS—▶▶

---

## NAMES

Returns information about symbol names.

▶▶—EXTRACT—NAMES—▶▶



---

## OPTIONS

A list of all ASMF options and their current settings.

▶▶—EXTRACT—OPTIONS—▶▶

Sets OPTION

---

## PER (CMS only)

Value of the PER setting.

▶▶—EXTRACT—PER—▶▶

Sets PER

---

## PFK

Current PFK definitions.

```
▶▶—EXTRACT—PFK—◀◀
```

Sets PFK.n

---

## PLIST

Arguments at the time of ASMIDF invocation.

```
▶▶—EXTRACT—PLIST—◀◀
```

Sets PLIST

---

## PLOCATES

Returns information about the variables that may be located with Locator (pointer) variables.

```
▶▶—EXTRACT—PLOCATES—locator-variable-name—◀◀
```



---

## QUALIFY

Name of the currently qualified module

```
▶▶—EXTRACT—QUALIFY—◀◀
```

Sets QUALIFY

---

## QUERY SETTING

Returns the current value of an indicator or option item

▶▶—EXTRACT—QUERY—*argument*—▶▶

Sets QUERY.n

---

## REGS

The GPRs, FPRs, and PSW

▶▶—EXTRACT—REGS—▶▶

Sets GPR.n, OGPR.n, FPR.n, OFPR.n, PSW, OPSW, FPC

---

## REGSTOPS (CMS only)

List of registers that are being monitoring.

▶▶—EXTRACT—REGSTOPS—▶▶

Sets GPR.0 - GPR.15

---

## SCOPE

Returns information about the statement scope block that corresponds to a memory address.

▶▶—EXTRACT—SCOPE—

*address*

---

## SCRVAR

Returns the contents of an LSM information window

▶▶—EXTRACT—SCRVAR—▶▶

---

## SELFNUCX

Current value of the self-load offset

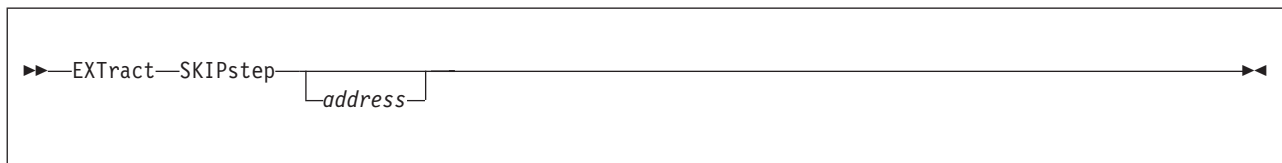


Sets SELFNUCX

---

## SKIPSTEP

One or all currently skipped subroutines.

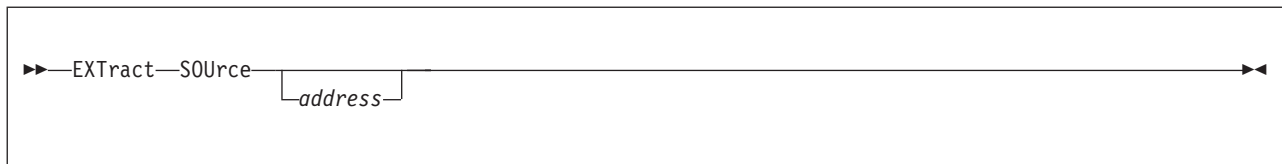


Sets SKIP.n

---

## SOURCE

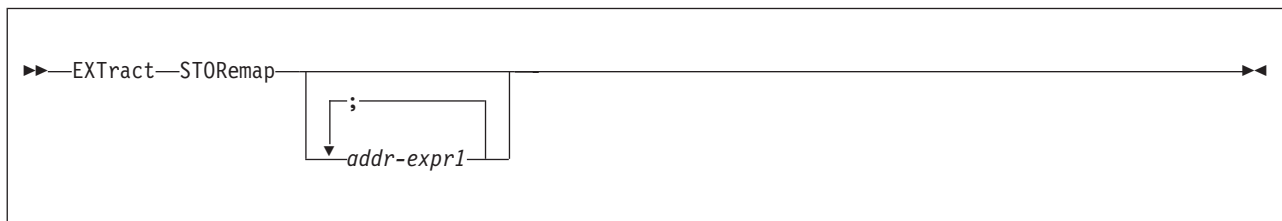
Returns the source records that correspond to a memory address



---

## STOREMAP

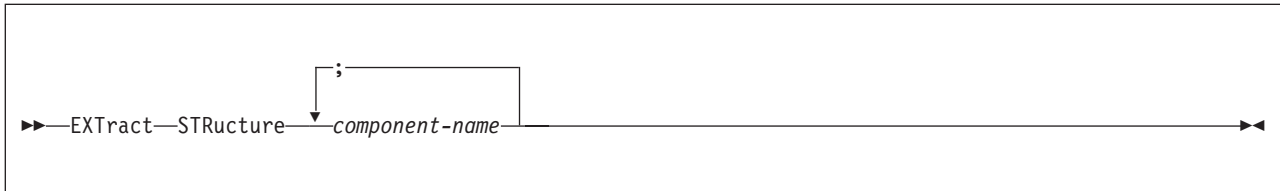
Return Storage Allocation Map information.



---

## STRUCTURE

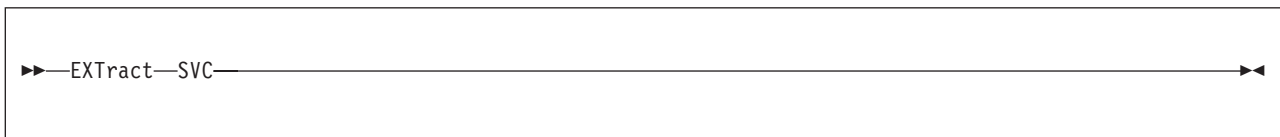
Returns information about structure and union components



---

## SVC (CMS only)

Current SVC tracing state.

