# ASG-SmartTest™ Reference Guide

Version 7.0
Publication Number: STX0400-70
Publication Date: February 2003

The information contained herein is the confidential and proprietary information of Allen Systems Group, Inc. Unauthorized use of this information and disclosure to third parties is expressly prohibited. This technical publication may not be reproduced in whole or in part, by any means, without the express written consent of Allen Systems Group, Inc.

©2003 Allen Systems Group, Inc. All rights reserved.
All names and products contained herein are the trademarks or registered trademarks of their respective holders.

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 800.932.5536 (USA only)

# Contents

**Preface** . . . . . **xi**

**About this Publication** . . . . . **xi**

**Related Publications** . . . . . **xii**

**ASG-Existing Systems Workbench (ASG-ESW)** . . . . . **xiii**

**Invoking ESW Products** . . . . . **xvi**

**ESW Product Integration** . . . . . **xvii**
Example 1 . . . . . xviii
Example 2 . . . . . xix

**Publication Conventions** . . . . . **xxi**

**ASG Customer Support** . . . . . **xxi**
Intelligent Support Portal (ISP) . . . . . xxii
Telephone Support . . . . . xxii

**ASG Documentation/Product Enhancements** . . . . . **xxiv**

**1 File** . . . . . **1**

**File Pull-down** . . . . . **2**

**File - Setup Test Environment Pop-up** . . . . . **3**

**File - TCA Test Plan Selection Pop-up** . . . . . **5**

**File - Open Program (Qualify to Program) Pop-up** . . . . . **6**

**Environment - AKR Directory Pop-up** . . . . . **7**

**File - Analyze Submit Pop-up** . . . . . **10**

**File - AKR Utility Pop-up** . . . . . **13**

**File - AKR Directory Pop-up** . . . . . **15**

**File - AKR Allocate/Expand Pop-up** . . . . . **17**

**Options - COPY/Include Libraries Screen** . . . . . **20**

**File - Execute Script Pop-up** . . . . . **22**

2 View . . . . . 25

View Pull-down . . . . . 26

View - Execution Counts Pop-up . . . . . 28

View - Execution Tracking Pop-up . . . . . 29

View - Memory Pop-up . . . . . 30

View - Qualify Request Pop-up . . . . . 31

View - Zoom Data Request Pop-up . . . . . 32

View - Keep Request Pop-up . . . . . 34

View - Display Request Pop-up . . . . . 35

View - Paragraph Cross-reference Request Pop-up . . . . . 36

View - Paragraph Cross Reference Pop-up . . . . . 38

View - Reset Request Pop-up . . . . . 40

View - Exclude Request Pop-up . . . . . 42

View - Using Request Pop-up . . . . . 43

View - Drop Request Pop-up . . . . . 44

3 Test . . . . . 45

Test Pull-down . . . . . 46

Test - Run Request Pop-up . . . . . 48

Test - Run Request (Backtrack Recording Mode Active) Pop-up . . . . . 49

Test - Run Request (Backtrack Review Mode Active) Pop-up . . . . . 51

Test - Run Request (CICS Environment-Test Active) Pop-up . . . . . 52

Test - Run Request (CICS - Backtrack Recording Mode Active) Pop-up . . . . . 53

Test - Step Request Pop-up . . . . . 54

Test - Step Request (Backtrack Recording Mode) Pop-up . . . . . 56

Test - Step Request (Backtrack Review Mode) Pop-up . . . . . 58

Test - Step Request (CICS Environment-Test Active) Pop-up . . . . . 59

Test - Step Request (CICS - Backtrack Recording Mode Active) Pop-up . . . . . 61

Test - Break Request Pop-up . . . . . 63

Break - Data Name Pop-up . . . . . 64

Break - Line Number Pop-up . . . . . 66

Break - Label/Paragraph Name Pop-up . . . . . 67

**Break - on Subset Pop-up** . . . . . 68
**Break - Program Name Pop-up** . . . . . 69
**Break - Pattern Pop-up** . . . . . 71
**Test - Stop Request Pop-up** . . . . . 73
**Test - Go Request Pop-up** . . . . . 74
**Test - Move Request Pop-up** . . . . . 75
**Test - Add Request Pop-up** . . . . . 76
**Test - Subtract Request Pop-up** . . . . . 77
**Test - Where Request Pop-up** . . . . . 78
**Test - SmartTest Testpoint Generation Pop-up** . . . . . 79
**SmartTest/Alliance Interface Pop-up** . . . . . 80
**Test - Newcopy Request Pop-up** . . . . . 81

**4 Search** . . . . . 83

**Introduction** . . . . . 84
Find . . . . . 84
Highlight . . . . . 84
Scroll . . . . . 84
Print . . . . . 84
Punch . . . . . 84
Exclude . . . . . 84
**Search Pull-down** . . . . . 85
**Search - Data Name Pop-up** . . . . . 86
**Search - Label Name Pop-up** . . . . . 89
**Search - Paragraph Name Pop-up** . . . . . 91
**Search - Pattern String Pop-up** . . . . . 93
**Search - Subset Name Pop-up** . . . . . 96
**Search - Perform Range Name Pop-up** . . . . . 99
**Search - Program Name Pop-up** . . . . . 101
**Search - User Mark Name Pop-up** . . . . . 103
**Search - Line Range Pop-up** . . . . . 105
**Search - Any/Unknown Type Pop-up** . . . . . 107
**Search - Branch Request Pop-up** . . . . . 109
**Branch Previous Options Pop-up** . . . . . 111

Selection List Pop-ups . . . . . 111
IN-Clause Option Pop-up . . . . . 112

5 List . . . . . 115

List Pull-down . . . . . 117
Test Facilities List Screen . . . . . 120
Access Registers Screen . . . . . 121
Address Stop Entry Screen . . . . . 122
Environment AKR Directory Pop-up . . . . . 124
List - BackTrack Variable History Pop-up . . . . . 127
List - BackTrack Variable History Screen . . . . . 128
Breakpoints List Screen . . . . . 129
List - CALL Statements Pop-up . . . . . 132
Compiler Options Screen . . . . . 134
Execution Counts Screen . . . . . 135
List - Equates Pop-up . . . . . 138
Floating Point Registers Screen . . . . . 139
Load Module Intercept List Pop-up . . . . . 140
List - User Marks Pop-up . . . . . 141
Memory Display Screen . . . . . 143
Load Module Directory Screen . . . . . 152
Module Map Screen . . . . . 153
List - Perform Range Names Pop-up . . . . . 155
Profile Data Set Member List Screen . . . . . 156
List - Program/Subprogram Names Screen . . . . . 159
Pseudo Code List Screen . . . . . 160
General Registers Screen . . . . . 162
List - COBOL Subsets Names Screen . . . . . 164
Test Session Tailoring Screen . . . . . 165
Execution Tracking Screen . . . . . 168
When Conditions List Screen . . . . . 170
List - CICS Features Pop-up . . . . . 172

List - COBOL Features Pop-up . . . . . 173
List - Execution Counts Pop-up . . . . . 174
List - Execution Tracking Pop-up . . . . . 175
List - Registers Pop-up . . . . . 177

6 Help . . . . . 179

Help Pull-down . . . . . 180
Help - Specific ASG Command Pop-up . . . . . 181
Help - Specific ASG Message Pop-up . . . . . 182
Help Table of Contents . . . . . 182
Help Index . . . . . 183
Help - About Pop-up . . . . . 184

7 Options . . . . . 185

Options Pull-down . . . . . 186
Options - Product Parameters Pop-up . . . . . 187
Options - Product Allocations Pop-up . . . . . 188
Options - Log/List/Punch Definition Pop-up . . . . . 190
Options - Script File Allocations Pop-up . . . . . 192
Options - PF Key Definition Pop-up . . . . . 193
Options - Modes Screen . . . . . 195
Options - Scratchpad Pop-up . . . . . 202
Options - Scratchpad Equate Pop-up . . . . . 203
Options - Scratchpad Mark Pop-up . . . . . 204
Options - Scratchpad Copy Pop-up . . . . . 205
Options - Scratchpad Delete Pop-up . . . . . 207
Options - Scratchpad Merge Pop-up . . . . . 207
Options - Scratchpad Rename Pop-up . . . . . 209

8 Commands . . . . . 211

Command Processing . . . . . 216
Command Diagrams . . . . . 216
Command Syntax . . . . . 218

**Repeating Commands** . . . . . 219
**Cursor Position** . . . . . 222
**Cursor Substitution Character** . . . . . 222
**Double Byte Character Set (DBCS) Strings** . . . . . 223
**& (Retain) Command** . . . . . 224
**ADD Command** . . . . . 225
**ALLIANCE Command** . . . . . 226
**ALLOCDEF Command** . . . . . 227
**ANALYZE Command** . . . . . 228
**BRANCH Command** . . . . . 229
**BREAK Command** . . . . . 232
**CANCEL Command** . . . . . 241
**CONVERT Command** . . . . . 242
**COPY Command** . . . . . 243
**CURRENT Command** . . . . . 246
**DELETE Command** . . . . . 247
**DISPLAY Command** . . . . . 250
**DROP Command** . . . . . 253
**DUMP Command** . . . . . 254
**END Command** . . . . . 258
**ENVIRONMENT Command** . . . . . 259
**EQUATE Command** . . . . . 260
**EXCLUDE Command** . . . . . 262
**EXECUTE Command** . . . . . 270
**FIND Command** . . . . . 274
**FINDXTND Command** . . . . . 279
**FLOW Command (without Insight)** . . . . . 292
**FLOW Command (Insight Only)** . . . . . 294
**FORCE Command** . . . . . 299
**GO Command** . . . . . 300
**HELP Command** . . . . . 301
**HIGH Command** . . . . . 303

**KEEP Command** . . . . . 310
**KEYS Command** . . . . . 314
**LIST Command** . . . . . 316
**LOCATE Command** . . . . . 330
**LPRINT Command** . . . . . 333
**LPUNCH Command** . . . . . 342
**MARK Command** . . . . . 350
**MERGE Command** . . . . . 353
**MOVE Command** . . . . . 356
**NEWCOPY Command** . . . . . 357
**PARMDEF Command** . . . . . 358
**PREF Command** . . . . . 359
**PRINTLOG Command** . . . . . 362
**PRINTLST Command** . . . . . 363
**PROCESS Command** . . . . . 364
**PRODLVL Command** . . . . . 368
**QUALIFY Command** . . . . . 369
**RECALL Command** . . . . . 371
**REDO Command** . . . . . 374
**REFRESH Command** . . . . . 376
**RENAME Command** . . . . . 377
**REPEAT Command** . . . . . 379
**RESET Command** . . . . . 380
**RETURN Command** . . . . . 382
**RFIND Command** . . . . . 383
**RHIGH Command** . . . . . 384
**RPREF Command** . . . . . 385
**RSCROLL Command** . . . . . 386
**RTRACE Command** . . . . . 387
Trace Decision Options Pop-up . . . . . 388
**RUN Command** . . . . . 390
**RUN Command (CICS Only)** . . . . . 393

**RUN Command (BACKTRACK ON)** . . . . . **396**
**SAVE Command** . . . . . **399**
**SCROLL Command** . . . . . **401**
**SELECT Command** . . . . . **408**
**SET Command** . . . . . **409**
LEARN Mode - Generated Command Pop-up . . . . . 418
**SETUP Command** . . . . . **419**
**SHOW Command (CICS Only)** . . . . . **420**
**STEP Command (BACKTRACK OFF)** . . . . . **421**
**STEP Command (BACKTRACK ON)** . . . . . **425**
**STOP Command** . . . . . **429**
**SUBTRACT Command** . . . . . **431**
**TCA DEFINE Command** . . . . . **432**
**TCA LIST Command** . . . . . **433**
**TCA RECORD Command** . . . . . **434**
**TCA REPORT Command** . . . . . **435**
**TCA RUN Command** . . . . . **436**
**TEST Command** . . . . . **437**
**TESTPOINT Command** . . . . . **438**
**TOGGLE Command** . . . . . **439**
**TRACE Command** . . . . . **440**
**UPDATE Command** . . . . . **446**
**USING Command** . . . . . **448**
**UTILITY Command** . . . . . **450**
**VIEW Command** . . . . . **451**
**WHEN Command** . . . . . **452**
**WHERE Command** . . . . . **456**
**WIZARD Command** . . . . . **458**
**ZOOMDATA Command** . . . . . **459**
**ZOOMIN/ZOOMOUT Commands** . . . . . **462**
**SmartTest Line Commands** . . . . . **464**
. (Label) Line Command . . . . . 467
A (After) Line Command . . . . . 468

B (Before) Line Command . . . . . 469
BR (Break) Line Command . . . . . 470
C (Copy) Line Command . . . . . 471
CC (Copy Block) Line Command. . . . . 472
D (Delete) Line Command . . . . . 473
DD (Delete Block) Line Command . . . . . 474
F (First) Line Command . . . . . 475
GO Line Command . . . . . 476
H (Highlight) Line Command . . . . . 478
HH (Highlight Block) Line Command . . . . . 479
I (Insert) Line Command. . . . . 480
K (Keep) Line Command . . . . . 481
KG (Keep Group) Line Command . . . . . 483
KGH (Keep Group Hexadecimal) Line Command. . . . . 485
KH (Keep Hexadecimal) Line Command. . . . . 487
L (Last) Line Command . . . . . 489
M (Move) Line Command . . . . . 490
MM (Move Block) Line Command . . . . . 491
R (Repeat) Line Command . . . . . 492
RR (Repeat Block) Line Command . . . . . 493
S (Show) Line Command . . . . . 494
SS (Show Block) Line Command. . . . . 495
X (Exclude) Line Command. . . . . 496
XX (Exclude Block) Line Command . . . . . 497
ZA (Zoom Assembler) Line Command . . . . . 498
ZD (Zoom Data) Line Command . . . . . 500
ZG (Zoom Group) Line Command. . . . . 502
ZGH (Zoom Group Hexadecimal) Line Command . . . . . 504
ZH (Zoom Hexadecimal) Line Command . . . . . 506
ZI (Zoom In) Line Command . . . . . 508
ZO (Zoom Out) Line Command. . . . . 509

**9 Pseudo Code . . . . . 511**

**Using Pseudo Code . . . . . 512**

**Entering Pseudo Code into Source Code . . . . . 513**

**Execution of Pseudo Code . . . . . 515**

**Saving Pseudo Code . . . . . 516**

**Pseudo Code Commands and Statements. . . . . 516**
77 Statement . . . . . 517
ADD Statement. . . . . 519
BREAK Statement . . . . . 520
GO Statement . . . . . 521
IF Statement . . . . . 522

MOVE Statement . . . . . 525
pslabel. Statement . . . . . 526
SUBTRACT Statement . . . . . 527
WHEN Statement . . . . . 528

**COBOL Reserved Words . . . . . 531**

**Pseudo Code Examples . . . . . 532**

**Glossary . . . . . 533**

**Index . . . . . 543**

# Preface

This *ASG-SmartTest Reference Guide* provides information about the ASG-SmartTest (herein called SmartTest) commands. SmartTest is the Testing/Debugging component of the ASG Existing Systems Workbench (ESW). It automates the time-consuming and error-prone process of testing and debugging application programs. This publication is a guide for installing and maintaining SmartTest.

Allen Systems Group, Inc. (ASG) provides professional support to resolve any questions or concerns regarding the installation or use of any ASG product. Telephone technical support is available around the world, 24 hours a day, 7 days a week.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on any ASG product.

## About this Publication

This publication consists of these chapters:

- Chapter 1, "File," describes the SmartTest functions available from the File pull-down.
- Chapter 2, "View," explains the SmartTest functions available from the View pull-down.
- Chapter 3, "Test," describes the SmartTest functions available from the Test pull-down.
- Chapter 4, "Search," describes the SmartTest functions available from the Search pull-down.
- Chapter 5, "List," describes the SmartTest functions available from the List pull-down.
- Chapter 6, "Help," describes the SmartTest functions available from the Help pull-down.
- Chapter 7, "Options," describes the SmartTest functions available from the Options pull-down.

- Chapter 8, "Commands," includes detailed information about the SmartTest commands.
- Chapter 9, "Pseudo Code," explains how to use COBOL compatible statements and commands called pseudo code.

# Related Publications

The documentation library for ASG-SmartTest consists of these publications (where *nn* represents the product version number):

- *ASG-Center Installation Guide* (CNX0300-*nn*) contains installation and maintenance information for ASG-Center, the common set of libraries shared by the ASG-ESW suite of products.
- *ASG-ESW Enhancement Summary* (ESW1000-*nn*) highlights the new functionality for this release.
- *ASG-SmartTest CICS User's Guide* (STC0200-*nn*) contains specific commands and test session setup information for the CICS environments.
- *ASG-SmartTest for COBOL and Assembler User's Guide* (STA0200-*nn*) contains introductory and usage information for COBOL and Assembler. It also contains test session setup information for the TSO, ISPF, IMS/DB, DB/2, BTS, and batch environments.
- *ASG-SmartTest IMS User's Guide* (STM0200-*nn*) contains specific commands and test session setup information for the IMS/DC environments.
- *ASG-SmartTest Installation Guide* (STX0300-*nn*) contains information for installing and maintaining ASG-SmartTest.
- *ASG-SmartTest PLI User's Guide* (STL0200-*nn*) contains introductory and usage information about how to use ASG-SmartTest with the PL/I language. It also contains test session setup information for the TSO, ISPF, IMS/DB, DB/2, BTS, and batch environments.
- *ASG-SmartTest Quick Start for COBOL/ASM* (STA0900-*nn*) summarizes how to use ASG-SmartTest with the COBOL or Assembler language.
- *ASG-SmartTest Quick Start for PL/I* (STL0900-*nn*) summarizes how to use ASG-SmartTest with the PL/I language.
- *ASG-SmartTest Reference Guide* (STX0400-*nn*) contains detailed reference information about CUA pull-downs and pop-ups, ASG-SmartTest command syntax, and pseudo code.
- *ASG-SmartTest Reference Summary* (STX0600-*nn*) summarizes the syntax and usage of ASG-SmartTest commands.

- *ASG-SmartTest TCA User's Guide* (STT0200-*nn*) contains procedures for using the ASG-SmartTest-TCA (Test Coverage Analysis) option.

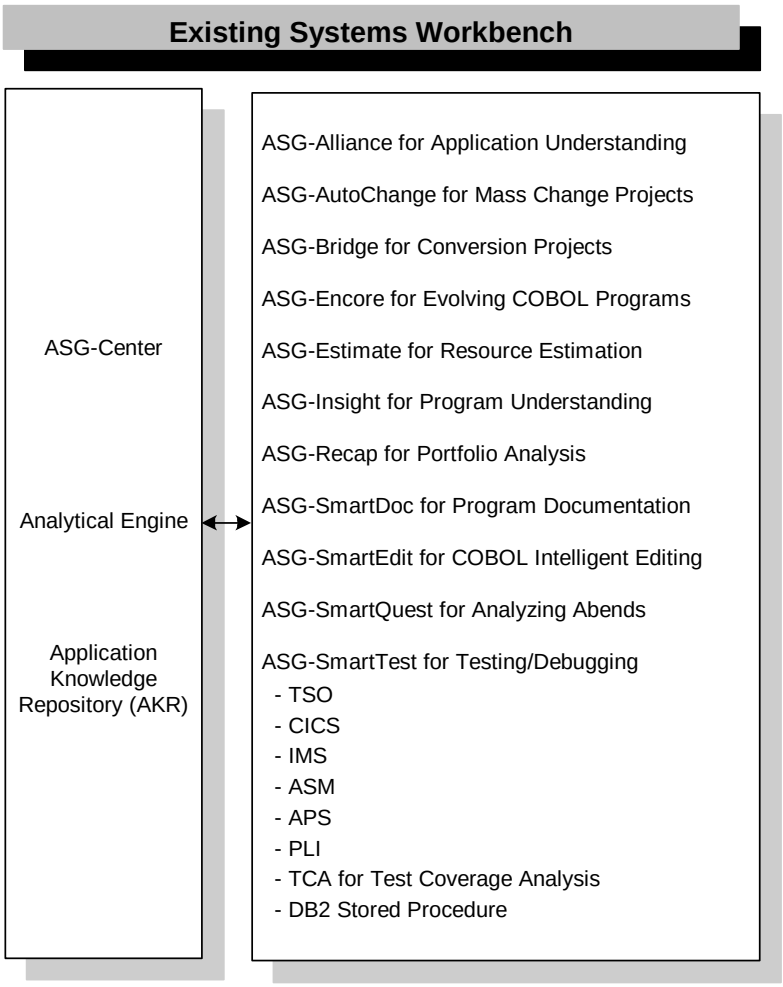
**Note:**

To obtain a specific version of a publication, contact ASG Customer Support.

# ASG-Existing Systems Workbench (ASG-ESW)

ASG-ESW (herein called ESW) is an integrated suite of components designed to assist organizations in enhancing, redeveloping, or re-engineering their existing systems. ESW products use the Application Knowledge Repository (AKR) to store source program analysis information generated by the Analytical Engine. Figure 1 represents the components of ESW.

**Figure 1 • ASG Existing Systems Workbench**



This table contains the name and description of each ESW component:

| ESW Product | Herein Called | Description |
|---|---|---|
| ASG-Alliance | Alliance | The application understanding component that is used by IT professionals to conduct an analysis of every application in their environment. Alliance supports the analysis and assessment of the impact of change requests upon an entire application. Alliance allows the programmer/ analyst to accurately perform application analysis tasks in a fraction of the time it would take to perform these tasks without an automated analysis tool. The impact analysis from Alliance provides application management with additional information for use in determining the resources required for application changes. |
| ASG-AutoChange | AutoChange | The COBOL code change tool that makes conversion teams more productive by enabling quick and safe changes to be made to large quantities of code. AutoChange is an interactive tool that guides the user through the process of making source code changes. |
| ASG-Bridge | Bridge | The bridging product that enables field expansion for program source code, without being required to simultaneously expand the fields in files or databases. Because programs are converted in smaller groups, or on a one-by-one basis, and do not require file conversion, testing during the conversion process is simpler and more thorough. |
| ASG-Center | Center | The common platform for all ESW products. Center provides the common Analytical Engine to analyze the source program and store this information in the AKR. This common platform provides a homogeneous environment for all ESW products to work synergistically. |

| ESW Product | Herein Called | Description |
|---|---|---|
| ASG-Encore | Encore | The program re-engineering component for COBOL programs. Encore includes analysis facilities and allows you to extract code based on the most frequently used re-engineering criteria. The code generation facilities allow you to use the results of the extract to generate a standalone program, a callable module, a complement module, and a CICS server. Prior to code generation, you can view and modify the extracted Logic Segment using the COBOL editor. |
| ASG-Estimate | Estimate | The resource estimation tool that enables the user to define the scope, determine the impact, and estimate the cost of code conversion for COBOL, Assembler, and PL/I programs. Estimate locates selected data items across an application and determines how they are used (moves, arithmetic operations, and compares). Time and cost factors are applied to these counts, generating cost and personnel resource estimates. |
| ASG-Insight | Insight | The program understanding component for COBOL programs. Insight allows programmers to expose program structure, identify data flow, find program anomalies, and trace logic paths. It also has automated procedures to assist in debugging program abends, changing a computation, and resolving incorrect program output values. |
| ASG-Recap | Recap | The portfolio analysis component that evaluates COBOL applications. Recap reports provide function point analysis and metrics information, program quality assessments, intra-application and inter-application comparisons and summaries, and historical reporting of function point and metrics information. The portfolio analysis information can also be viewed interactively or exported to a database, spreadsheet, or graphics package. |
| ASG-SmartDoc | SmartDoc | The program documentation component for COBOL programs. SmartDoc reports contain control and data flow information, an annotated source listing, structure charts, program summary reports, exception reports for program anomalies, and software metrics. |

| ESW Product | Herein Called | Description |
|---|---|---|
| ASG-SmartEdit | SmartEdit | The COBOL editing component that can be activated automatically when the ISPF/PDF Editor is invoked. SmartEdit provides comprehensive searching, inline copybook display, and syntax checking. SmartEdit allows you to include an additional preprocessor (for example, the APS generator) during syntax checking. SmartEdit supports all versions of IBM COBOL, CICS, SQL, and CA-IDMS. |
| ASG-SmartQuest | SmartQuest | The diagnostic tool for analyzing batch and CICS transaction abends. SmartQuest has been designed to make the maximum use of simple point-and-shoot techniques to enable fast and easy navigation through any data dump. |
| ASG-SmartTest | SmartTest | The testing/debugging component for COBOL, PL/I, Assembler, and APS programs in the TSO, MVS Batch, CICS (including file services), and IMS environments. SmartTest features include program analysis commands, execution control, intelligent breakpoints, test coverage, pseudo code with COBOL source update, batch connect, disassembled object code support, and full screen memory display. |

# Invoking ESW Products

The method you use to invoke an ESW product depends on your system setup. If you need assistance to activate a product, see your systems administrator. If your site starts a product directly, use the ISPF selection or CLIST as indicated by your systems administrator. If your site uses the ESW screen to start a product, initiate the ESW screen using the ISPF selection or CLIST as indicated by your systems administrator and then typing in the product command on the command line.

The product names can also vary depending on whether you access a product directly or through ESW. See "ESW Product Integration" on page xvii for more information about using ESW.

To initialize ESW products from the main ESW screen, select the appropriate option on the action bar pull-downs or type the product shortcut on the command line.

| Product Name (ESW Name) | Shortcut | ESW Pull-down Options |
|---|---|---|
| Alliance (Application Understanding) | AL | Understand ▸ Application |
| AutoChange (Conversion Set) | CC | Change ▸ Conversion Set |
| Bridge | BR | Change ▸ ASG-Bridge |
| Encore (Program Re-engineering) | EN | Re-engineer ▸ Program |
| Estimate | ES | Measure ▸ ASG-Estimate |
| Insight (Program Understanding) | IN | Understand ▸ Program |
| Recap (Portfolio Analysis) | RC | Measure ▸ Portfolio |
| SmartDoc (Program Documentation) | DC | Document ▸ Program |
| SmartEdit | SE | Change ▸ Program<br>**Or**<br>Change ▸ Program with Options |
| SmartQuest | SQV | Understand ▸ Abend/Dump |
| SmartTest (Testing/Debugging) | ST | Test ▸ Module/Transaction |

# ESW Product Integration

Because ESW is an integrated suite of products, you are able to access individual ESW products directly, or through the main ESW screen. As a result, different fields, values, action bar options, and pull-down options display on a screen or pop-up depending on how you accessed the screen or pop-up.

Certain ESW products also contain functionality that interfaces with other ESW products. Using SmartTest as an example, if Alliance is installed, SmartTest provides a dynamic link to Alliance that can be used to display program analysis information. If Insight is installed and specified during the analyze, the Insight program analysis functions are automatically available for viewing logic/data relationships and execution path. For example, the Scratchpad option is available on the Options pull-down if you have Insight installed.

Access to these integrated products requires only that they be installed and executed in the same libraries.

## Example 1

Figure 2 shows the Encore Primary screen that displays when you access Encore directly.

The Encore Primary screen contains these eight action bar menu items: File, View, Extract, Generate, Search, List, Options, and Help.

**Figure 2 • Encore Primary Screen**

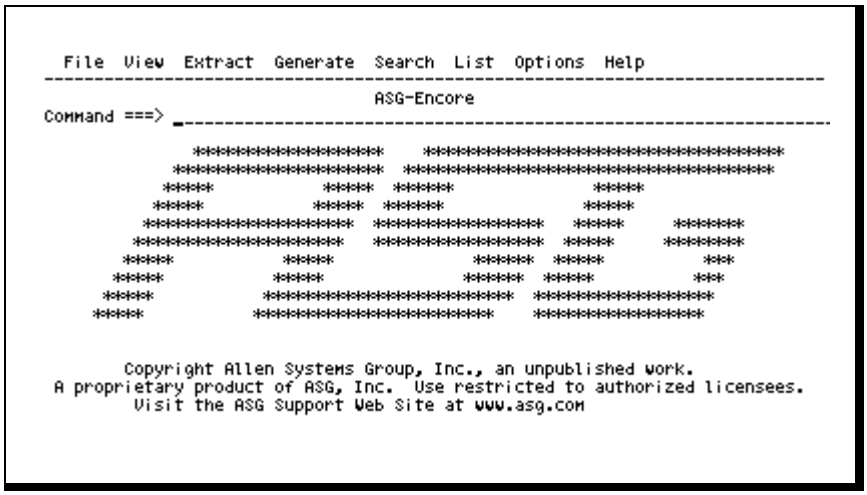


Figure 3 shows the Encore Primary screen that displays when you access Encore through ESW by selecting Re-engineer ▸ Program from the ESW action bar menu. Notice that the Primary screen name changes to ASG-ESW - Program Re-engineering when you enter Encore through ESW. Also, the Logic menu item displays if Insight is installed.

**Figure 3 • ESW Encore Primary Screen**

```
 File  View  Extract  Generate  Search  Logic  List  Options  Help
--------------------------------------------------------------------------------
                        ASG-ESW - Program Re-engineering
Command ===> ____________________________________________________________________

                 ******************    ***********************************
               ********************  ************************************
             *****            *****  ******               *****
            *****           *****  ******               *****
           ********************  *****************   *****     *******
          ********************   *****************  *****     ********
         *****            *****               ******  *****          ***
        *****            *****               ******  *****          ***
       *****            ************************  ******************
      *****            ************************    ****************

         Copyright Allen Systems Group, Inc., an unpublished work.
  A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
          Visit the ASG Support Web Site at www.asg.com
```

## Example 2

Figure 4 shows the File - Analyze Submit pop-up that displays when you access SmartTest directly. Figure 5 on page xx shows the File - Analyze Submit pop-up that displays when you access SmartTest through ESW.

**Figure 4 • File - Analyze Submit Screen**

```
                          File - Analyze Submit
Command ===> ____________________________________________________________

          E - Edit JCL                                S - Submit JCL

Compile and link JCL (PDS or sequential):
   Data set name  ____________________________________________________________

Analyze features (Y/N):
   ASG-SmartTest:  Y   Extended Analysis:  N

AKR data set name ____________________________________________
AKR program name  NEWDEMO              (if overriding PROGRAM-ID)

Analyze options:
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y
```

The actions shown on these screens can also vary. For example, the D - Doc Options action is only available on the File Prepare Program screen (or File - Analyze Submit screen) if SmartDoc is installed on your system. In Figure 4 on page xix, the Doc Options action is not displayed.

**Figure 5 • ASG-ESW - Prepare Program Screen (accessed through ESW)**

```
                         ASG-ESW - Prepare Program
Command ===> ____________________________________________________________

             E - Edit JCL     S - Submit JCL     D - Doc Options

Compile and link JCL (PDS or sequential):
   Data set name  ________________________________________________________

Analyze features (Y/N):
   Understand:  N   Test:  Y   Extended Analysis:  N   Document:  N
   Re-engineer:  N   Abend/Dump:  N
AKR data set name ______________________________________________
AKR program name  NEWDEMO              (if overriding PROGRAM-ID)

Analyze options:
   ______________________________________________________________________
   ______________________________________________________________________
   ______________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y     (Test and Abend/Dump only)
```

Notice that the Analyze features field in Figure 5 lists additional ESW products than shown on Figure 4 on page xix. This field is automatically customized to contain the ESW products you have installed on your system. These are the names of the analyze types:

| Analyze Type | Analyze Type (ESW) |
|---|---|
| ASG-Encore | Re-engineer |
| ASG-Insight | Understand |
| ASG-SmartDoc | Document |
| ASG-SmartQuest | Abend/Dump |
| ASG-SmartTest | Test |
| Extended Analysis (ASG-SmartTest with Insight installed) | Extended Analysis |

# Publication Conventions

ASG uses these conventions in technical publications:

| Convention | Represents |
|---|---|
| ALL CAPITALS | Directory, path, file, dataset, member, database, program, command, and parameter names. |
| Initial Capitals on Each Word | Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab). |
| *`lowercase italic monospace`* | Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file. |
| `Monospace` | Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax.<br>Also used for denoting brief examples in a paragraph. |
| Vertical Separator Bar ( \| ) with underline | Options available with the default value underlined (e.g., Y\|N). |
| Underline | Denotes a cursor-selectable field or line. |

# ASG Customer Support

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours.

**ASG Third-party Support.** ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

## Intelligent Support Portal (ISP)

Online product support is available at: http://www.asg.com/support/support.asp via the ASG Intelligent Support Portal (ISP). Your logon information for ISP online support is:

Customer ID = *NNNNNNNNN*
Password = *XXXXXXXXXX*

where:

*NNNNNNNNNN* is your customer ID supplied by ASG Product Distribution.
*XXXXXXXXXX* is your unique password supplied by ASG Product Distribution.

The *ASG-Intelligent Support Portal User's Guide* provides instructions on how to use the ISP and is located on the ASG Support web page.

## Telephone Support

To expedite response time, please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely as displayed
- A description of the specific steps that immediately preceded the problem
- Verify whether you received an ASG Service Pack or cumulative service tape for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack. You can access the latest software corrections and Service Packs via the ISP.
- The severity code (ASG Customer Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

**Severity Codes and Expected Support Response Times**

| Severity | Meaning | Expected Support Response Time |
|---|---|---|
| 1 | Production down, critical situation | Within 30 minutes |
| 2 | Major component of product disabled | Within 2 hours |
| 3 | Problem with the product, but customer has work-around solution | Within 4 hours |
| 4 | "How-to" questions and enhancement requests | Within 4 hours |

## The Americas

| | Phone | Fax | E-mail |
|---|---|---|---|
| **United States and Canada** | 800.354.3578 | 1.703.464.4901 | support@asg.com |

## Europe, Middle East, and Africa (EMEA)

During normal business hours, we recommend that you call the Central Support number first (except in South Africa).

| | Phone | Fax | E-mail |
|---|---|---|---|
| **Central Support** | 00.800.3544.3578 | 44.1727.812018 | support.emea@asg.com |
| **English** | 44.1727.736305 | 44.1727.812018 | support.uk@asg.com |
| **French** | 33.141.028590 | 33.141.028589 | support.fr@asg.com |
| **German** | 49.89.45716.200 | 49.89.45716.400 | support.de@asg.com |
| **Italian** | 39.0290450025 | | support.it@asg.com |
| **Dutch** | 31.30.241.6133 | | support.nl@asg.com |
| **Spanish** | 34.913.523.800 | 34.917.156.961 | support.es@asg.com |
| **South Africa** | 800.201.423 | | support.sa@asg.com |

## Asia Pacific (APAC)

| | Phone | Fax | E-mail |
|---|---|---|---|
| **Central Support** | 61.3.9645.8500 | 61.3.9645.8077 | support.au@asg.com |
| **Australia** | 800.637.947 | 61.3.9645.8077 | support.au@asg.com |
| **Hong Kong** | 800.96.2800 | | support.hk@asg.com |
| **Japan** | 81.3.5326.3684 | 81.3.5326.3001 | support.au@asg.com |
| **Singapore** | 65.224.3080 | 65.224.8516 | support.sg@asg.com |

## All Other Countries (Also for any non-working numbers)

| | Phone | Fax | E-mail |
|---|---|---|---|
| **All other countries** | 1.239.435.2201 | | support@asg.com |

If you receive a voice mail message, follow the instructions to report a production-down or critical problem. Leave a detailed message including your name and phone number. An ASG Customer Support representative will be paged and will return your call as soon as possible. Please have available the information described previously when the ASG Customer Support representative contacts you.

# ASG Documentation/Product Enhancements

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

# 1 File

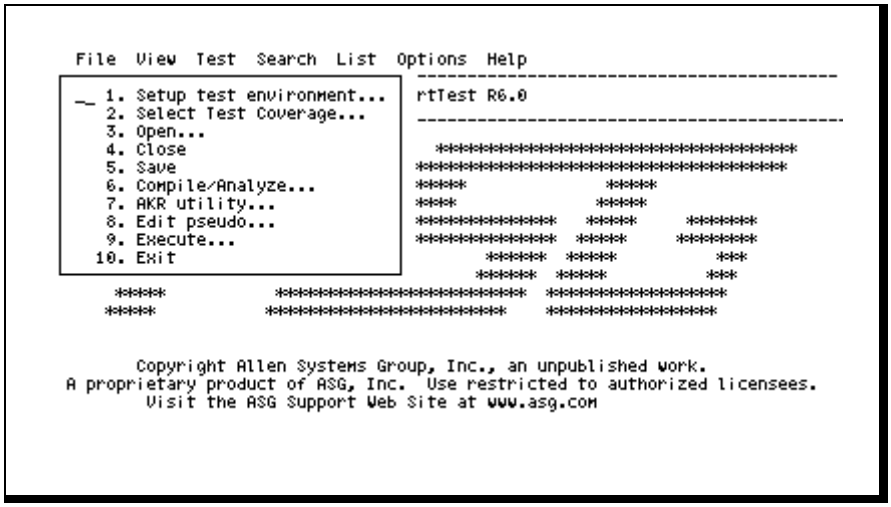
This chapter describes the options available on the File pull-down menu and contains these sections:


| Section | Page |
|---|---|
| File Pull-down | 2 |
| File - Setup Test Environment Pop-up | 3 |
| File - TCA Test Plan Selection Pop-up | 5 |
| File - Open Program (Qualify to Program) Pop-up | 6 |
| Environment - AKR Directory Pop-up | 7 |
| File - Analyze Submit Pop-up | 10 |
| File - AKR Utility Pop-up | 13 |
| File - AKR Directory Pop-up | 15 |
| File - AKR Allocate/Expand Pop-up | 17 |
| Options - COPY/Include Libraries Screen | 20 |
| File - Execute Script Pop-up | 22 |

# File Pull-down

Selecting File on the action bar displays the File pull-down shown in Figure 6. Use the File pull-down to setup the test environment, manage the AKR, compile a program, update a source file, and to exit SmartTest.

**Figure 6 • File Pull-down**




## Actions

| Field | Description |
|---|---|
| Setup test environment | Displays the File - Setup Test Environment pop-up used to setup the execution environment, specify the AKR, LOADLIB, PROCLIB, module intercepts, and to tailor the test session. |
| Select Test Coverage | Displays the File - TCA Test Plan Selection pop-up used to specify the AKR and test plan name, and to select a TCA option.<br>TCA is an optional feature in SmartTest. If you select this action and the TCA option is not installed at your site, an information message displays. |
| Open | Displays the File - Open Program (Qualify to Program) pop-up used to display a different program on the Program View screen. |
| Close | Releases a qualified program so that it becomes available for other processing such as compile and batch analysis. |
| Save | Displays the Save Options pop-up used to save pseudo code, Marks, and/or Equates in the AKR. |

| Field | Description |
|---|---|
| Compile/Analyze | Displays the File - Analyze Submit pop-up used to submit an Analyze job. |
| AKR Utility | Displays the File - AKR Utility pop-up used to display the Program Directory, allocate or expand an AKR, rename a program, and delete a program. |
| Edit pseudo | Displays the editor entry screen used to specify the source program and change pseudo code lines to actual COBOL source lines making them part of the program. If you use ASG-SmartEdit, the Options - COPY/Include Libraries Screen displays. |
| Execute | Displays the File - Execute Script pop-up used to execute a script file. |
| Exit | Ends the current SmartTest session. |

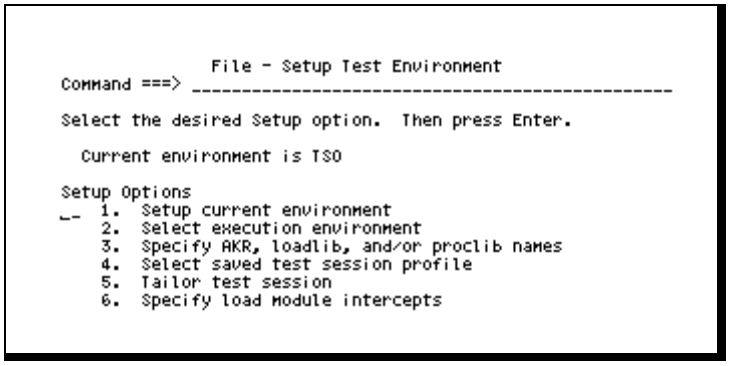
# File - Setup Test Environment Pop-up

To set up the execution environment, follow this step:

- Select File ▸ Setup test environment.

The File - Setup Test Environment pop-up, shown in Figure 7, displays. Use this screen to specify the AKR, LOADLIB, PROCLIB, and module intercepts, as well as to tailor the session.

**Figure 7 • File - Setup Test Environment Pop-up**



## Options

| Option | Description |
|---|---|
| Setup current environment | Displays the appropriate setup screen for the current environment. |
| Select execution environment | Displays the Environment Selection pop-up to select the current execution environment.<br>See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about the Environment Selection pop-up and setup screens. |
| Specify AKR, LOADLIB, and/or PROCLIB names | Displays the Environment Selection pop-up to specify the AKR, LOADLIB and/or PROCLIB names. |
| Select saved test session profile | Displays the Profile Data Set Member List screen used to select a saved test session profile. |
| Tailor test session | Displays the Test Session Tailoring screen used to turn SmartTest features selectively ON or OFF for a set of modules and/or programs.<br>See "Test Session Tailoring Screen" on page 165 for additional information. |
| Specify load module intercepts | Displays the Load Module Intercept List pop-up used to specify names of the load modules to be dynamically intercepted.<br>See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about the Load Module Intercept List pop-up. |

# File - TCA Test Plan Selection Pop-up

To enter the AKR and plan name of the test coverage plan and to select the TCA option, follow this step:

- Select File ▸ Select Test Coverage.

The File - TCA Test Plan Selection pop-up, shown in Figure 8, displays.

**Figure 8 • File - TCA Test Plan Selection Pop-up**

```
                         File - TCA Test Plan Selection
Command ===> ____________________________________________________________

Enter the AKR that contains the COVERAGE PLAN and the PLAN Name. Select the
desired TCA option and press enter.


TCA AKR Name.. 'USER12.TCA.AKR
TCA PLAN Name. TESTNTFD

  Current environment is TSO

TCA Options
__   1.  Change TCA Execution Environment
     2.  Define/Display TCA PLAN
     3.  Setup TCA TEST
     4.  TCA Test Result Utility
     5.  Generate TCA Reports
```

## Fields

| Field | Description |
|---|---|
| AKR Name | Specifies the name of the AKR that is to contain the test coverage plan. |
| PLAN Name | Specifies the name of the test coverage plan. |
| TCA Option | Specifies the number of the desired TCA option. |
| Change TCA Execution Environment | Displays the Environment Selection pop-up that is used to select the operating environment for the test. |
| Define/Display TCA Plan | Displays the T.C.A. - COVERAGE PLAN screen that is used to specify the test coverage modules and run reports. |
| Setup TCA TEST | Displays the Session Setup for the environment selected for the test coverage plan. |

| Field | Description |
|---|---|
| TCA Test Result Utility | Displays the T.C.A. - TEST UTILITY screen that is used to select and process the test results. |
| Generate TCA Reports | Displays the T.C.A. - REPORT SELECTION screen that is used to select the reports to be run. |

# File - Open Program (Qualify to Program) Pop-up

To open a program on the Program View screen, follow this step:

▶ Select File ▶ Open.

The File - Open Program (Qualify to Program) pop-up, shown in Figure 9, displays.