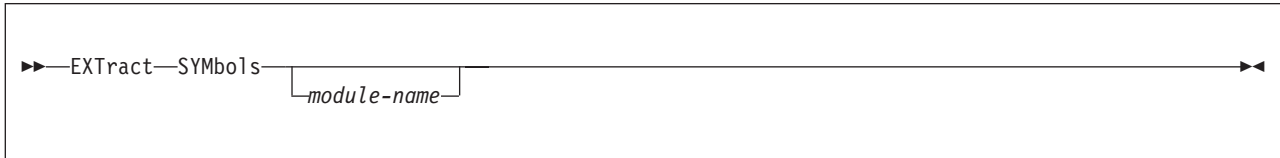


Sets SVC

---

## SYMBOLS

Information about symbols known to ASMIDF

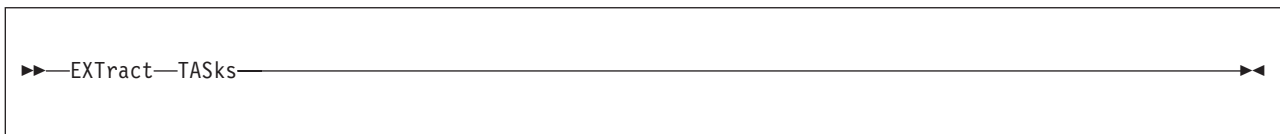


Sets SYMBOL.n

---

## TASKS

Returns information about the currently executing tasks

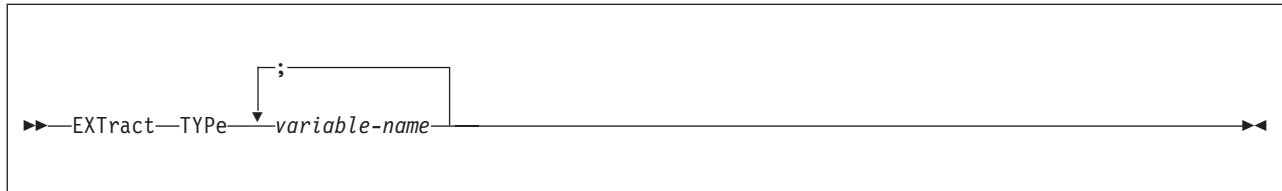




---

## TYPE

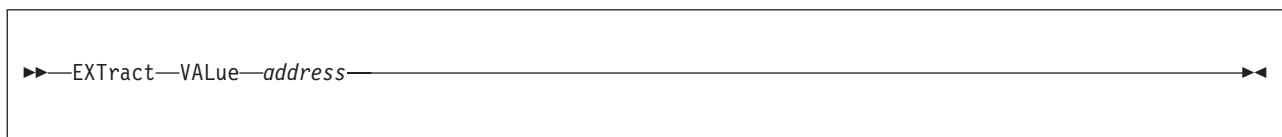
Returns information about the type attributes for variables.



---

## VALUE

Value of an expression.



Sets EXPR

---

## VARIABLE

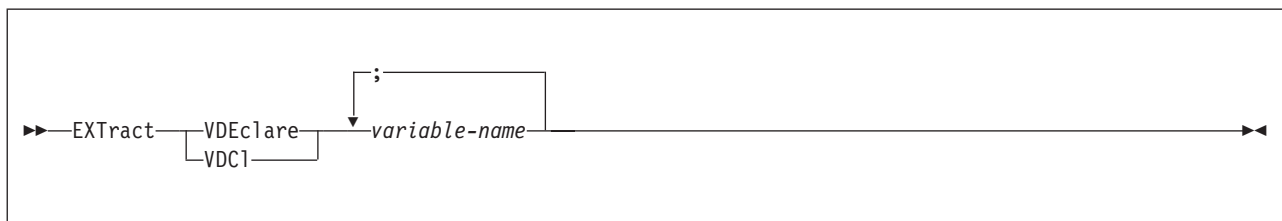
Returns information about variables



---

## VDECLARE

Returns attribute information about variables



---

## VERSION

ASMIDF Version message

▶▶—EXTRACT—LANGUAge—VERsion————▶▶

Sets VERSION

---

## VLOC

Returns location information about variables

▶▶—EXTRACT—VLOC—*variable-name*————▶▶  
                          ↓  
                          ;

---

## VVALUE

Returns data value information about variables

▶▶—EXTRACT—VVALues—*variable-name*————▶▶  
                          ↓  
                          ;

---

## WINDOWS

Information about the screen and open windows

▶▶—EXTRACT—WINDows————▶▶

Sets WINDOW.n

---

## Chapter 4. ASMIDF Options

Options may be turned on by:

```
<option> ON  
SET <option> ON  
SET OPT ON <option>
```

Options may be turned off by:

```
<option> OFF  
SET <option> OFF  
SET OPT OFF <option>
```

### **1ADSTop (CMS)**

When PER is enabled, treats the four address ranges as a single address range.

### **AMODE24**

Forces the target program to run in AMODE-24.

### **AMODE31**

Forces the target program to run in AMODE-31.

### **AMODE64 (z/OS)**

Forces the target program to run in AMODE-64.

**ASCII** Displays a dump in ASCII.

### **AUTOLoad**

ASMIDF should/should not automatically try to load LSM extract files when Statement stepping.

### **AUTOSize**

ASMIDF should/should not resize window.

**BCX** Displays branches in extended mnemonics.

### **CKSubcm**

Insures ASMIDF's Subcom is valid before running macros.

### **CMDLog**

Logs user entered commands.

### **CMPExit**

Indicates Exit is written in compiled code.

### **COLors mhti**

msg/head/text/input

Blue/Green/Pink/Red/Turquoise/Yellow/White

### **COMmand**

PLIST for target is actually a command to invoke.

### **DMS0 (CMS)**

Loads symbols that start with "DMS0".

### **EXItexec execname**

Specifies the name of the EXIT EXEC that should be used to determine breakpoint applicability.

### **FASTPath**

Uses fast version of PATH.

### **FULLQual**

Symbolic addresses should always be fully qualified.

**HEXDisp**  
Displays offset in hexadecimal.

**HEXInput**  
Numbers without explicit base are hexadecimal.

**IMPMacro**  
Permits implicit macros from command line.

**INVPsw**  
Accepts invalid PSWs on a SET PSW command.

**ISA address (CMS)**  
Defines the address of a 16-byte double-aligned interrupt save area.

**LIBE fn/\$ (CMS and z/OS)**  
Loads from specified DDname.

**LINE X'*nmn*' (CMS)**  
Uses a terminal other than the virtual console.

**LSMDebug**  
Displays LSM debugging information.

**LUname *lu\_unit* (z/VSE and z/OS)**  
Defines the VTAM logical unit name of the terminal used by IDF.

**MACROLog**  
Logs commands entered from macros.

**MODE *xx* (CMS)**  
The CMDLOG and PATHDATA files are read from, or written to, the minidisk at the specified file mode.

**MODMap (CMS)**  
Uses fn MAP before LOAD MAP for symbol information.

**NOAUTOLD**  
Do not automatically try to load LSM extract files when Statement stepping.

**NOAUTOSz**  
Do not automatically resize windows.

**NOBcx**  
Do not display branches in extended mnemonics.

**NODSects**  
Do not load symbols in DSECTs.

**NOIMPMac**  
Disallows the implied execution of macros from the command line.

**NOINVPsw**  
Does not accept invalid PSWs on a SET PSW command.

**NOMODMap (CMS)**  
Prefers the "LOAD MAP" file to the "modname MAP" file.

**NOProfil**  
Do not run a profile macro.

**NOSTOPNp**  
Do not put internal breakpoints at a NOP(R) after a BAL(R).

**NOSTOPSt**  
Do not stop statement stepping when not in a statement.

**NOSVC97 (z/OS)**  
Do not use SVC 97 for events.

**NUCext (CMS)**  
Runs the program as a CMS nucleus extension.

**OFFSet**  
Displays address in offset format.

**OLDBREAK**  
Uses the old operation of the Break command.

**PASspgm**  
Passes program interrupts to the target.

**PATH** Displays the number of times each instruction has executed.

**PATHFile**  
Writes the number of times each instruction has executed to a file.

**PROfile name**  
Runs REXX procedure 'name' as the profile.

**QWDump**  
Forces unformatted Dump display to begin on a fullword.

**RISk** Ignores as many "errors" as possible.

**RLog** Replays all previously logged user commands.

**ROWstyle**  
Uses row style for display of registers.

**SBORDer**  
Uses simple border characters.

**SCDactiv**  
Collapses ASMIDF Subcom before running target.

**SELFNucx *symbol* (CMS)**  
The code is self-nucxloading.

**STOPNOP**  
Places internal breakpoints at a NOP(R) after a BAL(R).

**STOPStnt**  
Stops statement stepping when not in a statement.

**SVC97 (TSO)**  
Uses SVC 97 for events.

**SWAp** Enables the capture of a target program's screen image.

**SYStem (CMS)**  
Runs the program in system key (key=0).

**TRACeall**  
All instructions are traced in single stepped mode.

**TRANs (CMS)**  
Runs the program as a transient.

**UNFtdump**  
Displays Dump in unformatted mode.



---

## Chapter 5. ASMIDF Language Support

Some examples in this section use the < > characters as follows:

< **item** >

Item (such as parameter or word) is optional

-> Represents the based-pointer notation

---

### Introduction

ASMLANG is a Language Support Module (LSM) subsystem which acts as an extension to ASMIDF and provides source-level debugging capabilities for assembler programs. ASMLANG uses extract files which contain the language source and variable information. The extract files are created by the ASMLANGX utility using the SYSADATA files provided by the IBM High Level Assembler (HLASM).

---

### A word about variables

Information for all variables in the user's program is extracted into a common format by ASMLANGX.

ASMLANG displays variables using terminology like that of PL/I. Where necessary, extensions have been made - for example, FLOAT, PACKED DECIMAL and ZONED DECIMAL are terms used by ASMLANGX.

---

### Invocation

ASMLANG is integrated with the base debugger module, ASMIDF, and the LSM support is activated during ASMIDF initialization.

To *explicitly* load ASMLANG extract files from the ASMIDF command line or via a macro, you can issue the following command:

```
LOAD LANGUAGE efn eft efm (options)
```

To *implicitly* load ASMLANG extract files you can use an LSM command such as STMTSTEP.

Parameters:

- EFN** Extract file name
- On z/OS, the PDS member name of the extract file created by ASMLANGX.
  - On CMS, the file name of the extract file created by ASMLANGX.
  - On z/VSE, the file name of the extract file created by ASMLANGX.
- EFT** Extract file type
- On z/OS, the DD name allocated to the extract file created by ASMLANGX.
  - On CMS, the file type of the extract file created by ASMLANGX.
  - On z/VSE, not used.
  - Specifying this option eliminates the search using the XPATH file types (DD names).
  - The default XPATH is "ASMLANGX".
- EFM** Optional
- On CMS, the FM of the extract file created by ASMLANGX.
  - On z/OS, not used, ignored if specified.
  - On z/VSE, not used, ignored if specified.

---

## Options

### MODULE *modname*

Module with which to associate the extract file.

If this option is *not* specified:

- If extract file contains information which requires load-time resolution it defaults to the qualified target module.
- Otherwise, the extract file is "generic" where it is freely associated with any relevant CSECTs in all MODULEs.

---

## Displaying source

Use the ASMIDF DISASM command:

```
DISASM (module.csect)stmt#nnn
DISASM 0(PSW)
```

---

## Displaying variables

Use the VAR command:

```
VAR var
```

Multiple variables may be displayed:

```
VAR var1<;var2<;var3;...;varn>>
```

Locating expressions may be used:

```
VAR ptr1->ptr2->based_var
VAR array_var(1,3,4)
VAR struct_var.member[1,3]
VAR triglyphs.too??(1,3??)
```

Substring specification may be used:

```
VAR chrstr(1);chrstr(2:3);chrstr(1::4)
VAR cstring[0::4]
```

ADDR() function can be used:

```
VAR Addr(buff(1))->based_var
```

---

## Displaying structures

Use the STRUCT command:

```
STR strname
STR strname.substructure
```

The expression syntax is the same as for the VAR command.

---

## Displaying array elements

Use the ARRAY command:

```
Arr arrayvar1(2);array2[3,55]
```

The expression syntax is the same as for the VAR command.

When the display is scrolled, the array indexing is also scrolled.



---

## Altering variables

1. Display variables using any of the previous commands.
2. Type over the current variable contents
3. Press the Enter key

---

## Displaying type attributes

Use the TYPE command:

```
TYPE var
```

Type attribute information includes:

- Fundamental data type
- User defined data type
- Type hierarchy

Type attributes for multiple variables may be displayed:

```
TYPE var1<;var2<;var3<;...;varn>>
```

---

## LANGUAGE command aliases

With ASMIDF a facility called "Command Alias" is available which allows ASMLANG to add additional commands *without* the LANGUAGE prefix.