**Figure 9 • File - Open Program Pop-up**

```
                    File - Open Program (Qualify to Program)
Command ===> ____________________________________________________

   Type program name.  Then press Enter.

   The AKR name may be changed on the Environment Panel. Use the
   ENV command or select 'SETUP' on the 'FILE' pulldown.

Application Knowledge Repository (AKR):

   Data set name 'VIAUSER.GENERALAKR'

   Program name  _________  (blank for selection list)
```

## Fields

| Field | Description |
|---|---|
| Data set name | Specifies the dataset name of the AKR. You can change this name by using the ENVIRONMENT command or by selecting File ▶ Setup test environment. |
| Program name | Specifies the program to be viewed on the Program View screen. Leave this field blank to display and select a program name from the File - Open Program Directory pop-up. |

# Environment - AKR Directory Pop-up

The Environment - AKR Directory pop-up, shown in Figure 10, lists all programs in the specified AKR. This pop-up displays when you leave the Program name field blank on the File - Open Program (Qualify to Program) pop-up. You can scroll to view programs that are not visible, or type the L (Locate) command with a character string to scroll to a specific program. The screen is scrolled to the program that most closely matches the character string.

**Figure 10 • Environment - AKR Directory Pop-up**

```
                         Environment - AKR Directory      TESTCOBA.TESTCOBA
Command ===> ____________________________________________ Scroll ===> CSR

Select desired Program and press Enter.   Current environment is TSO

      Name     Lib  Alias of     Type     Lines    Date     Time    Jobname
_  ---------- --- ----------  ---------  ------ --------- -------- --------
   $$GENERAL    1             C.A. SET   563232 12MAR1997 06:53:33 VIAJLC4
_  *SIMPLE1     1 SIMPLE      ST,PLI         11 21AUG1997 11:12:29 VIAJLCI
_  ABBEY-AL     1             AL              0 12NOV1996 06:15:09 VIAJLC4
_  APCONV       1             ST,ASM         19 22NOV1996 13:43:37 VIAJLC4
_  ASMBASE      1             ST,ASM        105 05NOV1997 08:18:34 VIAWGFPM
_  CD05A384     1             IN           2525 12NOV1996 07:49:07 VIAJLCA
_  CD05B        1 CD05B333    IN            563 31MAR1997 06:59:21 VIAJLC8
_  CD05B333     1             IN            563 31MAR1997 06:59:21 VIAJLC8
_  CD05C        1 CD05C384    ST            632 30OCT1997 15:13:14 VIAJLC4C
_  CD05C384     1             ST            632 30OCT1997 15:13:14 VIAJLC4C
_  CD10A        1 CD10A333    IN           1662 31MAR1997 06:59:33 VIAJLC8
_  CD10A333     1             IN           1662 31MAR1997 06:59:33 VIAJLC8
_  CD10B        1 CD10B179    IN            680 31MAR1997 06:59:38 VIAJLC8
_  CD10B179     1             IN            680 31MAR1997 06:59:38 VIAJLC8
_  CD110254     1             IN           1248 31MAR1997 06:59:44 VIAJLC8
```

## Fields

| Field | Description |
|---|---|
| S - Line command area | Selects the program to be qualified. Type S to the left of the Name field to select the program. The program selected displays on the Program View screen. |
| Name | Specifies the names of the program in the AKR from the PROGRAM-ID statement.<br><br>If the analyzed program contains an ENTRY point, Name is the ENTRY point name.<br><br>If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Name is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.<br><br>See Figure 11 on page 10 for more information about entries in the AKR. |

| Field | Description |
|---|---|
| Alias of | Specifies the name of the program that contains the ENTRY point, if the analyzed program contains an ENTRY point.<br>If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.<br>See Figure 11 on page 10 for more information about alias names in the AKR. |
| Type | Indicates the type of analysis that was performed on the program. The type of analysis is specified on the File - Analyze Submit pop-up. These are the valid values: |
| AL | Indicates that an Alliance analysis was performed. |
| ASM | Indicates that the program is an Assembler source program. |
| DA | Indicates that an Extended SmartDoc analysis with a COBOL compile was performed. |
| DC | Indicates that a SmartDoc analysis with a COBOL compile was performed. |
| DS | Indicates that a SmartDoc analysis was performed. |
| DX | Indicates that an Extended SmartDoc analysis was performed. |
| EN | Indicates that an Encore analysis was performed.<br>**Note:**<br>Encore has replaced the Renaissance product and the EN analysis type replaces the former RN analysis type. However, for this release of SmartTest, RN is still a valid analysis option for the Analyze Submit pop-up. |
| IN | Indicates that an Insight analysis was performed. |
| PLI | Indicates that the program is a PL/I source program. |

| Field | Description |
|---|---|
| PROFILE | Indicates that the member contains profile information. Generally, the member name is the user ID for which the profile was created. Profile information is automatically saved when the SmartTest Session Tailoring screen is used to specify the testing environment options. Once program level testing options are specified on the Session Tailoring screen (and saved in the AKR profile member), the profile member is used by SmartTest when a test session is initiated. A profile is only used (and updated) by the user for which it is created. Any modifications to the Session Tailoring screen are automatically reflected in the profile member. |
| RC | Indicates that a Recap analysis was performed. |
| SQ | Indicates that a SmartQuest analysis was performed. |
| ST | Indicates that a SmartTest analysis was performed. |
| STX | Indicates that an Extended SmartTest analysis was performed. |
| Lines | Specifies the number of lines in the program. |
| Date | Specifies the date on which the program was analyzed. |
| Time | Specifies the time at which the program was analyzed. |
| Jobname | Specifies the job name used to analyze the program. |

# File - Analyze Submit Pop-up

Use the Analyze Submit pop-up, shown in Figure 11, to edit and submit JCL, specify AKR information for the job to analyze your program with the specified features and options, and to store it on the AKR.

To display the Analyze Submit pop-up, follow this step:

- Select File ▸ Compile/Analyze or type ANALYZE in any command input area.

**Figure 11 • File - Analyze Submit Pop-up**

```
                          File - Analyze Submit
Command ===> ____________________________________________________________

           E - Edit JCL                              S - Submit JCL

Compile and link JCL (PDS or sequential):
   Data set name  'ASG123.REL.CNTL(HTEST)'

Analyze features (Y/N):
   ASG-SmartTest:  Y   Extended Analysis:  N

AKR data set name 'ASG123.GENERAL.AKR'
AKR program name  __________          (if overriding PROGRAM-ID)

Analyze options:
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y
```

**Note:**

If you started your SmartTest session using the ESW Primary Screen, the product names listed under the Analyze features (Y/N) field may differ from the names shown in this example.

## Options

| Option | Description |
|---|---|
| E - Edit JCL | Enables you to review or change the compile/analyze JCL, if necessary. When the E option is selected, the JCL to be edited is generated from the JCL member specified in the Data set name field. The generated JCL is then displayed on the Edit screen.<br><br>When editing is complete, the ISPF SUBMIT command must be entered to submit the edited JCL for execution. Optionally, the edited JCL can be saved in a partitioned dataset by using the CREATE command. Otherwise, any changes made at this time are not saved. |
| D - SmartDoc Options | Displays only if SmartDoc is installed. Type `D` to display the SmartDoc Options screen that is used to request an Extended SmartDoc analysis and to specify which reports (if any) are to be generated. |
| S - Submit JCL | Submits the JCL to compile/analyze the specified program. The JCL submitted is generated from the JCL member specified in the Data set name field. |

## Fields

| Field | Description |
|---|---|
| Data set name | Specifies the PDS member or sequential dataset containing the JCL to compile and link the program. If the JCL resides in a source manager such as Librarian or Panvalet, use the VIASUB edit macro to submit the compile/analyze job. This is a required field. |
| Analyze features | Specifies the types of analysis you can run. |
| ASG-Encore | Displays only if Encore is installed. This type of analysis provides the logic and program execution flow capabilities of Encore. YES specifies that an Encore compile/analysis is to be performed. The default is N. |
| Extended Analysis | Provides comprehensive program analyzing capabilities in addition to the testing and debugging capabilities of SmartTest. The default is Y.<br><br>If SmartDoc is installed, an Extended SmartDoc analysis is specified on the SmartDoc Options screen. |

| Field | Description |
|---|---|
| ASG-Insight | Displays only if Insight is installed. This type of analysis provides the logic and program execution flow capabilities of Insight. The default is N. |
| ASG-SmartDoc | Displays only if SmartDoc is installed. This type of analysis provides the report information generated by SmartDoc. The default is N. |
| ASG-SmartQuest | Displays only if SmartQuest is installed. This type of analysis lets you analyze the source used in diagnosing dumps. The default value is N. |
| ASG-SmartTest | Specifies whether a SmartTest compile/analysis is to be performed. This type of analysis provides the testing and debugging information required by SmartTest. If SmartTest is the only product installed, this field contains YES and cannot be changed. The default is Y. |
| AKR data set name | Specifies the AKR that is to contain the information for the analyzed program. This is a required field. |
| AKR program name | Specifies an alias name to be used by the analyze process to save its results in the AKR.<br>If a value is not entered in this field, the analyze job uses the program name from the PROGRAM-ID statement in the COBOL source as the name under which to save results in the AKR.<br>If an AKR program name is entered, the analyzed program is saved in the AKR as the entered name, and also as an alias of the PROGRAM-ID.<br>If a program contains ENTRY points, the analyze job also saves in the AKR a member for each ENTRY point, with an alias of the PROGRAM-ID.<br>**Note:**<br>This field is only used for the AKR program name and does not change the COBOL, ASM, or PL/I program name in the source. |
| Analyze options | Specifies analyze options that are to be overridden. Default options for the analyze job are established at installation time. Analyze options that can be entered in this field are described in the *ASG-SmartTest for COBOL and Assembler User's Guide* and the *ASG-SmartTest PLI User's Guide*. |

| Field | Description |
|---|---|
| Compile? | Specifies whether to compile the program. A program need not be compiled or linked if Insight, Encore, or SmartDoc are the only analyze features specified. The compile step and link steps of your JCL can be suppressed by entering N in this field. This field is forced to a value of Y if SmartTest and/or Extended analysis is selected. |
| Link load module reusable | Tests a program under SmartTest that is dynamically loaded and is tested with RUN NOMONITOR. It is necessary to mark the load module as reusable so that the Breakpoints are retained across calls. The default is Y. |

# File - AKR Utility Pop-up

The File - AKR Utility pop-up, shown in Figure 12, is used to display the program directory, allocate or expand an AKR, rename a program, and delete a program.

To display this pop-up, follow this step:

- Select File ▸ AKR Utility or type `UTILITY` in any command input area.

**Figure 12 • File - AKR Utility Pop-up**

```
                          File - AKR Utility
Command ===> ___________________________________________________________

       Blank - Display member list       D - Delete member
       A     - Allocate/expand AKR       R - Rename member

Application Knowledge Repository (AKR):

   Data set name . . 'USER12.GENERAL.AKR'
   Member  . . . . . __________  (if "R" or "D" selected)
   New name  . . . . __________  (if "R" selected)

   Volume serial . . ______      (if not cataloged)
   Password  . . . .             (if password protected)
```

## Options

| Option | Description |
|---|---|
| Blank - Display member list | Displays the File - AKR Directory pop-up that lists the programs stored in the AKR. Programs can also be deleted or renamed on the File - AKR Directory pop-up. |
| A - Allocate/Expand AKR | Allocates a new AKR or to expand an existing AKR. Enter the AKR name in the Data Set Name field. The File - AKR Allocate/Expand pop-up is then displayed. If expanding an existing AKR, enter YES in the Expand existing AKR field on that pop-up. |
| D - Delete member | Deletes a program. Prior to selecting option D, enter the AKR name in the Data Set Name field, and the program name in the Member field. |
| R - Rename member | Renames a program. Prior to selecting option R, enter the AKR name in the Data Set Name field, the program name in the Member field, and the new name in the Newname field. |

## Fields

| Field | Description |
|---|---|
| Data Set Name | Required. Enter the name of the AKR directory. If the TSO ID qualifier is the same as the user ID, enter the library, type and program without quotes. If the TSO ID qualifier is different than the user ID, enter the project, library, type, and program within quotes. Alternately, the program name may be entered in the Member field. |
| Member | Optional. Enter the name of the program in the AKR. Required when deleting a program from the AKR. |
| Newname | Required when renaming a program. Enter the new name of the program. The name must be 1 to 10 alphanumeric characters. |
| Volume Serial | Required if the dataset specified in the Data Set Name field is not cataloged. Enter the volume serial number. If the dataset is cataloged, this field is optional. |
| Password | Optional. Enter the dataset password if the dataset is protected. |

# File - AKR Directory Pop-up

The File - AKR Directory pop-up, shown in Figure 13, lists all members in the specified AKR. This pop-up displays when you press Enter on the File - AKR Utility pop-up without entering anything in the command input area. You can scroll to view members that are not visible, or enter the L (Locate) command with a character string to scroll to a specific member. The screen is scrolled to the member that most closely matches the character string.

**Figure 13 • File - AKR Directory Pop-up**

```
                              File - AKR Directory
Command ===> ____________________________________________ Scroll ===> CSR

AKR: VIADVAP.APPL.AKR                                          Row:    158
                                 Total members:   1001   Total entries:   1003
D - Delete   R - Rename          Total records: 247330      Free space:   0.3%

     Name     New name   Alias of    Type      Date     Time  Jobname  Space
_ ---------- ---------- ---------- --------- --------- ----- -------- -----
_ DE25911Z   ----------            RC,AL,TM  05APR1999 14:50 VIAJAZ2   0.0%
_ DE26082    ----------            AL        12MAY1999 12:45 VIAKDR4   0.0%
_ DE26328    ----------            AL        14DEC1999 09:37 VIAJAZ2   0.0%
_ DE26363    ----------            AL        26MAY1999 07:51 VIAKDR4   0.1%
_ DE26843    ----------            AL        16SEP1999 13:58 VIAJAZ2   0.0%
_ DE26851    ----------            RC,AL,TM  21SEP1999 10:30 VIAJAZ2   0.0%
_ DE26891    ----------            AL        24SEP1999 16:30 VIAJAZ2   0.0%
_ DE27096    ----------            AL        09DEC1999 15:40 VIAJAZ2   0.0%
_ DE27141    ----------            AL        23DEC1999 12:41 VIAJAZ2   0.0%
_ DE27143    ----------            AL        29DEC1999 09:51 VIAM1J2   0.1%
_ DE27152    ----------            RC,AL     12JAN2000 14:16 VIAJAZ2   0.0%
_ DE27269    ----------            AL        06MAR2000 09:37 VIAJAZ2   0.0%
_ DE27375    ----------            AL        20APR2000 17:22 VIAJAZ2   0.0%
```

## Fields

| Field | Description |
|---|---|
| AKR | Specifies the complete dataset name of the requested AKR. |
| Row | Specifies the relative number in the AKR of the first member displayed on this pop-up. |
| Total members | Indicates the total number of members in this AKR, not including aliases. |
| Total entries | Specifies the total number of members in this AKR, including aliases. |
| Total records | Specifies the number of records allocated to this AKR, as entered in the Space Amount field on the File - AKR Allocate/Expand pop-up when the AKR was allocated or expanded. If the Space Units was entered as Tracks or Cylinders on that pop-up instead of Records, the amount is converted to Records for display in the Total records field. |

| Field | Description |
|---|---|
| Free space | Indicates the amount of available space in this AKR, rounded to the nearest .1 percent. |
| Line command area | The line command area, to the left of the Name field, accepts these line commands: |
| D – Delete | Deletes the member from the AKR. Alias members cannot be deleted. Programs can also be deleted on the File - AKR Utility pop-up. |
| R – Rename | Renames the member. Type the new name in the New name field. The alias member cannot be renamed. When a primary member name is changed, the alias name is automatically changed. Programs can also be renamed on the File - Utility pop-up. |
| Name | Specifies the name of the member in the AKR. Program names are taken from the PROGRAM-ID statement.<br>If an analyzed program contains an ENTRY point, Name is the ENTRY point name.<br>If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Name is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.<br>See Figure 11 on page 10 for more information about entries in the AKR. |
| New name | Provides space to rename a member. This field is required when renaming a member. The member name must be 1 to 10 alphanumeric characters. |
| Alias of | Specifies the name of the program that contains the ENTRY point, if an analyzed program contains an ENTRY point.<br>If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.<br>See Figure 11 on page 10 for more information about alias names in the AKR. |
| Type | Indicates the type of analysis that was performed on the program. The type of analysis is specified on the File - Analyze Submit pop-up and can be: |
| AL | Indicates that an Alliance analysis was performed. |

| Field | Description |
|---|---|
| ASM | Indicates that the program is an Assembler source program. |
| DA | Indicates that an Extended SmartDoc analysis with a COBOL compile was performed. |
| DC | Indicates that a SmartDoc analysis with a COBOL compile was performed. |
| DS | Indicates that a SmartDoc analysis was performed. |
| DX | Indicates that an Extended SmartDoc analysis was performed. |
| EN | Indicates that an Encore analysis was performed. |
| IN | Indicates that an Insight analysis was performed. |
| PLI | Indicates that the program is a PL/I source program. |
| RC | Indicates that a Recap analysis was performed. |
| SQ | Indicates that a SmartQuest analysis was performed. |
| ST | Indicates that a SmartTest analysis was performed. |
| STX | Indicates that an Extended SmartTest analysis was performed. |
| Date | Specifies the date on which the program was analyzed. |
| Time | Specifies the time at which the program was analyzed. |
| Jobname | Specifies the Job name used to analyze the program. |
| Space | Specifies the percentage of space the program is using on this AKR. This percentage is rounded to the nearest .1 percent. |

## File - AKR Allocate/Expand Pop-up

The File - AKR Allocate/Expand pop-up is used to allocate a new AKR or to expand an existing AKR. This pop-up displays by selecting option A on the File - AKR Utility pop-up.

If expanding an existing AKR, type `YES` in the Expand Existing AKR field on this pop-up. If allocating a new AKR, type `NO` in this field.

The space needed for the AKR depends on the size and the number of COBOL programs to be analyzed and placed in it. See the Space Amount field description for more information.

Figure 14 shows the File - AKR Allocate/Expand pop-up using SMS and a VSAM AKR.

**Figure 14 • File - AKR Allocate/Expand Pop-up**

```
                        File - AKR Allocate/Expand
Command ===> ____________________________________________________________

        S - Submit JCL       E - Edit JCL       C - Specify Catalog

Expand existing AKR . . . NO_            (Yes or No)

AKR data set name . . . . 'USER12.GENERAL.AKR'___________________________
  Management Class  . . . ________       (Blank for default MGMTCLAS)
  Storage Class . . . . . ________       (Blank for default STORCLAS)
    Volume  . . . . . . . ______
  Data Class  . . . . . . ________       (Blank for default DATACLAS)
    Space units . . . . . _________      (Records, Tracks or Cylinders)
    Space amount  . . . . ______         (Total AKR size in above units)
    Unique  . . . . . . . ___            (Enter NO for a VSAM data space)

Job statement information:
  ------------------------------------------------------------------------
  ------------------------------------------------------------------------
  ------------------------------------------------------------------------
  ------------------------------------------------------------------------
```

## Options

| Option | Description |
|---|---|
| S - Submit JCL | Submits the generated JCL to allocate and/or expand the AKR shown in the AKR Data Set Name field. |
| E - Edit JCL | Enables you to edit the generated JCL to allocate and/or expand the AKR shown in the AKR Data Set Name field. |
| C - Specify Catalog | Displays the AKR Catalog Information pop-up that is used to specify the dataset name and password of a catalog, if required. |

## Fields

| Field | Description |
|---|---|
| Expand existing AKR | Expands the AKR shown in the AKR Data Set Name field. Type `NO` to allocate the AKR. The default is NO. |
| AKR data set name | Identifies the AKR name specified on the File - AKR Utility pop-up. |
| Management Class | Specifies the management class for the AKR, if your site uses SMS. The default is set during installation. |
| Storage Class | Specifies the storage class for the AKR, if your site uses SMS. The default is set during installation. |
| Volume | Specifies the volume serial number where the AKR is to reside. This field is required if you are not using SMS. |
| Data Class | Specifies the data class for the AKR, if your site uses SMS. The default is set during installation. |
| Space units | Specifies the type of units to allocate. The units must be records, tracks or cylinders. The default is records.<br>**Note:**<br>This field is optional. Do not enter if you are using SMS. |
| Space amount | Specifies the amount of space to allocate in the type of units specified in the Primary space Units field. Enter the amount of secondary space if required. This field is required if you are not using SMS.<br>The space needed for the AKR depends on the size and the number of programs that will be analyzed and placed in it. See the *ASG-Center Installation Guide* for space estimates. |
| Unique | Specifies whether the AKR occupies a unique data space. This field is optional. Type `NO` if the AKR is a suballocation in a VSAM data space on the volume. Otherwise, type `YES`. The default is YES. |
| Job statement information | Specifies the appropriate JOB statement information for your site. This field is required to submit JCL.<br>If you need to specify a private catalog or the password for your catalog, type `C` to display the AKR Catalog Information pop-up to specify the Catalog DSN and Password. |

| Field | Description |
|---|---|
| Catalog DSN | Specifies the catalog dataset name if the AKR dataset name is to be added to a private catalog.This field is optional. |
| Password | Specifies the password for the catalog dataset name specified in the CATALOG DSN field. This field is required if the CATALOG DSN is protected. |

# Options - COPY/Include Libraries Screen

The Options - COPY/Include Libraries screen, shown in Figure 15, is used to specify the source manager and the Copy/Include dataset names. This screen displays if you are using SmartEdit and select File ▸ Edit pseudo.

**Figure 15 • Options - COPY/Include Libraries Screen**

```
  File  COBOL  SourceManager  Options
--------------------------------------------------------------------------------
                        Options - COPY/Include Libraries
Command ===> ____________________________________________ Scroll ===> CSR

Specify Source Manager, COBOL Version and COPY/Include Libraries.

Source Manager . . : PDS/Sequential   COBOL Version . : COBOL II R3

   COPY List:
  Data Set Name  . :
  Member . . . . . :
  Description  . . :

        Copy/Include Data Set Name(s)             Type VolSer   Unit   Password
- ------------------------------------------ ---- ------ -------- --------
/ 'VIAINS.TEST.CNTL'                          PDS  ______ ________
****************************** BOTTOM OF DATA ******************************
```

## Fields

| Field | Description |
|---|---|
| Source Manager | Specifies the source manager that contains the program to be updated. These are the valid source managers:<br>• PDS/Sequential - partitioned or sequential dataset<br>• Librarian<br>• Panvalet<br>• SCLM - Software Configuration and Library Manager<br>• Other (User)- user-defined source manager |
| COBOL Version | This field is not used in the Update Facility. |
| COPY List | These fields are not used in the Update Facility. |
| Line command area | Commands may be entered in the field to the left of the COPY/INCLUDE DATA SET NAME(S) field, to edit the dataset names. These commands are accepted:<br>C - Copy M - Move I - Insert<br>D - Delete A - After O - Overlay<br>R - Repeat B - Before<br>**Note:**<br>Enter the Overlay command on a line where the dataset name field is blank. If there is data on that line, it is replaced. |
| Copy/Include Data Set Name(s) | Enter COPY or INCLUDE dataset names as necessary for your program. Include Source Type as PDS for PDS, LIB for Librarian, or PAN for Panvalet. Complete VOLSER, UNIT, and PASSWORD, as required by your installation. |
| Type | Specifies the COPY or INCLUDE dataset source type. These are the valid entries:<br>• PDS - partitioned dataset or sequential dataset<br>• LIB - Librarian<br>• PAN - Panvalet |
| VolSer | Specifies the volume serial number containing the COPY or INCLUDE dataset if it is not catalogued. |

| Field | Description |
|---|---|
| Unit | Specifies the generic type of unit containing COPY/INCLUDE dataset if it is not catalogued. |
| Password | Specifies the password for the COPY or INCLUDE dataset, if required. |

After the information is entered, it is saved and displays on the screen the next time it is presented.

If you choose Librarian or Panvalet as the Source Manager, that manager displays its entry screen. You may then enter the data required to retrieve your source program.

**Note:**

When editing a Panvalet or Librarian member, the ++INCLUDE or -INC expansion is set to NO. This allows SmartTest to do the expansion when necessary.

# File - Execute Script Pop-up

To begin processing a script file, follow this step:

- Select File ▸ Execute. The File - Execute Script pop-up, shown in Figure 16, displays.

**Figure 16 • File - Execute Script Pop-up**

```
                    File - Execute Script

To execute a script file, type a Data Set Name, a Data Set Name with a
member name, or just a member name in parentheses. Next specify any
desired Options. Then press Enter.

Data Set Name  _______________________________________________________

Options
__  1.  STEP
    2.  RESUME
```

## Fields

| Field | Description |
|---|---|
| Data Set Name | Specifies the name of the script file to be processed. This may be a dataset name, a dataset and member name, or a member name (from a default script dataset). This field is required. |
| Options | Specifies the processing mode for the script file. The default is 1, single step. |
| STEP | Invokes the debug option that allows stepping through each command in the script file. Each command in the script file displays in the command input area. The displayed command can be changed if desired or erased from the command input area. Press Enter to actually execute the displayed command. Processing of the script file continues in this manner until all commands have been displayed and/or executed. |
| RESUME | Executes the remaining script file commands without stepping through each.Type 2 after a script file has been executed with STEP. |

# 2 View

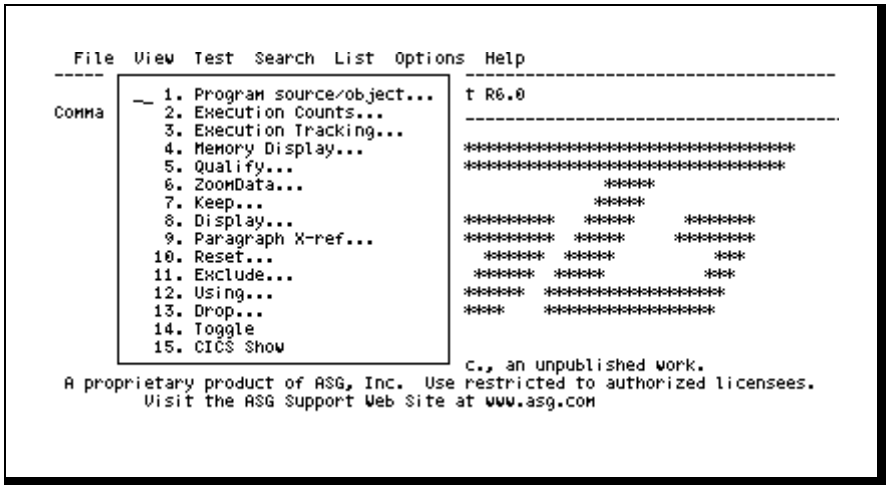
This chapter describes the options available on the View pull-down menu and contains these sections:


| Section | Page |
|---|---|
| View Pull-down | 26 |
| View - Execution Counts Pop-up | 28 |
| View - Execution Tracking Pop-up | 29 |
| View - Memory Pop-up | 30 |
| View - Qualify Request Pop-up | 31 |
| View - Zoom Data Request Pop-up | 32 |
| View - Keep Request Pop-up | 34 |
| View - Display Request Pop-up | 35 |
| View - Paragraph Cross-reference Request Pop-up | 36 |
| View - Paragraph Cross Reference Pop-up | 38 |
| View - Reset Request Pop-up | 40 |
| View - Exclude Request Pop-up | 42 |
| View - Using Request Pop-up | 43 |
| View - Drop Request Pop-up | 44 |

# View Pull-down

Selecting View on the action bar displays the View pull-down. The View pull-down, shown in Figure 17, is used to access the various methods used to view a program in SmartTest.

**Figure 17 • View Pull-down**




## Actions

| Action | Description |
|---|---|
| Program source/object | Displays the Program View screen used to display COBOL, Assembler, or PL/I source or disassembled object code and data.<br>See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about the Program View screen. |
| Execution Counts | Displays the View - Execution Counts pop-up used to customize the Execution Counts screen display. |
| Execution Tracking | Displays the View - Execution Tracking pop-up used to access the Execution Tracking screen. |
| Memory Display | Displays the View - Memory pop-up used to access the Display screen. |
| Qualify | Displays the View - Qualify Request pop-up used to display a different program on the Program View screen. |

| Action | Description |
|---|---|
| Zoomdata | Displays the View - Zoom Data Request pop-up used to scroll to the definition of a specific dataname and display its value and address. |
| Keep | Displays the View - Keep Request pop-up used to keep the value and address of a specific data item displayed at the top of the Program View screen. |
| Display | Displays the View - Display Request pop-up used to display the current value of a specific data item in the long message area. |
| Paragraph X-ref | Displays the View - Paragraph Cross-reference Request pop-up used to access the View - Paragraph Cross-reference pop-up. |
| Reset | Displays the View - Reset Request pop-up used to select view options to be reset. |
| Exclude | Displays the View - Exclude Request pop-up exclude specified lines from the display. |
| Using | Displays the View - Using Request pop-up used to specify which Register to use as a base for determining the address of fields within Assembler DSECTs. |
| Drop | Displays the View - Drop Request pop-up used to end addressability to any Assembler DSECT currently being addressed by a specific Register. |
| Toggle | Toggles from the SmartTest ISPF environment to the connected CICS or IMS environment. |
| CICS Show | Displays the last user application screen. This option is used in the CICS environment. |

# View - Execution Counts Pop-up

To customize the Execution Counts screen, follow this step:

- Select View ▸ Execution Counts. The View - Execution Counts pop-up, shown in Figure 18, displays.

**Figure 18 • View - Execution Counts Pop-up**

```
        View - Execution Counts

Select List format and Sort order.
Then press Enter.

    List Format
    1   1.  Source
        2.  Paragraph/Label

    Sort Order
    1   1.  Statement/Line
        2.  Ascending
        3.  Descending
```

See "Execution Counts Screen" on page 135 for additional information.

## Fields

| Field | Description |
|---|---|
| List Format | |
| Source | Shows current execution counts by source statement, sorting statements according to the sort option selected. This is the default if COUNTS PARAGRAPHS or COUNTS LABELS has not been specified during the current test session. |
| Paragraph/Label | Shows current execution counts by LABEL point or PARAGRAPH name, sorting labels or paragraphs according to the sort option specified. |
| Sort Order | |
| Statement/Line | Shows current execution counts, sorting source by line. This is the default if COUNTS ASCENDING or COUNTS DESCENDING has not been specified during the current test session. |

| Field | Description |
|---|---|
| Ascending | Shows current execution counts, sorting source by ascending counts. Once specified this becomes the default until another COUNTS operand is specified. |
| Descending | Shows current execution counts, sorting source by descending counts. Once specified this becomes the default until another COUNTS operand is specified. |

# View - Execution Tracking Pop-up

To customize the Execution Tracking pop-up, follow this step:

- Select View ▸ Execution Tracking.

The View - Execution Tracking pop-up, shown in Figure 19, displays. See "Execution Tracking Screen" on page 168 for additional information.

**Figure 19 • View - Execution Tracking Pop-up**

```
                View - Execution Tracking

Select format of Execution Tracking display.
Then press Enter.

  Execution Tracking
  1    1.  Source
       2.  Program
       3.  Csects
       4.  Paragraphs/Labels
       5.  Statements
       6.  Offsets
       7.  Disassembled
```

## Fields

| Execution Tracking | Description |
|---|---|
| Source | Displays the Execution Tracking screen listing programs, CSECTS, paragraphs, and source statements that were last executed for the program being tested. |
| Programs | Displays the Execution Tracking screen listing programs that were last executed for the program being tested. |
| Csects | Displays the Execution Tracking screen listing programs and CSECTS that were last executed for the program being tested. |

| Execution Tracking | Description |
|---|---|
| Paragraphs/Labels | Displays the Execution Tracking screen listing programs, CSECTS, and Labels/Paragraphs that were last executed for the program being tested. |
| Statements | Displays the Execution Tracking screen listing programs, CSECTS, paragraphs, and line numbers of statements that were last executed for the program being tested. |
| Offsets | Displays the Execution Tracking screen listing programs, CSECTS, paragraphs, and statement offsets that were last executed for the program being tested. This operand is used with the TRACKING operand. |
| Disassembled | Displays the Execution Tracking screen listing programs, CSECTS, paragraphs, and disassembled source statements that were last executed for the program being tested. This operand is used with the TRACKING operand. |

# View - Memory Pop-up

To access the Memory Display pop-up, follow this step:

- Select View ▸ Memory Display.

The View - Memory pop-up, shown in Figure 20, displays. See "Memory Display Screen" on page 143 for additional information.

**Figure 20 • View - Memory Pop-up**

```
                              View - Memory

To specify an area of memory to be displayed, type an address
specification in the address area, or leave it blank for no area
specification.  Then press Enter.

Area Specification ____________________________________________________
Map Data Name  . . ____________________________________________________
                    (Data Name, Blank for NONE, or ?  for External)
```

### Field

| Field | Description |
|---|---|
| Area Specification | Specifies the area of memory to be displayed. Type an address specification or memory keyword. If you leave this field blank, you may specify the area on the Memory Display screen. |
| Map Data Name | Specifies the high-level dataname for the structure and/or the PDS name containing the structure to be used for mapping. If you leave this field blank, you can specify the map dataname on the Memory Display screen. The supported structures that you can map with are external COBOL copybooks and PL/I Includes, as well as standard in-source structures. Map structures must also be in the native source of the program you are testing, or have been analyzed separately. |

## View - Qualify Request Pop-up

Use the View - Qualify Request pop-up to display a different program on the Program View screen. The program being tested remains active; that is if a STEP or RUN command is entered, it is executed for the active program (the program being tested), NOT the qualified program. All other SmartTest commands are performed for the qualified program.

To display another program, follow this step:

- Select View ▸ Qualify. The View - Qualify Request pop-up, shown in Figure 21, displays.

The status box changes when you enter a STEP or RUN command to indicate the status of the program being tested; not the qualified program.

**Figure 21 • View - Quality Request Pop-up**

```
                View - Qualify Request

To qualify to a program, type the Program Name, or
'*' for the active test program.  To cancel the
current qualification, leave the Program Name
blank, and select the Cancel option.  Then press
Enter.

Program Name  ________________________________

Cancel Options
__  1.  Cancel Current Qualification
    2.  Cancel All Qualifications
```

## Fields

| Field | Description |
|---|---|
| Program Name | Specifies the name of the program to be qualified. Include the module name if the program does not reside in the current load module. If you leave this field blank, the Selection List - Program Names pop-up displays. |
| Cancel Current Qualification | Releases the current qualified program so it becomes available for other processing such as compile and batch analysis. Pseudo code and equates entered into the program while it is qualified are saved/disregarded based on the values specified on the Parameter Definition pop-up. |
| Cancel All Qualification | Releases all qualified programs so they become available for other processing such as compile and batch analysis. Pseudo code and equates entered into the program while it is qualified are saved/disregarded based on the values specified on the Parameter Definition pop-up. |

# View - Zoom Data Request Pop-up

To view the value and address of a specific dataname, follow this step:

- Select View ▸ ZoomData.

The View - Zoom Data Request pop-up, shown in Figure 22, displays.

**Figure 22 • View - Zoom Data Request Pop-up**

```
                    View - Zoom Data Request

Type a fully qualified data name with any applicable subscripts,
or an address expression in the Target area, and specify
options.  For a list of data names, type a "pattern" in the
Target area. Then press Enter.

Target  _______________________________________________________

Display Options      Group Option
1   1.  Default      _  Display all
    2.  Hex            items of a group
    3.  Char

Length ___  (Optional)
```

## Fields

| Field | Description |
|---|---|
| Target | Specifies a fully-qualified dataname with any applicable subscripts, or an address expression. Leave this field blank or type a pattern using a question mark (?) to represent one character and/or an asterisk (*) to represent one or more characters in the string to display the Selection List - Data Names pop-up.<br>See "Selection List Pop-ups" on page 111 for additional information. |
| Display Options | |
| Default | Displays data values in hexadecimal format when the SET HEX mode is ON or in character format when the SET HEX mode is OFF. |
| Hex | Displays the value and address of the specified dataname in hexadecimal format. |
| Char | Displays the value and address of the specified dataname in character format. |
| Group Option | Displays all items of a group, including the levels, values, and addresses of the group items associated with the specified dataname. |
| Length | Displays the value of the data item for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. |

# View - Keep Request Pop-up

To keep the value and address of a specific data item displayed at the top of the Program View screen, follow this step:

- Select View ▸ Keep. The View - Keep Request pop-up, shown in Figure 23, displays.

**Figure 23 • View - Keep Request Pop-up**

```
                    View - Keep Request

Type a fully qualified data name with any applicable subscripts,
or an address expression in the Target area, and specify
options.  For a list of data names, type a "pattern" in the
Target area. Then press Enter.

Target  ____________________________________________________

Display Options      Group Option
1   1.  Default      _  Display all
    2.  Hex            items of a group
    3.  Char

Length ___  (Optional)
```

## Fields

| Field | Description |
|---|---|
| Target | Specifies a fully-qualified dataname with any applicable subscripts, or an address expression. Leave this field blank or type a pattern using a question mark (?) to represent one character and/or an asterisk (*) to represent one or more characters in the string to display the Selection List - Data Names pop-up.<br>See "Selection List Pop-ups" on page 111 for additional information. |
| Display Options | |
| Default | Displays data values in hexadecimal format when the SET HEX mode is ON or in character format when the SET HEX mode is OFF. |
| Hex | Displays the value and address of the specified dataname in hexadecimal format. |
| Char | Displays the value and address of the specified dataname in character format. |

| Field | Description |
|---|---|
| Group Option | Displays all items of a group, including the levels, values, and addresses of the group items associated with the specified dataname. |
| Length | Displays the value of the data item for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. |

# View - Display Request Pop-up

To display the current value of a specific data item in the long message area, follow this step:

- Select View ▸ Display. The View - Display Request pop-up, shown in Figure 24, displays.

**Figure 24 • View - Display Request Pop-up**

```
                    View - Display Request

Type a fully qualified data name with any applicable
subscripts, or an address expression in the Target area, and
specify options.  For a list of data names, type a "pattern" in
the Target area. Then press Enter.

Target  ____________________________________________________

Display Options
1   1.  Default          Length __  (Optional)
    2.  Hex
    3.  Char
```

## Fields

| Field | Description |
|---|---|
| Target | Specifies a fully-qualified dataname with any applicable subscripts, or an address expression. Leave this field blank or type a pattern using a question mark (?) to represent one character and/or an asterisk (*) to represent one or more characters in the string to display the Selection List - Data Names pop-up.<br>See "Selection List Pop-ups" on page 111 for additional information. |
| Display Options | |
| Default | Displays data values in hexadecimal format when the SET HEX mode is ON or in character format when the SET HEX mode is OFF. |

| Field | Description |
|---|---|
| Hex | Displays the value and address of the specified dataname in hexadecimal format. |
| Char | Displays the value and address of the specified dataname in character format. |
| Length | Displays the value of the data item for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. |

# View - Paragraph Cross-reference Request Pop-up

To select a target to be viewed, follow this step:

- Select View ▸ Paragraph X-ref. The View - Paragraph Cross-Reference Request pop-up, shown in Figure 25, displays.

See "Selection List Pop-ups" on page 111 for additional information.

**Figure 25 • View Paragraph Cross Reference Request Pop-up**

```
                View - Paragraph Cross-Reference Request

To list transfers of control to or from paragraphs in the program,
type a target name and select a target type and direction.  Then
press Enter.  For a name selection list for a target type (other
than types 1 and 6), type a pattern (e.g. ABC*) in the target name
input field.

Target name  ____________________________________________________

Target type                    Direction
__  1.  None - use cursor      1   1.  Previous
    2.  Mark name                  2.  Next
    3.  Perfrange name
    4.  Program name
    5.  Subset name
    6.  Line range
    7.  Label name
```

## Fields

| Field | Description |
|---|---|
| Target Name | Specifies the mark name, perform range name, program name, subset name, line range, or label name that will be used as a basis for the Paragraph Cross Reference. If you want to use cursor positioning to select the target, make no entry in this field. The entry in the Target name field must correspond to the entry in the Target type field. Read the Target type field description for valid entries in the Target name field.<br><br>If this field is left blank, and a Target type other than None is chosen, a selection list pop-up displays to allow selection of the desired target name. Wildcard characters (? for 1 character; * for zero or more characters) can be entered in this field to reduce the size of the selection list.<br><br>See "Selection List Pop-ups" on page 111 for additional information. |
| Target Type | Specifies the type of target desired. |
| None - use cursor | Indicates that the target is specified by positioning the cursor on the desired target. When this Target type is selected, the previous View screen is redisplayed to allow cursor placement. |
| Mark name | Specifies that the entry in the Target name field is a Mark name; (i.e., a set of source lines created using the scratchpad pop-ups or the COPY, MARK, MERGE, or RENAME command). All paragraph names containing the specified Mark name are displayed on the View - Paragraph Cross Reference pop-up. |
| Perfrange name | Specifies that the entry in the Target name field is a perform range name. All paragraphs containing the perform range are displayed on the View - Paragraph Cross Reference pop-up. |
| Program name | Specifies that the entry in the Target name field is a program name. All paragraph names containing the program are displayed on the View - Paragraph Cross Reference pop-up. |
| Subset name | Specifies that the entry in the Target name field is a subset. For descriptions of the subsets, select List ▸ Subsets. |
| Line range | Specifies that the entry in the Target name field is a single line number or range of lines. Line numbers are displayed in columns 1 through 6 of Source View. All paragraph names containing the specified line(s) are displayed on the View - Paragraph Cross Reference pop-up. |

| Field | Description |
|---|---|
| Label name | Specifies that the entry in the Target name field is a label (i.e., any PROCEDURE DIVISION paragraph or section name) or the literals PROCEDURE and PROC. All paragraph names containing the specified label are displayed on the View - Paragraph Cross Reference pop-up. |
| Direction | Specifies the type of transfer. |
| Previous | Displays paragraphs from which control is transferred to the target paragraph. |
| Next | Displays paragraphs to which control is transferred from the target paragraph. |

# View - Paragraph Cross Reference Pop-up

The View - Paragraph Cross Reference pop-up, shown in Figure 26, shows how control is transferred to or from the target paragraphs and provides this information:

- The location of this paragraph.
- Which paragraphs transfer control to this paragraph.
- The paragraphs to which this paragraph transfers control and how.

This pop-up displays when you press Enter on the View - Paragraph Cross-Reference Request pop-up, or by typing PREF on any screen. You can enter the LPRINT * command on this pop-up to copy the data to the List file.

**Figure 26 • View - Paragraph Cross Reference Pop-up**

```
                    View - Paragraph Cross Reference
Command ===> ________________________________________ Scroll ===> CSR

Target(s): PROCEDURE DIVISION
Direction: PREVIOUS

   A : Add to target       P : Execute PREF       S : Select for viewing

S   Comes from                  How            Target paragraph(s)
- ---------------------------- -------------- ---------------------------
_                             ENTRY          PROCEDURE DIVISION
****************************** BOTTOM OF DATA ****************************
```

## Fields

| Field | Description |
|---|---|
| Target(s) | Specifies the paragraph name(s) of the requested target. |
| Direction | Specifies the type of transfer. PREVIOUS highlights paragraphs from which control is transferred. NEXT highlights paragraphs to which control is transferred. The default is PREVIOUS. |
| Line command area | Specifies the area to the left of the Comes from or Target paragraph(s) field. This field accepts these entries: |
| A | Adds the selected paragraph to the target paragraph(s). If the direction is PREVIOUS, the target paragraph is added at the top of the list. If the direction is NEXT, the target paragraph is added to the end of the list. |
| P | Executes another paragraph cross reference. |
| S | Returns to the source code with the first line of the paragraph displayed at the top of the screen. |
| Comes from | Displays when the direction is PREVIOUS. It lists the paragraph names that transfer control to the target paragraph(s). All paragraphs that relate to the target paragraph are grouped together. |
| Goes to | Displays when the direction is NEXT. It lists the paragraph names that receive control from the target paragraph(s). |

| Field | Description | |
|---|---|---|
| How | Indicates the way in which control is transferred from or to the target paragraph. | |
| | Method | Abbreviations |
| | PERFORM | PERF(3), PERF, P(2), P |
| | RETURN | RET, R |
| | GOTO | GO(3), G(2), G |
| | ALTERED GOTO | ALTER, AG |
| | GOTO DEPENDING | GODEP, GD |
| | FALLTHRU | FALL, F |
| | USE ON ERROR | USE ERROR, U |
| | USE FOR DEBUG | USE DEBUG, U |
| | USE BEFORE REPORT | USE REPORT, U |
| | *CALL, INTERNAL | CALL_INT, CALL RPT, CA |
| | *CALL INTERNAL | CLL INT, CI |
| | *CALL RETURN | CLL RTN, CR |
| Target Paragraph(s) | Specifies the paragraphs requested as the target in the PREF command. Ditto marks (") are shown when multiple paragraphs transfer control to the same paragraph | |

# View - Reset Request Pop-up

Use the View - Reset Request pop-up to perform these functions:

- Turn off highlighting.
- Erase tags and line labels.
- Redisplay excluded lines.
- Cancel pending line commands.
- Terminate message line displays.
- Delete pseudo code statements and lines created as a result of one of the SmartTest Zoom line commands (ZA, ZD, or ZH).
- Delete WHEN statements and kept lines.

To display this pop-up, follow this step:

- Select View ▸ Reset. The View - Reset Request pop-up, shown in Figure 27, displays.

**Figure 27 • View - Reset Request Pop-up**

```
                    View - Reset Request

To reset display features, select an Option.  Then press Enter.

  Reset Options
  /  All (Excluded lines, Highlighted lines, Tags, Messages)
  _  All Lines
  _  Excluded Lines
  _  Highlighted Lines
  _  Tag Comments
  _  Messages
  _  Labels
  _  Keep Windows
  _  Zoom Windows
  _  Pseudo Code (Including Breaks and Whens)
  _  Breaks
  _  Whens
```

## Fields

| Reset Options | Description |
|---|---|
| All | Resets all conditions indicated by the HI, TAG, EXCLUDED, LINES and MESSAGE operands.<br>The ALL option does not reset conditions indicated by the PSEUDO, ZOOM, BREAKS, WHENS, LABELS, or KEEPS operands. |
| All Lines | Clears any pending line commands. |
| Excluded Lines | Displays any lines that have been excluded. |
| Highlighted Lines | Removes the highlighting from any line that has been highlighted as a result of another SmartTest command. |
| Tag Comments | Erases tags in columns 73 through 80 that have been set as a result of another SmartTest command. |
| Messages | Erases short or long messages that are displayed. |
| Labels | Erases all line labels entered in the prefix area (columns 1 through 6). A line label can also be erased by typing over it with blanks on the Program View screen. |
| Keep Windows | Deletes all lines kept at the top of the screen as a result of the KEEP command. |

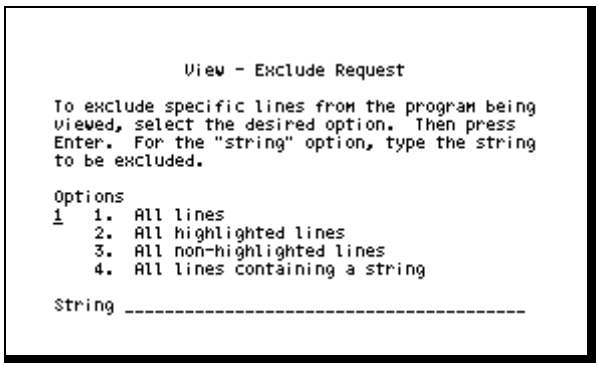
| Reset Options | Description |
|---|---|
| Zoom Windows | Deletes all lines created from using one of the SmartTest Zoom line commands such as ZA, ZD, ZG, ZGH, or ZH. |
| Pseudo Code | Deletes all pseudo code statements within the current active or qualified program. |
| Breaks | Deletes all BREAK statements within the current program if they are the first pseudo code verbs in the block of pseudo code. |
| Whens | Deletes all WHEN statements within the current active or qualified program. |

# View - Exclude Request Pop-up

To exclude specific lines from the program, follow this step:

- Select View ▸ Exclude. The View - Exclude Request pop-up, shown in Figure 28, displays.

**Figure 28 • View Exclude Request Pop-up**




## Fields

| Field | Description |
|---|---|
| All lines | Excludes all lines from the display. |
| All highlighted lines | Excludes all lines that have been highlighted by another action from the display. |
| All non-highlighted | Excludes all lines that have not been highlighted from the display. |

| Field | Description |
|---|---|
| All lines containing a string | Excludes all lines that contain the specified string. |
| String | Specifies the string to be used to exclude lines for Option 4. You can use the question mark (?) to represent one character and the asterisk (*) to represent one or more characters. |

# View - Using Request Pop-up

The View - Using Request pop-up specifies which Register to use as a base for determining the address of fields within Assembler DSECTs. The ZOOMDATA command uses this information to display data fields that are DSECT relative.

To display this pop-up, follow this step:

- Select View ▸ Using. The View - Using Request pop-up, shown in Figure 29, displays.

**Figure 29 • View - Using Request Pop-up**

```
            View - Using Request

Type Label Name and Register number, then
select Addressing Mode, if appropriate.
Then press Enter.

DSECT Name ______________________________

Register    ___   (R0-R15)

Addressing Mode
_  31 Bit (AMODE 31)
```

## Fields

| Field | Description |
|---|---|
| DSECT Name | Specifies the Assembler DSECT name. |
| Register | Specifies a register R0 through R15. |
| Addressing Mode 31 Bit (AMODE) | Indicates the register address is 31-bit. |

# View - Drop Request Pop-up

To end addressability to any Assembler DSECT currently being addressed by a specified register, follow this step:

- Select View ▸ Drop. The View - Drop Request pop-up, shown in Figure 30, displays.

**Figure 30 • View - Drop Request Pop-up**

```
        View - Drop Request

Enter Register number to Drop.
Then press Enter.

Register  __  (R0-R15)
```

## Field

**Register.** A register R0 through R15 for which the address was set.

# 3 Test

This chapter describes the options available on the Test pull-down menu and contains these sections:

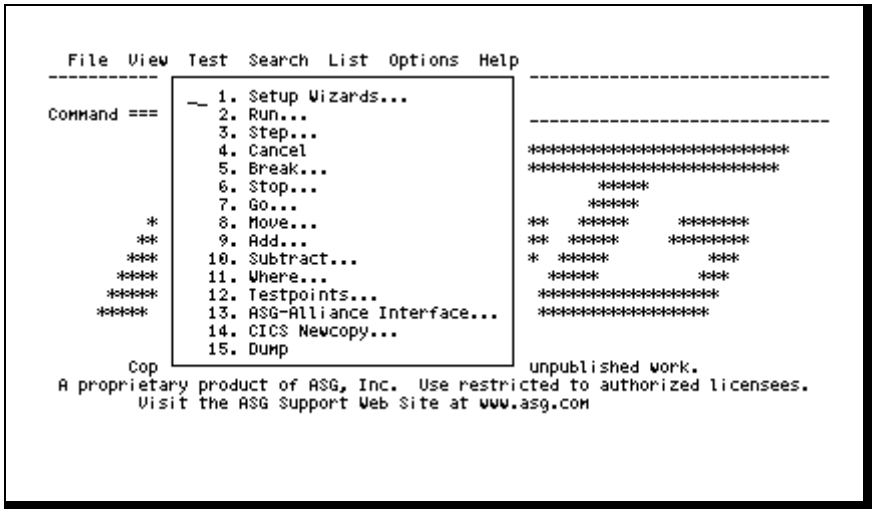

| Section | Page |
|---|---|
| Test Pull-down | 46 |
| Test - Run Request Pop-up | 48 |
| Test - Run Request (Backtrack Recording Mode Active) Pop-up | 49 |
| Test - Run Request (Backtrack Review Mode Active) Pop-up | 51 |
| Test - Run Request (CICS Environment-Test Active) Pop-up | 52 |
| Test - Run Request (CICS - Backtrack Recording Mode Active) Pop-up | 53 |
| Test - Step Request Pop-up | 54 |
| Test - Step Request (Backtrack Recording Mode) Pop-up | 56 |
| Test - Step Request (Backtrack Review Mode) Pop-up | 58 |
| Test - Step Request (CICS Environment-Test Active) Pop-up | 59 |
| Test - Step Request (CICS - Backtrack Recording Mode Active) Pop-up | 61 |
| Test - Break Request Pop-up | 63 |
| Break - Data Name Pop-up | 64 |
| Break - Line Number Pop-up | 66 |
| Break - Label/Paragraph Name Pop-up | 67 |
| Break - on Subset Pop-up | 68 |
| Break - Program Name Pop-up | 69 |

| Section | Page |
|---|---|
| Break - Pattern Pop-up | 71 |
| Test - Stop Request Pop-up | 73 |
| Test - Go Request Pop-up | 74 |
| Test - Move Request Pop-up | 75 |
| Test - Add Request Pop-up | 76 |
| Test - Subtract Request Pop-up | 77 |
| Test - Where Request Pop-up | 78 |
| Test - SmartTest Testpoint Generation Pop-up | 79 |
| SmartTest/Alliance Interface Pop-up | 80 |
| Test - Newcopy Request Pop-up | 81 |


# Test Pull-down

Selecting Test on the action bar displays the Test pull-down shown in Figure 31. The Test pull-down is used to access the various methods used to execute a program and manipulate data values.

**Figure 31 • Test Pull-down**



## Actions

| Action | Description |
|---|---|
| Setup Wizard | Displays the Test- Setup Wizard pop-up used to assist in setting up test sessions in IMS, CICS, and TSO foreground or batch connect environments. |
| Run | Displays the Test - Run Request pop-up used to begin or continue testing of a program after an interrupt. The Test - Run Request pop-up will appear differently depending on whether CICS is the active environment and whether Backtrack Recording is active. |
| Step | Displays the Test - Step Request pop-up used to begin or continue testing of a program after an interrupt or to continue stepping through a program after a halt. The Test - Step Request pop-up will appear differently depending on whether CICS is the active environment and whether Backtrack Recording is active. |
| Cancel | Terminates the current test session. |
| Break | Displays the Test - Break Request pop-up used to insert a Breakpoint before or after a specified target. |
| Stop | Displays the Test - Stop Request pop-up used to set an address stop for a specified data item. |
| Go | Displays the Test - Go Request pop-up used to transfer control to the statement containing a specified COBOL or Assembler label name, pseudo code label, or line. |
| Move | Displays the Test - Move Request pop-up used to assign a value contained in or represented by the first operand to the specified data item. |
| Add | Displays the Test - Add Request pop-up used to add the value contained in or represented by the first operand to the specified data item. |
| Subtract | Displays the Test - Subtract Request pop-up used to subtract the value contained in or represented by the first operand from the specified data item. |
| Where | Displays the Test - Where Request pop-up used to identify the Storage location using an address expression. |
| Testpoints | Displays the Test - SmartTest Testpoint Generation pop-up used to set breakpoints with an impact dataset. |

| Action | Description |
|---|---|
| ASG-Alliance Interface | Displays the ASG-SmartTest/ASG-Alliance Interface pop-up used to link to Alliance. |
| CICS Newcopy | Displays the Test - Newcopy Request pop-up used to load a new copy of the PPT module specified into CICS. |
| Dump | Generates an MVS symptom and snap dump, or a CICS transaction dump, of the current suspended program or transaction. |

# Test - Run Request Pop-up

To begin or resume testing of a program, follow this step:

- Select Test ▸ Run in environments other than CICS when Backtrack Recording mode is not active. The Test - Run Request pop-up, shown in Figure 32, displays.

**Figure 32 • Test - Run Request Pop-up**

```
                    Test - Run Request

Enter "Run To" line number and Monitor option,
as needed.  Then press Enter.

Run to Line Number ______

Select Monitor option
__  1.  Monitor
    2.  No Monitor
```

## Fields

| Field | Description |
|---|---|
| Run to Line Number | Indicates that execution will be completed at the specified line number. |

| Field | Description |
|---|---|
| Select Monitor option | |
| Monitor | Indicates all features are available for testing, including Address Stop Entry, Statement Counts, Execution Tracking, and pseudo code monitoring. The default is MONITOR. |
| No Monitor | Reduces the CPU overhead during a test session. This is particularly useful when testing large programs or programs that read thousands of records. |

# Test - Run Request (Backtrack Recording Mode Active) Pop-up

To enter, exit, and simulate execution backward or forward in the execution history, follow this step:

- Select Test ▸ Run (in environments other than CICS) when Backtrack Recording mode is active. The Test - Run Request (Backtrack Recording Mode Active) pop-up, shown in Figure 33, displays.

**Note:**

An asterisk (*) indicates an option that invokes the Backtrack Review mode.

**Figure 33 • Test - Run Request (Backtrack Recording Mode Active) Pop-up**

```
          Test - Run Request (Backtrack Recording Mode Active)

Enter Direction and any "Run To" choices, as needed. Then press Enter.

NOTE: * indicates an option that invokes BackTrack Review Mode.

Direction:
__  1.  Forward (Default)
    2.  *Backward

Run To Choices:
  *Run To Top . . . . _  Enter non-blank to select
   Run to Line Number ______
  *Run to Data Name . ________________________________________________
```

## Fields

| Field | Description |
|---|---|
| Direction | |
| Forward | Directs the RUN in the forward direction to the specified target. |
| *Backward | Directs the RUN in the backward direction to the specified target. |
| Run to Choices | |
| *Run to Top | Positions the Backtrack Review facility to the top (first or oldest entry) of the execution history. The Backtrack arrow (BKTR=>) points to the first statement to have been executed in Backtrack Recording mode. |
| Run to Line Number | Defines the target line number in the execution history. If the line number cannot be found in the execution history (in the current direction), SmartTest displays an error message. If the target number exists more than once in the execution history, the target will be the first occurrence of the line number in the direction you are going. |
| *Run to Data Name | Defines a target dataname in the execution history. This dataname must have been modified by a statement in the execution history. If the dataname cannot be found, SmartTest displays an error message. If the dataname exists more than once, the target will be the first occurrence of the dataname in the current direction. |

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for detailed information about the BACKTRACK facility.

# Test - Run Request (Backtrack Review Mode Active) Pop-up

To enter, exit, and simulate execution backward or forward in the execution history while in Backtrack Review mode, follow this step:

- Select Test ▸ Run when Backtrack Review mode is active. The Test - Run Request (Backtrack Review Mode Active) pop-up, shown in Figure 34, displays.

**Figure 34 • Test - Run Request Pop-up**

```
        Test - Run Request (Backtrack Review Mode Active)

Enter a Direction, if needed, and fill in the appropriate "Run To"
option. Then press Enter.

Direction (Default is current direction)
__  1.  Forward
    2.  Backward

Run To ...
  T, B or *  . . . . _  Top, Bottom or current stmt/exit review
  Line Number  . . . ______
  Modified Data Name ____________________________________________
```

## Fields

| Field | Description |
|---|---|
| Direction | |
| Forward | Directs the RUN in the forward direction to the specified target. |
| Backward | Directs the RUN in the backward direction to the specified target. |
| Run To... | |
| T, B, or * | Enter T to position the Backtrack Review facility at the top of the execution history. Type B to position the Backtrack Review facility at the bottom of the execution history. Type * to terminate the Backtrack Review facility and return to the next instruction to be executed. |

| Field | Description |
|---|---|
| Line Number | Defines the target line number in the execution history. If the line number cannot be found in the execution history (in the current direction), SmartTest displays an error message. If the target number exists more than once in the execution history, the target will be the first occurrence of the line number in the direction you are going. |
| Modified Data Name | Defines a target dataname in the execution history. This dataname must have been modified by a statement in the execution history. If the dataname cannot be found, SmartTest displays an error message. If the dataname exists more than once, the target will be the first occurrence of the dataname in the current direction. |

# Test - Run Request (CICS Environment-Test Active) Pop-up

To begin or resume testing of a program, follow this step:

- Select Test ▸ Run.

The Test - Run Request (CICS Environment-Test active) pop-up, shown in Figure 35, displays. This pop-up displays in the CICS environment when Backtrack Recording mode is not active.

**Figure 35 • Test - Run Request (CICS Environment-Test Active) Pop-up**

```
       Test - Run Request (CICS Environment-Test active)

Enter a "Run To" line number, if desired, and indicate whether
a FORCE operand is required.  Then press enter.

Run to Line Number . ______

CICS Option:
  FORCE update  . _  Enter a non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Run to Line Number | Indicates that execution will be completed at the specified line number. |
| CICS Option | FORCE update. Enter any character to cause the monitor to ignore the current storage violation warning allowing the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. You must have proper authorization to use the FORCE option. |

# Test - Run Request (CICS - Backtrack Recording Mode Active) Pop-up

The Test - Run Request (CICS - Backtrack Recording Mode Active) pop-up, shown in Figure 36, is used to enter, exit, and simulate execution backward or forward in the execution history. An asterisk (*) indicates an option that invokes the Backtrack Review mode.

To access this pop-up, follow this step:

- Select Test ▸ Run in a CICS environment when Backtrack Recording mode is active.

**Figure 36 • Test - Run Request (CICS-Backtrack Recording Mode Active) Pop-up**

```
      Test - Run Request (CICS - Backtrack Recording Mode Active)

Select Direction, "Run To" choices, or CICS Option, as needed.
Then press Enter.

NOTE: * indicates an option that invokes BackTrack Review Mode.

Direction
__  1.  Forward (Default)
    2.  *Backward

Run To ...
  *Top of History . . _  Enter non-blank to select
   Line Number  . . . ______
  *Modified Data Name ____________________________________________

CICS Option:
  FORCE update _  Enter non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Direction | |
| Forward | Directs the RUN in the forward direction to the specified target. |
| *Backward | Directs the RUN in the backward direction to the specified target. |
| Run To | |
| *Top of History | Positions the Backtrack Review facility to the top (first or oldest entry) of the execution history. The backtrack arrow (BKTR=>) will point to the first statement to have been executed in Backtrack Recording mode. |

| Field | Description |
|---|---|
| Line Number | Defines the target line number in the execution history. If the line number cannot be found in the execution history (in the current direction), SmartTest displays an error message. If the target number exists more than once in the execution history, the target will be the first occurrence of the line number in the direction you are going. |
| *Modified Data Name | Defines a target dataname in the execution history. This dataname must have been modified by a statement in the execution history. If the dataname cannot be found, SmartTest displays an error message. If the dataname exists more than once, the target will be the first occurrence of the dataname in the current direction. |
| CICS Option | FORCE update. Enter any character to cause the monitor to ignore the current storage violation warning allowing the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. You must have proper authorization to use the FORCE option. |

# Test - Step Request Pop-up

The Test - Step Request pop-up, shown in Figure 37, is used to begin a test of a program, to resume testing after an interrupt occurs, or to continue stepping through program execution after the Step process has been initiated.

To access the Test - Step Request pop-up, follow this step:

- Select Test ▸ Step in environments other than CICS when Backtrack Recording mode is not active.

**Figure 37 • Test - Step Request Pop-up**

```
                    Test - Step Request

Select Step type, Step Options, or Monitor Option, as needed.
Then press Enter.

Step ...
__  1.  to next STATEMENT (Default)
    2.  to next PARAGRAPH/LABEL
    3.  OVER (COBOL PERFORM, Assembler Linkage, or
        PL/I CALL)

Step Options:
  Step Count _____  Enter 1 to 65535 (999 for Auto)
  Step Auto  _      Enter non-blank to select

Monitor Option: (If OVER specified)
__  1.  Monitor
    2.  No Monitor
```

## Fields

| Field | Description |
|---|---|
| Step | |
| to next STATEMENT | Specifies that the next COBOL statement, Assembler instruction, or PL/I statement is executed. Execution pauses after the statement or instruction is executed. |
| to next PARAGRAPH/ LABEL | Causes SmartTest to step to the next paragraph in a COBOL program, the next label in an Assembler program, or the next procedure in a PL/I program. |
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. Code within the PERFORMed or CALLed routine is executed and the execution stops at the statement following the PERFORM or CALL. If a program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement following the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test. |
| Step Options | |
| Step Count | Specifies the number of statements or instructions to execute. |
| Step Auto | Displays each statement before it is executed. |
| Monitor Option | |
| Monitor | Indicates all features are available for testing, including Address Stop Entry, Statement Counts, Execution Tracking, and pseudo code monitoring. The default is MONITOR. |
| No Monitor | Reduces the CPU overhead during a test session. This is particularly useful when testing large programs or programs that read thousands of records. |

# Test - Step Request (Backtrack Recording Mode) Pop-up

The Test - Step Request (Backtrack Recording Mode) pop-up, shown in Figure 38, is used to resume testing after an interrupt occurs or enter, and step backward through the execution history while in Backtrack Recording Mode.

To access the Test - Step Request pop-up, follow this step:

- Select Test ▸ Step in environments other than CICS when Backtrack Recording mode is active.

**Figure 38 • Test - Step Request (Backtrack Recording Mode) Pop-up**

```
     Test - Step Request (Backtrack Recording Mode)

Select Step type, Direction, or Step Options, as needed.
Then press Enter.

Step ...
__  1.  to next STATEMENT (Default)
    2.  to next PARAGRAPH/LABEL
    3.  OVER (COBOL PERFORM or Assembler Linkage)

Direction:
__  1.  Forward (Default)
    2.  *Backward (invokes BackTrack Review Mode)

Step Options:
  Step Count ___  Enter 1 to 999
  Step Auto  _    Enter non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Step | |
| to next STATEMENT | Specifies that the next COBOL statement, Assembler instruction, or PL/I statement is executed. Execution pauses after the statement or instruction is executed. |
| to next PARAGRAPH /LABEL | Causes SmartTest to step to the next paragraph in a COBOL program, the next label in an Assembler program, or the next procedure in a PL/I program. |

| Field | Description |
|---|---|
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. Code within the PERFORMed or CALLed routine is executed and the execution stops at the statement following the PERFORM or CALL. If a program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement following the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test. |
| Direction | |
| Forward | Steps in the forward direction to the specified target. |
| *Backward | Steps in the backward direction to the specified target. This option invokes the Backtrack Review Mode. |
| Step Options | |
| Step Count | Specifies the number of statements or instructions to execute. |
| Step Auto | Displays each statement before it is executed. |

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for detailed information about the BACKTRACK facility.

# Test - Step Request (Backtrack Review Mode) Pop-up

To exit and step backward or forward through the execution history while in Backtrack Review mode, follow this step:

- Select Test ▸ Step. The Test - Step Request (Backtrack Review Mode) pop-up, shown in Figure 39, displays. This pop-up displays in all environments when Backtrack Review mode is active.

**Figure 39 • Test - Step Request (Backtrack Review Mode) Pop-up**

```
        Test - Step Request (Backtrack Review Mode)

Select Step type, Direction, or Step Options, as needed.
Then press Enter.

Step ...
__   1.  to next STATEMENT (Default)
     2.  to next Paragraph/Label

Direction (Default is current direction)
__   1.  Forward
     2.  Backward

Step Options:
  Step Count ___  Enter 1 to 999
  Step Auto  _    Enter non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Step | |
| to next STATEMENT | Specifies that the next COBOL statement, Assembler instruction, or PL/I statement is executed. Execution pauses after the statement or instruction is executed. |
| to next PARAGRAPH/ LABEL | Causes SmartTest to step to the next paragraph in a COBOL program, the next label in an Assembler program, or the next procedure in a PL/I program. |
| Direction | |
| Forward | Steps in the forward direction to the specified target. |
| *Backward | Steps in the backward direction to the specified target. This option invokes the Backtrack Review Mode. |
| Step Options | |
| Step Count | Specifies the number of statements or instructions to execute. |
| Step Auto | Displays each statement before it is executed. |

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for detailed information about the BACKTRACK facility.

# Test - Step Request (CICS Environment-Test Active) Pop-up

The Test - Step Request (CICS Environment-Test active) pop-up, shown in Figure 40, is used to begin a test of a program, to resume testing after an interrupt occurs or to continue stepping through program execution after the Step process has been initiated.

To access this pop-up, follow this step:

- Select Test ▸ Step in a CICS environment when Backtrack Recording mode is not active.

**Figure 40 • Test - Step Request (CICS Environment-Test Active) Pop-up**

```
Test - Step Request (CICS Environment-Test active)

Select Step type, Step Options, or CICS Option, as
needed.  Then press Enter.

Step ...
__   1.  to next STATEMENT (Default)
     2.  to next PARAGRAPH/LABEL
     3.  OVER (COBOL PERFORM or Assembler Linkage)

Step Options:
  Step Count ___  Enter 1 to 999
  Step Auto  _    Enter non-blank to select

CICS Option:
  FORCE update _  Enter non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Step | |
| to next STATEMENT | Specifies that the next COBOL statement, Assembler instruction, or PL/I statement is executed. Execution pauses after the statement or instruction is executed. |
| to next PARAGRAPH/ LABEL | Causes SmartTest to step to the next paragraph in a COBOL program, the next label in an Assembler program, or the next procedure in a PL/I program. |

| Field | Description |
|---|---|
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. Code within the PERFORMed or CALLed routine is executed and the execution stops at the statement following the PERFORM or CALL. If a program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement following the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test. |
| Step Options | |
| Step Count | Specifies the number of statements or instructions to execute. |
| Step Auto | Displays each statement before it is executed. |
| CICS Option | FORCE update. Enter any character to cause the monitor to ignore the current storage violation warning allowing the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. You must have proper authorization to use the FORCE option. |

# Test - Step Request (CICS - Backtrack Recording Mode Active) Pop-up

The Test - Step Request (CICS - Backtrack Recording Mode Active) pop-up, shown in Figure 41, is used to resume testing after an interrupt occurs, or enter and step backward through the execution history while in Backtrack Recording Mode.

To access this pop-up, follow this step:

- Select Test ▸ Step in a CICS environment when Backtrack Recording mode is active.

**Figure 41 • Test - Step Request (CICS-BackTrack Recording Mode Active) Pop-up**

```
 Test - Step Request (CICS - BackTrack Recording Mode Active)

Select Step type, Direction, Step Options, or CICS option, as
needed.  Then press Enter.

Step ...
__  1.  to next STATEMENT (Default)
    2.  to next PARAGRAPH/LABEL
    3.  OVER (COBOL PERFORM or Assembler Linkage)

Direction:
__  1.  Forward (Default)
    2.  *Backward (Invokes BackTrack Review Mode)

Step Options:
  Step Count _____  Enter 1 to 65535 (999 if Auto)
  Step Auto  _      Enter non-blank to select

CICS Option:
  FORCE update _  Enter non-blank to select
```

## Fields

| Field | Description |
|---|---|
| Step | |
| to next STATEMENT | Specifies that the next COBOL statement, Assembler instruction, or PL/I statement is executed. Execution pauses after the statement or instruction is executed. |
| to next PARAGRAPH/ LABEL | Causes SmartTest to step to the next paragraph in a COBOL program, the next label in an Assembler program, or the next procedure in a PL/I program. |

| Field | Description |
|---|---|
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. Code within the PERFORMed or CALLed routine is executed and the execution stops at the statement following the PERFORM or CALL. If a program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement following the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test. |
| Direction | |
| Forward | Steps in the forward direction to the specified target. |
| *Backward | Steps in the backward direction to the specified target. This option invokes the Backtrack Review Mode. |
| Step Options | |
| Step Count | Specifies the number of statements or instructions to execute. |
| Step Auto | Displays each statement before it is executed. |
| CICS Option | FORCE update. Enter any character to cause the monitor to ignore the current storage violation warning allowing the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. You must have proper authorization to use the FORCE option. |

# Test - Break Request Pop-up

A Breakpoint creates an interrupt in the program execution. Program execution stops when the BREAK statement is encountered.

To insert a Breakpoint before or after the statement containing a specified target, follow this step:

- Select Test ▸ Break. The Test - Break Request pop-up, shown in Figure 42, displays.

**Figure 42 • Test - Break Request Pop-up**

```
        Test - Break Request

Specify Break Type, and Location.
Then press Enter.

    Break Type
    1   1.  Data...
        2.  Line...
        3.  Label/Paragraph...
        4.  Subset...
        5.  Program...
        6.  Pattern...

    Break Location
    1   1.  Before
        2.  After
```

## Fields

| Field | Description |
|---|---|
| Break Type | |
| Data | Displays the Break - Data Name pop-up used to specify datanames on which to break. |
| Line | Displays the Break - Line Number pop-up used to specify line numbers on which to break. |
| Label/Paragraph | Displays the Break - Label/Paragraph Name pop-up used to specify label names on which to break. |
| Subset | Displays the Break - COBOL Subset pop-up used to specify COBOL subsets on which to break. |
| Program | Displays the Break - Program Name pop-up used to specify a (sub)program on which to break. |
| Pattern | Displays the Break - Pattern pop-up used to specify pattern strings on which to break. |

| Field | Description |
|---|---|
| Break Location | |
| Before | Inserts the Breakpoint before the statement containing the specified target. |
| After | Inserts the Breakpoint after the statement containing the specified target. |

# Break - Data Name Pop-up

Use the Break - Data Name pop-up, shown in Figure 43, to set Breakpoints before or after specified datanames. This pop-up displays by selecting the Data action on the Test - Break Request pop-up.

**Figure 43 • Break - Data Name Pop-up**

```
                         Break - Data Name

To break on a data name reference, specify a fully qualified name
and options. Then press Enter.  For a data name list, type a
"pattern" (or blanks) in the Data Name area.

Data Name  ____________________________________________________

Data Reference Type         Indirect Impact
1   1.  Uses                1   1.  None                Levels ___
    2.  Modifications           2.  Of Size Change
    3.  Uses+Modifications      3.  Of Value Change

Direction                   Options
2   1.  All                 _  No Data Aliasing
    2.  Next                _  IN-Clause...
    3.  Previous
    4.  First
    5.  Last
```

## Fields

| Field | Description |
|---|---|
| Data Name | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. To display the Selection List pop-up, leave the dataname field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). This field is required. |
| Data Reference Type | Specifies the datanames at which Breakpoints are set can be restricted to only those datanames that are used, or modified, by specifying the type of dataname reference. |
| Uses | Includes occurrences of the target name where its value is being tested or used. |
| Modifications | Includes occurrences of the target name where its value is being set or modified. |
| Uses+ Modifications | Includes occurrences of the target name where its value is being tested, used or modified. |
| Indirect Impact | Restricts the search to include only the occurrence of datanames that are indirectly affected by the specified dataname. |
| None | Includes occurrences of the target name (and aliases if specified) where it is directly tested, used, set, modified, or defined. |
| Of Size Change | Includes target names that could be directly or indirectly affected by a change in the size of the specified dataname. |
| Of Value Change | Includes target names that are directly or indirectly affected by a change in the value of the specified dataname. |
| Levels | Includes all data items that are affected within the specified number of indirect levels. |
| Direction | |
| All | Includes all occurrences of the requested target. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the requested target. |

| Field | Description |
|---|---|
| First | Searches from the top of the source file to the first occurrence of the requested target. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the requested target. |
| Options | |
| No Data Aliasing | Specifies the alias dataname. If you select by entering a slash (/), aliases for the specified dataname are ignored. These are the valid aliases: Parent (higher level group item), Child (lower level item), and Rename/Redefinition (renamed, redefined, or 88 level items). |
| IN-Clause | Displays the IN-clause Option pop-up used to restrict the source lines to be considered. |

# Break - Line Number Pop-up

Use the Break - Line Number pop-up, shown in Figure 44, to place Breakpoints before or after specified line numbers. This pop-up displays by selecting the Line action on the Test - Break Request pop-up.

**Figure 44 • Break - Line Number Pop-up**

```
                    Break - Line Number

To break on (a) specific line, specify a line number. Then press
Enter.

Line Number ____________________________________________________

Options
_  IN-Clause...
```

## Fields

| Field | Description |
|---|---|
| Line Number | Specifies a single line number. Line numbers are displayed in columns 1 through 6 on the Program View screen. This field is required. |
| IN-Clause Options | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered. |

# Break - Label/Paragraph Name Pop-up

Use the Break - Label/Paragraph Name pop-up, shown in Figure 45, to place a Breakpoint before or after a specified label name. This pop-up displays by selecting the Label action on the Test - Break Request pop-up.

**Figure 45 • Break - Label/Paragraph Name Pop-up**

```
                    Break - Label/Paragraph Name

To break on a label or paragraph name reference, specify
label/paragraph, direction, and options. Then press Enter.  For a
label/paragraph list, type a "pattern" (or blanks) in the Label/Para
Name area.

To break on all paragraphs, select 'Options', 3.

Label/Para Name ____________________________________________________

Direction                Options
2   1.  All              __  1.  None
    2.  Next                 2.  IN-Clause...
    3.  Previous             3.  Break on All Paragraphs
    4.  First
    5.  Last
```

## Fields

| Field | Description |
|---|---|
| Label/Para Name | Specifies any PROCEDURE DIVISION paragraph name or section name. The PROCEDURE and PROC literals can also be specified. This field is required.<br><br>To display the Selection List pop-up, leave the Label Name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |
| Direction | |
| All | Includes all occurrences of the requested target. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| First | Searches from the top of the source file to the first occurrence of the requested target. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the requested target. |

| Field | Description |
|---|---|
| Options | |
| None | No options are selected. |
| IN-Clause | Displays the IN-clause Option pop-up is used to restrict the source lines to be considered. |
| Break on All Paragraphs | Places a Breakpoint on the first executable statement following each paragraph or label in the current program. |

# Break - on Subset Pop-up

Use the Break - on Subset pop-up, shown in Figure 46, to place Breakpoints before or after specified subset names. This pop-up displays by selecting the Subset action on the Test - Break Request pop-up.

**Figure 46 • Break - COBOL Subset Pop-up**

```
                         Break - on Subset

To break on a  subset reference, specify subset and options, then
press Enter.  For a subset list, enter a "pattern" (or blanks) in
the Subset Name area.

Subset Name  ____________________________________________________

Direction                Options
2   1.  All              _  IN-Clause...
    2.  Next
    3.  Previous
    4.  First
    5.  Last
```

## Fields

| Field | Description |
|---|---|
| Subset Name | Specifies the name of a predefined COBOL subset. To display the Selection List pop-up, leave the Subset Name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). This field is required. |
| Direction | |
| All | Includes all occurrences of the requested target. |

| Field | Description |
|---|---|
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| First | Searches from the top of the source file to the first occurrence of the requested target. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the requested target. |
| IN-Clause Options | Displays the IN-clause Option pop-up is used to restrict the source lines to be considered. |

# Break - Program Name Pop-up

Use the Break - Program Name pop-up, shown in Figure 47, to place Breakpoints before or after subprograms. This pop-up displays by selecting the Program action on the Test - Break Request pop-up.

**Figure 47 • Break - Program Name Pop-up**

```
                         Break - Program Name

To break on a (sub)program, specify Program Name and options. Then
press Enter.  For a name list, type a "pattern" (or blanks) in the
Program Name area.

Program Name  ____________________________________________________

Direction                Options
2   1.  All              _  IN-Clause...
    2.  Next
    3.  Previous
    4.  First
    5.  Last
```

## Fields

| Field | Description |
|---|---|
| Program Name | Specifies the name of the main program or any nested program. To display the Selection List pop-up, leave the Program Name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). This field is required. |
| Direction | |
| All | Includes all occurrences of the requested target. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| First | Searches from the top of the source file to the first occurrence of the requested target. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the requested target. |
| IN-Clause Options | Displays the IN-clause Option pop-up is used to restrict the source lines to be considered. |

# Break - Pattern Pop-up

Use the Break - Pattern String pop-up, shown in Figure 48, to place break points before or after a specified pattern. This pop-up displays by selecting the Pattern action on the Test - Break Request pop-up.

**Figure 48 • Break - Pattern Pop-up**

```
                          Break - Pattern

To break on a statement containing a "character string", specify the
String and options. Then press Enter.

String  _------------------------------------------------------------

String Type              Location
1   1.  Simple           1   1.  Any
    2.  Hexadecimal          2.  Word
    3.  Text                 3.  Prefix
    4.  Picture              4.  Suffix

Direction                Options
2   1.  All              _  IN-Clause...
    2.  Next
    3.  Previous
    4.  First
    5.  Last
```

## Fields

| Field | Description |
|---|---|
| String | Specifies any PROCEDURE DIVISION paragraph name, section name, Assembler label, PL/I label, or Block name. The PROCEDURE and PROC literals can also be specified. This field is required. |
| String Type | Specifies the type of pattern string. The default is Simple. |
| Simple | Indicates a string that is not a hexadecimal, text, or a picture string. |
| Hexadecimal | Allows specific unprintable characters to be specified by giving the EBCDIC hexadecimal value. |
| Text | Indicates that a character sting may be entered regardless of upper or lowercase by using the text option. |

| Field | Description |
|---|---|
| Picture | Indicates that a string profile may be entered instead of exact characters. Nine special characters are defined by SmartTest for use in picture strings. These special characters may be combined with other characters. These are the special characters:<br>= Any character<br>¬ Any nonblank character<br>. Any nondisplay character<br># Any numeric character<br>- Any non-numeric character<br>@ Any alphabetic character (upper or lowercase)<br>< Any lowercase alphabetic character<br>> Any uppercase alphabetic character<br>$ Any special character (not alphabetic or numeric) |
| Location | Specifies the location to search for the pattern string |
| Any | Specifies any string anywhere in the source code. |
| Word | Specifies a string preceded and followed by any non-alphanumeric character (except hyphen). |
| Prefix | Specifies a word that begins with the specified string. |
| Suffix | Specifies a word that ends with the specified string. |
| Direction | Indicates the direction for the search. The default is All. |
| All | Searches for all occurrences of the requested target and displays the number found in the short/long message field. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| First | Searches from the top of the source file to the first occurrence of the requested target. |

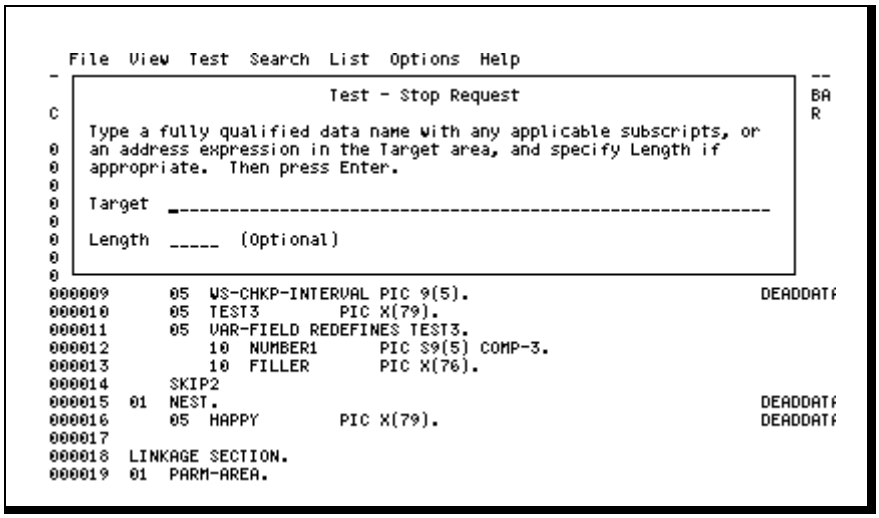
| Field | Description |
|---|---|
| Last | Searches backward from the bottom of the source file for the first occurrence of the requested target. |
| IN-Clause Options | Displays the IN-clause Option pop-up is used to restrict the source lines to be considered. |

# Test - Stop Request Pop-up

To set an address stop for a specified data item, follow this step:

- Select Test ▸ Stop. The Test - Stop Request pop-up, shown in Figure 49, displays. An address stop interrupts test execution before each modification of the specified data item.

**Figure 49 • Test - Stop Request Pop-up**



## Fields

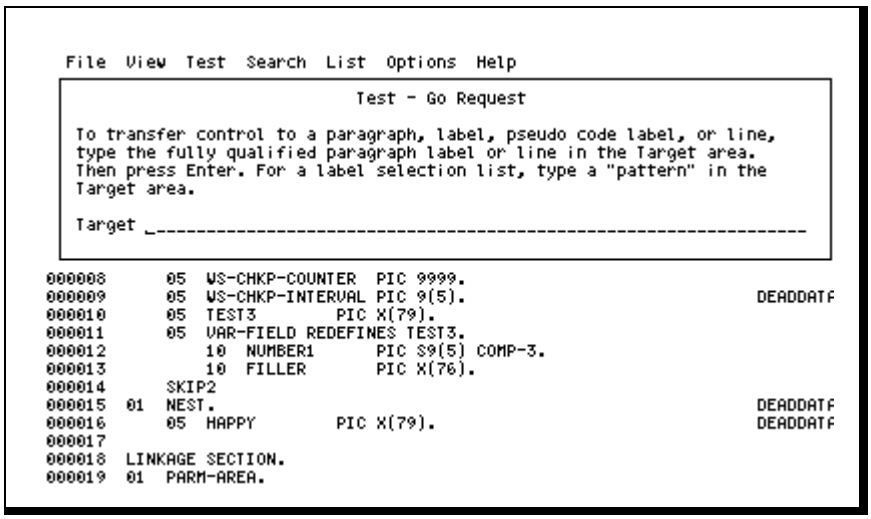
| Field | Description |
|---|---|
| Target | Specifies a data item for which an address stop is to be set. Subscript information may be specified using the standard COBOL syntax, (*subscript1, subscript2, ... subscriptn*). Left and right parentheses and at least one blank between subscripts are required. Commas between subscripts are optional. A subscript may be a numeric literal, an index data item, a numeric data item, or a numeric data item plus or minus a numeric literal. An address expression may also be entered. |
| Length | Sets an address stop for the data item specified to the length specified. The default length for the symbolic data items is the defined length of the data items. The default length for address expressions is 4. |

# Test - Go Request Pop-up

To transfer control to the statement containing the specified COBOL, Assembler, or PL/I procedure or label name, pseudo code label, or line, follow this step:

- Select Test ▸ Go. The Test - Go Request pop-up, shown in Figure 50, displays.

**Figure 50 • Test - Go Request Pop-up**




## Field

**Target.** Specifies a standard COBOL paragraph or section name, Assembler label, PL/I procedure or label, pseudo code label name, or COBOL or Assembler source line number.

# Test - Move Request Pop-up

To move the value contained in or represented by the first operand to the specified data item, follow this step:

- Select Test ▸ Move. The Test - Move Request pop-up, shown in Figure 51, displays. The value is converted to the proper format for the data item.

**Figure 51 • Test - Move Request Pop-up**

```
                    Test - Move Request

To move the value contained in or represented by one
operand to another, specify the Move operands.  Then press
Enter. To specify a figurative constant as the source
operand, select a Figurative Constant. For a list of data
names, type a "pattern" in the operand area(s).