Command Alias is an ASMIDF facility that enables:

**Alias**    **Equivalent ASMLANG command**

**/**        LANGUAGE LOCATE /

**-/**       LANGUAGE LOCATE -/

**BOTtom**  
          LANGUAGE BOTTOM

**CALLers**  
          LANGUAGE CALLERS

**F**        LANGUAGE FIND

**FIRst**   LANGUAGE FIRST

**GOTo**    PSW

**HIDe**    LANGUAGE HIDE

**LAST**    LANGUAGE LAST

**LLocate**  
          LANGUAGE LOCATE

**Locate**   LANGUAGE LOCATE

**MAP**     LANGUAGE MAP

**MPAck**  
          LANGUAGE MPACK

**MStatus**  
          LANGUAGE MSTATUS

**NAMes**  
          LANGUAGE NAMES

**PARms**  
          LANGUAGE PARMS

**PLOcates**  
          LANGUAGE PLOCATES

**SHOw**  
          LANGUAGE SHOW

**TOP**     LANGUAGE TOP

**TYPE**    LANGUAGE TYPE

## Hints and tips

1. Maximum LSM window display lines

When displaying multiple variables use the LANGUAGE VSEP OFF command. This increases the number of screen lines available to display the variable information.

2. z/OS Dataset Conventions

On z/OS, ASMIIDF and ASMLANGX commands do not change. The CMS conventions are used, with the following mapping of the CMS file conventions to z/OS:

**CMS    z/OS Equivalent**

**fn**     PDS member name (ignored if using sequential file)

**ft**     DDNAME, which in turn points to the z/OS dataset name

**fm**     not used on z/OS

You must allocate the DDs using ALLOC (CLIST or EXEC) or DD (JCL).

3. z/VSE Dataset Conventions

On z/VSE, ASMIIDF and ASMLANGX are invoked from JCL, with the following mapping of the CMS file conventions to z/VSE:

**CMS    z/VSE Equivalent**

**fn**     z/VSE librarian member name.

**ft**     DLBL name, which in turn points to the z/VSE file name.

**fm**     not used on z/VSE

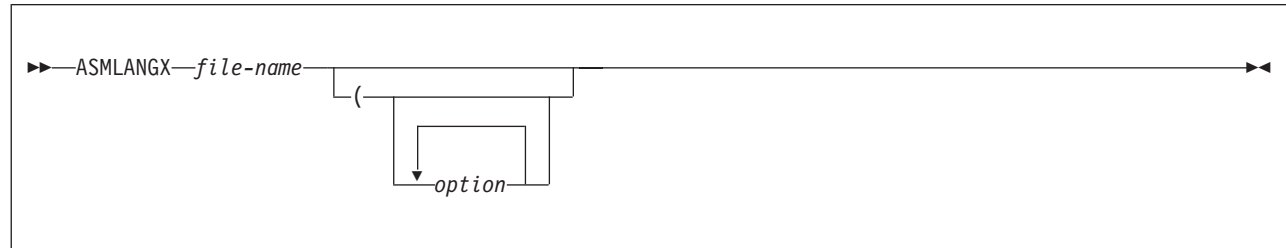
---

## Chapter 6. Using ASMLANGX

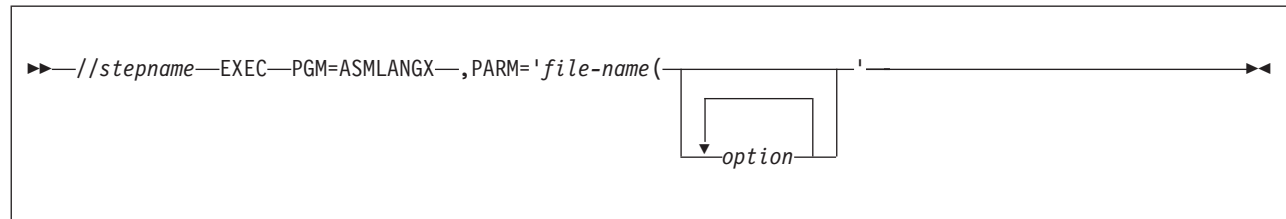
---

### Invocation

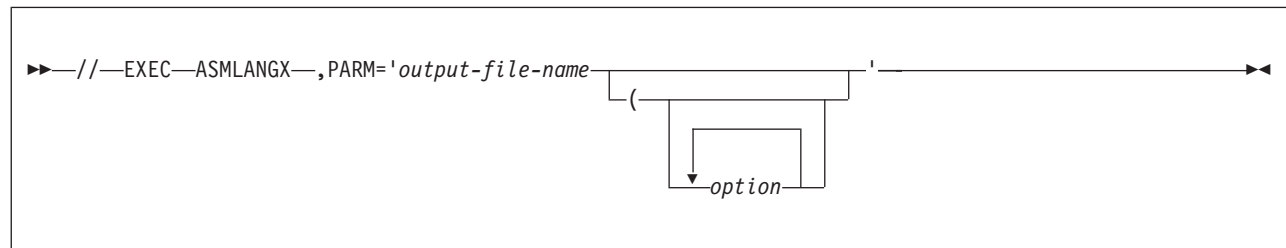
#### ASMLANGX CMS and TSO syntax



#### ASMLANGX z/OS EXEC syntax



#### ASMLANGX z/VSE EXEC syntax



Parameters:

#### input\_file\_name

Input file name

- On TSO, the PDS member name of the SYSADATA input file.
- On CMS, the file name (FN) of the SYSADATA input file.
- On z/OS in batch, this defaults to the SYSADATA file created by the High Level Assembler.
- On z/VSE, this defaults to the SYSADATA file created by the High Level Assembler.
- Dummy token required *only* if the extract file name is to be modified.

#### output\_file\_name

Output file name.

- On TSO, or CMS, this is optional and defaults to the same name as the input file with a file type (CMS) of ASMLANGX, or the PDS name of the ASMLANGX DD (TSO) name.
- On z/OS, this must be specified with PARM='output\_file\_name'.
- On z/VSE, this must be specified with PARM='output\_file\_name'.