Figurative Constant
__  1.  HIGH-VALUE(S)     _  ALL Figurative Constant
    2.  LOW-VALUE(S)
    3.  SPACE(S)
    4.  ZERO(ES)

Move  . . ------------------------------------------

   TO

Data Name ------------------------------------------
```

## Fields

| Field | Description |
|---|---|
| Figurative Constant | Specifies the number of the appropriate figurative constant to move HIGH-VALUE(S), LOW-VALUE(S), SPACE(S) or ZERO(ES) to the specified data item. |
| ALL Figurative Constant | Repeats a constant specified in the Move field to fill the specified data item. |
| Move | Specifies a COBOL dataname, qualified COBOL dataname, Assembler data area, PL/I variable name, level 77 data item defined within a block of pseudo code, numeric constant, COBOL reserved word (figurative constant), or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |
| Data Name | Specifies a COBOL dataname, qualified COBOL dataname, Assembler data area, PL/I variable name, level 77 data item defined within a block of pseudo code, or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |

# Test - Add Request Pop-up

To add the value contained in or represented by the first operand from the specified data item, follow this step:

- Select Test ▸ Add. The Test - Add Request pop-up, shown in Figure 52, displays. The value is converted to the proper format for the data item.

**Figure 52 • Test - Add Request Pop-up**

```
                    Test - Add Request

To add the value contained in or represented by one
operand to another, specify the Add operands. Then
press Enter.  For a list of data names, type a
"pattern" in the operand area(s).

Add . . . ________________________________________

   TO

Data Name ________________________________________
```

## Fields

| Field | Description |
|---|---|
| Add | Specifies a COBOL dataname, qualified COBOL dataname, Assembler numeric data area, PL/I variable name, level 77 data item defined within a block of pseudo code, numeric constant, COBOL reserved word (figurative constant), or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |
| Data Name | Specifies a COBOL dataname, qualified COBOL dataname, Assembler numeric data area, PL/I variable name, level 77 data item defined within a block of pseudo code, or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |

# Test - Subtract Request Pop-up

To subtract the value contained in or represented by the first operand from the specified data item, follow this step:

- Select Test ▸ Subtract. The Test - Subtract Request pop-up, shown in Figure 53, displays. The value is converted to the proper format for the data item.

**Figure 53 • Test - Subtract Request Pop-up**

```
                Test - Subtract Request

To subtract the value contained in or represented by
one operand from another, specify the Subtract
operands.  Then press Enter.  For a list of data
names, type a "pattern" in the operand area(s).

Subtract  ------------------------------------------

  FROM

Data Name ------------------------------------------
```

## Fields

| Field | Description |
|---|---|
| Subtract | Specifies a COBOL dataname, qualified COBOL dataname, Assembler numeric data area, PL/I variable name, level 77 data item defined within a block of pseudo code, numeric constant, COBOL reserved word (figurative constant), or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more). |
| Data Name | Specifies a COBOL dataname, qualified COBOL dataname, Assembler numeric data area, PL/I variable name, level 77 data item defined within a block of pseudo code, or a register 0 through 15. To display the Selection List pop-up leave this field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). |

# Test - Where Request Pop-up

To identify a storage location, follow this step:

- Select Test ▸ Where. The Test - Where Request pop-up, shown in Figure 54, displays. A message displays that indicates the offset and the area of the program is loaded.

**Figure 54 • Test - Where Request Pop-up**

```
  File  View  Test  Search  List  Options  Help
-                                                          -----------
                      Test - Where Request                 BA.TESTCOBA
C                                                          l ===> CSR
  To identify a storage location, type an address expression.
0 Then press Enter.
0
0 Address Expression _______________________________________
0
0
000006  WORKING-STORAGE SECTION.
000007  01  TESTER.
000008      05  WS-CHKP-COUNTER  PIC 9999.
000009      05  WS-CHKP-INTERVAL PIC 9(5).                      DEADDATI
000010      05  TEST3        PIC X(79).
000011      05  VAR-FIELD REDEFINES TEST3.
000012          10  NUMBER1      PIC S9(5) COMP-3.
000013          10  FILLER       PIC X(76).
000014      SKIP2
000015  01  NEST.                                               DEADDATI
000016      05  HAPPY        PIC X(79).                         DEADDATI
000017
000018  LINKAGE SECTION.
000019  01  PARM-AREA.
```

## Field

**Address Expression.** Specifies an address or a register number followed by 32 indirection indicators and/or offsets.

# Test - SmartTest Testpoint Generation Pop-up

To set breakpoints using an impact dataset, follow this step:

- Select Test ▸ Testpoints. The Test - ASG-SmartTest Testpoint Generation pop-up, shown in Figure 55, displays.

**Figure 55 • Test - SmartTest Generation Pop-up**

```
                    Test - ASG-SmartTest Testpoint Generation
Command ===> ____________________________________________________________

Enter Program, Attribute, Impact type and PDS, then ENTER to generate point
Press END to return.

    Generate Testpoints:

    Program(s)........... IGZEBST
    Data name Attribute. MOD     (MOD/REF)

    Impact type......... AUTO    (AUTO/ALL/EST/USER)
    PDS......... ''
    Member............... IGZEBST
```

## Fields

| Field | Description |
|---|---|
| Program(s) | Selects programs listed in the impact dataset. Enter the name of a single program or use an asterisk (*) to select them all. The program names must match the names of the programs to which breakpoints are being applied. |
| Data name Attribute | Specifies a dataname attribute. One of two entries is allowed, MOD or REF. MOD causes a breakpoint to be set wherever an impacted data item is modified. REF causes a breakpoint to be set wherever an impacted data item is referenced. |
| Impact type | Indicates the source of the impact dataset. Valid values are AUTO, ALL, EST, and USER, indicating these sources:<br>• AUTO = AutoChange<br>• ALL = Alliance<br>• EST = Estimate<br>• USER = User-supplied |

| Field | Description |
|---|---|
| PDS | Specifies the PDS containing the impact dataset to import. |
| Member | Lists the PDS member name containing the impact dataset for Impact Types ALL, EST, or USER. |

# SmartTest/Alliance Interface Pop-up

To link to Alliance and run a query script upon entry, follow this step:

- Select Test ▸ ASG-SmartTest Interface. The SmartTest/Alliance Interface pop-up, shown in Figure 56, displays.

**Figure 56 • SmartTest/Alliance Interface Pop-up**

```
                    ASG-SmartTest/ASG-Alliance Interface
Command ===> ____________________________________________________________
Please enter the following :

Alliance AKR.... . . ASG-SmartTest/ASG-Alliance Interface.
Application Name . . __________
Program Name ....... ____________________
Query Name.......    VIAPQIMS
```

## Fields

| Field | Description |
|---|---|
| Alliance AKR | Specifies the AKR containing the analysis information produced by Alliance for the specified application and program. |
| Application Name | Specifies name of the application containing the program to be tested. |

| Field | Description |
|---|---|
| Program Name | Specifies program to be tested. |
| Query Name | Specifies the name of the query script. Upon entering the SmartTest/Alliance Interface pop-up, the Query Name field is automatically filled with the name of a query script appropriate for the current environment. SmartTest selects a query script from the PDS specified in the Options - Script File Allocations pop-up, which is activated by selecting the Script file allocations item from the Options menu. You must have previously copied the VIAPALSC script and the VIAPQ* scripts into the PDS defined in the Options - Script File Allocations pop-up and edited the scripts appropriately. The VIAPALSC script and the VIAPQ* scripts are normally installed in the ESW CNTL library during the installation of SmartTest. |

# Test - Newcopy Request Pop-up

To load a new copy of the PPT module into the CICS environment, follow this step:

- Select Test ▸ CICS Newcopy. The Test - Newcopy Request pop-up, shown in Figure 57, displays. This pop-up is available only in the CICS environment.

**Figure 57 • Test - Newcopy Request Pop-up**

```
                    Test - Newcopy Request
Type a PPT module name to be loaded. Then press Enter.
Module Name _________
```

## Field

**Module Name.** Specifies the PPT module name to be loaded.

# 4 Search

This chapter describes the options available on the Search pull-down menu and contains these sections:


| Section | Page |
|---|---|
| Introduction | 84 |
| Search Pull-down | 85 |
| Search - Data Name Pop-up | 86 |
| Search - Label Name Pop-up | 89 |
| Search - Paragraph Name Pop-up | 91 |
| Search - Pattern String Pop-up | 93 |
| Search - Subset Name Pop-up | 96 |
| Search - Perform Range Name Pop-up | 99 |
| Search - Program Name Pop-up | 101 |
| Search - User Mark Name Pop-up | 103 |
| Search - Line Range Pop-up | 105 |
| Search - Any/Unknown Type Pop-up | 107 |
| Search - Branch Request Pop-up | 109 |
| Branch Previous Options Pop-up | 111 |
| Selection List Pop-ups | 111 |
| IN-Clause Option Pop-up | 112 |

# Introduction

The Search facility is used to find, highlight, scroll, print, punch, or exclude a specified target, which may be a dataname, label name, paragraph name, pattern, COBOL subset, perform range, program, user mark, or a line number.

## Find

Searches the source code for one or all occurrences of the specified target. The cursor is positioned at the first target in the specified direction, all occurrences found are highlighted, and tags are placed to the right of each statement indicating the type of target found.

## Highlight

Highlights source code lines containing the specified target. Previously highlighted lines remain unchanged. The cursor is positioned at the first target in the specified direction, all occurrences found are highlighted, and tags are placed to the right of each statement indicating the type of target found.

## Scroll

Positions the screen to the line or box containing the specified target. Previously highlighted lines remain unchanged. The screen is positioned at the first line containing the specified target.

## Print

Copies lines containing the specified target to the List file.

## Punch

Copies lines containing the specified target to the Punch file for subsequent processing.

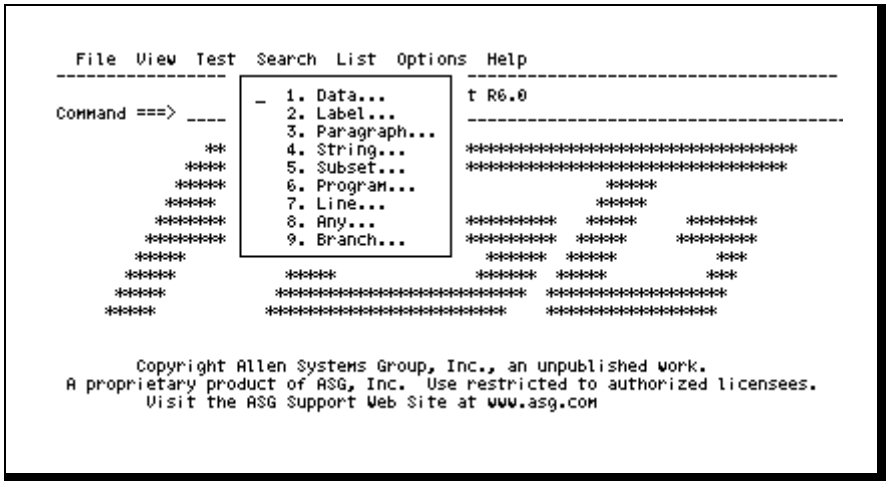
## Exclude

Searches the source code for one or all occurrences of the target and omits the resulting source code lines.

# Search Pull-down

Selecting Search on the action bar displays the Search pull-down. The Search pull-down is used to conduct a COBOL intelligent search of the source code for one or all occurrences of a specified target. Figure 58 shows the Search pull-down.

**Figure 58 • Search Pull-down**




## Actions

**Note:**

You cannot select actions on the Search pull-down unless there is an active program (i.e., a program has been opened).

| Action | Description |
|---|---|
| Data | Displays the Search - Data Name pop-up used to search for data items. |
| Label | Displays the Search - Label Name pop-up used to search for a specified label name. |
| Paragraph | Displays the Search - Paragraph Name pop-up used to search for statements in a paragraph. |
| String | Displays the Search - Pattern String pop-up used to search for a character string. |
| Subset | Displays the Search - COBOL Subset Name pop-up used to search for COBOL subsets. |

| Action | Description |
|---|---|
| Perform | Displays the Search - Perform Range Name pop-up used to search for statements within a perform range.<br>This action displays only if Insight is installed. |
| Program | Displays the Search - Program Name pop-up used to search for statements in subprograms. |
| Mark | Displays the Search - User Mark Name pop-up used to search for statements in a set of marked lines.<br>This action displays only if Insight is installed. |
| Line | Displays the Search - Line Range pop-up used to search for statements within line ranges. |
| Any | Displays the Search - Any/Unknown Type pop-up used to search for targets of unknown types. |
| Branch | Displays the Search - Branch Request pop-up used to branch to a specific location. |

# Search - Data Name Pop-up

To find, highlight, scroll, print, punch, or exclude datanames, follow this step:

- Select Search ▸ Data. The Search - Data Name pop-up, shown in Figure 59, displays.

**Figure 59 • Search - Data Name Pop-up**

```
                         Search - Data Name

Type a data name and select search options.  Then press Enter.  For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data name  ____________________________________________________

References            Indirect impact            Size change
1   1.  All           1   1.  None               levels . . . ___
    2.  Defs              2.  Of size change
    3.  Uses              3.  Of value change
    4.  Mods

Direction             Options                    Action
1   1.  All           _  No data aliasing        1   1.  Find
    2.  Next          _  IN-clause...                2.  Highlight
    3.  Previous                                     3.  Scroll
    4.  First                                        4.  Print
    5.  Last                                         5.  Punch
                                                     6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Data name | Specifies a dataname or qualified program source name. Dataname refers to any valid program reference for a data element. This field is required.<br><br>To display the Selection List pop-up, leave the dataname field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.<br><br>Datanames may be concatenated by placing a plus sign (+) between the datanames. |
| References | Restricts the search to only those datanames that are defined, used, or modified, by specifying the type of dataname reference. The default is All. |
| All | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default. |
| Defs | Includes definitions of the dataname in the DATA DIVISION. |
| Uses | Includes occurrences of the dataname where its value is being tested or used. |
| Mods | Includes occurrences of the dataname where its value is being set or modified. |
| Indirect Impact | Restricts the search to include occurrences of datanames that are indirectly affected by the specified dataname. |
| None | Includes only occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on value of the References field). This is the default. |
| Of size change | Includes datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. |
| Of value change | Includes datanames that are directly or indirectly affected by a change in the value of the specified dataname. |
| Size change levels | Specifies a number that identifies the depth of the indirect impact of a size change that is desired. When this field is used, all references that are affected within the specified level are shown. If left blank, all levels are searched. |

| Field | Description |
|---|---|
| Direction | Restricts the search to a specific direction or occurrence by specifying a direction. |
| All | Includes all occurrences of the dataname and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the dataname. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the dataname. |
| First | Searches from the top of the source file to the first occurrence of the dataname. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the dataname. |
| No data aliasing | Indicates that aliases for the specified dataname are ignored. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>If this field is left blank, aliases of the dataname are included. This is the default. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If this field is left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the dataname. |
| Find | Searches for one or all occurrences of the dataname. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. This is the default. |
| Highlight | Highlights source code lines containing the datanames. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the dataname. Previously highlighted lines remain unchanged. |

| Field | Description |
|---|---|
| Print | Copies lines containing the dataname to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the dataname to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the dataname from the display. |

# Search - Label Name Pop-up

To find, highlight, scroll, print, punch, or exclude a specified label name and all transfers of control to that label name, follow this step:

- Select Search ▸ Label. The Search - Label Name pop-up, shown in Figure 60, displays.

**Figure 60 • Search - Label Name Pop-up**

```
                        Search - Label Name

To find label name references, type a name and select search
options.  Then press Enter.  For selection list, type a pattern
(e.g. ABC*) in the name area.

Label name  ____________________________________________________

Direction               Options               Action
1   1.  All             _  IN-clause...       1   1.  Find
    2.  Next                                      2.  Highlight
    3.  Previous                                  3.  Scroll
    4.  First                                     4.  Print
    5.  Last                                      5.  Punch
                                                  6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Label name | Specifies any PROCEDURE DIVISION paragraph name section name, or PL/I label. The PROCEDURE and PROC literals can also be specified. This field is required.<br>To display the Selection List pop-up, leave the Label name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.<br>Label names may be concatenated by placing a plus sign (+) between the Label names. |
| Direction | Restricts the search to a specific direction or occurrence by specifying a direction. |
| All | Includes all occurrences of the label name and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the label name. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the label name. |
| First | Searches from the top of the source file to the first occurrence of the label name. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the label name. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the label name. |
| Find | Searches for one or all occurrences of the label name. This is the default. |
| Highlight | Highlights source code lines containing the label names. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the label name. Previously highlighted lines remain unchanged. |

| Field | Description |
|---|---|
| Print | Copies lines containing the label name to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the label name to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the label name from the display. |

# Search - Paragraph Name Pop-up

To find, highlight, scroll, print, punch, or exclude statements in a specified paragraph, follow this step:

- Select Search ▸ Paragraph. The Search - Paragraph Name pop-up, shown in Figure 61 displays.

**Figure 61 • Search - Paragraph Name Pop-up**

```
                    Search - Paragraph Name

To find all statements in a paragraph, type a name and select the
search options.  Then press Enter.  For a name list, type a pattern
(e.g. ABC*) in the name area.

Paragraph name  _------------------------------------------------

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Paragraph name | Specifies the name of a paragraph on which to perform the search.To display the Selection List pop-up, leave the Paragraph name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.Paragraph names may be concatenated by placing a plus sign (+) between the Paragraph names. This field is required. |
| Direction | Restricts the search to a specific direction or occurrence by specifying a direction. |
| All | Includes all occurrences of the paragraph name and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the paragraph name. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the paragraph name. |
| First | Searches from the top of the source file to the first occurrence of the paragraph name. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the paragraph name. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the paragraph name. |
| Find | Searches for one or all occurrences of the paragraph name. This is the default. |
| Highlight | Highlights source code lines containing the paragraph names. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the paragraph name. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the paragraph name to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |

| Field | Description |
|---|---|
| Punch | Copies lines containing the paragraph name to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the paragraph name from the display. |

# Search - Pattern String Pop-up

To find, highlight, scroll, print, punch, or exclude a specified pattern, follow this step:

- Select Search ▸ String. The Search - Pattern String pop-up, shown in Figure 62, displays.

**Figure 62 • Search - Pattern String Pop-up**

```
                    Search - Pattern String

Type a string and select search options.  Then press Enter.

String  _____________________________________________________________

String type            Location
1   1.  Simple         1   1.  Any
    2.  Hexadecimal        2.  Word
    3.  Text               3.  Prefix
    4.  Picture            4.  Suffix

Direction              Options              Action
1   1.  All            _  IN-clause...      1   1.  Find
    2.  Next                                    2.  Highlight
    3.  Previous                                3.  Scroll
    4.  First                                   4.  Print
    5.  Last                                    5.  Punch
                                                6.  Exclude
Begin column ___       End column ___
```

## Fields

| Field | Description |
|---|---|
| String | Specifies a string of alphanumeric or DBCS characters. It is not necessary to enclose the string in single or double quotes. All characters entered are used for the pattern search. This field is required. |
| String Type | Specifies the type of pattern string. |
| Simple | Indicates a character. This is the default. |

| Field | Description |
|---|---|
| Hexadecimal | Specifies that specific unprintable characters can be specified by giving the EBCDIC hexadecimal value. |
| Text | Specifies that a character sting can be entered regardless of upper or lowercase using the text option. |
| Picture | Specifies that a string profile can be entered instead of exact characters. Nine special characters are defined by SmartTest for use in picture strings. These special characters may be combined with other characters. These are the valid special characters:<br>= Any character<br>¬ Any nonblank character<br>. Any nondisplay character<br># Any numeric character<br>- Any non-numeric character<br>@ Any alphabetic character (upper or lowercase)<br>< Any lowercase alphabetic character<br>> Any uppercase alphabetic character<br>$ Any special character (not alphabetic or numeric) |
| Location | Specifies the location to search for the pattern string. |
| Any | Specifies any string anywhere in the source code. This is the default. |
| Word | Specifies a string preceded and followed by any non-alphanumeric character (except hyphen). |
| Prefix | Specifies a word that begins with the specified string. |
| Suffix | Specifies a word that ends with the specified string. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the string and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the string. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the string. |

| Field | Description |
|---|---|
| First | Searches from the top of the source file to the first occurrence of the string. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the string. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the string name. |
| Find | Searches for one or all occurrences of the string name. This is the default. |
| Highlight | Highlights source code lines containing the string names. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the string name. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the string name to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the string name to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the string name from the display. |
| Begin column | Specifies the column number where the search is to begin. The default is column 7. |
| End column | Specifies the column number where the search is to end. The default is column 66. |

# Search - Subset Name Pop-up

To find, highlight, scroll, print, punch, or exclude statements in COBOL subsets, follow this step:

- Select Search ▸ Subset. The Search - Name Subset pop-up, shown in Figure 63, displays.

**Figure 63 • Search - Name Subset Pop-up**

```
                    Search - Subset Name

Type a subset name and select search options.  Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

Subset name  _____________________________________________________

Direction              Options              Action
1   1.  All            _  IN-clause...      1   1.  Find
    2.  Next                                    2.  Highlight
    3.  Previous                                3.  Scroll
    4.  First                                   4.  Print
    5.  Last                                    5.  Punch
                                                6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Subset name | Specifies the name of a predefined COBOL subset.<br><br>To display a Selection List pop-up, leave the Subset name field blank, or enter a pattern and press Enter. The pattern may include wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.<br><br>Subset names may be concatenated by placing a plus sign (+) between the subset names.<br><br>These are the COBOL subsets:<br><br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br><br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br><br>A screen subset:<br><br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br><br>A tagged lines subset of tags appearing in columns 73 through 80. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the subset and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the subset. |

| Field | Description |
|---|---|
| Previous | Searches backward from the current cursor position to the previous occurrence of the subset. |
| First | Searches from the top of the source file to the first occurrence of the subset. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the subset. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the subset. |
| Find | Searches for one or all occurrences of the subset. This is the default. |
| Highlight | Highlights source code lines containing the subset. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the subset. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the subset to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the subset to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the subset from the display. |

# Search - Perform Range Name Pop-up

To find, highlight, scroll, print, punch, or exclude statements in a perform range, follow this step:

- Select Search ▸ Perform. The Search - Perform Range Name pop-up, shown in Figure 64, displays.

**Figure 64 • Search - Perform Range Name Pop-up**

```
                    Search - Perform Range Name

Type a perfrange name and select search options.  Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

Perfrange name  ___________________________________________________

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Perfrange name | Specifies the name specified in a PERFORM statement, or the name of any section contained in the DECLARATIVES. This field is required.<br>To display the Selection List pop-up, leave the Perfrange name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the perform range and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the perform range. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the perform range. |
| First | Searches from the top of the source file to the first occurrence of the perform range. |

| Field | Description |
|---|---|
| Last | Searches backward from the bottom of the source file for the first occurrence of the perform range. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the perform range. |
| Find | Searches for one or all occurrences of the perform range. This is the default. |
| Highlight | Highlights source code lines containing the perform range. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the perform range. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the perform range to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the perform range to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the perform range from the display. |

# Search - Program Name Pop-up

To find, highlight, scroll, print, punch, or exclude statements in subprograms, follow this step:

- Select Search ▸ Program. The Search - Program Name pop-up, shown in Figure 65, displays.

**Figure 65 • Search - Program Name Pop-up**

```
                        Search - Program Name

Type a (sub)program name and select search options.  Then press
Enter.  For a selection list, type a pattern (e.g. ABC*) in the name
area.

Program name  _____________________________________________________

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Program name | Specifies the name of the main program or any nested program. This field is required.<br>To display the Selection List pop-up, leave the Program name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.<br>Program names may be concatenated by placing a plus sign (+) between the program names. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the program name and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the program name. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the program name. |
| First | Searches from the top of the source file to the first occurrence of the program name. |

| Field | Description |
|---|---|
| Last | Searches backward from the bottom of the source file for the first occurrence of the program name. |
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the program name. |
| Find | Searches for one or all occurrences of the program name. This is the default. |
| Highlight | Highlights source code lines containing the program name. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the program name. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the program name to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the program name to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the program name from the display. |

# Search - User Mark Name Pop-up

To find, highlight, scroll, print, punch, or exclude sets of marked lines, follow this step:

- Select Search ▸ Mark. The Search - User Mark Name pop-up, shown in Figure 66, displays.

**Figure 66 • Search - Mark Name Pop-up**

```
                    Search - User Mark Name

Type a mark name and select search options.  Then press Enter.  For
a selection list, type a pattern (e.g. ABC*) in the name area.

Mark name  _____________________________________________________

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Mark name | Specifies a 1 to 10 alphanumeric character name given to a set or path. This field is required.<br>To display the Selection List pop-up, leave the Mark name field blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the mark name and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the mark name. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the mark name. |
| First | Searches from the top of the source file to the first occurrence of the mark name. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the mark name. |

| Field | Description |
|---|---|
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the mark name. |
| Find | Searches for one or all occurrences of the mark name. This is the default. |
| Highlight | Highlights source code lines containing the mark name. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the mark name. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the mark name to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the mark name to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the mark name from the display. |

# Search - Line Range Pop-up

To find, highlight, scroll, print, punch, or exclude a single line or range of lines, follow this step:

- Select Search ▸ Line. The Search - Line Range pop-up, shown in Figure 67, displays.

**Figure 67 • Search - Line Range Pop-up**

```
                         Search - Line Range

Type a line range (or line) and select search options.  Then press
Enter.

Line range  _____________________________________________________

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Line range | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6 of the Program View screen. This field is required. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the line range and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the line range. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the line range. |
| First | Searches from the top of the source file to the first occurrence of the line range. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the line range. |

| Field | Description |
|---|---|
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the mark name. |
| Find | Searches for one or all occurrences of the line range. This is the default. |
| Highlight | Highlights source code lines containing the line range. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the line range. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the line range to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the line range to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the line range from the display. |

# Search - Any/Unknown Type Pop-up

To find, highlight, scroll, print, punch, or exclude targets when the type is unknown, does not fit one of the predefined categories, or is a combination of predefined categories, follow this step:

- Select Search ▸ Any. The Search - Any/Unknown Type pop-up, shown in Figure 68, displays.

**Figure 68 • Search - Any/Unknown Type Pop-up**

```
                    Search - Any/Unknown Type

To find references to a target of "unknown" type (data, label,
etc.), type the target and select search options.  Then press Enter.

Target  _________________________________________________________

Direction                Options                Action
1   1.  All              _  IN-clause...        1   1.  Find
    2.  Next                                        2.  Highlight
    3.  Previous                                    3.  Scroll
    4.  First                                       4.  Print
    5.  Last                                        5.  Punch
                                                    6.  Exclude
```

## Fields

| Field | Description |
|---|---|
| Target | Specifies the name of the target. Target names may be concatenated by placing a plus sign (+) between the target names.For COBOL II or later programs, a dataname that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. This field is required. |
| Direction | Indicates the direction for the search. |
| All | Searches for all occurrences of the target and displays the number found in the short/long message field. This is the default. |
| Next | Searches forward from the current cursor position to the next occurrence of the target. |
| Previous | Searches backward from the current cursor position to the previous occurrence of the target. |
| First | Searches from the top of the source file to the first occurrence of the target. |
| Last | Searches backward from the bottom of the source file for the first occurrence of the target. |

| Field | Description |
|---|---|
| IN-clause | Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See "IN-Clause Option Pop-up" on page 112 for more information. |
| Action | Specifies the action to be performed on the target. |
| Find | Searches for one or all occurrences of the target. This is the default. |
| Highlight | Highlights source code lines containing the target. Lines that are already highlighted are not reset. |
| Scroll | Positions the screen to the first line containing the target. Previously highlighted lines remain unchanged. |
| Print | Copies lines containing the target to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up. |
| Punch | Copies lines containing the target to the Punch file for subsequent processing. The Punch file is processed on the Options - Log/List/Punch Definition pop-up. |
| Exclude | Omits source code lines containing occurrences of the target from the display. |

# Search - Branch Request Pop-up

To position the cursor to the specified target on the Program View screen, follow this step:

- Select Search ▸ Branch. The Search - Branch Request pop-up, shown in Figure 69, displays.

**Figure 69 • Search - Branch Request Pop-up**

```
                    Search - Branch Request

 To branch to another area of the program, select the Option desired.
 For Option 1, type the branch location (Target) information.  Then
 press Enter.  For a name selection list (for Target type 2, 3 or 4),
 type a pattern (e.g. ABC*) in the name field.

 Option
 1   1.  Branch to target
     2.  Return to previous "Branch to target" location
     3.  Branch to transfer(s) of control to cursor position

 Target name  ____________________________________________________

 Target type
 1   1.  None - use cursor
     2.  Label name
     3.  Perfrange name
     4.  Program name
```

## Fields

| Field | Description |
|---|---|
| Option | Specifies the type of branch to perform. The default is Branch to target. This field is required. |
| Branch to target | Positions the cursor to the name specified in the Target name field. If this Option is selected, the Target name field is required. This is the default. |
| Return to Branch location | Positions the cursor to the statement from which the previous branch occurred. |
| Display all transfers of control to cursor position | Redisplays the Program View screen with a long message stating to position the cursor to the desired branch location and press Enter. The cursor is positioned to the statement that branches to the specified branch location.<br>If more than one statement transfers control to the specified branch location, the Branch Previous Options pop-up displays to select the desired statement.<br>If you select this Option, the Target name and Target type fields are invalid. |

| Field | Description |
|---|---|
| Target name | Specifies the name of the target and is required if you selected Option 1. Enter a Perform Range name, label name, paragraph name, or section name. For COBOL II or later programs, a subprogram name can also be entered.<br><br>For Perform Ranges and label names, if the specified name is not fully qualified and there are multiple Perform Ranges or labels with the specified name, the cursor is positioned on the first statement found.<br><br>For COBOL II or later programs, a data item, label or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name.<br><br>To display the Selection List pop-up, leave the target name blank, or enter a pattern with wildcard characters (? for one character; * for zero or more characters), and select Target type 2, 3, or 4. On the Selection List pop-up, select the desired name with the S line command. See "Selection List Pop-ups" on page 111 for more information. |
| Target type | Specifies the number of the target type corresponding to the entry in the Target name field. This field is valid for Option 1 (Branch to target) only. |
| None - use cursor | Redisplays the Program View screen with a long message stating to position the cursor to the selected statement and press Enter. This is the default.<br><br>If the statement is a GO TO, PERFORM or internal CALL, the cursor is positioned to the target of that statement. If any other statement type is selected, the cursor is positioned to the next sequential statement to be executed. The PROCEDURE DIVISION label is located if the cursor is positioned outside the PROCEDURE DIVISION. |
| Label name | Positions the cursor to the specified paragraph or section name. If the specified name is not fully qualified and there are multiple paragraphs or sections with the specified name, the cursor is positioned to the first paragraph or section found. |
| Perfrange name | Positions the cursor to the specified Perform Range. If the specified name is not fully qualified and there are multiple Perform Ranges with the specified name, the cursor is positioned to the first Perform Range found. |
| Subprogram name | Positions the cursor to the specified nested program. For COBOL II and later programs, a data item, label, or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. |

# Branch Previous Options Pop-up

The Branch Previous Options pop-up, shown in Figure 70, is used to select a statement for branching. This pop-up displays when more than one statement transfers control to the specified branch location on the Search - Branch Request pop-up, or when you type `BRANCH PREVIOUS`.

**Figure 70 • Branch Previous Options Pop-up**

```
                          Branch Previous Options        9 STATEMENTS FOUND
Command ===> ____________________________________________ Scroll ===> CSR

Select the statement to be viewed.  Then press Enter.

S  Line    Source statement.
- ------ ------------------------------------------------------------------
_ 000292      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000406      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000428      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000516      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000581      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000760      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000857      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 000899      GO TO MAIN-ROUTINE-CHECK-ALL.
_ 001041      GO TO MAIN-ROUTINE-CHECK-ALL.
****************************** BOTTOM OF DATA ****************************
```

## Fields

| Field | Description |
|---|---|
| S line command | Selects the identified source statement and positions the cursor to that statement on the Program View screen. |
| Line | Specifies the source statement line number. |
| Source statement | Specifies the source code statement. |

# Selection List Pop-ups

Selection list pop-ups are available for the Data, Perform Range, Paragraph, Program, and Subset searches. To display the selection list, leave the name field blank or enter a pattern with wildcard characters (? for one character; * for zero or more characters) in the name field or the Start line/label field on various pop-ups.

**Note:**

Selection List pop-ups are not available for PL/I programs.

The selection list pop-ups are the same except for the type of list displayed. Figure 71 shows the Selection List - Data Names pop-up.

**Figure 71 • Selection List - Data Names Pop-up**

```
                         Selection List - Data Names          LINE 1 OF 15
Command ===> ____________________________________________ Scroll ===> CSR

S    Data name
-  ---------------------------------------------------------------------
_  DIBDBDNM
_  HAPPY (DEAD)
_  NEST (DEAD)
_  NUMBER1
_  PARM-AREA
_  PARM-CHARACTER
_  PARM-DEMO-CHAR
_  PARM-LENGTH
_  PARM-TEXT
_  RETURN-CODE
_  TESTER
_  TEST3
_  VAR-FIELD
_  WS-CHKP-COUNTER
_  WS-CHKP-INTERVAL (DEAD)
 ****************************** BOTTOM OF DATA   ************************
```

Type S to select the desired name from the list. The selected name(s) are placed in the name field of the requesting pop-up. Use the LOCATE command to scroll through the list of names. You can select more than one name on these pop-ups:

- Selection List - Data Names
- Selection List - Label/Paragraph Names
- Selection List - Subset Names
- Selection List - Program Names

# IN-Clause Option Pop-up

The IN-Clause Option pop-up, shown in Figure 72, is used to restrict the source lines to be considered for a search. This pop-up displays by specifying the IN-Clause option on a Search pop-up.

**Figure 72 • IN-Clause Option Pop-up**

```
                          IN-Clause Option

To restrict the source lines to be considered, type one or more
IN-Clause names.  Then press Enter.  For a selection list, type a
pattern in the appropriate name field.

Line range  . . . ________________________________________________
Subset name . . . ________________________________________________
Paragraph Name    ________________________________________________
Program name  . . ________________________________________________
```

## Fields

| Field | Description |
|---|---|
| Line range | Specifies a single line number or range of lines. |
| Subset name | Specifies the name of a predefined COBOL subset.<br><br>To display a Selection List pop-up, leave the Subset name field blank, or enter a pattern and press Enter. The pattern may include wildcard characters (? for one character; * for zero or more characters). See "Selection List Pop-ups" on page 111 for more information.<br><br>Subset names may be concatenated by placing a plus sign (+) between the subset names.<br><br>These are the valid COBOL subsets:<br><br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br><br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br><br>A screen subset:<br><br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br><br>A tagged lines subset of tags appearing in columns 73 through 80. |
| Paragraph Name | Specifies a PROCEDURE DIVISION paragraph or section name. The PROCEDURE and PROC literals can also be specified. |

| Field | Description |
|---|---|
| Perfrange name | Specifies the name specified in a PERFORM statement, or the name of any section contained in the DECLARATIVES. The search includes all statements that are executed as a result of the PERFORM statement. |
| Program name | Specifies the name of the main program or any nested program. The search includes all code contained in the program, including all programs physically nested inside the specified program. |

## Usage Notes

In most cases, any combination of IN-Clause options can be specified. However, some defined Mark names are incompatible with other logical sets of COBOL statements and the combinations are not allowed.

All fields on this pop-up are optional. If all fields are left blank, the search is not restricted (i.e., the entire program is searched).

# 5 List

This chapter describes the options available on the List pull-down menu and contains these sections:

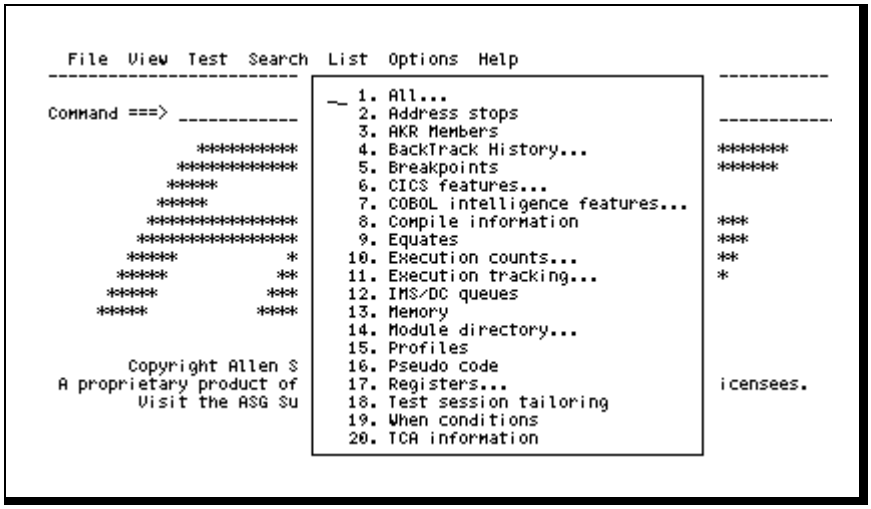

| Section | Page |
|---|---|
| List Pull-down | 117 |
| Test Facilities List Screen | 120 |
| Access Registers Screen | 121 |
| Address Stop Entry Screen | 122 |
| Environment AKR Directory Pop-up | 124 |
| List - BackTrack Variable History Pop-up | 127 |
| List - BackTrack Variable History Screen | 128 |
| Breakpoints List Screen | 129 |
| List - CALL Statements Pop-up | 132 |
| Compiler Options Screen | 134 |
| Execution Counts Screen | 135 |
| List - Equates Pop-up | 138 |
| Floating Point Registers Screen | 139 |
| Load Module Intercept List Pop-up | 140 |
| List - User Marks Pop-up | 141 |
| Memory Display Screen | 143 |
| Load Module Directory Screen | 152 |

| Section | Page |
|---|---|
| Module Map Screen | 153 |
| List - Perform Range Names Pop-up | 155 |
| Profile Data Set Member List Screen | 156 |
| List - Program/Subprogram Names Screen | 159 |
| Pseudo Code List Screen | 160 |
| General Registers Screen | 162 |
| List - COBOL Subsets Names Screen | 164 |
| Test Session Tailoring Screen | 165 |
| Execution Tracking Screen | 168 |
| When Conditions List Screen | 170 |
| List - CICS Features Pop-up | 172 |
| List - COBOL Features Pop-up | 173 |
| List - Execution Counts Pop-up | 174 |
| List - Execution Tracking Pop-up | 175 |
| List - Registers Pop-up | 177 |

# List Pull-down

Selecting List on the action bar displays the List pull-down, shown in Figure 73. The List pull-down is used to access pop-ups and screens that list information about the program and the testing environment.

**Figure 73 • List Pull-down**




## Actions

| Action | Description |
|---|---|
| All | Displays the Test Facilities List screen used to select the pop-up or screen that displays pertinent information about the program being tested and/or being viewed on the Program View screen. |
| Address stops | Displays the Address Stop Entry screen used to specify an absolute address and length of storage to be monitored during a test session. |
| AKR Members | Displays the Environment - AKR Directory pop-up used to list all programs in the current AKR, including concatenated ones. On this pop-up, you can also select a program for display in Program View. |
| BackTrack History | Displays the List - BackTrack History pop-up used to specify a data item for use in searching the BackTrack history. The search displays statements in the BackTrack history that modified the specified data item. |

| Action | Description |
|---|---|
| Breakpoints | Displays the Breakpoints List screen used to list all Breakpoints that have been set in the program being tested and/or viewed on the Program View screen. |
| CICS features | Displays the List - CICS Features pop-up used to access list screens and pop-ups applicable only to CICS.<br>See the *ASG-SmartTest CICS User's Guide* for examples and information about the CICS list screens. |
| COBOL intelligence features | Displays the List - COBOL Features pop-up used to access list screens for information about CALLS, PERFORMS, programs, subsets, and marks. |
| Compile information | Displays the Compiler Options screen used to display compiler information about the current program. |
| Equates | Displays the List - Equates pop-up used to show all equates for the program being tested or for the qualified program. |
| Execution counts | Displays the List - Execution Counts pop-up used to customize and access the Execution Counts screen. |
| Execution tracking | Displays the List - Execution Tracking pop-up used to customize and access the Execution Tracking screen. |
| IMS/DC queues | Displays the Processing Queue List screen used to list the types of IMS/DC processing queues.See the *ASG-SmartTest IMS User's Guide* for examples and information about the IMS/DC processing queue list screens. |
| Memory | Displays the List - Memory pop-up used to access the Memory Display screen. |
| Module directory | Displays the Load Module Directory screen used to list the modules residing in the specified libraries. |
| Profiles | Displays the Profile Data Set Member List screen used to copy from the dataset of another user, delete a member from the profile, select the environment, or write the current environment ISPF profile information. |
| Pseudo code | Displays the Pseudo Code List screen used to display all pseudo code statements in the program being tested and/or displayed on the Program View screen. |

| Action | Description |
|---|---|
| Registers | Displays the List - Registers pop-up used to access the Access Registers, Floating Point Registers and General Registers screens. |
| Test session tailoring | Displays the Test Session Tailoring screen used to selectively turn SmartTest features ON or OFF for a set of modules and/or programs. |
| When conditions | Displays the When Conditions List screen used to display the When Conditions in the program being tested and/or displayed on the Program View screen. |
| TCA information | Displays the File - TCA Test Plan Selection pop-up that is used to select a TCA option for setting up or modifying a test coverage plan, setting up a TCA test, or generating reports about the test results. Test Coverage Analysis is an optional feature in SmartTest.<br><br>**Note:**<br>This action is only available if you have the SmartTest TCA option installed. |

# Test Facilities List Screen

***To select the specified screen that displays pertinent information about the program being tested and/or viewed on the Program View screen***

**1** Select List ▸ All or type `LIST` on any SmartTest screen. The Test Facilities List screen, shown in Figure 74, displays.

**Figure 74 • Test Facilities List Screen**

```
                          Test Facilities List            TESTCOBA.TESTCOBA
Command ===> ____________________________________________ Scroll ===> CSR
ASG2343I USE THE DOWN COMMAND FOR ADDITIONAL TEST SESSION OPTIONS.
   Enter S before the keyword to select a list category

     Keyword    Description
   _ --------   ---------------------------------------------------------
   _ AKRMEM     List the AKR MEMBERS in the concatenated data bases
   _ ACCESS     List the ACCESS REGISTERS for the active program
   _ ADSTOP     List ADDRESS STOPS for the current test session
   _ BACKTRACK  List BACKTRACK variable history by variable name
   _ BREAKS     List BREAKpoints in the current program
   _ CALLS      List programs that are CALLed by the current program
   _ COMPILE    List COMPILEr or Assembler options for the current program
   _ COUNTS     List statement execution COUNTS and histogram
   _ EQUATES    List EQUATES for the current program
   _ FLOATING   List the FLOATING point registers for the active program
   _ INTERCEPTS List the load modules to be INTERCEPTed
   _ MEMORY     List an area of MEMORY in a dump format
   _ MODULES    List directory of load MODULES in current load libraries
   _ PERFORMS   List PERFORM ranges in the current COBOL program
   _ PROFILE    List the members of the PROFILE data set
   _ PROGRAMS   List PROGRAMS and subprograms in the current COBOL program
   _ PSEUDO     List PSEUDO code in the current program
```

**2** Type `S` in the line command area to the left of the Keyword field and press Enter to select a category.

## Usage Notes

EIB, FILE, LIMITS, and TABLES are only available if SmartTest-CICS is installed and is the currently selected environment. INTERCEPTS is not available in the CICS environment. QUEUE is only available if SmartTest-IMS is installed and is the currently selected environment. MARKS is available only if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

See the *ASG-SmartTest IMS User's Guide* for examples and information about the IMS/DC processing queue list screens. See the *ASG-SmartTest CICS User's Guide* for more information about CICS list screens.

# Access Registers Screen

To list the contents of the access registers for the program being tested, follow this step:

**1** Choose one of these options:

- Select ACCESS on the Test Facilities List screen.
- Type `LIST ACCESS` on any SmartTest screen.
- Select List ▸ Registers and specify Access Registers on the Test - Registers pop-up.

The Access Registers screen, shown in Figure 75, displays.

**Note:**

Register contents are displayed in hexadecimal and decimal formats and can be modified.

**Figure 75 • Access Registers Screen**

```
                                Access Registers            TESTCOBA.TESTCOBA -Q
Command ===> ______________________________________________ Scroll ===> CSR

PSW address: 800CFD1A   Instr: 07FE           AMODE: 31  Mode: PRIMARY-SPACE

Reg    Hex       GP Reg                                        Data
---- --------  --------                                 --------------------
AR00 007FB01F  00000000                                * << PROTECTED >>  *
AR01 00000000                                          *                  *
AR02 00000000                                          *                  *
AR03 00000000                                          *                  *
AR04 00000000                                          *                  *
AR05 00000000                                          *                  *
AR06 00000000                                          *                  *
AR07 00000000                                          *                  *
AR08 00000000                                          *                  *
AR09 00000000                                          *                  *
AR10 00000000                                          *                  *
AR11 00000000                                          *                  *
AR12 00000000                                          *                  *
AR13 00000000                                          *                  *
AR14 00000000                                          *                  *
AR15 00000000                                          *                  *
```

**2** Make any necessary modifications and press PF3/PF15 to exit.

## Fields

| Field | Description |
|---|---|
| PSW address | Specifies the Program Status Word address. |
| Instr | Specifies the current instruction. |
| AMODE | Specifies the current condition addressing mode. |

| Field | Description |
|---|---|
| Mode | Specifies the Address Space Control (ASC) mode. In PRIMARY mode, data that the program can access resides in the program's primary address space. In ACCESS-REGISTER mode, the data that the program can access resides in the address/data space indicated by the access registers. |
| Reg | Specifies the Access Registers 0 through 15. |
| Hex | Specifies the register contents in hexadecimal format. |
| GP Reg | Specifies the general register associated with the access register. |
| Data | Specifies the data at the target address. |

# Address Stop Entry Screen

***To specify an absolute address and length of storage to be monitored during a test session***

**1** Choose one of these options:

- Select ADSTOP on the Test Facilities List screen.
- Type `LIST ADSTOP` on any SmartTest screen.
- Select List ▸ Address Stop.

The Address Stop Entry screen, shown in Figure 76, displays.

**Figure 76 • Address Stop Entry Screen**

```
                              Address Stop Entry              TESTCOBA.TESTCOBA
Command ===> ______________________________________________ Scroll ===> CSR

Enter the address and length of storage areas to be monitored.  Use the STOP
command to enter this data automatically.  The stops are automatically deleted
at the end of the test session, or manually with a 'D'.

     Active            Length    Length
     Y/N/D   Address   in hex    in decimal   Description
     ------  --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
       -     --------  --------  -----------  --------------------------------
```

All storage areas listed on this screen are monitored by SmartTest and the test execution is interrupted before a specified storage area is actually updated. The Address Stop facility is unavailable during RUN NOMONITOR processing. See the "RUN Command" on page 390 for additional information.

**2** Press PF3/PF15 to exit.

## Fields

| Field | Description |
|---|---|
| Active Y/N/D | Indicates the address stop is to be enabled at this time. N disables the address stop for the current test. Type `D` over the Y or N to delete an entry. |
| Address | Specifies an absolute address in storage to be monitored during a test session. |
| Length in hex | Specifies the length of the storage area to monitor. This length is specified as a hexadecimal value. The equivalent decimal value of an entry is automatically displayed in the Length in decimal field. |

| Field | Description |
|---|---|
| Length in decimal | Specifies the length of the storage area to monitor. This length is specified as a decimal value. When an entry is made in this field, the equivalent hexadecimal value is automatically displayed in the Length in hex field. |
| Description | Specifies the description of the address stop. You can enter a maximum of 32 characters to describe the address stop or the reason for setting the address stop. This is an optional field. |

## Usage Notes

Address stops are automatically deleted at the end of the test session.

See the "STOP Command" on page 429 for more information about setting address stops.

# Environment AKR Directory Pop-up

***To list all members in the current AKR, including concatenated AKRs***

**1** Choose one of these options:

- Select AKRMEM on the Test Facilities List screen.
- Type `LIST AKRMEMBERS` on the Program View screen.
- Select List ▸ AKR Members.

The Environment - AKR Directory pop-up, shown in Figure 77, displays.

**Figure 77 • Environment - AKR Directory Pop-up**

```
                          Environment - AKR Directory      TESTCOBA.TESTCOBA
Command ===> ____________________________________________ Scroll ===> CSR

Select desired Program and press Enter.   Current environment is TSO

      Name     Lib  Alias of     Type     Lines    Date      Time    Jobname
_  ---------- --- ---------- --------- ------ --------- -------- --------
   $$GENERAL    1             C.A. SET  563232 12MAR1997 06:53:33 VIAJLC4
_  *SIMPLE1     1 SIMPLE      ST,PLI        11 21AUG1997 11:12:29 VIAJLCI
_  ABBEY-AL     1             AL             0 12NOV1996 06:15:09 VIAJLC4
_  APCONV       1             ST,ASM        19 22NOV1996 13:43:37 VIAJLC4
_  ASMBASE      1             ST,ASM       105 05NOV1997 08:18:34 VIAWGFPM
_  CD05A384     1             IN          2525 12NOV1996 07:49:07 VIAJLCA
_  CD05B        1 CD05B333    IN           563 31MAR1997 06:59:21 VIAJLC8
_  CD05B333     1             IN           563 31MAR1997 06:59:21 VIAJLC8
_  CD05C        1 CD05C384    ST           632 30OCT1997 15:13:14 VIAJLC4C
_  CD05C384     1             ST           632 30OCT1997 15:13:14 VIAJLC4C
_  CD10A        1 CD10A333    IN          1662 31MAR1997 06:59:33 VIAJLC8
_  CD10A333     1             IN          1662 31MAR1997 06:59:33 VIAJLC8
_  CD10B        1 CD10B179    IN           680 31MAR1997 06:59:38 VIAJLC8
_  CD10B179     1             IN           680 31MAR1997 06:59:38 VIAJLC8
_  CD110254     1             IN          1248 31MAR1997 06:59:44 VIAJLC8
```

## Fields

| Field | Description |
|---|---|
| Line command area | Selects a program. Type S in the line command area to the left of the Name field to display the selected program on the Program View screen. |
| Name | Specifies the name of the program in the AKR, from the PROGRAM-ID statement. If the analyzed program contains an ENTRY point, Name is the ENTRY point name. If the default name in the PROGRAM-ID statement was overridden on the File - Analyze Submit pop-up during the analyze, Name is the specified override name. |
| Lib | Specifies the ordinal number of the AKR that contains the program. Numbers are assigned in the order that the AKRs are specified on the Environment Selection and the System AKR Libraries pop-ups. |
| Alias of | Specifies the name of the program that contains the ENTRY point if the analyzed program contains an ENTRY point. If the default name in the PROGRAM-ID statement was overridden on the File - Analyze Submit pop-up during the analyze, Alias of is the specified override name. |
| Type | Indicates the type of analysis that was performed on the program. The type of analysis is specified on the File - Analyze Submit pop-up. |
| AL | Indicates that an Alliance analysis was performed. |
| ASM | Indicates that the program is an Assembler source program. |
| DA | Indicates that an Extended SmartDoc analysis with a COBOL compile was performed. |
| DC | Indicates that a SmartDoc analysis with a COBOL compile was performed. |
| DS | Indicates that a SmartDoc analysis was performed. |
| DX | Indicates that an Extended SmartDoc analysis was performed. |
| EN | Indicates that an Encore analysis was performed. |
| IN | Indicates that an Insight analysis was performed. |
| RC | Indicates that a Recap analysis was performed. |

| Field | Description |
|---|---|
| SQ | Indicates that a SmartQuest analysis was performed. |
| ST | Indicates that a SmartTest analysis was performed. |
| STX | Indicates that an Extended SmartTest analysis was performed. |
| INTERNAL | Indicates that the member contains logic segment information. The Name field contains $$SEGMENTS or the name assigned to the member using the Rename function. The $$SEGMENTS member contains the logic segment information. This is an Encore feature. |
| METRICS | Indicates that the member contains metrics information. The Name field contains $$METRIC or the name assigned to the metrics using the Rename function. The $$METRIC member contains metrics that have been calculated for the programs in this AKR. |
| PLI | Indicates that the program is a PL/I source program. |
| PROFILE | Indicates that the member contains profile information. Generally, the member name is the user ID for which the profile was created. Profile information is automatically saved when the SmartTest Session Tailoring screen is used to specify the testing environment options. Once program level testing options are specified on the Session Tailoring screen (and saved in the AKR profile member), the profile member is used by SmartTest when a test session is initiated. A profile is only used (and updated) by the user for which it is created. Any modifications to the Session Tailoring screen are automatically reflected in the profile member. |
| Lines | Specifies the number of lines in the program. |
| Date | Specifies the date on which the program was analyzed. |
| Time | Specifies the time at which the program was analyzed. |
| Jobname | Specifies the JOB NAME used to analyze the program. |

# List - BackTrack Variable History Pop-up

***To select a variable that has been modified while in BackTrack recording mode***

**1** Choose one of these options:

- Select BACKTRACK on the Test Facilities List screen
- Type `LIST BACKTRACK` without specifying a variable on the Program View screen
- Select List ▸ BackTrack History.

The List - BackTrack Variable History pop-up, shown in Figure 78, displays.

**Figure 78 • List - BackTrack Variable History Pop-up**

```
                 List - BackTrack Variable History

Type a fully qualified data name with any applicable subscripts
in the Target area. For a list of data names, type a "pattern"
in the Target area. Then press Enter.

Target  _____________________________________________________
```

**2** Type a fully-qualified dataname in the Target field and press Enter.

## Fields

**Target.** Enter the name of the variable to select. You can use a question mark (?) or an asterisk (*) to enter a pattern to select from a list of variables in the program. For example, this command finds all occurrences of variables that begin with END-FILE.

```
END-FILE*
```

## Usage Notes

After selecting a variable, the BackTrack facility lists all statements in the execution history that modified the variable. This provides a quick look at statements that may be causing unexpected data values. You may then select the suspect statement to be viewed in the context of the program.

# List - BackTrack Variable History Screen

The List - BackTrack Variable History screen, shown in Figure 79, is used to view statements in the execution history that modified the selected data item. You can use this screen to select a statement for viewing in the context of the program. This screen displays by selecting a variable on the List - BackTrack Variable History pop-up.

To view a statement in context within the program source, select a statement on this screen. This initiates the BackTrack review mode allowing you to step backward and forward through the execution history of the program and view data values as they existed when the statement was executed.

**Figure 79 • List - BackTrack Variable History Screen**

```
                    List - BackTrack Variable History    VIASUB37.VIASUB37 -A
Command ===> ______________________________________________ Scroll ===> CSR

Variable: COMPARISON-CODE                 Desc: PIC 9
   Index: ____________________________________________________________

S RELMOD VALUE                LINE   STATEMENT CONTENTS
- ------ -------------------- ------ ----------------------------------------
******************************** TOP OF DATA ********************************
  VIAMRG37.VIAMRG37
_      -3 .                   000187     MOVE ZEROS TO   COMPARISON-KEY-1, COMP
  VIASUB37.VIASUB37
_      -2 0                   000038         MOVE 1 TO COMPARISON-CODE
_      -1 1                   000038         MOVE 1 TO COMPARISON-CODE
_       0 1                   >>>>>>    <CURRENT VALUE OF VARIABLE>
****************************** BOTTOM OF DATA ******************************


+-------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK                PROGRAM: VIASUB37  BACKTRACK:  |
|  STMT: 000023  OFF: 000284  AMODE: 31      MODULE: VIASUB37      RECORDING |
|SOURCE: IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY                      |
+-------------------------------------------------------------------------+
```

## Fields

| Field | Description |
|---|---|
| Variable | Specifies the name of the variable selected on the List - BackTrack Variable History pop-up. |
| Desc | Specifies the PIC clause describing the selected variable. |
| Index | Specifies the occurrence number of the selected variable if the variable is indexed or subscripted. This field is blank if the selected variable is not indexed or subscripted. |
| Line command area | Identifies the line command area. Type S to the left of the line to select a statement. The statement selected displays on the Program View screen in the context of the program. |

| Field | Description |
|---|---|
| RELMOD | Specifies the relative number of the statement that modified the selected variable. The relative number of the current value is 0. As you step back through the execution history, the relative number of the most recently executed statement that modified the selected variable is -1; the next most recent statement is -2, and so forth. |
| VALUE | Specifies the contents of the selected variable at entry to the statement shown. This field shows a history of the values the variable had when each statement was executed. |
| LINE | Specifies the source program line number of the statement modifying the selected variable. |
| STATEMENT CONTENTS | Specifies the source statement that modified the selected variable. |

### Usage Notes

Use the List - BackTrack Variable History screen to display the set of values for a variable during program execution. This screen quickly shows which statements might be causing an invalid value in a data item.

## Breakpoints List Screen

***To list all Breakpoints that have been set in the program being tested and/or viewed on the Program View screen***

**1** Choose one of these options:

- Select BREAKS on the Test Facilities List screen
- Type `LIST BREAKS` command on the Program View screen
- Select List ▸ Breakpoints.

The Breakpoints List screen, shown in Figure 80, displays.

**Figure 80 • Breakpoints List Screen**

```
                                Breakpoints List            VIASUB37.VIASUB37 -A
Command ===> ______________________________________________ Scroll ===> CSR

         Set pseudo  . . ON       Pseudo active: 1       Inactive: 0
         Set breaks  . . ON       Breaks active: 1       Inactive: 0
         Set whens . . . ON       Whens  active: 0       Inactive: 0

$ Line   Pseudo Code (breakpoints highlighted)                         A Count
- ------ ------------------------------------------------------------ - -----
_ 000022  PROCEDURE DIVISION USING COMPARISON-KEY COMPARISON-CODE.
_ '''''1      BREAK.                                                   Y 000003
_ >>>>>>      IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY
  ************************** BOTTOM OF DATA ***************************
```

The status of the SET PSEUDO, SET BREAKS, and SET WHENS modes are displayed and can be changed if desired. The number of active and inactive breakpoints, pseudo code, and When conditions are also indicated.

**2** Enable and disable any or all breakpoints.

**3** Press PF3/PF15 to exit.

## Fields

| Field | Description |
|---|---|
| Set pseudo | Enables pseudo code when this mode is ON. If this mode is OFF, any attempt to execute Pseudo code is ignored, including Breakpoints regardless of the values in the Set breaks or the Breaks active/Inactive fields. Breakpoints and When Conditions are considered to be pseudo code. |
| Set breaks | Enables the Breakpoints facility when this mode is ON. If this mode is OFF, all Breakpoints are ignored. If Set pseudo is OFF, Breakpoints are ignored regardless of the Set breaks mode. |
| Set whens | Enables the When Conditions facility when this mode is ON. When this mode is OFF, all When Conditions are ignored. If Set pseudo is OFF, When Conditions are ignored regardless of the Set whens mode. |
| Pseudo active/ Inactive | Indicates the number of pseudo code lines in the program that are enabled or disabled. |
| Breaks active/ Inactive | Indicates the number of Breakpoints that are enabled or disabled. |
| Whens active/ Inactive | Indicates the number of When Conditions that are enabled or disabled. |

| Field | Description |
|---|---|
| S | Displays the Program View screen with the specified line at the top of the screen. Typing `D` in this field deletes the corresponding pseudo code statement. |
| Pseudo Code (breakpoints highlighted) | Displays each block of pseudo code containing a BREAK statement, as well as the source code lines preceding and following the pseudo code. A dashed line separates each block of pseudo code that displays. The rest of the pseudo code that does not fit on the screen can be viewed by scrolling to it. |
| A | Activates the BREAK statement so that it is executed when the program is tested. If the Set pseudo or Set breaks fields contain OFF, the BREAK statement is ignored even if Y is entered in this field. Typing `N` in this field disables the BREAK statement so that it is ignored when the program is tested. The default is Y for this field. |
| Count | Indicates the number of times the BREAK statement has been executed during the test session. |

# List - CALL Statements Pop-up

***To display the programs that are CALLed by the current program***

**1** Choose one of these options:

- Select CALLS on the Test Facilities List screen.
- Type `LIST CALLS` on the Program View screen.
- Select List ▸ COBOL intelligence features and selecting Calls on the List - COBOL Features pop-up.

The List - CALL Statements pop-up, shown in Figure 81, displays.

**Figure 81 • List - CALL Statements Pop-up**

```
                          List - CALL Statements                    1 CALL
Command ===> ____________________________________________ Scroll ===> CSR

Select the program CALL(s) to be viewed.  Then press Enter.

$    Called program                                                Mode
-   ------------------------------------------------------------ --------
_    'TESTCOBB'                                                   STATIC
*************************** BOTTOM OF DATA ******************************
```

**2** Select the program you want to view and press Enter.

## Fields

| Field | Description |
|---|---|
| S | Selects an internal CALLed subprogram for viewing. Type S in this field to redisplay the Program View screen with related CALL statements highlighted and tagged. |
| Called program | Specifies the name of the CALLed subprogram. |
| Mode | Specifies the type of CALL. These are the valid values: |
| STATIC | Specifies that the object of the CALL is a literal and the associated program is link-edited into the same load module as the CALLing program. |
| DYNAMIC | Specifies that the object of the CALL is an identifier and is determined at run time. Programs referenced may or may not be link-edited into the same load module as the CALLing program. |
| INTERNAL | Specifies a CALL to a nested source subprogram. |

## Usage Notes

When this pop-up is first displayed, the short message field indicates the number of CALLs listed. The rest of the CALLs that do not fit on the screen can be viewed by scrolling to them.

The LPRINT * command can be entered on the List - CALL Statements screen to copy the screen contents to the List file.

# Compiler Options Screen

### *To display compiler options for the program being monitored*

**1** Choose one of these options:

- Select COMPILE on the Test Facilities List screen
- Type `LIST COMPILE` on any SmartTest screen
- Select List ▸ Compile information.

The Compiler Options screen, shown in Figure 82, displays.

**Figure 82 • Compiler Options Screen**

```
                                Compiler Options
Command ===> ____________________________________________ Scroll ===> CSR

Program   : TESTCOBA
Compiler  : COBOL II Rel. 3.2          Analyze Date: 05JAN2000 Time: 13:01:31
Optimizer : NONE                       Compile Date: 05JAN2000 Time: 13:01:31

Compiler Options:
  ADV
  APOST
NOAWO
  BUFSIZE(4096)
NOCMPR2
NOCOMPILE(S)
  DATA(31)
NODBCS
NODECK
NODUMP
NODYNAM
NOEXIT
NOFASTSRT
NOFDUMP
  FLAG(I)
NOFLAGMIG
```

**2** Complete the compiler options and press Enter.

## Fields

| Field | Description |
|---|---|
| Program | Specifies the name of the current program. |
| Compiler | Specifies the name of the compiler used for the current program. |
| Optimizer | Specifies the type of optimization (if any) used for the current program. |
| Analyze Date | Specifies the date that the current program was analyzed. |
| Time | Specifies the time that the current program was analyzed. |

| Field | Description |
|---|---|
| Compile Date | Specifies the date that the current program was compiled. |
| Time | Specifies the time that the current program was compiled. |
| Compiler Options | Lists the options used by the compiler for the current program. |
| Optimizer Options | Lists the options used by an independent optimizer (if any) used for the current program. |
| CICS SYSID | Specifies the SYSID that represents the CICS region from which the current program was loaded (CICS only). |

# Execution Counts Screen

***To display the current execution count and a histogram that indicates the relative count for each executable PROCEDURE DIVISION statement, Assembler source statement, or PL/I statement***

**1** Choose one of these options:

- Select COUNTS on the Test Facilities List screen.
- Type `LIST COUNTS` on any SmartTest screen.
- Select List ▸ Execution counts.

The Execution Counts screen, shown in Figure 83, displays.

**Figure 83 • Execution Counts Screen**

```
                                Execution Counts                 SORTED BY LINE
Command ===> ______________________________________________ Scroll ===> CSR

VIAMRG37  Total STMTS:   209  100%     VIAMRG37  Execution count: 38
       Executed STMTS:   38    18%                Highest count: 1
     Unexecuted STMTS:   171   82%            Highest count line: 164

$ LINE   COUNT   %.........50%........100%  STMTS SORTED BY LINE NUMBER
- ------ ------- ------------------------- ----------------------------------
_ 000164       1 ************************* PROCEDURE DIVISION.
_ 000165       1 *************************     PERFORM 1000-INITIALIZE THRU
_ 000167       1 *************************     PERFORM 2000-PROCESSING-LOOP THF
_ 000170       -                               PERFORM 9000-TERMINATION THRU
_ 000172       -                               IF END-FILE-COUNT = 0 THEN
_ 000173       -                                  MOVE +8 TO ABEND-CODE
_ 000174       -                                  PERFORM 9999-ABEND-IT.
_ 000175       -                               MOVE +0 TO RETURN-CODE.
_ 000176       -                               GOBACK.
_ 000178       1 *************************     OPEN INPUT  INFILE1 INFILE2 INFI
+-------------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK                PROGRAM: VIASUB37  DATE: 04OCT2000 |
|  STMT: 000023  OFF: 000284  AMODE: 31      MODULE: VIASUB37  TIME: 09:25:33  |
|SOURCE: IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY                           |
+-------------------------------------------------------------------------------+
```

All executable source statements are shown on this screen. Statements that do not fit on the screen can be displayed by scrolling to them. Execution Counts are not captured for areas of programs that SmartTest is not monitoring, such as areas executed after a RUN NOMONITOR request.

**2** Press PF3/PF15 to exit.

See the "RUN Command" on page 390 for additional information.

Source statements on the Execution Counts screen may be displayed sorted in order of line number, in order of ascending counts, or in order of descending counts. Sort order is set by using the LINE, ASCENDING, or DESCENDING operands. The default sort order is by line number. See the "LIST Command" on page 316 for more information about setting the sort order.

**Note:**

The Counts Active field on the Session Tailoring screen is used to enable the Statements Counts facility. Statement counts are automatically monitored when the Counts Active field contains YES or when there is an active WHEN for the program.

## Fields

| Field | Description |
|---|---|
| Total STMTS | Specifies the number and percentage of COBOL verbs, Assembler instructions, or PL/I verbs in the program being tested. |
| Executed STMTS | Specifies the number and percentage of COBOL verbs, Assembler instructions, or PL/I verbs that have been executed in the program being tested. |
| Unexecuted STMTS | Specifies the number and percentage of COBOL verbs, Assembler instructions, or PL/I verbs that have not been executed in the program being tested. |
| Execution count | Specifies the total COBOL verbs, Assembler instructions, or PL/I verbs that have been executed in the program being tested. This total reflects the count of COBOL verbs, Assembler instructions, or PL/I verbs that have been executed multiple times. |
| Highest count | Represents the execution total of the COBOL verb, Assembler instruction, or PL/I verb that was executed the most in the program being tested. |
| Highest count line | Specifies the statement number containing the COBOL verb, Assembler instruction, or PL/I verb with the highest execution count. |

| Field | Description |
|---|---|
| S | Displays the Program View screen with the specified line at the top of the screen. |
| LINE | Specifies the statement number containing the COBOL verb, Assembler instruction, or PL/I verb that was executed (or that can be executed). |
| COUNT | Specifies the number of times the statement has been executed. |
| %...50%...100% | Specifies the histogram that shows the execution count as a percentage of the statement with the highest execution count. |
| STMTS SORTED BY *XXXX* | Specifies the source statement containing the COBOL verb, Assembler instruction, or PL/I verb that was executed (or that can be executed). Scrolling the screen to the right displays any portion of the statement that does not fit on the screen. Counts may be sorted by line, ascending count, or descending count. (See the "LIST Command" on page 316 for information about setting the sort order). |

## Usage Notes

The LPRINT * command can be entered on the Execution Counts screen to copy the screen contents to the List file.

For COBOL II and later optimized programs, the optimization phase of the compile may place PERFORMed paragraphs in-line causing repeated groups of out-of-sequence line numbers when statement counts are requested sorted by line (the default).

# List - Equates Pop-up

***To display all equates for the program being tested and/or displayed on the Program View screen***

**1** Choose one of these options:

- Select List ▸ Equates.
- Select EQUATES on the Test Facilities List screen.
- Type `LIST EQUATES` on any SmartTest screen.

The List - Equates pop-up, shown in Figure 84, displays.

**Note:**

The first time this pop-up displays, the short message field indicates the number of equates listed.

**Figure 84 • List - Equates Pop-up**

```
                                 List - Equates
Command ===> ____________________________________________ Scroll ===> CSR

   Name     Program           Substitution String     Columns 001-053
---------- ---------- ----------------------------------------------------
A          TESTCOBA   TEST3
****************************** BOTTOM OF DATA ******************************
```

**2** Press PF3/PF15 to exit.

## Fields

| Field | Description |
|---|---|
| Name | Specifies the name assigned using the EQUATE command. |
| Program | Specifies the program for which the equated name was defined. |
| Substitution String | Specifies the data represented by the equated name. This data is automatically substituted for the equated name prior to the execution of a SmartTest command that uses the equated name. The substitution string can be modified by typing over it. The right-most column contains a greater than (>) symbol when the substitution string extends the screen column capability.<br>Typing blanks over the substitution string deletes the equate. |

## Usage Notes

Use the ISPF RIGHT and LEFT commands to scroll the screen so all of the substitution string can be viewed.

Use the LPRINT * command on the Equates List screen to copy the screen contents to the List file.

# Floating Point Registers Screen

***To display the floating point registers for the program being tested***

**1** Choose one of these options:

- Select List ▸ Registers to display the List - Registers pop-up.
- Select FLOATING on the Test Facilities List screen.
- Type `LIST FLOATING` on any SmartTest screen.

**2** Select Floating Pt Registers on the List - Registers pop-up. The Floating Point Registers screen, shown in Figure 85, displays.

**Figure 85 • Floating Point Registers Screen**

```
                          Floating Point Registers        TESTCOBA.TESTCOBA -Q
Command ===> ____________________________________________ Scroll ===> CSR

        Register        Hex             Mantissa          Exponent
        --------  ----------------  -----------------       ---
            0     0000000000000000                +0.0     E +00
            2     0000000000000000                +0.0     E +00
            4     0000000000000000                +0.0     E +00
            6     0000000000000000                +0.0     E +00

+-----------------------------------------------------------------------------+
|STATUS: BREAK AT END OF TEST SESSION      PROGRAM: IGZEBST   DATE: 04OCT2000 |
|  STMT: 000244  OFF: 00035A  AMODE: 31     MODULE: DEADCODE  TIME: 08:46:21  |
|SOURCE: BCR   15,R14                          HEX: 07FE                      |
|R0-7 00000000 000B1210 00000000 00000000 00000000 00000000 00000000 00000000 |
|R8-F 00000000 00000000 00000000 00000000 00000000 000AFFB0 000C1976 00000004 |
+-----------------------------------------------------------------------------+
```

**Note:**

Register contents are displayed in both hexadecimal and decimal formats.

## Fields

| Field | Description |
|---|---|
| Register | Lists each floating point register. |
| Hex | Specifies the register contents in hexadecimal format. |
| Mantissa | Specifies the fraction portion of the number. This number may be changed by typing over it. |
| Exponent | Specifies the exponent, or characteristic, of the number. This number may be changed by typing over it. |

## Usage Notes

You can change values for each register by typing the new values in the Mantissa and Exponent fields. Also, you can enter a specific hexadecimal value into a floating point register on the Program View screen during program execution by typing `SET FLOAT ON` and entering the desired value in the floating point area within the status box.

# Load Module Intercept List Pop-up

To list load modules that may be intercepted by SmartTest for testing, follow this step:

- Select INTERCEPTS on the Test Facilities List screen or type `LIST INTERCEPTS` on any SmartTest screen. The Load Module Intercept List pop-up, shown in Figure 86, displays.

**Figure 86 • Load Module Intercept Pop-up**

```
                         Load Module Intercept List
Command ===> ____________________________________________________________

Type the names of the load modules to be dynamically intercepted.
Then press Enter.

                 Load                              Load
                 Module                            Module
                 --------                          --------
             1   --------                       2  --------
             3   --------                       4  --------
             5   --------                       6  --------
             7   --------                       8  --------
             9   --------                      10  --------
            11   --------                      12  --------
            13   --------                      14  --------
            15   --------                      16  --------
            17   --------                      18  --------
            19   --------                      20  --------
            21   --------                      22  --------
            23   --------                      24  --------
```

A load module that is linked, attached, or called can be entered on the Load Module Intercept List pop-up. Entries must be made before the start of the test.

**Note:**

The Load Module Intercept List pop-up is not available in the CICS environment.

### Fields

**Load Module.** Specifies the name of the load module. You can enter a maximum of 24 load modules to be intercepted by SmartTest. Entries with wildcard characters are not permitted on this pop-up. It is necessary to specify exactly which modules will be intercepted.

### Usage Notes

When running in MONITOR mode it is not necessary to list load modules on the Load Module Intercept List pop-up since SmartTest knows when an external module is called and maintains control of the test session. Load modules must be listed on the Load Module Intercept List pop-up when running in NOMONITOR mode. In NOMONITOR mode SmartTest is not controlling the test and cannot determine when control is transferred from one load module to another. See the *ASG-SmartTest for COBOL and Assembler User's Guide* and the *ASG-SmartTest PLI User's Guide* for more information about intercepts.

Programs that are ATTACHed by another program can only be tested by placing an entry in the Load Module Intercept List pop-up.

## List - User Marks Pop-up

**Note:**

The List - User Marks pop-up is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

***To display the List - User Marks pop-up***

**1** Choose one of these options:

- Select MARKS on the Test Facilities List screen.
- Type `LIST MARKS` on any SmartTest screen.
- Select List ▸ COBOL Features and then select Marks on the List COBOL features pop-up.

The List - User Marks pop-up, shown in Figure 87, displays.

**Figure 87 • List - User Marks Pop-up**

```
                              List - User Marks
Command ===> ____________________________________________ Scroll ===> CSR

 C : Copy set/path 1 to set/path 2.   D : Delete existing set/path 1.
 M : Merge set/path 1 with set 2.     R : Rename set/path 1 to set/path 2.
 S : Select the mark to be viewed.

  Set/path 1      Set/path 2      Comments
_ ---------- ---- ---------- --------------------------------------------
****************************** BOTTOM OF DATA ******************************
```

2 Use the line commands to complete these functions:

- Display a directory or list of all existing marks
- Change comments
- Merge sets
- Copy, delete, and rename sets or paths

3 Press PF3/PF15 to exit.

## Fields

| Field | Description |
|---|---|
| Line command area | Specifies the desired function. Enter the appropriate letter for the function to the left of the Set/path 1 field, then complete the required fields. The valid functions are listed at the top of the screen. |
| C | Copies a set or path. Enter the target of the copy in the Set/path 2 field. |
| M | Merges sets. Enter the target of the merge in the Set/path 2 field. |
| S | Selects an item for viewing the screen from which the List - User Marks pop-up was requested is redisplayed with the marks highlighted and tagged. |
| D | Deletes an existing set or path. |
| R | Renames a set or path. Enter the new name in the Set/path 2 field. |
| Set/path 1 | Specifies the name of the set or path. |

| Field | Description |
|---|---|
| Set/path 2 | Specifies the mark name if the C, M, or R line command is entered. Enter the appropriate mark name. |
| Comments | Provides comments. Existing comments can be changed by typing over them. |

### Usage Notes

The LPRINT * command can be entered on the List - User Marks pop-up to copy the screen contents to the List file.

## Memory Display Screen

The LIST MEMORY mapping feature enables you to display memory mapped through language-specific structures. COBOL, PL/I, and Assembler language structures are supported.

***To display non-fetch protected area of storage in hexadecimal and character formats***

**1** Choose one of these options:

- Select List ▸ Memory.
- Select MEMORY on the Test Facilities List screen.
- Type `LIST MEMORY` on any SmartTest screen.

**2** The List - Memory pop-up shown in Figure 88 displays.

**Figure 88 • List - Memory Pop-up**

```
                              List - Memory

To specify an area of memory to be displayed, type an address
specification in the address area, or leave it blank for no area
specification.  Then press Enter.

Area Specification _______________________________________________
Map Data Name  . . _______________________________________________
                    (Data Name, Blank for NONE, or ?  for External)
```

**3** Specify the area of memory to be displayed by typing an address specification or memory keyword in the Area Specification field. If you leave this field blank, you can specify the area on the Memory Display screen.

**4** Specify the high-level dataname for the structure and/or the PDS name containing the structure to be used for mapping in the Map Data Name field. If you leave this field blank, you can specify the map dataname on the Memory Display screen.

**Note:**

The supported structures that you can map with are external COBOL copybooks and PL/I Includes, as well as standard in-source structures. Map structures must also be in the native source of the program you are testing, or have been analyzed separately.

If you enter a question mark (?) in the Map Data Name field, the Structure Analyze pop-up, shown in Figure 89, displays.

**Figure 89 • Structure Analyze Pop-up**

```
                              STRUCTURE ANALYZE
COMMAND ===> ____________________________________________________________

STRUCTURE ANALYZE FOR COBOL COPY MEMBER or PL/1 INCLUDE

STRUCTURE LIBRARY: ____________________________________________
                   (PDS DATASET NAME, NO QUOTES)

STRUCTURE MEMBER:  ________

STRUCTURE LANGUAGE TYPE: _ 1. COBOL COPY MEMBER
                         _ 2. PL/1 INCLUDE MEMBER

 Press Enter to process, or (Cancel) to exit without processing
```

**5** Complete these required fields:

**a** Specify the name of the PDS or sequential library containing the COBOL copy member or PL/I Include that you want to use to map memory.

**b** Specify the member name of the structure to be used to map memory. If you enter an incorrect name, SmartTest uses the closest 01 structure.

**c** Select the language type.

**6** Press Enter to display the Memory Display screen, shown in Figure 90.

**Figure 90 • Memory Display Screen**

```
                                Memory Display
Command ===> ____________________________________________________ Scroll ===> CSR

Area TGT                                  Offset 000000      Length 000400
 Member          Data Name
000E7070  000000  00300000 000D4FB0 00000000 00000000  * ......|......... *
000E7080  000010  00000000 00000000 00000000 00000000  * ................ *
000E7090  000020  00000000 00000000 00000000 00000000  * ................ *
000E70A0  000030  00000000 00000000 00000000 00000000  * ................ *
000E70B0  000040  00000000 00000000 3002204B 00000000  * ................ *
000E70C0  000050  00000000 000E7A24 00000000 00000000  * ......:......... *
000E70D0  000060  00000000 00000000 00000000 500E8F56  * ............&... *
000E70E0  000070  0005321A 0005321A 00000191 00029258  * ...........j..k. *
000E70F0  000080  000E74C8 00000006 000D4FB0 00000000  * ...H......|..... *
000E7100  000090  800E8F02 800E6D5C 000E8F70 000E6D50  * ......_*......_& *
000E7110  0000A0  000E6D50 000E74B0 5005342E 000BB02A  * .._&....&....... *
000E7120  0000B0  000BB02A 000000CD 00029258 800E6D5C  * ..........k..._* *
000E7130  0000C0  00000000 800C3B16 000E02C8 00000001  * ...........H.... *
000E7140  0000D0  00000003 000D4E24 000D4E24 0005321A  * ......+...+..... *
+-------------------------------------------------------------------------------+
|STATUS: STOPPED BY STEP REQUEST           PROGRAM: VIAPCOB   DATE: DDMMMYYYY |
|  STMT: 000362  OFF: 000F1E  AMODE: 24     MODULE: VIAPCOB   TIME: 09:10:41  |
|SOURCE: ACCEPT TIME-OF-DAY-WORK FROM TIME.                                   |
+-------------------------------------------------------------------------------+
```

**7** Use the Area and Offset fields to specify memory locations that are to be displayed. Once the data at the specified memory location displays, you can change it if the storage area is not store protected.

If you specified a dataname on the Structure Analyze screen, it is displayed in the Member field.

The Data Name field displays the high-level dataname for the structure and/or the PDS name containing the structure to be used for mapping that you specified on the List - Memory pop-up. If you did not specify a dataname on the List - Memory pop-up, you can type a name in on this screen.

## Fields

| Field | Description |
|---|---|
| Area | Specifies the area of memory to be displayed. Absolute addresses (24- and 31-bit), general registers, or SmartTest storage related keywords can be specified. |
| address | Specifies the absolute address. Only valid hexadecimal characters A through F and 0 through 9 can be entered (e.g., 2C41A0). If the address conflicts with a keyword such as FCB, precede the address with a zero (e.g., 0FCB). |
| MOD-name | Specifies the load module or PPT module (CICS environment) for any loaded module in the system (defaults to the qualified load module when entered without the dash). |

| Field | Description |
|---|---|
| PGMA | Specifies the active program for a test session. |
| PGMQ | Specifies the qualified module.program. |
| PGM-name | Specifies the program for any CSECT of the qualified load module (defaults to the qualified program when entered without the dash). |
| R0-R15 | Specifies the General Purpose Registers (e.g., R0, R1, ... R15). |
| AR0-AR15 | Specifies the Access Registers (e.g., AR0, AR1 ... AR15). |
| | These COBOL II related keywords are supported for qualified programs analyzed by SmartTest: |
| BLK-*nn* | Specifies the nth COBOL II base locator for LOCAL STORAGE. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| BLF-*nn* | Specifies the nth COBOL II base locator for files. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| BLL-*nn* | Specifies the nth COBOL base locator for the LINKAGE SECTION. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| BLW-*nn* | Specifies the nth COBOL II base locator for WORKING-STORAGE. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| DSA | Specifies the COBOL Dynamic Storage Area. |
| FCB-*nn* | Specifies the nth COBOL II base locator for file FCB storage. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| ILS-*nn* | Specifies the nth COBOL index cell in LOCAL STORAGE. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| INX-*nn* | Specifies the nth COBOL index cell. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| TGT | Specifies the COBOL Task Global Table. |
| WKS | Specifies the COBOL WORKING-STORAGE. |
| These CICS storage related keywords are supported: | |

| Field | Description |
|---|---|
| AFCB | Specifies the Authorized Function Control Block. |
| AICB | Specifies the CICS Application Interface Control Block. |
| CIA | Specifies the Terminal User Area (same as TUA). |
| COM | Specifies the Command Level Common Storage Area. This keyword is only available during an active test session. |
| CSA | Specifies the Common System Area. |
| DCA | Specifies the Dispatch Control Area. This keyword is only available during an active test session. |
| DCT-*name* | Specifies the Destination Control Table entry, where *name* represents the transient data ID. |
| DWE-*nn* | Specifies the nth Deferred Work Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| EIB | Specifies the EXEC Interface Block. This keyword is only available during an active test session. |
| EIS | Specifies the EXEC Interface Structure. This keyword is only available during an active test session. |
| FCT-*name* | Specifies the File Control Table entry, where *name* represents the file DD name. |
| FIO-*nn* | Specifies the nth File I/O Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| FWA-*nn* | Specifies the File Work Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| ICE-*nn* | Specifies the Interval Control Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| JCA-*nn* | Specifies the Journal Control Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |

| Field | Description |
|---|---|
| LLA-*nn* | Specifies the Load List Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| OPF | Specifies the Optional Feature List. |
| PAM | Specifies the Page Allocation Map. |
| PCT-*name* | Specifies the Program Control Table entry, where *name* is the transaction-ID. |
| PLB | Specifies the CICS Partition Lower Boundary Address. This keyword causes the fullword in the CSA that contains the address to be displayed, rather than the storage at that address. |
| PPT-*name* | Specifies the Processing Program Table entry, where *name* represents the module name. |
| PUB | Specifies the CICS Partition Upper Boundary Address. This keyword causes the fullword in the CSA that contains the address to be displayed, rather than the storage at that address. |
| RSA-*nn* | Specifies the nth Register Storage Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| SIT | Specifies the System Initialization Table. |
| SYS | Specifies the Task Control Area (System). This keyword is only available during an active test session. |
| TCA | Specifies the Task Control Area (User). This keyword is only available during an active test session. |
| TCE | Specifies the Terminal Control Table Terminal Entry. |
| TCT-*name* | Specifies the Terminal Control Table entry, where *name* is the terminal-ID. |
| TIA | Specifies the Current Terminal Input/Output Area. This keyword is only available during an active test session. |
| TIO-*nn* | Specifies the nth Terminal Input/Output Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| TSA | Specifies the Temporary Storage Allocation Table. |

| Field | Description |
|---|---|
| TSI-*nn* | Specifies the nth Temporary Storage Area. This keyword is only available during an active test session. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| TSM | Specifies the Temporary Storage Map. |
| TUA | Specifies the Terminal User Area (same as CIA). |
| TWA | Specifies the Task Work Area. |
| U24-*nn* | Specifies the nth 24-bit address User Transaction Area. |
| U31-*nn* | Specifies the nth 31-bit address User Transaction Area. |
| XCLS | Specifies the SmartTest Global Exclude CSECT Table. |
| *xx*P | Specifies the entry point of the CICS NUCLEUS program, where *xx* is the initials of the program name (i.e., DCP, FCP, ICP, KCP, PCP, PIP, SCP, SPP, SRP, SSP, TCP, TDP, TRP, TSP, and ZCP). |
| These System/Assembler related keywords are supported. They represent control blocks in the address space of the current processing environment, not necessarily in TSO. | |
| ASCB | Specifies the Address Space Control Block. |
| CDE-*name* | Specifies the Contents Directory Entry (same as MOD-*name*). |
| CVT | Specifies the Communication Vector Table. |
| DEB-*nn* | Specifies the nth Data Extent Block. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| LLE-*nn* | Specifies the nth Load List Element. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| PSW | Specifies the Program Status Word (formatted) and registers for the last program interrupt. |
| RB-*nn* | Specifies the nth Request Block. The default for *nn* is 1, if *nn* is left blank or is greater than the number of areas available. |
| SDWA | Specifies the System Diagnostic Work Area at the time of an abend. |
| TCB | Specifies the Task Control Block. |
| TIOT | Specifies the Task Input/Output Table. |

| Field | Description |
|---|---|
| Offset | Displays the storage area beginning at a particular displacement. Storage before the beginning of the specified storage area can be displayed by entering a negative offset [i.e., by preceding a valid hexadecimal offset with a dash (-)]. |
| Length | Specifies the length of the area identified in the Area field. If the length is unknown, a default hexadecimal value of 1000 (4096 bytes) displays. The value in this field can be overridden at any time, to display more or less data, by entering a valid hexadecimal number. |
| Member | Displays the member name you specified on the Structure Analyze pop-up. If you did not specify structure analyze information, this field is blank. |
| Data Name= | Displays the high-level dataname for the structure and/or the PDS name containing the structure to be used for mapping that you specified on the List - Memory pop-up. You can also specify the map dataname directly on the Memory Display screen.<br><br>The supported map structures are external COBOL copybooks, PL/I Includes, and standard in-source structures. Map structures must also be in the native source of the program you are testing, or have been analyzed separately. |
| dump area | Specifies the rest of the screen, which is the dump area that consists of seven columns of data. The first column contains the memory address, the second column contains the offset, the next four columns contain the actual hexadecimal values of the storage at the specified address, and the last column contains the EBCDIC equivalent (bordered by columns of asterisks). |

| Field | Description |
|---|---|
| | Data in the dump area can be changed using one of these methods:<br>• Type over the hexadecimal values with new hexadecimal values.<br>• Type over the character values with new character values.<br>• Enter pseudo hexadecimal data in the hexadecimal dump area by typing character data interspersed with blanks. For example, C A T would cause the hexadecimal value of C3C1E3 to be entered into memory. Pseudo hexadecimal data may also be entered in hexadecimal screen characters and is stored in hexadecimal format.<br>If an attempt is made to change a field that cannot be modified due to storage protection, the field can be reset to the original data by clearing it to spaces (i.e., typing blanks) or by pressing the ERASE EOF key. This method can be used to reverse a change, restoring the original data.<br>Address commands can also be entered in the hexadecimal area. These commands must be entered in the leftmost byte(s) of any word (four bytes aligned) in storage. These address commands are supported: |
| asterisk (*) | Scrolls to position the current word at the top of the screen. |
| L | Calls the Memory Display screen, recursively, to display storage at the 24-bit address, specified by the fullword at the L location, at the top of the screen. |
| LX | Calls the Memory Display screen, recursively, to display storage at the 31-bit address, specified by the fullword at the LX location, at the top of the screen. |
| W | Displays a message identifying the Program or named area represented by the 24-bit address where the W command was entered. Use this command when the current program has a 24-bit addressing mode. |
| WX | Displays a message identifying the Program or named area represented by the 24-bit address where the WX command was entered. Use this command when the current program has a 31-bit addressing mode. |

# Load Module Directory Screen

To list the modules residing in the specified load libraries, follow this step:

▶ Choose one of these options:

- Select List ▶ Module directory.
- Type `LIST MODULES` on any SmartTest screen.
- Select MODULES on the Test Facilities List screen.

Press Enter to display the Load Module Directory screen, shown in Figure 91.

**Note:**

Information such as module size, entry point address, alias name, authorization code, and link-edit attributes displays next to the module name.

**Figure 91 • Load Module Directory Screen**