---

## Options

General Options:

**ASM** Extract is for Assembler.

**CONDASM**

Include conditional assembly statements.

**DCL** Suppress source for declarations. This is the default option.

**DEBUG**

Log standard and internal diagnostic messages.

To be used as directed by IBM service personnel.

**ERROR**

List invalid or incomplete extract records.

**IFM file mode**

input file mode

- On CMS, the file mode to search for the input files. Standard search order used if not found.
- On z/OS, not used.
- On z/VSE, not used.

**INCL** Extract source from INCLUDE files. This is the default option.

**LOUD**

Issue progress or error messages.

**MACDEF**

Include inline macro definitions.

**NOCONDASM**

Exclude conditional assembly statements.

**NOINCL**

Suppress source from INCLUDE files. Variable information is still extracted.

**NOMACDEF**

Exclude inline macro definitions.

**NOPACK**

Disables packing of source statement text

**NODCL**

Suppress source for declarations (including associated block comments). Variable information is still extracted.

**NOSEQ**

Suppress source record sequence numbers.

**OFM file mode**

Output file mode

- On CMS, the file mode of the output file, default "A1".
- On z/OS, not used.
- On z/VSE, not used.

**OFN filename**

Output file name

- On CMS, the file name of the output file.
- On z/OS, the PDS member name of the output file.
- On z/VSE, the librarian member name of the output member.

**OFT file type**

Output file type

- On CMS, the file type of the output file.
- On z/OS, the DD name of the output file.
- On z/VSE, not used.

**PACK** Compress redundant characters in source statement text

**PFM file mode**

Primary input file mode

- On CMS, the initial file mode to search for the primary input file. Standard search order used if not found. Overrides IFM
- On z/OS, not used.
- On z/VSE, not used.

**PFT file type**

Primary input file type

- On CMS, the file type (FT) of the input file.
- On z/OS, the DD name of the input file.
- On z/VSE, the DLBL name of the input file.

**QUIET**

Suppress display of progress and error messages.

**SEQ** Retain source record sequence numbers.

Default Options:

- ASM
- PFT SYSADATA
- OFT ASMLANGX
- PACK
- NOSEQ
- DCL
- INCL

---

**Examples**

Here is a sample CMSASMLANGX invocation.

- using "*fn* SYSADATA"  
ASMLANGX *fn* (ASM LOUD ERROR



---

## Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

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

IBM Corporation  
Mail Station P300  
2455 South Road  
Poughkeepsie New York 12601-5400  
U.S.A.

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

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

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

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

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

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



---

## Bibliography

### High Level Assembler Documents

*HLASM General Information*, GC26-4943  
*HLASM Installation and Customization Guide*, SC26-3494  
*HLASM Language Reference*, SC26-4940  
*HLASM Programmer's Guide*, SC26-4941

### Toolkit Feature document

*HLASM Toolkit Feature User's Guide*, GC26-8710  
*HLASM Toolkit Feature Debug Reference Summary*, GC26-8712  
*HLASM Toolkit Feature Interactive Debug Facility User's Guide*, GC26-8709  
*HLASM Toolkit Feature Installation and Customization Guide*, GC26-8711

### Related documents (Architecture)

*z/Architecture Principles of Operation*, SA22-7832

### Related documents for z/OS

#### z/OS:

*z/OS MVS JCL Reference*, SA23-1385  
*z/OS MVS JCL User's Guide*, SA23-1386  
*z/OS MVS Programming: Assembler Services Guide*, SA23-1368  
*z/OS MVS Programming: Assembler Services Reference, Volume 1 (ABE-HSP)*, SA23-1369  
*z/OS MVS Programming: Assembler Services Reference, Volume 2 (IAR-XCT)*, SA23-1370  
*z/OS MVS Programming: Authorized Assembler Services Guide*, SA23-1371  
*z/OS MVS Programming: Authorized Assembler Services Reference, Volumes 1 - 4*, SA23-1372 - SA23-1375  
*z/OS MVS Program Management: User's Guide and Reference*, SA23-1393  
*z/OS MVS System Codes*, SA38-0665  
*z/OS MVS System Commands*, SA38-0666  
*z/OS MVS System Messages, Volumes 1 - 10*, SA38-0668 - SA38-0677  
*z/OS Communications Server: SNA Programming*, SC27-3674

#### UNIX System Services:

*z/OS UNIX System Services User's Guide*, SA23-2279

#### DFSMS/MVS:

*z/OS DFSMS Program Management*, SC27-1130  
*z/OS DFSMSdfp Utilities*, SC23-6864

#### TSO/E (z/OS):

*z/OS TSO/E Command Reference*, SA32-0975

#### SMP/E (z/OS):

*SMP/E for z/OS Messages, Codes, and Diagnosis*, GA32-0883  
*SMP/E for z/OS Reference*, SA23-2276  
*SMP/E for z/OS User's Guide*, SA23-2277

## **Related documents for z/VM**

*z/VM: VMSES/E Introduction and Reference, GC24-6243*  
*z/VM: Service Guide, GC24-6247*  
*z/VM: CMS Commands and Utilities Reference, SC24-6166*  
*z/VM: CMS File Pool Planning, Administration, and Operation, SC24-6167*  
*z/VM: CP Planning and Administration, SC24-6178*  
*z/VM: Saved Segments Planning and Administration, SC24-6229*  
*z/VM: Other Components Messages and Codes, GC24-6207*  
*z/VM: CMS and REXX/VM Messages and Codes, GC24-6161*  
*z/VM: CP System Messages and Codes, GC24-6177*  
*z/VM: CMS Application Development Guide, SC24-6162*  
*z/VM: CMS Application Development Guide for Assembler, SC24-6163*  
*z/VM: CMS User's Guide, SC24-6173*  
*z/VM: XEDIT User's Guide, SC24-6245*  
*z/VM: XEDIT Commands and Macros Reference, SC24-6244*  
*z/VM: CP Commands and Utilities Reference, SC24-6175*