```
                              Load Module Directory              TESTCOBA.TESTCOBA
Command ===> ______________________________________________ Scroll ===> CSR

    Name   Lib   Size    EPA    ALIAS of  AC  Attributes
_ -------- --  ------  ------  --------  --  ---------------------------------
_ ADDONE     1  000128  000000            00
_ ASMBASE    1  0000C8  000000            00
_ AZVIBC5    1  0024C8  000000            00  REUS
_ BLDQS      2  001840  000000  ILBOQSU   00
_ CCALL      1  000B90  000000            00
_ CD05C384   1  002F88  000000            00  REUS
_ COBDBG     2  035C60  00006E  IGZETEST  00  RENT REUS
_ COBTEST    2  035C60  000000  IGZETEST  00  RENT REUS
_ CSAMPLE    1  0010B0  000000            00
_ CTEST      1  000F00  000000            00
_ CTEST2     1  000B78  000000            00
_ DEADCODE   1  000800  000000            00
_ DLGMNU     1  001D90  000000            00
_ DSNAB1     1  0007F8  000000            00
_ EXAMPLE    1  0017E8  000000            00
_ IGZCACP    2  0002B0  000000            00  RENT REUS
_ IGZCACS    2  0003D0  000000            00  RENT REUS
_ IGZCANE    2  000290  000000            00  RENT REUS
_ IGZCANF    2  000170  000000            00  RENT REUS
```

## Fields

| Field | Description |
|---|---|
| Line command area | Displays the Module Map when you type S to the left of the module name. The Module Map screen lists attributes and CSECTs for the load module. See "Module Map Screen" on page 153 for more information about the Module Map screen. |
| Name | Specifies the load module for which information is to be displayed. |
| Lib | Specifies the relative number of the application load library on the Environment Setup Menu. |
| Size | Specifies the size of the load module in hexadecimal format. |
| EPA | Specifies the entry point address of the load module in hexadecimal format. |
| ALIAS of | Specifies the link-edited name for which this is an alias name. |
| AC | Specifies the link edited authorization code. |
| Attributes | Specifies the link edited attributes, such as RENT, REUS, and OVLY. |

# Module Map Screen

To list attributes and CSECTs for a load module, follow this step:

- Select the module on the Load Module Directory screen by typing S to the left of the CSECT name. The Module Map screen, shown in Figure 92, displays.

**Figure 92 • Module Map Screen**

```
                                   MODULE MAP
Command ===> ______________________________________________ Scroll ===> CSR

Library: COB2.V400.COB2LIB                                  Module: BLDQS
Module size: 001840    Link date: 25MAY1993  AMODE: 24  Entry point: ILBOQSU
Attributes :

         Enter S to view the program source or disassembled object

  CSECT        Source              Module
S Name         on AKR    Length    Offset    Type      Date         Time
- --------     ------    ------    ------    ------    ---------    --------
_ ILBOQSU        NO      001840    000000    ASM         N/A        N/A
******************************** BOTTOM OF DATA ********************************
```

The COBOL, Assembler, or PL/I source code for the selected CSECT displays if available in the AKR. If the source is not available in the AKR, the CSECT is shown in a disassembled format.

## Fields

| Field | Description |
|---|---|
| Library | Specifies the load library that contains the load module to be displayed. |
| Module | Specifies the program or CSECT for which information is to be displayed. |
| Module size | Specifies the Total length of the load module in hexadecimal format. |
| Link date | Specifies the last linkage editor execution date for this load module. |
| AMODE | Specifies the addressing mode in which the module receives control. |
| Entry point | Specifies the entry point of the load module. |
| Attributes | Lists these attributes supplied to the linkage editor:<br>AUTH RENT RMODE<br>OVLY REUS TEST<br>These fields are associated with each CSECT in the list: |
| S | Displays the COBOL, Assembler, or PL/I source, or the disassembled object code for the CSECT. |
| CSECT Name | Specifies a CSECT name from the specified load module. The data displayed on the line with the CSECT corresponds to the load module. |
| Source on AKR | Indicates whether symbolic information is available on the current AKR. |
| Length | Specifies the length of the CSECT in hexadecimal format. |
| Module Offset | Specifies the relative offset for this CSECT from the start of the load module. |
| Type | Specifies the language type of COBOL II, ASM, or PL/I (if available). |

| Field | Description |
|---|---|
| Date | Specifies the COBOL, Assembler, or PL/I date the CSECT was created (if available). |
| Time | Specifies the COBOL, Assembler, or PL/I time the CSECT was created (if available). |

# List - Perform Range Names Pop-up

***To display all COBOL PERFORM ranges in the program being tested and/or displayed on the Program View screen***

**1** Choose one of these options:

- Select PERFORMS on the Test Facilities List screen.
- Type `LIST PERFORMS` on any SmartTest screen.
- Select List ▸ COBOL intelligence features and then selecting Performs on the List - COBOL Features pop-up.

The List - Perform Range Names pop-up, shown in Figure 93, displays.

**Figure 93 • List - Perform Range Names Pop-up**

```
                          List - Perform Range Names        1 PERFORM RANGE
Command ===> ___________________________________________ Scroll ===> CSR

Select the perform range to be viewed.  Then press Enter.

    Perform range name
_   ------------------------------------------------------------------------
_   PARA1 THRU PARAEXIT
****************************** BOTTOM OF DATA ******************************
```

When this pop-up is first displayed, a short message indicates the number of PERFORM ranges found.

**Note:**

This screen is available only when Insight is installed and an Insight analysis has been run on the COBOL program being tested.

**2** Select the perform range you want to view and press Enter.

### Fields

| Field | Description |
|---|---|
| Line command area | Select a PERFORM label for viewing by typing S to the left of the Perform range name field. |
| Perform range name | Specifies the PERFORM label of the COBOL PERFORM range. |

### Usage Notes

PERFORM ranges that are recursively CALLed are indicated with an asterisk (*) in column one.

The LPRINT * command can be entered on the List - Perform Range Names screen to copy the screen contents to the List file.

## Profile Data Set Member List Screen

The Profile Data Set Member List screen is used to perform these functions:

- Copy from the profile dataset of another user
- Delete a member from the profile
- Select the environment
- Write (Save) the current environment ISPF profile information

### *To display the Profile Data Set Member List screen*

**1** Choose one of these options:

- Select List ▸ Profiles.
- Select PROFILE on the Test Facilities List screen.
- Type `LIST PROFILE` on any SmartTest screen.

The Profile Data Set Member List screen, shown in Figure 94, displays.

**Figure 94 • Profile Data Set Member List Screen**

```
                         Profile Data Set Member List      TESTCOBA.TESTCOBA -Q
Command ===> ____________________________________________________________________

The current environment is:  TSO
Profile dataset name  :  'USER.ISPF.PROFILE'
COPY TO dataset name  :  ____________________________________________
  S - Select member to restore    W - Write current environment to member
  C - Copy selected member        R - Replace member (Pending status)
  D - Delete member                   * denotes TCA Profile
  Profile   Environ    User profile description (optional)
_ --------  ---------  ------------------------------------------------------
_ VIAPST01  AVAILABLE  ------------------------------------------------------
_ VIAPST02  AVAILABLE  ------------------------------------------------------
_ VIAPST03  AVAILABLE  ------------------------------------------------------
_ VIAPST04  TSO        test1
_ VIAPST05  TSO        test for autotester
_ VIAPST06  TSO        test viapcob and keep window
_ VIAPST07  TSO        tca test tso profile
_ VIAPST08  TSO        this is new stuff
_ VIAPST09  TSO        Q/A transfer for TESTCOB RELEASE 3.3
_ VIAPST10  TSO        test count for tom
_ VIAPST11  TSO        analz test w/br in r$utatch
_ VIAPST12  CICS       cics test
_ VIAPST13  ISPF       ISPF setup 09/21/95
```

The LIST PROFILE command is available at all times except during connection to a batch environment.

**2** Type the line command for the action you want to perform and press Enter.

## Fields

| Field | Description |
|---|---|
| The current environment is | Indicates which execution environment is selected. |
| Profile data set name | Specifies the current user's ISPPPROF profile dataset name displays in this field as the default. The default may be replaced by another user's profile dataset name to copy or select from another user's profile. |
| COPY TO dataset name | Specifies a non-ISPF dataset as the target of a COPY line command. |

| Field | Description |
|---|---|
| C line command | Copies a member from the profile dataset to the specified COPY TO dataset. After you enter the COPY line command, the COPY TO dataset member list displays. Use the REPLACE line command to select where to place the copy member. The COPY TO dataset is either a non-ISPF dataset or your own ISPF profile dataset. |
| D line command | Deletes a member from your profile and places AVAILABLE in the Environ field. An error message displays if an attempt is made to delete another user's profile dataset. |
| S line command | Selects an environment from your profile dataset or from another user's profile dataset and displays that environment's session setup screen. This command cannot be used during an active test session, with an active program, or while connected to CICS. |
| W line command | Writes the current environment ISPF profile information to your profile dataset member. Only your own profile dataset can be written to. |
| R line command | Completes the copy operation. Use the R line command to select where you want to put the copy member. The R line command may only be used after the C line command is entered. |
| Profile | Specifies the user profile member name for a specific environment. |
| Environ | Specifies the environment for the user profile member. This may be one of any of the SmartTest testing environments (TSO, CICS, ISPF DIALOG, IMS/DC, IMS/DB, BTS, DB2, MVS BATCH, IMS BATCH, BTS BATCH, or DB2 BATCH). |
| User profile description | Displays the module from which a profile has been saved. Descriptive information can be added or the module name may be replaced. You can save a maximum of 99 profiles in a profile dataset. |

# List - Program/Subprogram Names Screen

***To display the internal subprograms defined with a COBOL II Release 3 or later program***

**1** Choose one of these options:

- Select PROGRAMS on the Test Facilities screen.
- Type `LIST PROGRAMS` on any SmartTest screen.
- Select COBOL intelligence features on the List pull-down and then select Programs on the List - COBOL Features pop-up.

The List - Program/Subprogram Names screen, shown in Figure 95, displays.

**Note:**

The internal subprograms are indented in relation to their hierarchy.

**Figure 95 • List - Program/Subprogram Names Screen**

```
                    List - Program/Subprogram Names        1 PROGRAM(S)
Command ===> ____________________________________________ Scroll ===> CSR

Select the program name to be viewed.  Then press Enter.

    Program name
_   ----------------------------------------------------------------------
_   TESTCOBA
*************************** BOTTOM OF DATA ***************************
```

**2** Select the program you want to view and press Enter.

## Fields

| Field | Description |
|---|---|
| Line command area | Selects the program for viewing when you type S to the left of the Program name field. |
| Program name | Specifies the name of the programs defined in a COBOL II or later program. Subprograms are indented. |

## Usage Notes

The LPRINT * command can be entered to copy the screen contents to the List file.

When this pop-up is first displayed, a short message indicates the number of programs found. Any programs that do not fit on the screen can be viewed by scrolling to them.

# Pseudo Code List Screen

***To display all pseudo code statements in the program being tested and/or displayed on the Program View screen***

**1** Choose one of these options:

- Select PSEUDO on the Test Facilities List screen.
- Type `LIST PSEUDO` on any SmartTest screen.
- Select List ▸ Pseudo code.

The Pseudo Code List screen, shown in Figure 96, displays.

**Figure 96 • Pseudo Code List Screen**

```
                                Pseudo Code List            TESTCOBA.TESTCOBA -Q
Command ===> ________________________________________________ Scroll ===> CSR

         Set pseudo  . . ON       Pseudo active: 1       Inactive: 0
         Set breaks  . . ON       Breaks active: 1       Inactive: 0
         Set whens . . . ON       Whens  active: 0       Inactive: 0

S Line   Pseudo code                                                  A Count
- ------ ------------------------------------------------------------ - -----
_ 000029  MAIN-PROCESS.
_ '''''1      BREAK.                                                  Y 000006
_ 000030      MOVE 0 TO WS-CHKP-COUNTER.
  ************************** BOTTOM OF DATA ****************************
```

**2** Make any desired changes. The status of the SET PSEUDO, SET BREAKS, and SET WHENS modes are displayed and can be changed if desired. The number of active and inactive breakpoints, pseudo code, and When conditions are also indicated. You can enable and disable any or all pseudo code statements using this screen. This includes breakpoints and When Conditions.

## Fields

| Field | Description |
|---|---|
| Set pseudo | Enables the pseudo code facility when this mode is ON. When OFF, any attempt to execute pseudo code is ignored, including Breakpoints and When Conditions regardless of the values in the Set breaks and Set whens fields. Breakpoints and When Conditions are considered to be pseudo code. |
| Set breaks | Enables the Breakpoints facility when this mode is ON. When this mode is OFF, all Breakpoints are ignored. If Set pseudo is OFF, Breakpoints are ignored regardless of the Set breaks mode. |

| Field | Description |
|---|---|
| Set whens | Enables the When Conditions facility when this mode is ON. When this mode is OFF, all When Conditions are ignored. If Set pseudo is OFF, When Conditions are ignored regardless of the Set whens mode. |
| Pseudo active/Inactive | Indicates the number of pseudo code lines in the program that are enabled or disabled. |
| Breaks active/ Inactive | Indicates the number of Breakpoints that are enabled or disabled. |
| Whens active/ Inactive | Indicates the number of When Conditions that are enabled or disabled. |
| S | Displays the Program View screen with the specified line at the top of the screen when you type `S` in this field. Typing `D` in this field deletes the corresponding pseudo code statement. |
| Line | Indicates the source code and pseudo code line numbers. |
| Pseudo code | Displays each block of pseudo code, as well as the source code lines preceding and following the pseudo code. A dashed line separates each block of pseudo code that displays. The rest of the pseudo code that does not fit on the screen can be viewed by scrolling to it. |
| A | Activates the pseudo code statement so that it is executed when the program is tested. Typing `N` in this field deactivates the pseudo code statement so that it is ignored when the program is tested. The default is Y.<br>If a conditional pseudo code statement (IF, WHEN) is deactivated, all of the subordinate statements are also deactivated. For example:<br>`IF A > B THEN`<br>`MOVE C TO D`<br>`BREAK.`<br>**Note:**<br>When the IF A > B THEN statement is deactivated, the MOVE C TO D and BREAK statements are also deactivated. |
| Count | Indicates the number of times the pseudo code statement has been executed during the test session. |

# General Registers Screen

The General Registers screen displays this information:

- Program Status Word (PSW)
- Current instruction
- Current addressing mode (AMODE)
- Condition code setting
- Program mask bit values
- General registers 0 through 15

### *To access the General Registers screen*

**1** Choose one of these options:

- Select List ▸ Registers.
- Select REGISTERS on the Test Facilities List screen.
- Type `LIST REGISTERS` on any SmartTest screen.

The General Registers screen, shown in Figure 97, displays.

**Figure 97 • General Registers Screen**

```
                              General Registers                    VIAPPLI -Q
Command ===> ______________________________________________ Scroll ===> CSR

PSW address: 800CFD1A   Instr: 07FE            AMODE: 31   CC: 8   Mask: 0

Reg   Hex       Decimal     Address identification      Data at address
---  --------  -----------  -------------------------  --------------------
R00  00000000           0                              *                  *
R01  000B1210      725520                              * ...V...V........ *
R02  00000000           0                              *                  *
R03  00000000           0                              *                  *
R04  00000000           0                              *                  *
R05  00000000           0                              *                  *
R06  00000000           0                              *                  *
R07  00000000           0                              *                  *
R08  00000000           0                              *                  *
R09  00000000           0                              *                  *
R10  00000000           0                              *                  *
R11  00000000           0                              *                  *
R12  00000000           0                              *                  *
R13  000AFFB0      720816                              * ..........7q.... *
R14  000C1976      792950  MOD-VIAPEMON + 343E         * .\.\.\...Fq...q. *
R15  00000004           4                              * a|. ............ *
```

**2** Make any changes. Register contents are displayed in both hexadecimal and decimal formats. These fields are modifiable. Target data displays and the target address is identified (if possible).

Registers modified as a result of the last RUN or STEP command being executed are highlighted.

**Note:**

The general registers always display in the status box (if displayed) if the program source does not reside in the AKR and the ASMVIEW mode is on.

## Fields

| Field | Description |
|---|---|
| PSW address | Specifies the Program Status Word address. |
| Instr | Specifies the current instruction. |
| AMODE | Specifies the current condition addressing mode. |
| CC | Specifies the current condition code setting. |
| Mask | Specifies the program mask bit values. |
| Reg | Specifies the general registers 0 through 15. |
| Hex | Specifies the register contents in hexadecimal format. |
| Decimal | Specifies the register contents in decimal format. |
| Address identification | Specifies the target address. |
| Data at address | Specifies the data at the target address. |

## Usage Notes

The Memory Display screen can be displayed by entering an L (24-bit address) or LX (31-bit address) in the Hex format area of a register.

# List - COBOL Subsets Names Screen

### *To display the subsets along with a brief description*

**1** Choose one of these options:

- Select SUBSETS on the Test Facilities List screen.
- Type `LIST SUBSETS` on any SmartTest screen.
- Select List ▸ COBOL intelligence features and then select Subsets on the List - COBOL Features pop-up.

The List - Subsets Names screen, shown in Figure 98, displays.

**Figure 98 • List - Subsets Names Screen**

```
                              List - Subset Names
Command ===> ____________________________________________ Scroll ===> CSR

Select the subset to be viewed.  Then press Enter.

$     Subset                              Description
- ---------------------------------------------------------------------------
_ ASsignment     - COBOL: ACCEPT, ADD, SUBTRACT, MULTIPLY, DIVIDE,
                          BY, COMPUTE, MOVE, FROM, INSPECT, EXAMINE,
                          STRING, UNSTRING, SET, SEARCH, TRANSFORM;
                   IDMS: ACCEPT, CHECK, ERASE, GET, INQUIRE, LOAD,
                         PUT, RETURN;
                   SQL: FETCH, SELECT INTO, SET.
_ CAll           - COBOL: CALL, CANCEL, ENTRY,
                          intrinsic function calls;
                   CICS: LINK, XCTL;
                   IDMS: ATTACH, LINK, XCTL.
_ CIcs           - EXEC CICS, EXEC DLI commands.
_ COBOLII        - CALL (COBOLII), PERFORM (COBOLII), CONTINUE, SET,
                   SERVICE LABEL, EVALUATE, INITIALIZE, end verbs.
_ COBOL/370      - Statements and clauses unique to COBOL/370, such as
                   intrinsic function calls, procedure pointers and
```

The subsets listed on this screen include COBOL language subsets, screen subsets, and tagged lines subsets. Some SmartTest commands accept subsets as operands, limiting their scope to one type of data. It may be necessary to scroll the screen forward to view all of the subsets.

**2** Select the subset you want to view and press Enter.

## Fields

| Field | Description |
|---|---|
| Line command area | Selects the first occurrence of the subset for viewing. |
| Subset | Specifies the subset name. |
| Description | Specifies the description of the subset. |

See the *ASG-SmartTest for COBOL and Assembler User's Guide* for more information and a list of the COBOL subsets. See the *ASG-SmartTest PLI User's Guide* for more information and a list of the subsets as they apply to PL/I.

# Test Session Tailoring Screen

***To turn SmartTest features selectively ON or OFF for a set of modules and/or programs***

**1** Choose one of these options:

- Select TAILOR on the Test Facilities List screen.
- Type `LIST TAILOR` on any SmartTest screen.
- Select List ▸ Test session tailoring.

The Test Session Tailoring screen, shown in Figure 99, displays.

**Figure 99 • Test Session Tailoring Screen**

```
                          Test Session Tailoring         TESTCOBA.TESTCOBA -Q
Command ===> ______________________________________________ Scroll ===> CSR

                        Monitor Track Count Break Break Break Pseudo Single
     Module.Program id   Act    Act   Act   Act  Entry  Rtn   Act    Step
     -----------------   ---    ---   ---   ---  -----  ---   ---    -----
**** ************************* TOP OF DATA ****************************
'''' TESTCOBB.TESTCOBB     YES    NO    YES   NO    NO    NO    NO     NO
'''' TESTCOBA.TESTCOBA     YES    NO    YES   NO    NO    NO    NO     NO
**** ************************ BOTTOM OF DATA ***************************
```

Session tailoring can reduce the execution time for a test and provides options at the program level.

**2** Make any changes and press Enter. A sample input line is provided on the first data line displayed, to indicate where information for a program is to be entered. The I (Insert), R (Repeat), and D (Delete) line commands can be used when editing the list of programs. (Block forms of these commands cannot be used.)

## Fields

| Field | Description |
|---|---|
| D*nnn* line command | Deletes *nnn* lines on this screen. If *nnn* is left blank, one line is deleted. |
| I*nnn* line command | Inserts *nnn* blank lines on this screen. If *nnn* is left blank, one line is inserted. |
| R*nnn* line command | Repeats the line *nnn* times. If *nnn* is left blank, the line is repeated one time. |
| Module.Program id | Specifies the load module or program name. Load module and program names can be entered in a generic manner using wild cards to reduce the number of entries. For example, the VIAS*.* entry on the preceding screen indicates all modules that begin with VIAS followed by any other characters, will not have the COBOL verbs, Assembler instructions, or PL/I verbs counted during this test session. However, the VIASUB.* module will have the COBOL verbs, Assembler instructions, or PL/I verbs counted, therefore it is listed before the generic VIAS*.* entry. The list of module.programs is processed from top to bottom. If two entries on the list conflict, the higher qualified line takes precedence.<br><br>For PL/I programs, Program ID must be the PL/I compiler generated name. You can use an asterisk (*) as a wildcard character at the end of the module ID or program ID, but not in a leading position. For example, if load module ABC calls the PL/I program XYZ and you want to break at the start of XYZ, you can use either:<br><br>`ABC.****XYZ1`<br><br>`or`<br><br>`ABC.****X*`<br><br>The trailing asterisk is the wildcard character. In the same example, you could not use this statement to break at the start of XYZ, because the leading asterisk represents a meaningful character, not a wildcard:<br><br>`ABC.*XYZ1` |
| Monitor Act | Indicates whether monitoring is active for the program. NO excludes monitoring for the program. This reduces system resource requirements for a test session. The default is YES. |
| Track Act | Indicates whether statement tracking is to be active for the specified program. NO excludes statement tracking for the program. This reduces system resource requirements for a test session. The default is NO. |

| Field | Description |
|---|---|
| Count Act | Indicates whether COBOL verbs, Assembler instructions, or PL/I verbs executed during a test session are to be counted for the program. The Statement Counts screen displays the execution count and a histogram indicating the relative count for each executable PROCEDURE DIVISION statement or Assembler source statement. The default is NO. |
| Breaks Act | Indicates whether Breakpoints are active or inactive for the specified program. YES specifies that a program interrupt occurs when a BREAK command is encountered. NO specifies that BREAK commands are ignored during the test session. The default is YES. If Pseudo active is NO, Breakpoints are ignored regardless of the entry in the Breaks active field. |
| Break Entry | Indicates whether an interrupt occurs upon entry into the specified program. YES causes program execution to stop at the first PROCEDURE DIVISION statement, or at offset zero if the program is not COBOL, or is not in the AKR and the ASMVIEW mode is enabled. The default is NO.<br><br>For TSO only, when a test session is initiated using the RUN command with the NOMONITOR operand, SmartTest breaks on entry only in CSECTs that have standard O/S entry point logic (i.e., offset 0). |
| Break Rtn | Indicates whether an interrupt occurs when a CALLed program returns control to the calling program. YES causes program execution to stop when a GOBACK or STOP RUN statement is encountered in the CALLed program. Note that this field is available only for analyzed COBOL programs. The default is NO. |
| Pseudo Act | Indicates whether pseudo code is active or inactive for the specified program. YES specifies that pseudo code is executed during the test session. NO specifies that all pseudo code including BREAKs are ignored during the test session. The default is YES. |
| Single Step | Indicates whether statement/offset stepping is allowed in the specified program(s). If SINGLE STEP YES was specified for Program A and NO for Program B, then a CALL to Program B is encountered while stepping through Program A, the next instruction stopped on (stepped to) would be the instruction following the CALL in Program A, even though Program B had been executed. The default is YES. |

# Execution Tracking Screen

***To list the modules, programs, paragraphs, procedures, statements, and offsets that were last executed***

**1** Choose one of these options:

- Select TRACKING on the Test Facilities List screen.
- Type `LIST TRACKING` on any SmartTest screen.
- Select List ▸ Execution tracking, follow this step.

The Execution Tracking screen, shown in Figure 100, displays.

**Figure 100 • Execution Tracking Screen**

```
                              Execution Tracking          VIASUB37.VIASUB37 -A
Command ===> ______________________________________________ Scroll ===> CSR

      000458      READ INFILE3 INTO INFILE3-WORK-REC
   3300-READ-INFILE3-X
      000466      EXIT.
   2000-PROCESSING-LOOP
      000214      MOVE SPACES TO OUTFILE-WORK-AREA,
      000216      COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF
      000219      IF READ-3-FILES
      000220          PERFORM 2100-COMPLEX-MERGE THRU
   2100-COMPLEX-MERGE
      000240      MOVE INFILE1-WORK-KEY TO COMPARISON-KEY-1.
      000241      MOVE INFILE2-WORK-KEY TO COMPARISON-KEY-2.
      000242      CALL 'VIASUB37' USING COMPARISON-KEYS, COMPARISON-CODES.
VIASUB37.VIASUB37
   PROCEDURE DIVISION OF VIASUB37
      000022  PROCEDURE DIVISION USING COMPARISON-KEY COMPARISON-CODE.
******************************* BOTTOM OF DATA ******************************
+-----------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK              PROGRAM: VIASUB37  DATE: 04OCT2000 |
|  STMT: 000023  OFF: 000284  AMODE: 31     MODULE: VIASUB37  TIME: 09:29:26  |
|SOURCE: IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY                          |
+-----------------------------------------------------------------------------+
```

**2** Customize the information viewed on the Execution Tracking screen using the LIST TRACKING command operands: PROGRAMS, CSECTS, PARAGRAPHS, SOURCE, STATEMENTS, OFFSETS, and DISASSEMBLED. See the "LIST Command" on page 316 for more information.

This screen is only used during a SmartTest test session. When this screen is accessed and a test session is not active, the `TEST SESSION NOT ACTIVE` message displays. The Execution Tracking information is not captured during RUN NOMONITOR processing. The Execution Tracking screen can be displayed, but tracking is performed only while monitoring, so the screen contents are not current.

Typically, a COBOL statement translates to several machine instructions. Statement tracking is recorded at the machine instruction level. The number of machine instructions available stored depends on the value entered with the SET TRACK option; the default value is 1024 with a maximum of 64K. Therefore, the maximum number of COBOL statements displayed depends on the number of machine instructions to which a COBOL statement is translated.

COBOL statement numbers, paragraphs, and program names are presented on this screen in the sequence in which they were executed. The last line of data displayed is the next line/offset to be executed (same as what is in the status box).

When the ASM mode is ON, the disassembled object code for each COBOL statement is also shown. When the ASMVIEW mode is ON, COBOL or PL/I subroutine CSECTS are also displayed.

See the "SET Command" on page 409 or the "Options - Mode Screen" on page 195 for more information.

Offsets and line numbers of programs that were monitored are displayed as input fields, and can be used as Select fields. Typing `S` on a line number displays the Program View screen with the selected line number at the top of the screen. Entering S on an offset displays the disassembled object program (if SET ASMVIEW is on) at the selected offset.

The message, `PROGRAM STORAGE NO LONGER ACCESSIBLE` may appear on the same line as the program identification when the program storage is no longer available to SmartTest. In this case, the program's execution history displays as a series of offsets.

**Note:**

APS Painter (S-COBOL) programs are shown on the Execution screen at the generated COBOL statement level. Assembler Macro programs are shown on the Execution screen at the generated Assembler instruction level.

### Fields

| Field | Description |
|---|---|
| *+nnnnnn* | Specifies that a plus sign (+) next to a number indicates an offset. |
| *nnnnnn* | Specifies that a number without a plus sign next to it is a statement number. |

## When Conditions List Screen

***To display the When Conditions in the program being tested and/or displayed on the Program View screen***

**1** Choose one of these options:

- Select WHENS on the Test Facilities List screen.
- Type `LIST WHENS` on any SmartTest screen.
- Select List ▸ When conditions. The When Conditions List screen, shown in Figure 101, displays.

**Figure 101 • When Conditions List Screen**

```
                              When Conditions List            VIAMRG37.VIAMRG37
Command ===> ______________________________________________ Scroll ===> CSR

         Set pseudo  . . ON       Pseudo active: 6       Inactive: 0
         Set breaks  . . ON       Breaks active: 3       Inactive: 0
         Set whens . . . ON       Whens  active: 3       Inactive: 0

$ Line   Pseudo code                                                  A Count
_ ------ ------------------------------------------------------------ - -----
_ ****** ********************** WHEN CONDITIONS **********************
_ '''''1      WHEN INFILE1-WORK-KEY > '1234567' THEN BREAK.           Y 000000
_ '''''2      WHEN INFILE2-WORK-KEY > '1234567' THEN BREAK.           Y 000000
_ '''''3      WHEN INFILE3-WORK-KEY > '1234567' THEN BREAK.           Y 000000
  ****************************** BOTTOM OF DATA ******************************



+-----------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK              PROGRAM: VIASUB37  DATE: 04OCT2000 |
|  STMT: 000023  OFF: 000284  AMODE: 31     MODULE: VIASUB37  TIME: 09:29:26  |
|SOURCE: IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY                          |
+-----------------------------------------------------------------------------+
```

**2** Enable or disable all or specific When Conditions. The number of active and inactive When Conditions are also indicated. WHEN conditions are ignored during RUN NOMONITOR processing.

## Fields

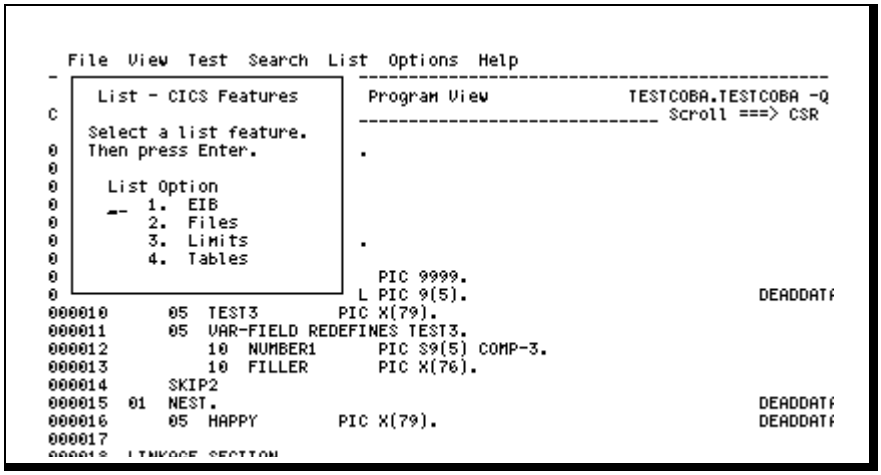
| Field | Description |
|---|---|
| Set pseudo | Enables the pseudo code facility when this mode is ON. When this mode is OFF, any attempt to execute pseudo code is ignored, including Breakpoints and When Conditions regardless of the values in the Set breaks and Set whens fields. Breakpoints and When Conditions are considered to be pseudo code. |
| Set breaks | Enables the Breakpoints facility when this mode is ON. When this mode is OFF, all Breakpoints are ignored. If Set pseudo is OFF, Breakpoints are ignored regardless of the Set breaks mode. |
| Set whens | Enables the When Conditions facility when this mode is ON. When this mode is OFF, all When Conditions are ignored. If Pseudo mode is OFF, all When Conditions are ignored regardless of the value of the Set whens field on this screen. |
| Pseudo active/Inactive | Indicates the number of pseudo code lines in the program that are enabled or disabled. |
| Breaks active/Inactive | Indicates the number of Breakpoints that are enabled or disabled. |
| Whens active/Inactive | Indicates the number of When Conditions that are enabled or disabled. |
| S | Displays the Program View screen with the specified line at the top of the screen. |
| Line | Indicates the source code and pseudo code line numbers. |
| Pseudo code | Specifies the When Condition statement as it displays at the end of the source code. If all of the When Condition statements do not fit on the screen the rest of the statements can be viewed by scrolling to them. |
| A | Activates the When Condition statement so that it is executed when the program is tested. If the Set whens field contains OFF or the pseudo mode is OFF, the When Condition statement is ignored even if you enter `Y` in this field. Typing `N` in this field deactivates the When Condition statement so that it is ignored when the program is tested. The default is Y. |
| Count | Indicates the number of times the When Condition statement has been executed during the test session. |

# List - CICS Features Pop-up

To access list screens available only in the CICS environment, follow this step:

- Select List ▸ CICS features. The List - CICS Features pop-up, shown in Figure 102, displays.

**Figure 102 • List - CICS Features Pop-up**




## Field

| List Option | Description |
|---|---|
| EIB | Displays the EXEC Interface Block (EIB) screen used to present a formatted display of the command level EIB for the purpose of review and updating. |
| Files | Displays the CICS File Services screen used to access CICS resource directories. |
| Limits | Displays the Transactions Limits and Options screen used to specify resource usage limits for each monitored transaction, and to specify SmartTest processing options. |
| Tables | Displays the User Protection Menu used to access the various screens that specify which terminals, tasks, programs, and storage areas are to be monitored and/or protected and any alternate resources to be used during a test. |

# List - COBOL Features Pop-up

To access list information about CALLS, PERFORMS, programs, subsets, and marks, follow this step:

- Select List ▸ COBOL intelligence features. The List - COBOL Features pop-up, shown in Figure 103, displays.

**Figure 103 • List - COBOL Features Pop-up**

```
  File  View  Test  Search  Logic  List  Options  Help
- .-------------------------. ---------------------------------------------
  |  List - COBOL Features  |  Program View                     VIAMERGE -Q
C |                         |  ----------------------------- Scroll ===> PAGE
  | Specify a List Option.  |
0 | Then press Enter.       |  .
0 |                         |  .
0 |   List Option           |  BY VIASOFT IN LANG LEVEL 2.
0 |   __  1.  Calls         |
0 |       2.  Performs      |
0 |       3.  Programs      |
0 |       4.  Subsets       |
0 |       5.  Marks         | IGN TO UT-S-INFILE1.
0 '-------------------------' IGN TO UT-S-INFILE2.
000010       SELECT INFILE3  ASSIGN TO UT-S-INFILE3.
000011       SELECT OUTFILE  ASSIGN TO UT-S-OUTFILE.
000012       SELECT OUTRPT   ASSIGN TO UT-S-OUTRPT.
000013 *
000014  DATA DIVISION.
+-------------------------------------------------------------------------+
|STATUS: BREAK AT END OF TEST SESSION      PROGRAM: IGZENRT   DATE: 13AUG2001 |
|  STMT: N/A     OFF: 000ABE  AMODE: 24     MODULE: TESTCOBA  TIME: 09:12:49 |
|SOURCE: BCR   15,R14                          HEX: 07FE                   |
+-------------------------------------------------------------------------+
```

## Field

| List Option | Description |
|---|---|
| Calls | Displays the List - CALL Statements pop-up that lists programs that are CALLed by the current program. |
| Performs | Displays the List - Perform Range Names pop-up that lists all COBOL perform ranges in the current program. |
| Programs | Displays the List - Program/Subprogram Names pop-up used to display the main program and any internal subprograms defined with a COBOL II Release 3 or later program. |
| Subsets | Displays the List - COBOL Subset Names pop-up that lists the subsets along with a brief description of each. |
| Marks | Displays the List - User Marks pop-up that lists all existing marks and is used to change comments, merge sets, and copy, delete, or rename sets or paths. |

# List - Execution Counts Pop-up

To customize and display the Execution Counts screen, follow this step:

- Select List ▸ Execution counts. The List - Execution Counts pop-up, shown in Figure 104, displays.

**Figure 104 • List - Execution Pop-up**

```
        List - Execution Counts

Select List format and Sort order.
Then press Enter.

    List Format
    1   1.  Source
        2.  Paragraph/Label

    Sort Order
    1   1.  Line
        2.  Ascending
        3.  Descending
```

## Fields

| Field | Description |
|---|---|
| List Format | |
| Source | Displays the current execution counts by source statement, sorting statements according to the sort option selected. This is the default if COUNTS PARAGRAPHS or COUNTS LABELS has not been specified during the current test session. |
| Paragraph/Label | Displays the current execution counts by LABEL point or PARAGRAPH name, sorting labels or paragraphs according to the sort option specified. |

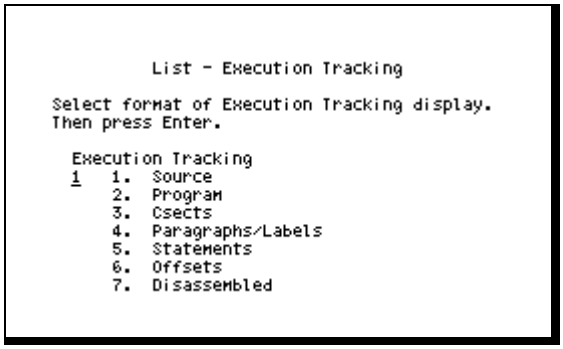
| Field | Description |
|---|---|
| Sort Order | |
| Line | Displays the current execution counts sorting source by line. This is the default if COUNTS ASCENDING or COUNTS DESCENDING has not been specified during the current test session. |
| Ascending | Displays the current execution counts, sorting source by ascending counts. Once specified this becomes the default until another COUNTS operand is specified. |
| Descending | Displays the current execution counts, sorting source by descending counts. Once specified this becomes the default until another COUNTS operand is specified. |

# List - Execution Tracking Pop-up

To customize the Execution Tracking display, follow this step:

- Select List ▸ Execution tracking. The List - Execution Tracking pop-up, shown in Figure 105, displays.

**Figure 105 • List - Execution Tracking Pop-up**



## Field

| Field | Description |
|---|---|
| Execution Tracking | |
| Source | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, procedures, and source statements that were last executed for the program being tested. |
| Program | Displays the Execution Tracking screen listing programs that were last executed for the program being tested. |
| Csects | Displays the Execution Tracking screen listing programs, CSECTs that were last executed for the program being tested. |
| Paragraphs/Labels | Displays the Execution Tracking screen listing programs, CSECTs, procedures, and paragraphs that were last executed for the program being tested. |
| Statements | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, procedures, and line numbers of statements that were last execute for the program being tested. |
| Offsets | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, procedures, and disassembled source statements that were last executed for the program being tested. This operand is used with the TRACKING operand. |
| Disassembled | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, procedures, and disassembled source statements that were last executed for the program being tested. This operand is used with the TRACKING operand. |

# List - Registers Pop-up

To display access the Access Registers, Floating Point Registers, and General Registers screens, follow this step:

- Select List ▸ Registers. The List - Registers pop-up, shown in Figure 106, displays.

**Figure 106 • List - Registers Pop-up**

```
List - Registers

Specify a List Option. Then press
Enter.

  List Option
  3   1.  Access Registers
      2.  Floating Pt. Registers
      3.  General Registers
```

## Field

| Field | Description |
|---|---|
| List Option | |
| Access Registers | Displays the Access Registers screen used to specify that lists the contents of the access registers for the program being tested. Register contents are displayed in hexadecimal and decimal formats and can be modified if desired. |
| Floating Pt. Registers | Displays Floating Point Registers screen used to display the floating point registers for the program being tested. Register contents are displayed in hexadecimal and decimal formats and can be modified if desired. |
| General Registers | Displays the General Registers screen used to list the contents of the general registers for the program being tested. Register contents are displayed in hexadecimal and decimal formats and can be modified if desired. |

# 6 Help

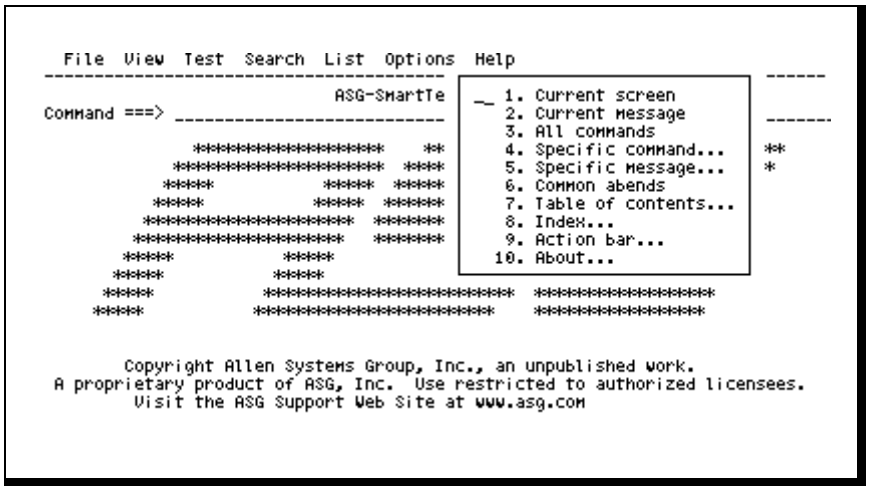
This chapter describes the options available on the Help pull-down menu and contains these sections:


| Section | Page |
|---|---|
| Help Pull-down | 180 |
| Help - Specific ASG Command Pop-up | 181 |
| Help - Specific ASG Message Pop-up | 182 |
| Help Table of Contents | 182 |
| Help Index | 183 |
| Help - About Pop-up | 184 |

# Help Pull-down

The Help pull-down is used to access the online help facility. Figure 107 shows the Help pull-down.

**Figure 107 • Help Pull-down**




## Actions

| Action | Description |
|---|---|
| Current Screen | Displays help for the screen that is currently displayed. |
| Current Message | Displays help for the message that is currently displayed. |
| All Commands | Displays a complete list of all SmartTest commands, from which information about a specific command can be displayed by selecting the appropriate number. |
| Specific Command | Displays the Help - Specific ASG Command pop-up used to obtain help about a specific SmartTest command. |
| Specific Message | Displays the Help - Specific ASG Message pop-up used to obtain help about a specific message number. |
| Common Abends | Displays the Product Abend Help panel, from which information about abends can be displayed. |
| Table of Contents | Displays the online help table of contents, which lists general information topics about SmartTest. See "Help Table of Contents" on page 182 for more information and an example of the online help table of contents. |

| Action | Description |
|---|---|
| Index | Displays the first online help index screen. See "Help Index" on page 183 for more information and an example of the online help index. |
| Action Bar | Displays the Help Tutorial for the SmartTest action bar. |
| About | Displays the Help - About pop-up that lists information about the current levels of SmartTest and Center. |

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about the Online Help facility.

# Help - Specific ASG Command Pop-up

To obtain help about a specific SmartTest command, follow this step:

- Select Help ▸ Specific Command. The Help - Specific ASG Command pop-up, shown in Figure 108, displays.

**Figure 108 • Help - Specific ASG Command Pop-up**

```
            Help - Specific ASG Command

To display a message that describes a ASG
command, type the command below.  Then press
Enter.