## **Related documents for z/VSE**

*z/VSE: Guide to System Functions, SC33-8312*  
*z/VSE: Administration, SC34-2627*  
*z/VSE: Installation, SC34-2631*  
*z/VSE: Planning, SC34-2635*  
*z/VSE: System Control Statements, SC34-2637*  
*z/VSE: Messages and Codes, Vol.1 , SC34-2632*  
*z/VSE: Messages and Codes, Vol.2, SC34-2633*  
*z/VSE: Messages and Codes, Vol.3, SC34-2634*  
*REXX/VSE Reference, SC33-6642*  
*REXX/VSE User's Guide, SC33-6641*

---

## Glossary

This glossary defines terms and abbreviations that are used in this book. If you do not find the term you are looking for refer to the index, to the glossary of the appropriate high-level language (HLL) manual, or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

### A

- abend** Abnormal end of application.
- accept** An SMP/E process that moves distributed code and programs to the distribution libraries.
- activate**  
To make a program available for use.
- addressing mode (AMODE)**  
An attribute that refers to the address length that a routine is prepared to handle upon entry. Addresses may be 24 or 31 bits long.
- address space**  
Domain of addresses that are accessible by an application.
- AMODE**  
Addressing mode.
- APAR** Authorized program analysis report.
- authorized program analysis report (APAR)**  
A request for correction of a problem caused by a defect in a current unaltered release of a program.
- authorized program facility (APF)**  
The authorized program facility (APF) is a facility that an installation manager uses to protect the system. In MVS, certain system functions, such as all or part of some SVCs, are sensitive; their use must be restricted to users who are authorized. An authorized program is one that executes in supervisor state, or with APF authorization.
- auxiliary file**  
In CMS, a file that contains a list of file types of update files to be applied to a particular source file.

### B

- base** The core product, upon which features may be separately ordered and installed.

- batch** Pertaining to activity involving little or no user action. Contrast with *interactive*.
- byte** The basic unit of storage addressability, normally with a length of 8 bits.

### C

- cataloged procedure**  
A set of control statements placed in a library and retrievable by name.
- CBPDO**  
Custom-Built Product Delivery Offering.
- CE** IBM customer engineer.
- CLIST** TSO command list.
- CMS** Conversational monitor system.
- compiler options**  
Keywords that can be specified to control certain aspects of compilation. Compiler options can control the nature of the load module generated by the compiler, the types of printed output to be produced, the efficient use of the compiler, and the destination of error messages.
- component**  
Software that is part of a functional unit.  
A set of modules that performs a major function within a system.
- condition code**  
A code that reflects the result of a previous input/output, arithmetic, or logical operation.
- control block**  
A storage area used by a computer program to hold control information.
- control file**  
In CMS, a file that contains records that identify the updates to be applied and the macrolibraries, if any, needed to assemble a particular source program.
- control program (CP)**  
A computer program designed to schedule and to supervise the execution of programs of a computer system.
- control section (CSECT)**  
The part of a program specified by the programmer to be a relocatable unit, all

elements of which are to be loaded into adjoining main storage locations.

**control statement**

In programming languages, a statement that is used to alter the continuous sequential execution of statements; a control statement can be a conditional statement, such as IF, or an imperative statement, such as STOP.

In JCL, a statement in a job that is used in identifying the job or describing its requirements to the operating system.

**conversational monitor system (CMS)**

A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under the control of the VM/370 control program.

**corrective maintenance**

Maintenance performed specifically to overcome existing problems.

**CP command**

In VM, a command by which a terminal user controls his or her virtual machine. The VM/370 control program commands are called CP commands.

**CPPL** Command processor parameter list.

**CP privilege class**

In VM, one or more classes assigned to a virtual machine user in the user's VM directory entry; each privilege class allows access to a logical subset of the CP commands.

**CSI** Consolidated software inventory data set. See *SMPCSI*.

**CSECT**

Control section.

**cumulative service tape**

A tape sent with a new function order, containing all current PTFs for that function.

**Custom-Built Installation Process Offering (CBIPO)**

A CBIPO is a tape that has been specially prepared with the products (at the appropriate release levels) requested by the customer. A CBIPO simplifies installing various products together.

**Custom-Built Product Delivery Offering (CBPDO)**

A CBPDO is a tape that has been specially prepared for installing a particular product and the related service requested by the customer. A CBPDO simplifies installing a product and the service for it.

**D**

**data definition name (DDNAME)**

The logical name of a file within an application. The DDNAME provides the means for the logical file to be connected to the physical file.

**data set**

On MVS, a named collection of related data records that is stored and retrieved by an assigned name. Equivalent to a CMS *file*.

**data set name (dsname)**

The data set name on the DD statement in the JCL or the dsname operand of the TSO ALLOC command.

**DBCS** Double-byte character set.

**DDDEF**

Dynamic data definition.

**DDNAME**

Data definition name.

**default**

A value that is used when no alternative is specified.

**DD statement**

In MVS, connects the logical name of a file and the physical name of the file.

**DELTA disk**

In VM, the virtual disk that contains program temporary fixes (PTFs) that have been installed but not merged.

**distribution libraries**

IBM-supplied partitioned data sets on tape containing one or more components that the user restores to disk for subsequent inclusion in a new system.

**distribution medium**

The medium on which software is distributed to the user; for example, 9-track magnetic tape, tape cartridge.

**distribution zone**

In SMP/E, a group of VSAM records that

describe the SYSMODs and elements in the distribution libraries.

**DITTO utility**

Data Interfile Transfer, Testing, and Operations utility.

**double-byte character set (DBCS)**

A collection of characters represented by a 2-byte code.

**driving system**

The system used to install the program. Contrast with target system.

**dsname**

Data set name.

**dynamic data definition (DDDEF)**

The process of defining a data set and allocating auxiliary storage space for it while, rather than before, a job step executes.

**dynamic storage**

Storage acquired as needed at run time. Contrast with *static storage*.

**E**

**ECMODE**

Extended control mode.

**executable program**

A program that has been link-edited and therefore can run in a processor.

The set of machine language instructions that constitute the output of the compilation of a source program.

**Extended control mode (ECMODE)**

A mode in which all features of a System/370 computing system, including dynamic address translation, are operational.

**Extended Service Option (ESO)**

A service option that gives a customer all the new fixes for problems in IBM licensed programs that operate under that customer's operating system.

**F**

**feature**

A part of an IBM product that may be ordered separately by a customer.

**feature number**

A four-digit code used by IBM to process hardware and software orders.

**file** A named collection of related data records

that is stored and retrieved by an assigned name. Equivalent to an MVS *data set*.

**FILEDEF**

File definition statement.

**file definition statement (FILEDEF)**

In CMS, connects the logical name of a file and the physical name of a file.

**fix**

A correction of an error in a program, generally a temporary correction or bypass of defective code.

**FMID** Function modification identifier.

**function**

A routine that is invoked by coding its name in an expression. The routine passes a result back to the invoker through the routine name.

**function modification identifier (FMID)**

The value used to distinguish separate parts of a product. A product tape or cartridge has at least one FMID.

**I**

**IBM customer engineer (CE)**

An IBM service representative who performs maintenance services for IBM hardware.

**IBM program support representative (PSR)**

An IBM service representative who performs maintenance services for IBM software at a centralized IBM location.

**IBM service representative**

An individual in IBM who performs maintenance services for IBM products or systems.

**IBM Software Distribution (ISD)**

The IBM department responsible for software distribution.

**IBM Support Center**

The IBM department responsible for software service.

**IBM systems engineer (SE)**

An IBM service representative who performs maintenance services for IBM software in the field.

**initial program load (IPL)**

The initialization procedure that causes an operating system to commence operation.

The process by which a configuration image is loaded into storage, as at the beginning of a work day or after a system malfunction or as a means to access updated parts of the system.

The process of loading system programs and preparing a system to run jobs.

**inline** Sequential execution of instructions, without branching to routines, subroutines, or other programs.

**IPL** Initial program load.

**interactive**

Pertaining to a program or system that alternately accepts input and responds. In an interactive system, a constant dialog exists between user and system. Contrast with *batch*.

**Interactive Interface**

A series of panels, allowing the user to use the facilities of the VSE/ESA operating. This interface runs within CICS®/VSE.

**ISD** IBM Software Distribution.

**J**

**JCL** Job control language.

**JCLIN data**

The JCL statements associated with the ++JCLIN statement or saved in the SMPJCLIN data set. They are used by SMP/E to update the target zone when the SYSMOD is applied. Optionally, SMP/E can use the JCLIN data to update the distribution zone when the SYSMOD is accepted.

**JES** Job Entry Subsystem

**Job Entry Subsystem**

A system facility for spooling, job queueing, and managing the scheduler work area.

**job control language (JCL)**

A sequence of commands used to identify a job to an operating system and to describe a job's requirements.

**job step**

You enter a program into the operating system as a job step. A job step consists of the job control statements that request and control execution of a program and request the resources needed to run the

program. A job step is identified by an EXEC statement. The job step can also contain data needed by the program. The operating system distinguishes job control statements from data by the contents of the record.

**L**

**librarian**

In VSE, the set of programs that maintains, services, and organizes the system and private libraries.

**library**

A collection of functions, subroutines, or other data.

**link pack area (LPA)**

In MVS, an area of main storage containing reenterable routines from system libraries. Their presence in main storage saves loading time when a reenterable routine is needed.

**linkage editor**

A program that resolves cross-references between separately assembled object modules and then assigns final addresses to create a single relocatable load module. The linkage editor then stores the load module in a program library in main storage.

**link-edit**

To create a loadable computer program by means of a linkage editor.

**load module**

An application or routine in a form suitable for execution. The application or routine has been compiled and link-edited; that is, address constants have been resolved.

**logical saved segment**

A portion of a physical saved segment that CMS can manipulate. Each logical saved segment can contain different types of program objects, such as modules, text files, execs, callable services libraries, language repositories, user-defined objects, or a single minidisk directory. A system segment identification file (SYSTEM SEGID) associates a logical saved segment to the physical saved segment in which it resides. See *physical saved segment* and *saved segment*.

**LPA** Link pack area.



## M

### **maintain system history program (MSHP)**

In VSE, a program used for automating and controlling various installation, tailoring, and service activities for a VSE system.

**MCS** Modification control statement

### **minidisk**

In VM, all, or a logical subdivision of, a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of stored data so that the data can be accessed. Synonymous with virtual disk.

### **module**

A language construct that consists of procedures or data declarations and can interact with other such constructs.

### **MSHP**

Maintain system history program.

**MVS** Multiple Virtual Storage operating system.

### **multicultural support**

Translation requirements affecting parts of licensed programs; for example, translation of message text and conversion of symbols specific to countries.

## N

### **Named Saved System**

A copy of an operating system that a user has named and saved in a file. The user can load the operating system by its name, which is more efficient than loading it by device number.

### **nonexecutable components**

Components of a product that cannot be run.

### **non reentrant**

A program that cannot be shared by multiple users.

### **nonreenterable**

See *non reentrant*.

**NSS** named saved system

## O

### **object code**

Output from a compiler or assembler which is itself executable machine code or

is suitable for processing to produce executable machine code.

### **object deck**

Synonymous with *object module*, *text deck*.

### **object module**

A portion of an object program suitable as input to a linkage editor. Synonymous with *text deck*, *object deck*.

**online** Pertaining to a user's ability to interact with a computer.

Pertaining to a user's access to a computer via a terminal.

### **operating system**

Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

## P

### **parameter**

Data items that are received by a routine.

### **partition**

A fixed-size division of storage.

**phase** In VSE, the smallest complete unit of executable code that can be loaded into virtual storage.

### **physical saved segment**

One or more pages of storage that have been named and retained on a CP-owned volume (DASD). When created, it can be loaded within a virtual machine's address space or outside a virtual machine's address space. Multiple users can load the same copy. A physical saved segment can contain one or more logical saved segments. A system segment identification file (SYSTEM SEGID) associates a physical saved segment to its logical saved segments. See *logical saved segment* and *saved segment*.

### **preventive maintenance**

Maintenance performed specifically to prevent problems from occurring.

### **preventive service planning (PSP)**

The online repository of program temporary fixes (PTFs) and other service information. This information could affect installation.

**procedure**

A named block of code that can be invoked, normally using a call.

**procedure library (PROCLIB)**

A program library in direct-access storage with job definitions. The reader/interpreter can be directed to read and interpret a particular job definition by an execute statement in the input stream.

**PROCLIB**

Procedure library.

**program level**

The modification, release, version, and fix level of a product.

**program number**

The seven-digit code (in the format *xxxx-xxx*) used by IBM to identify each program product.