        Command . . . . _________________

To display additional command information, enter
the HELP command (PF1/13) from the screen that
displays the command message.
```

## Fields

**Command.** Enter a primary command.

# Help - Specific ASG Message Pop-up

To display help for a specific message, follow this step:

- Select Help ▸ Specific Message. The Help - Specific ASG Message pop-up, shown in Figure 109, displays.

**Figure 109 • Help - Specific ASG Message Pop-up**

```
        Help - Specific ASG Message Number

To display the description of a message, type
the message number below.  Then press Enter.

            Message . . . . _-------
```

## Fields

**Message.** Enter an ASG message number.

# Help Table of Contents

The Help Table of Contents, shown in Figure 110, is used to display general help information. This screen displays by selecting Help ▸Table of contents or by typing `TOC` from the Help Tutorial.

**Figure 110 • Help Table of Contents**

```
ASG-SmartTest - R7.0 -----  Help Table of Contents  ---------------------- HELP
===> _

The topics below represent general categories of information about the ASG-ESW
Testing/Debugging component, ASG-SmartTest. To get help for a pull-down,
select the Action Bar topic. This Table of Contents may be redisplayed from
any HELP screen by entering the TOC command.

These topics are presented only if explicitly selected by number:

  1  Overview of ASG-SmartTest

  2  Introduction to CUA

  3  The Action Bar

  4  Customer Support

  5  ASG-SmartTest Release 7.0 Summary of Revisions
```

# Help Index

To display information about specific items, follow this step:

- Select Help ▸ Index or type `INDEX` from the Help Tutorial. The Help Index, shown in Figure 111, displays.

**Note:**

Typing an alphabetic character on any Index screen displays the Index screen corresponding to that character.

**Figure 111 • Help Index**

```
ASG-SmartTest - R7.0 ----------  Index A - B  ---------------------------  HELP
===> _

To select a topic, enter the 2- or 3-character identifier.

A1   - ABENDS
A2   - ADD Command                      B1  - BackTrack Facility
A3   - Address Stop Entry Screen        B2  - BackTrack Variable History Pop-up
A4   - AKR Directory                    B3  - BackTrack Variable History Screen
A5   - AKR Utilities                    B4  - BRANCH Command
A6   - ALLOCDEF Command                 B5  - BREAK Command
A7   - ALLIANCE Command                 B6  - Breakpoints List Screen
A8   - ALTPCB Message Queue List
       Screen
A9   - ALTPCB Segment List Screen
A10  - ANALYZE Command
A11  - Analyze Options


 Another index page can be displayed by entering its letter.
```

# Help - About Pop-up

To display information about the current releases of SmartTest and Center, follow this step:

▶ Select Help ▶ About. The Help - About pop-up, shown in Figure 112, displays.

**Figure 112 • Help - About Pop-up**

```
                    Help - About

 The following is release information for this ASG
 product.

  Product name . . . . . . . : ASG-SMARTTEST
   Release number  . . . . . : 7.0
   Maintenance level . . . . : 001

  ASG-CENTER release number  : 7.0
   Maintenance level . . . . : 001

  Operating system . . . . . : OS(390)
```

## Fields

| Field | Description |
|---|---|
| Product Name | Displays SmartTest, or the name of the ESW product you are currently running. |
| Release Number | Specifies the current release number of SmartTest you are currently running. |
| Maintenance Level | Specifies the current SmartTest PTF number. |
| Center Release Number | Specifies the current release number of Center. |
| Center Maintenance Level | Specifies the current Center PTF number. |
| Operating System | Specifies the operating system that you are running. |

This information is needed when contacting ASG Customer Support group for assistance.

# 7 Options

This chapter describes the options available on the Options pull-down menu and contains these sections:


| Section | Page |
|---|---|
| Options Pull-down | 186 |
| Options - Product Parameters Pop-up | 187 |
| Options - Product Allocations Pop-up | 188 |
| Options - Log/List/Punch Definition Pop-up | 190 |
| Options - Script File Allocations Pop-up | 192 |
| Options - PF Key Definition Pop-up | 193 |
| Options - Modes Screen | 195 |
| Options - Scratchpad Pop-up | 202 |
| Options - Scratchpad Equate Pop-up | 203 |
| Options - Scratchpad Mark Pop-up | 204 |
| Options - Scratchpad Copy Pop-up | 205 |
| Options - Scratchpad Delete Pop-up | 207 |
| Options - Scratchpad Merge Pop-up | 207 |
| Options - Scratchpad Rename Pop-up | 209 |

# Options Pull-down

Selecting Options on the action bar displays the Options pull-down shown in Figure 113. The Options pull-down is used to customize your SmartTest environment by setting certain parameters and options.

**Figure 113 • Options Pull-down**

```
 File  View  Test  Search  List  Options  Help
-------------------------------                          ---------
                                AS   _  1. Product parameters...
Command ===> ------------------------   2. Product allocations...      ----------
                                        3. Log/list/punch...
                                        4. Script file allocations...  *****
                                        5. PF keys...                  ****
                                        6. Modes - SET Options...
                                        7. Equate...
                                                                        *

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

## Actions

| Action | Description |
|---|---|
| Product parameters | Displays the Options - Product Parameters pop-up used to set parameters that affect the online operation of SmartTest. |
| Product allocations | Displays the Options - Product Allocations pop-up used to specify DASD volumes for the Log, List, Punch, and Work files, and to specify space for the Work file. |
| Log/list/punch | Displays the Options - Log/List/Punch Definition pop-up used to set values for allocating, formatting, and processing the SmartTest Log, List, and Punch files. |
| Script file allocations | Displays the Options - Script File Allocations pop-up used to modify the default script file dataset names for your session. |
| PF Keys | Displays the Options - PF Key Definition pop-up used to edit PF key values. |

| Action | Description |
|---|---|
| Modes | Displays the Options - Modes pop-up used to enable and disable SmartTest processing modes (LEARN, SCRIPT, XMODE). |
| Equate | Displays the Options - Equate pop-up used to define equates. |

**Note:**

If Insight is installed, this option displays as Scratchpad. Selecting this option displays the Options - Scratchpad pop-up, which is used to define equates, and to create, copy, delete, merge, or rename marks.

# Options - Product Parameters Pop-up

To set parameters that affect the online operation of SmartTest, follow this step:

- Select Options ▸ Product parameters, or type `PARMDEF`. The Options - Product Parameters pop-up, shown in Figure 114, displays.

**Figure 114 • Options - Product Parameters Pop-up**

```
                    Options - Product Parameters
Command ===> ________________________________________________

  Alarm  . . . . . . . YES    (YES or NO)

  Save pseudo code . . YES    (YES or NO)
  Save equates . . . . YES    (YES or NO)

  Cursor character . . %      (Token substitution character)
```

### Fields

| Field | Description |
|---|---|
| Alarm | Controls the audible alarm on the terminal. If YES, the alarm sounds when an error message displays. |
| Save pseudo code | Specifies the default for saving pseudo code (Pseudo code, Breakpoints, and When Conditions) when exiting from a program. |
| Save equates | Specifies the default for saving equates when exiting from a program. |
| Cursor Character | Sets the cursor token substitution character. The cursor substitution character can be used in any primary command during a test session. Any character can be specified as the token substitution character; however, it should be a character that is unique and rarely used in commands. The default substitution character is the percent sign (%). |

## Options - Product Allocations Pop-up

To specify the DASD volumes for the Log, List, Punch, and Work files, and to specify disk storage space for the Work file, follow this step:

- Select Options ▸ Product Allocations, or type `ALLOCDEF` on the command input line. The Options - Product Allocations pop-up, shown in Figure 115, displays.

**Figure 115 • Options - Product Allocations Pop-up**

```
                    Options - Product Allocations
Command ===> ____________________________________________________________

  Log file:
     Generic unit . . . SRTDA     (generic group name or unit address)
     Volume serial  . . SRT801    (blank for authorized default volume)
  List file:
     Generic unit . . . SYSDA     (generic group name or unit address)
     Volume serial  . . SRT801    (blank for authorized default volume)
  Punch file:
     Generic unit . . . SRTDA     (generic group name or unit address)
     Volume serial  . . SRT801    (blank for authorized default volume)
  Work file:
     Generic unit . . . SYSDA     (generic group name or unit address)
     Volume serial  . . ______    (blank for authorized default volume)
     Space units  . . . CYLS      (BLKS, TRKS or CYLS)
     Primary space  . . 5         (space units)
     Secondary space    5         (space units)
```

## Fields

| Field | Description |
|---|---|
| Log file | Specifies the SMS classes or the device type and volume serial number for the Log file that is allocated upon entry into SmartTest. The Log file is used for error messages and log commands. File characteristics are specified on the Options - Log/List/Punch Definition pop-up. |
| List file | Specifies the SMS classes or the device type and volume serial number for the List file that is allocated the first time a request is made to print output. The List file is used for all printed output. File characteristics are specified on the Options - Log/List/Punch Definition pop-up. |
| Punch file | Specifies the SMS classes or the device type and volume serial number for the Punch file that is allocated the first time a request is made to punch output. The Punch file is used for all punched output. File characteristics are specified on the Options - Log/List/Punch Definition pop-up. |
| Work file | Specifies the SMS classes or the device type, volume serial number, and space requirements for the Work file that is allocated upon entry into SmartTest. The Work file is a temporary file. The size of the programs being worked with should be considered when determining space requirements. Approximately one cylinder (on a 3380 device) is required for a 5,000 line program. |

# Options - Log/List/Punch Definition Pop-up

To set values for allocating, formatting, and processing the Log, List, and Punch files, follow this step:

- Select Options ▸ Log/list/punch, or type `PRINTLOG` or `PRINTLST` on the command input line. The Options - Log/List/Punch Definition pop-up, shown in Figure 116, displays.

**Figure 116 • Options - Log/List/Punch Definition Pop-up**

```
                    Options - Log/List/Punch Definition
Command ===> ____________________________________________________________

1 - Process log file  2 - Process list file  3 - Process punch file
                      4 - Customized data set name

Options                     Log            List           Punch
----------------            ---            ----           -----
Process option  . . . . . . K              PK             PK
Primary tracks     . . . . 1              1              1
Secondary tracks     . . . 2              5              5
Lines per page  . . . . . . 56             56             56
Sysout class  . . . . . . . *              *              *

Process options:  PK (print/keep), PD (print/delete), K, or D.


Job statement information:
  //USER1 JOB (DEVTTG,283200,SRT,00),'USER1',PRTY=6,
  //         MSGCLASS=X
  //*JOBPARM SYSAFF=CPUC
  //*
```

## Options

| Option | Description |
|---|---|
| 1 - Process log file | Processes the Log file. Verify or change the Options for the Log file, then select option 1 to process the Log file. A new file is allocated to collect additional data, if required. If the PK or PD processing option is specified, Job statement information must be entered prior to selecting option 1. The Log file is processed when you press Enter. It is not necessary to exit SmartTest to process the Log file. |
| 2 - Process list file | Processes the List file. Verify or change the Options for the List file, then select option 2 to process the List file. A new file is allocated to collect additional data, if required. The file name is in the format of *`userid.yyy.xxxxx`*.VIALIST where *`yyy`* is the product ID and *`xxxxx`* is a unique member. If the PK or PD processing option is specified, Job statement information must be entered prior to selecting option 2. The List file is processed when you press Enter. It is not necessary to exit SmartTest to process the List file. |

| Option | Description |
|---|---|
| 3 - Process punch file | Processes the Punch file. Verify or change the Options for the Punch file, then select option 3 to process the Punch file. This file is defined as fixed block with 80 byte records. The file name is in the format: *userid*.*yyyxxxxxx*.VIAPUNCH; where *yyy* is the product ID and *xxxxxx* is a unique member. Required Job statement information must be entered prior to selecting this option. It is not necessary to exit SmartTest to create the Punch file. |
| 4 - Customized data set name | Enables you to customize the dataset name for Log, List, and Punch files. If you specify the K or PK process option for the Options for the Log, List and Punch files, then you can specify a dataset where the Log, List, or Punch files will be allocated. Select option 4 to display the Options - Log/List/Punch Name Customization screen. |

## Fields

| Field | Description |
|---|---|
| Process option | Specifies one of the listed processing options: PK to print and keep, PD to print and delete, K to keep without printing, or D to delete without printing. The default is PD for the Log and List files, and PK for the Punch file. |
| Primary tracks | Specifies the number of primary tracks to allocate. A size change does not take effect until the next allocation occurs. The default for the Log, List, and Punch files is 1. |
| Secondary tracks | Specifies the number of secondary tracks to allocate. A size change does not take effect until the next allocation occurs. The default for the Log file is 2 and the default for the List and Punch files is 5. |
| Lines per page | Specifies the number of print lines per page. Typical maximum values are 60 for six lines per inch and 80 for eight lines per inch. The default is 56. |
| Sysout class | Specifies the SYSOUT class value. The default is *, which sends the SYSOUT to the destination specified in the MSGCLASS parameter on the JOB statement. |

| Field | Description |
|---|---|
| Process options | Lists the available options for the Log, List, and Punch files: PK - Print and keep, PD - Print and Delete, K - Keep without printing, and D - Delete without printing.<br><br>**Note:**<br>If you specify the K or PK process option, you can customize the dataset where the log, list, or punch file is allocated. By default, SmartTest allocates the Log, List, and Punch files as USERID.STT*nnnnn*.VIA*xxxxx*, where *nnnnn* is a sequential number from 00001 to 99999 and *xxxxx* is LOG for Log, LIST for List, and PUNCH for Punch files. If you have specified a TSO Prefix, the prefix will be appended to the beginning of the file name allocated for the Log, List, and Punch files. |
| *xxxxx* FILE IS ALLOCATED | Displays when the Log, List, or Punch file has been properly allocated. If the message does not appear, check the assignments on the Options - Allocation Definition pop-up. *xxxxx* is LOG, LIST, or PUNCH depending on the file that was allocated. |
| Job statement information | Specifies the appropriate JOB statement information for your installation. These JCL statements are required if the PK or PD processing option has been specified. |

# Options - Script File Allocations Pop-up

To display and modify the default script file concatenation sequence for your session, follow this step:

- Select Options ▸ Script file allocations. The Options - Script File Allocations pop-up, shown in Figure 117, displays.

**Figure 117 • Options - Script File Allocations Pop-up**

```
        Options - Script File Allocations
Command ===> ____________________________________

     R - Restore default Script allocations

Enter desired Script file concatenation.

Script file data set names:

  ----------------------------------------------
  ----------------------------------------------
  ----------------------------------------------
  ----------------------------------------------
```

## Fields

| Field | Description |
|---|---|
| Restore default Script allocations | Restores the defaults set during the install process. |
| Script file data set names | Specifies the dataset names for the desired default script files. The files are concatenated in the order entered. |

**Note:**

When initializing for defaults, the lines are filled from the bottom to allow concatenation.

## Usage Notes

When SmartTest is installed, the default script files are specified for the site. You can use the Options - Script File Allocations pop-up to modify the default dataset names or the concatenation sequence for the script files. The changes are maintained in your profile.

# Options - PF Key Definition Pop-up

To display and/or redefine the current PF key settings, follow this step:

- Select Options ▸ PF keys or type KEYS on the command line.

The Options - PF Key Definition pop-up, shown in Figure 118, displays. PF keys 1 through 12 are displayed initially. A similar pop-up exists for PF keys 13 through 24 and can be displayed by pressing Enter.

**Figure 118 • Options - PF Key Definition Pop-up**

```
                 Options - PF Key (01-12) Definition
Command ===> _____________________________________________________

Press Enter to process changes and/or to display alternate keys.
Press PF3/15 (END) to exit.

       Number of PF keys: 24        Terminal type: 3278

PF01 HELP
PF02 SPLIT
PF03 END
PF04 RUN
PF05 RFIND
PF06 STEP
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 BRANCH
PF11 BRANCH BACKUP
PF12 RECALL
```

## Fields

| Field | Description |
|---|---|
| Number of PF keys | Specifies the number of supported PF keys. |
| Terminal type | Indicates the type of terminal being used. SmartTest supports these 3270 type terminals:<br>• Model 2 (24 lines x 80 columns)<br>• Model 3 (27 lines x 80 columns)<br>• Model 4 (43 lines x 80 columns)<br>• Model 5 (27 lines x 133 columns) |
| PF01 - PF12 or PF13 - PF24 | Specifies the value assigned to the PF keys, which can be changed by typing over it. To reset a PF key to its default value, delete the entry and press Enter. To set a PF key to have no value (i.e., to turn off a PF key) enter the value NOP. This field is required. |

## Usage Notes

When SmartTest is installed, all PF keys are set to default values. This table lists the default values:

| PF Key Defaults | | | |
|---|---|---|---|
| Primary Key | Default | Alternate Key | Default |
| PF01 | HELP | PF13 | LIST MEMORY |
| PF02 | SPLIT | PF14 | SPLIT |
| PF03 | END | PF15 | END |
| PF04 | RUN | PF16 | RUN TO |
| PF05 | RFIND | PF17 | REPEAT |
| PF06 | STEP | PF18 | STEP OVER |
| PF07 | UP | PF19 | LIST |
| PF08 | DOWN | PF20 | SET |
| PF09 | SWAP | PF21 | SWAP |

| PF Key Defaults | | | |
|---|---|---|---|
| PF10 | BRANCH | PF22 | LEFT |
| PF11 | BRANCH BACKUP | PF23 | RIGHT |
| PF12 | RECALL | PF24 | RECALL |

**Note:**

The alternate PF keys may be PF01 through 12 or PF13 through 24, depending on your site's specifications.

# Options - Modes Screen

To enable or disable the mode indicated by the specified option, follow this step:

- Select Options ▸ Modes or type SET with no operands. The Options - Modes screen, shown in Figure 119 displays.

**Figure 119 • Options - Mode Screen**

```
                                Options - Modes
Command ===> ____________________________________________________ Scroll ===> CSR
ASG2343I USE THE DOWN COMMAND FOR ADDITIONAL TEST SESSION OPTIONS.
  Option     Set     Description
----------   ----   ------------------------------------------------------------
ASM          ON     Display Assembler code in status box, and instruction STEP
ASMVIEW      OFF    Display Assembler code for programs not on the AKR
AUTOQUAL     ON     QUALIFY to the Active Program after a RUN or STEP
BACKTRACK    OFF    BACKtrack Recording mode is disabled with a buffer of 1M
BREAKS       ON     The BREAKpoint facility is enabled
COLUMNS      OFF    The COLUMNS option is disabled
CUA          ON     The CUA Menu facility is enabled
DATA         AUTO   Number of lines for Zoom Data windows, or AUTO
DELAY        1      Number of seconds to delay between steps during STEP AUTO
FLOATING     OFF    Display floating point registers in the status box
GENERATED    ON     Display generated code with the original source code
HEX          OFF    Display data in hexadecimal format
KEEP         AUTO   Number of lines for KEEP window, or AUTO
LANGUAGE     COB    The current LANGUAGE for the test session is COBOL
LE           OFF    Normal Language Environment error processing
LEARN        OFF    Display internally generated Primary Commands
LINK         OFF    Monitor LINKed-to programs for the test session
MAIN         ON     The program being tested is a MAINLINE
MONITOR      ON     The default for the RUN command is MONITOR
```

You can directly enable or disable each mode shown on this screen using the SET command with the corresponding operand. The current setting for each Test Session option is saved between sessions with the exception of BREAK, PSEUDO, SCRIPT, and WHEN. Defaults for these options are restored when a new session is initiated.

**Note:**

Scroll the screen forward to display modes that do not appear. The STOPEXEC and STOPHAND modes only appear if the current environment is CICS.

## Fields

| Field | Description |
|---|---|
| Option | Specifies the mode to be enabled or disabled. All options can be set to their default values by entering `SET DEFAULT`. |
| Set | Indicates the status of the mode or operand (ON, OFF, AUTO, value); the AUTO status automatically assigns a value if you do not specify one. Type over the status in the SET column to change the status of the corresponding operand. |
| Description | Describes the mode (option) being enabled or disabled. Breakpoint, Monitor, Pseudo code, Script, and When Condition facilities indicate whether they are enabled or disabled since they control testing conditions. |
| ASM | Determines if the status box is to contain Assembler information. This information includes the offset, addressing mode, Assembler instruction, hexadecimal value, and the general register contents. When this mode is ON, Assembler information displays in the status box and the increment for the STEP command is at the Assembler instruction level. When this mode is OFF, the STEP command increment is at the COBOL, Assembler, or PL/I source statement level. The default is OFF. |
| ASMVIEW | Determines if Assembler code for a program that is not in the AKR is to be displayed automatically. When this mode is ON, Assembler code for the specified program displays if the program does not exist in the AKR. When this mode is OFF, a message displays indicating the source is unavailable displays. The default is OFF. With ASMVIEW set ON, STEPping into a program that is not on the AKR automatically STEPs at the Assembler instruction level and display registers in the status box. |

| Field | Description |
|---|---|
| AUTOQUAL | Determines if the program being tested is automatically displayed when a STEP or RUN command is issued after qualifying a program. When this mode is ON and a program has been qualified using the QUALIFY command, a STEP or RUN command causes the program being tested to be displayed instead of the qualified program. The calling program displays if an error condition occurs in a CALLed program when this mode is ON. When this mode is OFF, the qualified program remains displayed and the status box is updated for the program being tested, not the qualified program. The default is ON. |
| BACKTRACK | Controls the Backtrack Recording facility and the Backtrack Review facility. SET BACKTRACK sets the Backtrack Recording mode ON or OFF, and optionally sets the size of the Backtrack Recording buffer to other than the default size of 1 megabyte (100 KB for CICS). The buffer size affects how much statement execution history is available to the Backtrack Review facility. When the buffer fills, SmartTest begins overwriting the oldest information, thus saving the more recent execution history. Entering SET BACKTRACK with no other operands toggles the current mode ON and OFF. Backtrack Recording mode can be turned ON only during an active test. Toggling or turning the BACKTRACK Recording mode OFF causes the recording buffer to be discarded/destroyed. The default is OFF. |
| BREAKS | Enables or disables Breakpoints set in the program being tested and/or displayed on the Program View screen. When this mode is ON, Breakpoints are enabled unless the PSEUDO option is OFF. Breakpoints are considered to be pseudo code and therefore, are disabled regardless of the BREAKS setting when the PSEUDO option is OFF. The default is ON. |
| COLUMNS | Displays program source with a ruler running across the top of the screen. If the ruler displays, selecting COLUMNS removes it. |
| CUA | Enables or disables the CUA interface. During a test session a SmartTest user may toggle between the CUA and non-CUA interface. With CUA turned off, two extra lines of program source can be viewed on the screen. |
| DATA | Controls the number of lines used by all Zoom Data windows (excluding the lines used to display the outline around the window). When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. The default is AUTO. |

| Field | Description |
|---|---|
| DELAY | Controls the number of seconds to delay between steps when using the STEP AUTO command. The default is 1. |
| FLOATING | Displays the floating point registers in the program status box when this mode is ON. The STATUS option must also be ON or this option is ignored. The default is OFF. |
| GENERATED | Specifies whether the program functions as a standard COBOL program, with all SmartTest commands available and functioning normally. When OFF, the program functions at the Painter statement level. This operand is only available if SmartTest-APS is installed. The default is OFF. |
| HEX | Displays data values in hexadecimal format when this mode is ON. The default is OFF. |
| KEEP | Controls the number of lines used by the Keep window (excluding the separator line at the bottom). When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. The default is AUTO. |
| LANGUAGE | Specifies the SmartTest CUA interface language. SmartTest makes access available to debugging facilities that apply to the language of the active program being tested. Application programmers that maintain either COBOL, PL/I, or a mixture of both, find the user interface improves productivity because it is more intuitive for doing the debugging task at hand. |
| LEARN | Displays primary commands that are generated internally when actions on the pull-downs are executed. |
| LE370 | Specifies whether the LE370 run-time environment (Language Environment for 370 programs) is supported for programs compiled with the COBOL/370 or PL/I 370 compiler. This allows non-disruptive monitoring of an LE/370 program's initialization, termination, and condition handling. The default is OFF. |
| LINK | Monitors linked programs for the test session when this mode is ON. The default is OFF. This does not appear in the CICS environment. |

| Field | Description |
|---|---|
| MAIN | Identifies the program to be tested as a mainline routine executed by coding JCL EXEC PGM=, or as a subroutine that is called by another program. SET MAIN ON, the default, identifies the program as a mainline routine. SET MAIN OFF identifies the program to be tested as a subroutine. SET MAIN can only be turned OFF for COBOL programs with source code on the AKR. Entry to the COBOL program may be either a PROCEDURE DIVISION USING statement or an ENTRY USING statement.<br><br>When SET MAIN is OFF, data used by the subroutine is stored and initialized via the linkage section. Storage is initialized to zero for numeric items and to spaces for nonnumeric items. Data in the subroutine may be user modified using the ZOOMDATA command after stepping through the procedure division initialization code.<br><br>This operand is valid only in the TSO (Standard) environment. |
| MONITOR | Specifies whether the MONITOR option of the RUN command is the default. When this mode is OFF, the NOMONITOR option of the RUN command is the default. The default is ON. This does not appear in the CICS environment. See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about MONITOR. |
| OPERANDS | Specifies whether the current values for the data items on the current statement are displayed in the long message area after each STEP or RUN command. If the current statement has only one data item, its name and as much of the current value as possible are displayed. If the current statement contains more than one data item, a maximum of sixteen characters of the name and the current value for each data item are displayed. The default is OFF. |
| OUTLINE | Determines how command and/or function results are displayed on the Program View screen. When this mode is ON, these results are enclosed in a box. The default is ON. |
| PROMPT | Enables or disables the prompt to save the current Environment profile if there have been changes. |
| PSEUDO | Enables or disables the pseudo code facility for the program being tested and/or displayed on the Program View screen. When this mode is ON, the pseudo code facility is enabled. When this OFF, any attempt to execute pseudo code is ignored, including Breakpoints and When Conditions, regardless of the BREAKS or WHENS mode settings. The default is ON. |

| Field | Description |
|---|---|
| REFRESH | Refreshes the full ISPF screen after each STEP or RUN command when this mode is ON. The default is ON. This does not appear in the CICS environment. |
| REGISTERS | Displays the general registers in the program status box when this mode is ON. The STATUS option must also be ON or this option is ignored. The default is OFF. |
| SCALE | Specifies whether a scale line displays above alphanumeric and hexadecimal format data items within Zoomdata and KEEP windows. The default is OFF. |
| SCRIPT | Generates a script file automatically. When this mode is ON, all SmartTest primary commands entered during a test session are saved. Using this mode allows inclusion of a predefined command sequence in a test session. This option or the SET SCRIPT ON command allocates and opens a dataset and saves all SmartTest primary commands in this dataset. The dataset name is in the format: TSOUSERID.STT*nnnnn*.VIASCRIP where *nnnnn* is a number beginning with 00001 that is incremented. The name of the dataset displays when it is opened or closed. Scripts support all SmartTest primary commands. The script file is closed (ignored) when this mode is OFF or the SET SCRIPT OFF command is entered. Once a script file is closed, it can be used as the dataset name in the EXECUTE command. The default is OFF. |
| SHADOW | Determines if excluded lines are shown as a line of dashes. When this mode is ON, excluded lines are shown as a row of dashes with the number of excluded lines indicated at the right. When OFF, there is no indication of excluded lines other than the sequence numbers shown in the prefix area. The default is ON. |
| STATUS | Determines if the current status box displays during a test session. When this mode is ON, the status box displays. The content of the status box varies depending on the value of the ASM, FLOATING, and REGISTERS modes. A status box automatically displays when an error condition occurs even if this mode is OFF. The default is ON. |
| STOPEXEC | Stops execution before each EXEC CICS request. When execution is stopped, the Status Box description indicates the type of EXEC CICS request that is about to be made (e.g., STOPPED BEFORE SEND MAP). Issuing the STEP or RUN command continues execution. SET STOPEXEC affects all monitored programs for the current user including those that are not analyzed into the AKR. This displays only in the CICS environment. |

| Field | Description |
|---|---|
| STOPHAND | Stops execution after a CICS Handle condition has been raised. When execution is stopped, the Status Box description indicates the type of Handle condition that is being processed (e.g., `STOPPED AFTER HANDLE MAPFAIL`). Issuing the STEP or RUN command continues execution. SET STOPHAND affects all monitored programs for the current user including those that are not analyzed into the AKR. This mode displays only in the CICS environment. |
| TRACK | Sets the number of execution tracing entries. The number of execution entries can be set from 0 to 9999 or 0 KB to 512 KB. The default is 1024. |
| VALUES | Controls the number of lines used to display each alphanumeric data item and each data item displayed in hexadecimal format within a Zoom Data or the Keep window. When the number of lines required to display a data item exceeds the number of lines specified for the VALUE option, the window becomes horizontally scrollable. The number of lines can be set from 1 to 99. The default is AUTO. |
| WHENS | Enables or disables the When Conditions facility for the program being tested and/or viewed on the Program View screen. When ON, the When Conditions facility is enabled unless the PSEUDO option is OFF. When Conditions are regarded as pseudo code and therefore, are disabled regardless of the WHENS. |
| WRAP | Enables or disables a prompt when the tracking table is full. This allows you to save the tracking table before it is overlaid. |
| XMODE | Determines if lines are excluded from the screen before a primary command is executed. When this mode is ON, all lines are excluded before a primary command is executed. This mode affects all primary commands except BRANCH, LOCATE, RFIND, SCROLL, and WHEN. The default is OFF. |
| ZEROFILL | When this mode is ON, leading zeros display on numeric data displays. This option affects numeric data items displayed using the ZOOMDATA, KEEP, and DISPLAY commands. The default is OFF. |

# Options - Scratchpad Pop-up

To define equates and create, copy, delete, merge, or rename marks, follow this step:

- Select Options ▸ Scratchpad. The Options - Scratchpad pop-up, shown in Figure 120, displays.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 120 • Options - Scratchpad Pop-up**

```
                Options - Scratchpad

Select the desired option to create or modify
Equates and Marks.  Then press Enter.

        Options
        1   1.  Equate...
            2.  Mark...
            3.  Copy...
            4.  Delete...
            5.  Merge...
            6.  Rename...
```

## Fields

| Option | Description |
|---|---|
| Equate | Displays the Options - Scratchpad Equate pop-up used to define a name for a character string. |
| Mark | Displays the Options - Scratchpad Mark pop-up used to save the requested target as a mark set or path. |
| Copy | Displays the Options - Scratchpad Copy pop-up used to copy a mark set or path. |
| Delete | Displays the Options - Scratchpad Delete pop-up used to delete a mark set or path. |
| Merge | Displays the Options - Scratchpad Merge pop-up that is used to add the lines from the specified target to the specified mark name. |
| Rename | Displays the Options - Scratchpad Rename pop-up used to change the name of a mark path or set of lines. |

# Options - Scratchpad Equate Pop-up

The Options - Scratchpad Equate pop-up, shown in Figure 121, displays by selecting Equate on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 121 • Options - Scratchpad Equate Pop-up**

```
                    Options - Scratchpad Equate
To create an equate, type a name and string.  Then press Enter.
Equate name     __________
Equated string ____________________________________________________
```

## Fields

| Field | Description |
|---|---|
| Equate name | Specifies the name you want to assign to the character string. Names must be 1 to 10 alphanumeric characters; the first character must be alphabetic or a period; a hyphen is the only special character allowed; names may also be 1 to 4 DBCS characters. |
| Equated string | Specifies a character string to be substituted by the equate. A character string can be a long command, pattern, dataname, concatenated dataname, etc. Literals and blanks can be specified in the character string if desired. Character strings that include blanks must be enclosed in single or double quotes. The string can be a DBCS string. |

# Options - Scratchpad Mark Pop-up

The Options - Scratchpad Mark pop-up, shown in Figure 122, is used to save the requested target as a mark set or path. This pop-up displays by selecting Mark on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 122 • Options - Scratchpad Mark Pop-up**

```
                    Options - Scratchpad Mark

To create a mark, type a name and a target.  Then press Enter. For a
name selection list for a target type, type a pattern (e.g. ABC*) in
the target name area.

Mark name . . _________

Target name   ___________________________________________________

Target type
1   1.  Mark name
    2.  Perfrange name
    3.  Program name
    4.  Subset name
    5.  Line range
    6.  Label name

Comments  . . ______________________________________________
```

## Fields

| Field | Description |
|---|---|
| Mark name | Specifies the name you want to assign to the set of lines or path. Names must be 1 to 10 alphanumeric characters. |
| Target name | Specifies the name of the target you want to save. |
| Target type | |
| Mark name | Specifies a 1 to 10 alphanumeric character name given to a set or path, or one of these system-generated paths:<br>TRACK \| TRK<br>NETWORK \| NET<br>SUBNET \| SUB*n* |
| Perfrange name | Specifies the name assigned in a PERFORM statement, including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |

| Field | Description |
|---|---|
| Program name | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| Subset name | Specifies a predefined COBOL language subset. |
| Line range | Specifies a single line number or range of lines. |
| Label name | Specifies any PROCEDURE DIVISION paragraph name or section name. The PROCEDURE and PROC literals can also be specified. |
| Comments | Specifies a description or other comments for the mark. |

# Options - Scratchpad Copy Pop-up

The Options - Scratchpad Copy pop-up is used to copy the contents of a path or set of lines to a mark. Figure 123 displays by selecting Copy on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 123 • Options - Scratchpad Copy Pop-up**

```
                    Options - Scratchpad Copy

To create a mark, type a name and a target.  Then press Enter. For a
name selection list for a target type, type a pattern (e.g. ABC*) in
the target name area.

Mark name . . _________

Target name   ___________________________________________________

Target type
1   1.  Mark name
    2.  Perfrange name
    3.  Program name
    4.  Subset name
    5.  Line range
    6.  Label name

Comments  . . ___________________________________________________
```

## Fields

| Field | Description |
|---|---|
| Mark name | Specifies the name you want to assign to the set of lines or path. Names must be 1 to 10 alphanumeric characters. |
| Target name | Specifies the name of the target you want to save. |
| Target type | |
| Mark name | Specifies a 1 to 10 alphanumeric character name given to a set or path, or one of these system-generated paths:<br>TRACK \| TRK<br>NETWORK \| NET<br>SUBNET \| SUB*n* |
| Perfrange name | Specifies the name specified in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |
| Program name | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| Subset name | Specifies a predefined COBOL language subset. |
| Line range | Specifies a single line number or range of lines. |
| Label name | Specifies any PROCEDURE DIVISION paragraph name or section name. The PROCEDURE and PROC literals can also be specified. |
| Comments | Specifies a description or other comments for the mark. |

# Options - Scratchpad Delete Pop-up

The Options - Scratchpad Delete pop-up, shown in Figure 124, is used to delete marks. This pop-up displays by selecting Delete on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 124 • Options - Scratchpad Delete Pop-up**

```
                    Options - Scratchpad Delete

 To delete a mark, type the mark name.  Then press Enter.

                    Mark name . . _________
```

## Field

**Mark name.** Specifies the name of the mark you want to delete.

# Options - Scratchpad Merge Pop-up

The Options - Scratchpad Merge pop-up, shown in Figure 125, is used to add the lines from the specified target to the Mark name. This pop-up displays by selecting Merge on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 125 • Options - Scratchpad Merge Pop-up**

```
                    Options - Scratchpad Merge

 To expand a mark, type the name and target.  Then press Enter. For a
 name selection list for a target type, type a pattern (e.g. ABC*) in
 the target name area.

 Mark name . . _________

 Target name   ____________________________________________________

 Target type
 1   1.  Mark name
     2.  Perfrange name
     3.  Program name
     4.  Subset name
     5.  Line range
     6.  Label name

 Comments  . . ____________________________________________________
```

## Fields

| Field | Description |
|---|---|
| Mark name | Specifies the name you want to assign to the set of lines or path. Names must be 1 to 10 alphanumeric characters. |
| Target name | Specifies the name of the target you want to save. |
| Target type | |
| Mark name | Specifies a 1 to 10 alphanumeric character name given to a set or path, or one of these system-generated paths:<br>TRACK \| TRK<br>NETWORK \| NET<br>SUBNET \| SUB*n* |
| Perfrange name | Specifies the name assigned in a PERFORM statement, including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |
| Program name | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| Subset name | Specifies a predefined COBOL language subset. |
| Line range | Specifies a single line number or range of lines. |
| Label name | Specifies any PROCEDURE DIVISION paragraph name or section name. The PROCEDURE and PROC literals can also be specified. |
| Comments | Specifies a description or other comments for the mark. |

# Options - Scratchpad Rename Pop-up

The Options - Scratchpad Rename pop-up is used to rename marks. Figure 126 displays by selecting Rename on the Options - Scratchpad pop-up.

**Note:**

This pop-up is available only if Insight is installed.

**Figure 126 • Options - Scratchpad Rename Pop-up**

```
Options - Scratchpad Rename

To rename a mark, type the old and new
name.  Then press Enter.

Old mark name __________

New mark name __________
```

## Fields

| Field | Description |
|---|---|
| Old mark name | Specifies the mark name that you want to change. |
| New mark name | Specifies the new name that you want to assign to the mark. The new name must be 1 to 10 alphanumeric characters. |

# 8 Commands

This chapter describes how to use the SmartTest commands and contains these sections:


| Section | Page |
|---|---|
| Command Processing | 216 |
| Command Diagrams | 216 |
| Command Syntax | 218 |
| Repeating Commands | 219 |
| Cursor Position | 222 |
| Cursor Substitution Character | 222 |
| Double Byte Character Set (DBCS) Strings | 223 |
| & (Retain) Command | 224 |
| ADD Command | 225 |
| ALLIANCE Command | 226 |
| ALLOCDEF Command | 227 |
| ANALYZE Command | 228 |
| BRANCH Command | 229 |
| BREAK Command | 232 |
| CANCEL Command | 241 |
| CONVERT Command | 242 |
| COPY Command | 243 |

| Section | Page |
|---|---|
| CURRENT Command | 246 |
| DELETE Command | 247 |
| DISPLAY Command | 250 |
| DROP Command | 253 |
| DUMP Command | 254 |
| END Command | 258 |
| ENVIRONMENT Command | 259 |
| EQUATE Command | 260 |
| EXCLUDE Command | 262 |
| EXECUTE Command | 270 |
| FIND Command | 274 |
| FINDXTND Command | 279 |
| FLOW Command (without Insight) | 292 |
| FLOW Command (Insight Only) | 294 |
| FORCE Command | 299 |
| GO Command | 300 |
| HELP Command | 301 |
| HIGH Command | 303 |
| KEEP Command | 310 |
| KEYS Command | 314 |
| LIST Command | 316 |
| LOCATE Command | 330 |
| LPRINT Command | 333 |

| Section | Page |
|---|---|
| LPUNCH Command | 342 |
| MARK Command | 350 |
| MERGE Command | 353 |
| MOVE Command | 356 |
| NEWCOPY Command | 357 |
| PARMDEF Command | 358 |
| PREF Command | 359 |
| PRINTLOG Command | 362 |
| PRINTLST Command | 363 |
| PROCESS Command | 364 |
| PRODLVL Command | 368 |
| QUALIFY Command | 369 |
| RECALL Command | 371 |
| REDO Command | 374 |
| REFRESH Command | 376 |
| RENAME Command | 377 |
| REPEAT Command | 379 |
| RESET Command | 380 |
| RETURN Command | 382 |
| RFIND Command | 383 |
| RHIGH Command | 384 |
| RPREF Command | 385 |
| RSCROLL Command | 386 |

| Section | Page |
|---|---|
| RTRACE Command | 387 |
| RUN Command | 390 |
| RUN Command (CICS Only) | 393 |
| RUN Command (BACKTRACK ON) | 396 |
| SAVE Command | 399 |
| SCROLL Command | 401 |
| SELECT Command | 408 |
| SET Command | 409 |
| SETUP Command | 419 |
| SHOW Command (CICS Only) | 420 |
| STEP Command (BACKTRACK OFF) | 421 |
| STEP Command (BACKTRACK ON) | 425 |
| STOP Command | 429 |
| SUBTRACT Command | 431 |
| TCA DEFINE Command | 432 |
| TCA LIST Command | 433 |
| TCA RECORD Command | 434 |
| TCA REPORT Command | 435 |
| TCA RUN Command | 436 |
| TEST Command | 437 |
| TESTPOINT Command | 438 |
| TOGGLE Command | 439 |
| TRACE Command | 440 |

| Section | Page |
|---|---|
| UPDATE Command | 446 |
| USING Command | 448 |
| UTILITY Command | 450 |
| VIEW Command | 451 |
| WHEN Command | 452 |
| WHERE Command | 456 |
| WIZARD Command | 458 |
| ZOOMDATA Command | 459 |
| ZOOMIN/ZOOMOUT Commands | 462 |
| SmartTest Line Commands | 464 |


SmartTest is an interactive product that can be executed using simple, easily understood commands. SmartTest accepts primary and line commands that are entered in the same manner as ISPF commands. Primary commands are entered in the command input area (usually line two on a pop-up or line four on a screen); line commands are entered in the prefix area (columns 1 through 6) over the line numbers. SmartTest supports all ISPF system commands on the appropriate screens. All SmartTest primary commands are described in this chapter.

Some of the most frequently used primary commands have been assigned to PF keys. The KEYS command can be entered on any screen to change the PF key assignments.

# Command Processing

SmartTest line commands are processed before primary commands. Commands entered by pressing a PF key are handled as if entered in the command input area. If a command is entered by pressing a PF key and there is a command in the command input area, the contents of the command input area are appended to the PF key command. The combined commands are then executed as a whole. Multiple commands entered in the command input area are separated by a semicolon (;) and are processed in a left to right sequence. Command results are displayed when one of these conditions occurs:

- The last command is processed and all commands in the sequence completed successfully.
- A command is not successfully processed. If an error occurs in the sequence, processing stops and an error message displays. Results are displayed for the commands that successfully completed. The command causing the error and all remaining commands in the sequence are displayed in the command input area.

Each command in a command sequence is recognized separately for use with the RECALL command.