**program temporary fix (PTF)**

A temporary solution or bypass of a problem diagnosed by IBM as resulting from a defect in a current unaltered release of the program.

**PSP** Preventive service planning.

**PSR** IBM program support representative.

**PTF** Program temporary fix.

**Q****qualifier**

A modifier that makes a name unique.

**R****reentrant**

The attribute of a routine or application that allows more than one user to share a single copy of a load module.

**reenterable**

See *reentrant*

**relative file tape (RELFILE tape)**

A standard label tape made up of two or more files. It contains a file of the MCSs for one or more function SYSMODs and one or more relative files containing unloaded source data sets and unloaded, link-edited object data sets at the distribution library level. A relative file tape is one way of packaging SYSMODs, and is typically used for function SYSMODs.

**relative files (RELFILES)**

Files containing modification text and JCL input data associated with a SYSMOD.

**RELFILES**

Relative files

**RELFILE tape**

Relative file tape

**relocatable load module**

On CMS, a combination of object modules having cross references resolved and prepared for loading into storage for execution.

**residence mode (RMODE)**

The attribute of a load module that specifies whether the module, when loaded, must reside below the 16MB virtual storage line or may reside anywhere in virtual storage.

**resident modules**

A module that remains in a particular area of storage.

**return code**

A code produced by a routine to indicate its success. It can be used to influence the execution of succeeding instructions.

**RIM** Related installation materials

**RMODE**

Residence mode.

**run** To cause a program, utility, or other machine function to be performed.

**S****save area**

Area of main storage in which contents of registers are saved.

**SBCS** Single-byte character set.

**SE** IBM systems engineer.

**service level**

The modification level, release, version, and fix level of a program. The service level incorporates PTFs if there are any.

**saved segment**

A segment of storage that has been saved and assigned a name. Saved segments can be physical saved segments that CP recognizes or logical saved segments that CMS recognizes. The segments can be loaded and shared among virtual machines, which helps use real storage



- more efficiently, or a private, nonshared copy can be loaded into a virtual machine. See *logical saved segment* and *physical saved segment*.
- shared segment**  
In VM, a feature of a saved system that allows one or more segments of reenterable code in real storage to be shared among many virtual machines.
- shared storage**  
An area of storage that is the same for each virtual address space. Because it is the same space for all users, information stored there can be shared and does not have to be loaded in the user region.
- shared virtual area (SVA)**  
In VSE, a high address area of virtual storage that contains a system directory list (SDL) of frequently used phases, resident programs that can be shared between partitions, and an area for system support.
- severity code**  
A part of run-time messages that indicates the severity of the error condition (1, 2, 3, or 4).
- single-byte character set (SBCS)**  
A collection of characters represented by a 1-byte code.
- SMPCSI**  
The SMP/E data set that contains information about the structure of a user's system as well as information needed to install the operating system on a user's system. The SMPCSI DD statement refers specifically to the CSI that contains the global zone. This is also called the master CSI.
- softcopy**  
One or more files that can be electronically distributed, manipulated, and printed by a user.
- software inventory disk**  
In VM, the disk where the system level inventory files reside.
- source code**  
The input to a compiler or assembler, written in a source language.
- source program**  
A set of instructions written in a programming language that must be translated to machine language before the program can be run.
- SREL** System release identifier
- statement**  
In programming languages, a language construct that represents a step in a sequence of actions or a set of declarations.
- sublibrary**  
In VSE, a subdivision of a library.
- SUBSET**  
The value that specifies the function modifier (FMID) for a product level. It further specifies an entry in RETAIN\* for a product level.
- subsystem**  
A secondary or subordinate system, or programming support, normally capable of operating independently of or asynchronously with a controlling system. Examples are CICS and IMS.
- SVA** Shared virtual area.
- syntax** The rules governing the structure of a programming language and the construction of a statement in a programming language.
- SYSMOD**  
system modification.
- SYSMOD ID**  
system modification identifier.
- system abend**  
An abend caused by the operating system's inability to process a routine; can be caused by errors in the logic of the source routine.
- T**
- target disk**  
In VM, the disk to which a program is installed.
- target libraries**  
In SMP/E, a collection of data sets in which the various parts of an operating system are stored. These data sets are sometimes called system libraries.
- target zone**  
In SMP/E, a collection of VSAM records describing the target system macros, modules, assemblies, load modules,

source modules, and libraries copied from DLIBs during system generation, and the system modifications (SYSMODs) applied to the target system.

**text deck**

Synonym for *object module, object deck*.

**time sharing option/extended (TSO/E)**

An option on the operating system; for System/370, the option provides interactive time sharing from remote terminals.

**TSO/E** Time sharing option/extended.

**U**

**UCLIN**

In SMP/E, the command used to initiate changes to SMP/E data sets. Actual changes are made by subsequent UCL statements.

**UPGRADE**

An alphanumeric identifier that specifies a product level.

**user exit**

A routine that takes control at a specific point in an application.

**USERMOD**

User modification.

**user modification (USERMOD)**

A change to product code that the customer initiates.

**V**

**virtual machine (VM)**

A functional simulation of a computer and its associated devices. Each virtual machine is controlled by a suitable operating system.

In VM, a functional equivalent of either a System/370 computing system or a System/370-Extended Architecture computing system.

**VMFINS**

An installation aid supplied as part of VMSES/E to make installation on VM consistent.

**VM Serviceability Enhancements**

**Staged/Extended (VMSES/E)**

A program product for installing and maintaining products on VM.

**VMSES/E**

VM Serviceability Enhancements Staged/Extended.

**VOLSER**

Volume serial number.

**volume**

A certain portion of data, together with its data carrier, that can be handled conveniently as a unit.

A data carrier mounted and demounted as a unit; for example, a reel of magnetic tape, a disk pack.

**volume label**

An area on a standard label tape used to identify the tape volume and its owner. This area is the first 80 bytes and contains VOL 1 in the first four positions.

**volume serial number (VOLSER)**

A number in a volume label assigned when a volume is prepared for use in a system.

**VSAM**

Virtual storage access method. A high-performance mass storage access method. Three types of data organization are available: entry sequenced data sets (ESDS), key sequenced data sets (KSDS), and relative record data sets (RRDS).





GC26-8712-06