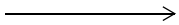
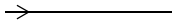
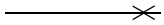
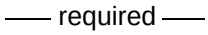
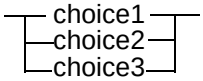
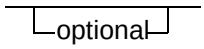
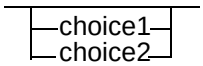
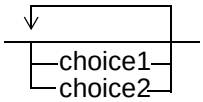
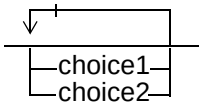
SmartTest is COBOL intelligent and many commands support COBOL-related keywords. However, these commands do not support pseudo code keywords. For example, a FIND command can be used to locate a qualified COBOL data item, but not a pseudo code data item.

See the online help or "Pseudo Code" on page 511 for information about pseudo code.

# Command Diagrams

These notational conventions and symbols are used to describe command syntax:

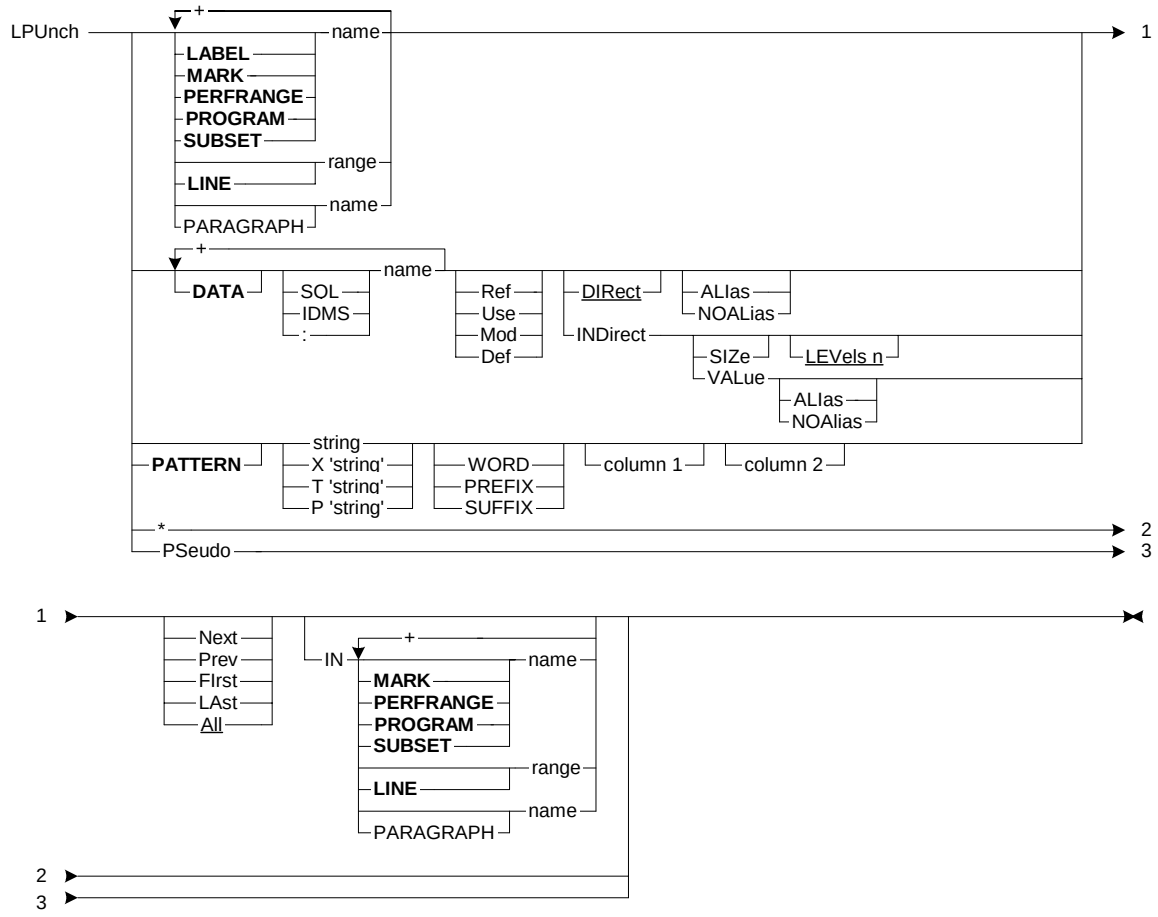
| Item | Description |
|---|---|
| ABBREViations | Illustrates the command abbreviation, which is shown in uppercase letters. Lowercase letters in the command are optional. |
| lowercase | Indicates user-supplied variable information. |
| UPPERCASE | Indicates commands or keywords. |
| Underline | Specifies the default value of an operand. |
| \| | Separates synonymous commands or operands. |

| Item | Description |
|---|---|
|  | Indicates that the command syntax is continued on the next line. |
|  | Indicates the command syntax is continued from the previous line. |
|  | Indicates the end of the command syntax. |
|  | Indicates that the operand or keyword appearing on the main command line is required. |
|  | Indicates that one operand is required. |
|  | Indicates that an operand or keyword appearing below the main command line is optional. |
|  | Indicates that operands are optional. |
|  | Indicates that more than one operand can be chosen. |
|  | Indicates that operands can be concatenated by placing a plus sign (+) between them. |

# Command Syntax

Figure 127 contains the syntax diagram for the LPUNCH command. LPUNCH or a minimum of LPU can be entered. LPUNCH must be entered with a target operand as indicated by the name variable on the main path of the line. If the path is followed down the first or second vertical line, any of the target operands, *, or PSEUDO can be specified. If more than one operand is specified, separate each with a space. Operands that can be concatenated are indicated by a returning arrow that includes a + in the line. Dataname operands (Ref, Use, etc.) pertain to all datanames in a concatenated series.

**Figure 127 • Command Syntax Diagram**




Bold operands are available only with Insight

Continuation lines are numbered 1 through 3, with 1 being the main path of the line. Continuation lines 2 and 3 show that the * and PSEUDO operands are entered with no other values since the line from them extends to the end of the path, bypassing the direction and intarget operands. The direction and intarget operands are optional since they are below the main path of the line.

## Repeating Commands

SmartTest stores the last twenty primary commands entered. Many of the primary commands may be recalled and repeated. These methods may be used to repeat them:

- Enter the RECALL command or press PF12/24 (the default PF key assignment) to display the last recallable command entered in the command input area. Once the recalled command displays, it can be modified, deleted or executed again. Press PF12/24 repeatedly to display the commands in reverse sequence without executing them. To execute a recalled command, press Enter while the command displays in the command input area.
- Enter REPEAT in the command input area to execute the last stacked primary command again (if it is a repeatable command).

These are the SmartTest primary commands that can be repeated:

| | | |
|---|---|---|
| ADD | LIST | SELECT |
| ALLIANCE | LOCATE | SET |
| BRANCH | LPRINT | SETUP |
| BREAK | LPUNCH | SHOW |
| CANCEL | MARK | STEP |
| COPY | MERGE | STOP |
| DELETE | MOVE | SUBTRACT |
| DISPLAY | NEWCOPY | TEST |
| DROP | PARMDEF | TESTPOINT |
| ENVIRONMENT | PREF | TRACE |
| EQUATE | QUALIFY | USING |
| EXCLUDE | REFRESH | UTILITY |
| EXECUTE | RENAME | VALIDATE |
| FIND | RESET | VIEW |
| FINDXTND | RPREF | WHEN |
| GO | RUN | WHERE |
| HIGH | SAVE | ZOOMDATA |
| KEEP | SCROLL | ZOOMIN |
| | | ZOOMOUT |

These methods may be used to repeat the last primary command entered:

| Use the | To... |
|---|---|
| & (Retain) command preceding a primary command | Keeps the command displayed in the command input area after execution. This is the quickest way to repeat a command when minor changes are required before execution. |
| REDO command | Reexecutes the last FIND, FINDXTND, HIGH, PREF, SCROLL, or TRACE command (see the REDO command for more information). |
| RFIND command | Repeats the last FIND or FINDXTND command from the current cursor location. PF5/17 is the default PF key for the RFIND command. |
| RHIGH command | Repeats the last HIGH command from the current cursor location. |
| RPREF command | Redisplays the last View - Paragraph Cross Reference pop-up, or to function as a PREF command with the cursor location as the target and default direction (see the RPREF command for more information). |
| RSCROLL command | Repeats the last SCROLL command from the current cursor location. |
| RTRACE command | Continues the last TRACE command from where it stopped (see the RTRACE command for more information). |

**Note:**

The COPY, DELETE, MARK, MERGE, RENAME, RTRACE, and TRACE commands are available only if Insight is installed.

# Cursor Position

The result of some commands depends on the starting point of the search. A starting point can be specified by entering a line number or label name, or by using the cursor as the starting point.

These are the search starts for each cursor position:

- The Command input area, the search is started in the first column of the first source line on the screen.
- The Line command area, the search is started from the first column on that line.
- The Program source, the search is started from the cursor location.

If the starting point is an unexecutable COBOL statement, these general rules apply:

- Comment lines and compiler directives occurring at the end of an executable code block are considered part of the paragraph or section that follows.
- Blank lines occurring at the end of an executable code block that are not preceded by a comment or compiler directive are considered part of the preceding paragraph or section.

# Cursor Substitution Character

The cursor character is a token substitution character that may be used in several primary commands on the Program View screen. A token is a contiguous set of characters preceded by and followed by a blank, period, comma, or parenthesis. The cursor substitution character is defined on the Options - Product Parameters pop-up. Any character may be chosen, but it should be one that is unique and rarely used in commands. The default is percent character (%).

The cursor substitution character saves time and typing in commands. The cursor substitution character is typed anywhere on the command line, then the screen cursor is placed under the token that is to take its place in the command. For example, to find all occurrences of HLD-ZIP-PREFIX, type this FINDXTND command in the command input area:

```
FX %
```

Place the screen cursor anywhere on HLD-ZIP-PREFIX in the source code and press Enter. The command FX HLD-ZIP-PREFIX is executed. Note that the cursor substitution character must have a space before and after it.

Multiple cursor tokens can be used to specify consecutive tokens in the source code. For example:

```
FX % % %
```

If the screen cursor is then placed on HLD-ZIP-PREFIX, the next two tokens are also picked up in the command, resulting in the command FX HLD-ZIP-PREFIX OF HLD-ZIP. The first token is HLD-ZIP-PREFIX, the second token is OF, and the third token is HLD-ZIP. The tokens must be consecutive.

## Double Byte Character Set (DBCS) Strings

SmartTest commands support DBCS character strings. DBCS character strings can be used in place of alphanumeric character strings. DBCS and alphanumeric characters must not be mixed in one string.

# & (Retain) Command

&any primary command

## Function

Executes the specified primary command and keeps it displayed in the command input area for repeated use or modification.

## Operands

& must be followed by a primary command.

## Usage Notes

The Retain command is useful if the same primary command is to be executed repeatedly or if minor changes to a command are desired.

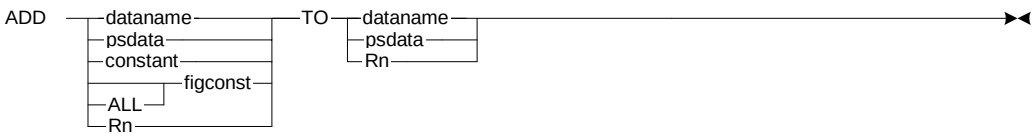
For example, after executing the FIND command for a data item, it may be desirable to execute an LPRINT command on the same data item. The Retain command could be used quickly and easily to perform this function.

## Example

This command finds the EOF-FLAG and keeps &FINDXTND EOF-FLAG displayed in the command input area.

```
&FINDXTND EOF-FLAG
```

# ADD Command




## Function

The ADD command adds the value contained in or represented by the first operand to the specified data item. The value is converted to the proper format for the data item.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, qualified PL/I variable name, or Assembler variable name. Dataname refers to any valid COBOL reference for a data element. |
| *psdata* | Specifies a level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| *constant* | Specifies a COBOL numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about coding pseudo code. |
| ALL | Specifies readability when used with a figurative constant. |
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to add specific values. The figurative constants ZERO, ZEROS, and ZEROES are supported and may be further qualified with the ALL figurative constant. |
| R*n* | Specifies a register 0 through 15. |
| TO | Specifies a required keyword used as a connective for the ADD statement operands. |

## Usage Notes

Selecting Test ▸ Add also allows you to add values to data.

# ALLIANCE Command

```
ALLIANCE ──────────────────────────────────────────────────────────►◄
```

## Function

The ALLIANCE command displays the SmartTest/Alliance Interface pop-up, which is used to configure a dynamic link to Alliance. After entering appropriate information and pressing Enter, SmartTest activates Alliance and runs the script specified on the Query Name field of the SmartTest/Alliance Interface pop-up.

## Operands

None.

## Usage Notes

The ALLIANCE command allows access to additional program information that assists in setting up a test session. From the ESW CNTL library, copy the VIAPALSC member and the VIAPQ* members into a user-defined PDS. Edit the members and their individual queries to customize them for your needs. Then specify a member name in the Query Name field of the SmartTest/Alliance Interface pop-up. Alliance displays information in response to the queries contained in the scripts.

Prior to issuing the ALLIANCE command, a complete Alliance application analyze must be done, including the load module libraries and execution JCL.

# ALLOCDEF Command

```
ALLOCDEF | ADEF ──────────────────────────────────────────────────────────────►◄
```

## Function

The ALLOCDEF command displays the Options - Product Allocations pop-up used to specify the DASD volumes for the Log, List, Punch, and Work files.

## Operands

None.

## Usage Notes

The ALLOCDEF command can be entered on any SmartTest screen.

The Options - Product Allocations pop-up can also be displayed by selecting Options ▸ Product Allocations.

# ANALYZE Command

```
ANalyze ──────────────────────────────────────────────────────────►◄
```

## Function

Displays the File - Analyze Submit pop-up that can be used to submit a compile/analyze job without ending the current SmartTest function. A program must be analyzed before it can be used in SmartTest.

## Operands

None.

## Usage Notes

The ANALYZE command can be entered on any SmartTest screen.

***To analyze a program during a test session***

**1** Type `ANALYZE` in the command input area.

**2** Submit the analyze job for the program.

**3** Type `END` in the command input area.

The test session is then resumed. The QUALIFY command can then be used to view the analyzed program during the current test session.

You can also display the File - Analyze Submit pop-up by selecting Options ▸ Compile/Analyze.

See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for more information about submitting an analyze job. See the online help for information about the File - Analyze Submit pop-up.

If a decision is made to reanalyze a program while it is being viewed, the test session must be completed or canceled (if active) and the program must be released (QUALIFY CANCEL). The analyze job requires that the program is not in use by any other SmartTest function.

# BRANCH Command

```
Branch ──┬──────────────────────────────────────────────────────────────►◄
         │                  ┌──name──┐
         ├──────────────────┤        │
         ├─PERFRANGE────────┤        │
         ├─PROGRAM──────────┤        │
         ├─LABEL────────────┤        │
         ├─PARAGRAPH────────┘        │
         ├─BACKup────────────────────┤
         └─PREVious──────────────────┘
```

Bold operands are available only with Insight

## Function

The BRANCH command is used to position the cursor at the specified target. This command can be used to scroll from a statement such as a PERFORM, to the paragraph being performed. The BACKUP operand can be used to return to the statement from which the BRANCH occurred.

## Operands

| Operand | Description |
|---|---|
| Blank | No operand is required if the cursor is positioned on a GO TO or PERFORM statement. The BRANCH command automatically locates the target of the GO TO or PERFORM statement. If the cursor is positioned on any other statement, the BRANCH command locates the next sequential statement to be executed. The PROCEDURE DIVISION label is located if the cursor is positioned outside the PROCEDURE DIVISION. |
| PERFRANGE *name* | Locates the specified PERFORM range. If the specified name is not fully qualified and there are multiple PERFORM ranges with the specified name, the cursor is positioned on the first PERFORM range found. A message displays indicating the number of PERFORM ranges found. This operand is available only if Insight is installed and an Insight analysis has been run on a COBOL program being tested. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| LABEL *name* | Locates the specified paragraph or section name. If the specified name is not fully qualified and there are multiple paragraphs or sections with the specified name, the cursor is positioned on the first paragraph or section found. A message displays indicating the number of paragraphs or sections found. |

| Operand | Description |
|---|---|
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| BACKup | Positions the cursor at the location from which the branch occurred. |
| PREVious | Positions the cursor, in conjunction with the current cursor location, at the GOTO, PERFORM, or CALL statement from which a target was reached. |

## Usage Notes

The BRANCH command is very helpful when tracking branching logic. This command can be used to track branching logic several levels deep into PERFORMed code, then return to each PERFORM statement. An effective way to use the BRANCH command is to position the cursor on a PERFORM or GO TO statement, then press PF10/22. (BRANCH is the default for the PF10/22 key; BRANCH BACKUP is the default for the PF11/23 key.)

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

The screen position is saved for use with the BACKUP operand based on these conditions:

- A PERFORM range or LABEL name is entered on the command line.
- The cursor is placed on a GO TO or PERFORM statement.

The PROCEDURE DIVISION label displays when you enter the BRANCH command with no operands and the cursor is positioned in a part of the program other than the PROCEDURE DIVISION.

If a PROGRAM EXIT or STOP RUN is encountered, a pause occurs and a message displays indicating that a logical end has been reached.

When entering an abbreviation of the BRANCH command, note that BR is the abbreviation for the BREAK command. Branch can be abbreviated as B or BRA.

Selecting Search ▸ Branch also allows you to branch to a specified target.

## Examples

To scroll the screen is scrolled and position the cursor on the READ-ALL paragraph, enter this command, place the cursor on PERFORM READ-ALL, then press Enter.

```
BRANCH
```

To scroll the screen and position the cursor on the P120-READ paragraph, enter this command:

```
B P120-READ
```

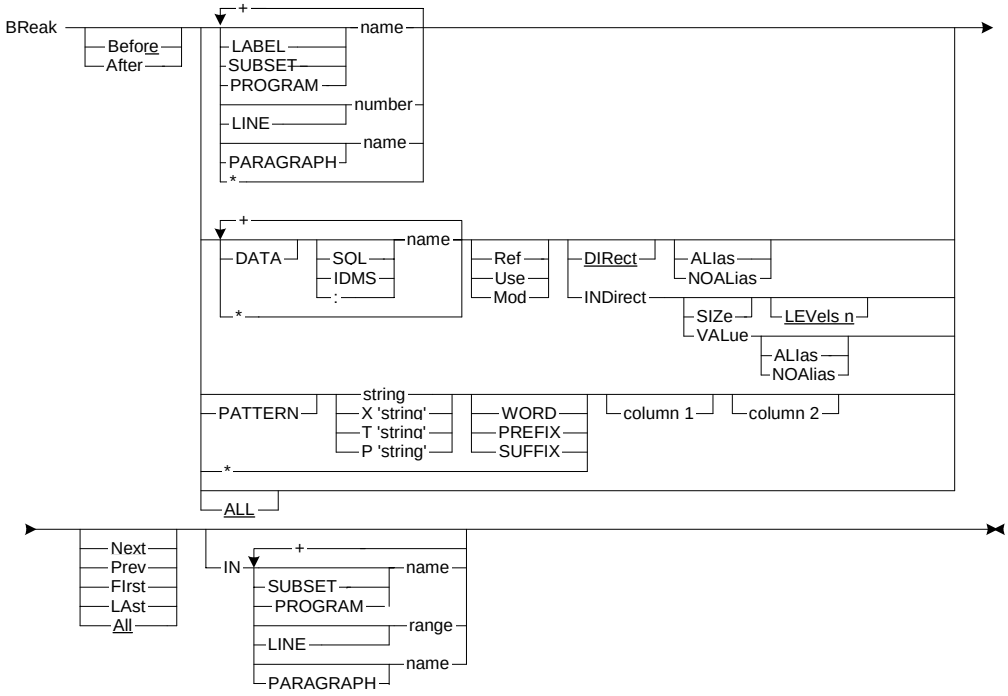
To scroll the screen and position the cursor on the PROGRAM-INIT paragraph, enter this command:

```
B PROGRAM-INIT
```

To scroll the screen and position the cursor at the location from where the B PROGRAM-INIT command was entered, type:

```
B BACKUP
```

# BREAK Command




## Function

The BREAK command is used to insert a Breakpoint before or after the statement containing the specified target. A Breakpoint creates an interrupt in the program execution. Program execution stops when the BREAK statement is encountered. The current status box displays with a STOPPED AFTER BREAK status. Typically, the BREAK command is used to automatically set numerous Breakpoints at various locations within the program, such as BREAK BEFORE CALL ALL. Variables can be checked, values can be changed, or pseudo code can be entered when an interrupt is encountered during a SmartTest test session.

## Operands

| Operand | Description |
|---|---|
| Before | The BREAK command inserts a Breakpoint before the statement containing the specified target. See the Usage Notes for information about the placement of Breakpoints when various targets are specified. BEFORE is the default value for the BREAK command. |
| After | The BREAK command inserts a Breakpoint after the statement containing the specified target unless the target is contained in the last statement of a paragraph or section. See the Usage Notes for information about the placement of Breakpoints. |
| LABEL *name* | A PROCEDURE DIVISION paragraph name or section name, an Assembler label name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. The PROCEDURE and PROC literals can also be specified. The BREAK command inserts a Breakpoint before or after execution of the specified section or paragraph name. Optionally, the ALL operand can be entered to insert Breakpoints before or after all section or paragraph names. |
| SUBSET *name* | The BREAK command inserts a Breakpoint before or after the specified subset. These are the valid COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information. |

| Operand | Description |
|---|---|
| | The ENTRY, EXIT and PARAGRAPH subsets cannot be specified with the AFTER operand.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code in the program. This includes all programs physically nested inside the specified program. |
| LINE *number* | Specifies a single line number. Line numbers are displayed in columns 1 through 6. The BREAK command inserts a Breakpoint before or after the specified line. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight or scroll action or command. |
| DATA *name* | Inserts a Breakpoint before or after the specified COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify where Breakpoints are to be set: REF, USE, MOD, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |
| SQL | Breaks on DB2/SQL variables only. |
| IDMS | Breaks on IDMS variables only. |
| : | Breaks on COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is tested, used, set, or modified. This is the default value for the DATA name operand. |

| Operand | Description |
|---|---|
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, or modified (based on the REF, USE, or MOD selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. |

| Operand | Description |
|---|---|
| *string* | Specifies that the BREAK command will insert a Breakpoint before or after the statement that contains the character string. Literals and blanks can be included in the character string if desired. If the string contains blanks, it must be enclosed in single or double quotes. The WORD, PREFIX, and SUFFIX subordinate operands pertain to the pattern string and can be used to further qualify where Breakpoints are to be set. |
| X'*string*' | Specifies the hexadecimal string option. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. The hexadecimal string corresponds to an offset into the program. |
| T'*string*' | Specifies the text string option. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |
| P'*string*' | Specifies the picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings.<br>These special characters can be combined with other characters.<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Specifies the pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| Column1 | Specifies the column number where a pattern search is to begin. |
| column2 | Specifies the column number where a pattern search is to end. |

| Operand | Description |
|---|---|
| Next | Searches forward from the current cursor position for the next occurrence of the requested target. This is the default value for the BREAK command. |
| Prev | Searches backward from the current cursor position for the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file for the first occurrence of the requested target. |
| LAst | Searches backward from the bottom of the source file for the first occurrence of the requested target. |
| All | Searches the entire source file for every occurrence of the requested target. |
| IN | Restricts the BREAK command to the specified target type. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | |
|---|---|---|
| LABEL | SUBSET | PARAGRAPH |
| LINE | PROGRAM | PATTERN |

- The * operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

The BREAK command is used to set program interrupts when testing a COBOL, Assembler, or PL/I program (residing in the AKR). One or more pseudo code BREAK statements are automatically inserted when the BREAK command is used. Other pseudo code statements can be inserted after the BREAK statement if desired. The Breakpoints List screen shows all Breakpoints for the current program and can be used to activate or deactivate Breakpoints.

See the online help for more information about the Breakpoints List screen. Additional information can also be found by referring to the "LIST Command" on page 316.

Enter the BREAK command with no operands when using it within a block of pseudo code.

Program execution is stopped (interrupted) when an active Breakpoint is encountered. All Breakpoints remain active until reset, deleted, or the test session ends. The SET command can also be used to activate or deactivate ALL Breakpoints. The D (Delete) line command can be used to delete Breakpoints on the Program View screen.

See the SET command in online help and in "SET Command" on page 409 and the Delete line command for additional information.

Use the Save Options pop-up to save any Breakpoints created during the current test session.

Selecting Test ▸ Break also allows you to insert breakpoints.

When using the BEFORE operand, these conditions apply:

- When a target of SUBSET CONDITIONAL is specified with the BEFORE operand, the Breakpoint is inserted before the conditional test. If the specified target contains multiple conditional tests, the Breakpoint is inserted before the first conditional test (unless one of the occurrence operands such as All is also specified). If the All operand is also specified, a Breakpoint is inserted before each nested conditional test.
- When a target of SUBSET ENTRY is specified with the BEFORE operand, the Breakpoint is inserted before the first executable statement of each entry point.
- When a target of SUBSET EXIT is specified with the BEFORE operand, the Breakpoint is inserted before the statements that pass control from the program.
- When a target of SUBSET LABEL is specified with the BEFORE operand, the Breakpoint is inserted before the first executable statement in the section or paragraph after the label.
- When a line number is specified with the BEFORE operand, the Breakpoint is inserted before the specified line if it is an executable statement. If the specified line is not an executable statement, the Breakpoint is inserted before the next executable statement that follows.

- When a target of SUBSET PARAGRAPH is specified with the BEFORE operand, the Breakpoint is inserted before the first executable statement in the paragraph after the paragraph name.
- When a PATTERN STRING is specified with the BEFORE operand, the Breakpoint is inserted before the executable statement that contains the specified pattern string. If the line containing the specified pattern string is not an executable statement, the Breakpoint is inserted before the next executable statement that follows.

When using the AFTER operand, these conditions apply:

- When a target of SUBSET CONDITIONAL is specified with the AFTER operand:
  - If the conditional test contains only statements for a true path (no ELSE or false path), a Breakpoint is inserted after the conditional test, before the first true path statement.
  - If the conditional test contains statements for true and false paths (IF statements, ELSE statements), a Breakpoint is inserted after the conditional tests (IF, ELSE), before the first true and false path statements.
  - If the specified target contains multiple conditional tests, a Breakpoint is inserted after the first conditional test (unless one of the occurrence operands such as All is also specified). If the All operand is also specified, a Breakpoint is inserted after each nested conditional test.
  - When NEXT SENTENCE is used as the path (IF, NEXT SENTENCE), it is also considered to be a conditional statement.
  - When a direction operand is used to insert a Breakpoint after a conditional (an IF statement), the corresponding ELSE statement is also considered to be a conditional. For example, if the cursor is positioned on an IF statement and the BREAK AFTER COND NEXT command is entered, a Breakpoint is inserted after the next ELSE statement encountered.
- When a target of SUBSET LABEL is specified with the AFTER operand, a Breakpoint is inserted after the PERFORM of the paragraph and section, or before the next executable statement of a paragraph and section if there is a fallthrough condition.
- When a LINE number is specified with the AFTER operand, a Breakpoint is inserted after the line.
- When a PATTERN STRING is specified with the AFTER operand, a Breakpoint is inserted after the statement containing the specified pattern string.
- These targets cannot be specified with the AFTER operand:

  SUBSET ENTRY  SUBSET EXIT

For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*.

## Notes for Assembler Programmers

This is the order in which the BREAK command places breakpoints in Assembler source programs:

1. Assembler labels with associated executable statements.
2. Opcode operands or comments on executable statements.

## *Examples*

To insert a breakpoint before each Input and Output statement in the current program, enter this command:

```
BREAK IO ALL
```

To insert a breakpoint before each occurrence of a statement containing ZIP-CODE where it is being modified, enter this command:

```
BREAK ZIP-CODE MOD ALL
```

To place a breakpoint on all of the statements that refer to GETREC as a label, an operand, or as a comment on an executable statement place quotes on GETREC, type this command:

```
BREAK 'GETREC' ALL
```

To break on every Assembler Branch And Link instruction in the source, type this command:

```
BREAK ' BAL ' ALL
```

To break on the first and every tenth execution of a specific instruction, type this pseudo code before the line containing the source of the instruction:

```
77 TEMP-VALUE PIC S9(7) COMP.
   IF &COUNT > TEMP-VALUE THEN
      BREAK
      ADD +10 TO TEMP-VALUE.
```

# CANCEL Command

```
CANcel ──────────────────────────────────────────────────────────►◄
```

## Function

The CANCEL command terminates the current test session.

## Operands

None.

## Usage Notes

The CANCEL command is used to terminate the current test session. When this command is entered on the Program View screen, the program being executed remains displayed but testing ends. The RUN or STEP command can be used to restart the test if desired. All pseudo code statements remain intact and Breakpoints remain active. All files used by the program are closed and memory is freed.

When the CANCEL command is entered while connected to a batch job, the batch job step is terminated with a U222 abend. Succeeding steps, if any, then execute under control of standard MVS JCL.

Exclusive control of a qualified program on the AKR is not released until the QUALIFY CANCEL command is used, the SmartTest session is terminated, or the AKR dataset name on the Environment Selection pop-up is released. A qualified program causes a batch analysis job to wait for exclusive control.

Selecting Test ▸ Cancel also allows you to cancel the current test session.

# CONVERT Command

```
CONvert ──────────────────────────────────────────────────────────►◄
```

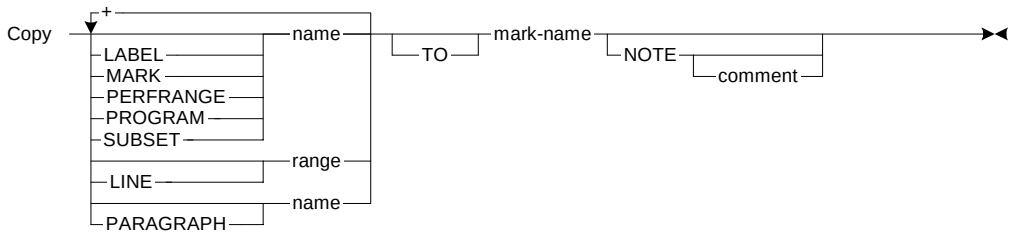
## Function

The CONVERT command displays the Convert Batch JCL screen.

## Operands

None.

# COPY Command




ASG-Insight only

## Function

Copies the contents of a path or set of lines to a mark. This provides a means of saving the same information under a different name for additional use. A new description can be included if desired. This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>You can also create the TRACK, NETWORK and SUBNET paths using the Logic Order Search pop-ups. See the online help or the *ASG-Insight User's Guide* for more information about Logic order. |

| Operand | Description |
|---|---|
| PERFRANGE *name* | Specifies the name in a PERFORM statement (including all statements that are executed as a result of a PERFORM statement) or the name of any section contained in the Declaratives. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |

| Operand | Description |
|---|---|
| TO | Specifies an optional keyword indicating the mark name that follows is where the target is copied. |
| *mark-name* | Specifies a name assigned to a path or set of lines using the MARK command or the Options - Scratchpad Mark pop-up. The specified name must meet these requirements:<br>• Must be a maximum of 10 alphanumeric characters and can include hyphens.<br>• Names longer than 10 characters are truncated.<br>• Name must begin with an alphabetic character.<br>This name is assigned to the new path or set. |
| NOTE comment | Includes an optional description about the name. Comment text can be a maximum of 50 alphanumeric characters. If the NOTE comment operand is not entered, the existing comment is copied to the new mark name. If NOTE is entered without comment text, the existing comment is not copied to the new mark name. |

## Usage Notes

The same path or set of lines can be copied to different names. The specified TO mark-name must be unique. An error message displays and the copy function is not performed if an existing mark-name is specified.

Selecting Options ▸ Scratchpad and then selecting Copy on the Options - Scratchpad pop-up also allows you to copy the contents of a path or set of lines available only if Insight is installed.

To concatenate targets, use a plus sign (+) between the target names. A concatenated target name can be used on these valid targets:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

## Example

After executing this command, the contents of TRACK exist in both TRACK and FICA.

```
COPY TRACK TO FICA
```

# CURRENT Command

CURRent

## Function

The CURRENT command is used to save the current cursor location in the program source. The LOCATE &CURRENT command can then be used to reposition the screen to the saved location.

Only one CURRENT command can be in effect. A new CURRENT command replaces the prior one.

The CURRENT command is valid only on the Program View screen.

## Operands

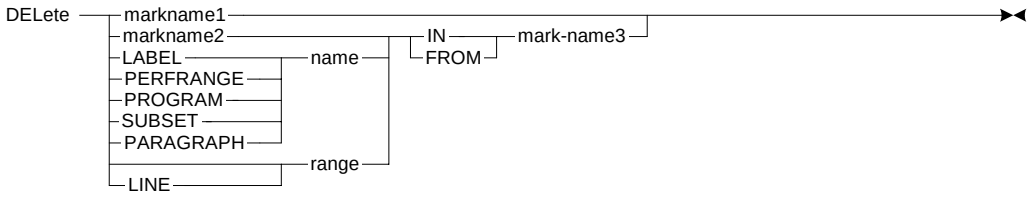
None

## Usage Notes

The CURRENT and LOCATE &CURRENT commands increase the ability to control the position in the source during script processing.

The CURRENT command can also be used to save your current location in the program source while you browse related data items.

# DELETE Command




ASG-Insight only

## Function

The DELETE command is used to delete a mark name and its contents. This command is also used to delete a mark name, path, or a set of lines from an existing mark name.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| *markname*1, *markname*2, *markname*3 | Specifies a name assigned to a path or set of lines using the COPY, MERGE, MARK, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>**Note:**<br>The TRACK and NETWORK paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are also deleted. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement and all statements that are executed as a result of a PERFORM statement or the name of any section contained in the Declaratives. |

| Operand | Description |
|---|---|
| PROGRAM *name* | Specifies the name of the main program or any nested program. All the code contained in the named program, including nested programs, is deleted. |
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| PARAGRAPH *name* | A PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. The entire paragraph or section is deleted. |
| LINE *range* | A single line number or range of lines. Line numbers are displayed in columns 1 through 6 of Program View. |
| IN/FROM *markname3* | This operand deletes the specified mark name, path, or set of lines from the existing markname3. |

## Usage Notes

Marknames that are no longer needed can be deleted using the DELETE command. If the markname does not exist, an error message displays. The NETWORK and TRACK system-generated paths can also be deleted. SUBNETs are automatically deleted when the corresponding NETWORK is deleted.

Selecting Options ▸ Scratchpad and then selecting Delete on the Options - Scratchpad pop-up also allows you to delete a Mark name and its contents.

The set of lines in one markname can be deleted from another markname. In addition, a path or line range can be deleted from an existing markname.

## Example

This command deletes the mark name FICA and its contents:

```
DELETE FICA
```

This command deletes line 17 from the markname FICA:

```
DELETE 17 FROM FICA
```

This command deletes the set of lines in the markname FICA from the markname WITHHOLD:

```
DELETE FICA FROM WITHHOLD
```

# DISPLAY Command

```
DIsplay ──┬─dataname──┬──────────┬──┬─CHAR─┬──┬─LENgth n─┬──────────────────────►◄
          │           └─subscripts─┘  └─HEX─┘  └──────────┘
          ├─address expression──────┤
          └─OPERANDs────────────────┘
```

## Function

The DISPLAY command is used to display the current value of a specified data item in the long message area.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a data item for which the value is to be displayed in the long message area. |
| *subscripts* | Specifies subscript information using the standard COBOL syntax:<br>(*subscript1, subscript2, ... subscriptn*)<br>Left and right parentheses and at least one blank between subscripts are required. Commas between subscripts are optional. A subscript may be a numeric literal, an index data item, a numeric data item, or a numeric data item plus or minus a numeric literal. |
| *address expression* | Specifies an address expression consisting of an address or a register number followed by a maximum of 32 indirection indicators and/or offsets.<br>These are definitions of the possible components of an address expression. The address expression must not contain spaces. |
| Address: | X'*nnnn*' An absolute address specified in hexadecimal notation. |
| Registers: | R*n* A register 0 through 15.<br>*n*R A register 0 through 15. |
| Indirection indicators: | % Indicates a 24-bit indirection.<br>? Indicates a 31-bit indirection. |

| Operand | Description | |
|---|---|---|
| Offsets: | + | Indicates that the following offset is to be added. |
| | - | Indicates that the following offset is to be subtracted. |
| | X'*nn*' | Specifies an offset specified in hexadecimal notation. |
| | *nn* | Specifies an offset specified in decimal notation. |
| | R*n* | Specifies a register 0 through 15 that contains the offset. |
| | *n*R | Specifies a register 0 through 15 that contains the offset. |
| OPERANDs | Displays the current values for the symbolic data items on the current statement. | |
| CHAR | Displays the value for the specified data item in character format. | |
| HEX | Displays the value of the specified data item in hexadecimal format. | |
| LENgth *n* | Displays the value of the data item specified for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. | |

## Usage Notes

The DISPLAY command supports pseudo code defined variables. It does not support the pseudo code variable &COUNT.

Selecting View ▸ Display also allows you to display the current value of a specified data item.

Display attempts to fit all referenced data items into the long message area. If necessary, DISPLAY truncates longer data values and datanames to a minimum of 14 characters. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed.

These notes apply to using the DISPLAY command with Assembler programs:

- The DISPLAY command can be used in Assembler source programs with CSECT relative named data items and literal data items. CSECT relative named data items are data items that occupy storage locations within the supported CSECT of the source module. Literal data items are those data items defined in the supported CSECT that begin with an equal (=) sign. Unnamed data items is shown with the name UNNAMED ITEM.
- Data items defined outside of the supported CSECT are not supported.

## Examples

To display the value of the ZIP-CODE data item in the long message area, type this command:

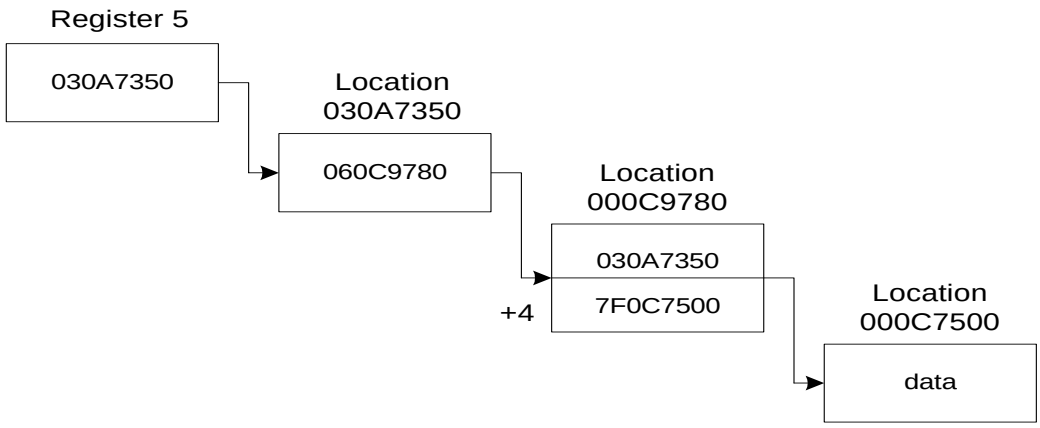
```
DISPLAY ZIP-CODE
```

To display data at the address expression specified, type this command:

```
DISPLAY R5?%+4% CHAR LEN 50
```

Figure 128 shows the above address expression graphically:

**Figure 128 • Example of Address Expression**



# DROP Command

## Function

The DROP command ends addressability to any Assembler DSECT currently being addressed by Register n, as set by the USING command.

Full-screen Assembler support is available only with the SmartTest-ASM option.

## Operands

| Operand | Description |
|---|---|
| *n* | Specifies a register (0 through 15) for which the address was set. |
| R*n* | Specifies a register R0 through R15. |
| *n*X | Indicates the register address is 31-bit. |
| R*n*X | Specifies a 31-bit register R0X through R15X. |

## Usage Notes

The X is not required, even if the associated USING command had specified it.

See "USING Command" on page 448 for more information.

Selecting View ▸ Drop also ends addressability to any Assembler DSECT currently being addressed by Register n.

# DUMP Command

```
DUMP ──────────────────────────────────────────────────────────────►◄
```

## Function

The DUMP command is used to generate an MVS symptom and snap dump, or a CICS transaction dump, of the current suspended task or transaction. Selecting Test ▸ Dump also allows you to generate these dumps.

## Operands

None.

## Usage Notes

During an active test session, a dump can be generated by entering the DUMP command in the command input area.

Issuing the DUMP command in an MVS environment produces a symptom and a snap dump of the program being tested, along with its control blocks. The symptom dump includes general program information and REGISTER information. The REGISTER information contained in the snap dump is limited to the current values at the time you issue the dump and might not include the program that is being tested's register values. The snap dump consists of all PDATA and some SDATA SNAPX options and is appended to the symptom dump data. All dump data is copied to the log file. See "Options - Log/List/Punch Definition Pop-up" on page 190 for more information about processing the log file.

## Examples

Figure 129 through Figure 133 on page 257 are samples of the various types of dump output for user-requested symptom dumps:

**Figure 129 • General Program Information**

```
12:50:43 ASG2502I             ASG-SmartTest            User Requested Symptom DUMP            Page   1
12:50:43 ASG2502I
==========================================================================================================
12:50:43 ASG2502I
12:50:43 ASG2502I General Program information
12:50:43 ASG2502I
12:50:43 ASG2502I
----------------------------------------------------------------------------------------------------------
12:50:43 ASG2502I
12:50:43 ASG2502I         Program:      NAME=TESTCOBL
12:50:43 ASG2502I                       PSW=8B3CE30E
12:50:43 ASG2502I                       TCB=0078F200
12:50:43 ASG2502I                      BASE=0B3CDDA0
12:50:43 ASG2502I                    OFFSET=0000056E
12:50:43 ASG2502I                    LENGTH=00000E10
12:50:43 ASG2502I                     AMODE=31
12:50:43 ASG2502I                INSTRUCTION=D209 92B1 9048
12:50:43 ASG2502I
12:50:43 ASG2502I  ASG-SMARTTEST/CURRENT STATUS=STOPPED BY STEP REQUEST
12:50:43 ASG2502I
12:50:43 ASG2502I
==========================================================================================================
12:50:43 ASG2502I
12:50:43 ASG2502I
```

**Figure 130 • General Purpose Register Information**

```
08:53:46 ASG2502I             ASG-SmartTest            User Requested Symptom DUMP            Page   2
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
08:53:46 ASG2502I General Purpose Register information
08:53:46 ASG2502I
08:53:46 ASG2502I
----------------------------------------------------------------------------------------------------------
08:53:46 ASG2502I
08:53:46 ASG2502I              REG    Hex        Decimal
08:53:46 ASG2502I              ---   --------   -----------
08:53:46 ASG2502I              R0  - 8B41FE0E   -1958609394
08:53:46 ASG2502I              R1  - 0009FF40       655168
08:53:46 ASG2502I              R2  - 0B41F242     188871234
08:53:46 ASG2502I              R3  - 0004122C       266796
08:53:46 ASG2502I              R4  - 0B2BD210     187421200
08:53:46 ASG2502I              R5  - 8B41F424   -1958611932
08:53:46 ASG2502I              R6  - 00000000           0
08:53:46 ASG2502I              R7  - 0009FF40       655168
08:53:46 ASG2502I              R8  - 0009F280       651904
08:53:46 ASG2502I              R9  - 0B2BD4F8     187421944
08:53:46 ASG2502I              R10 - 0B41EE28     188870184
08:53:46 ASG2502I              R11 - 0B41F140     188870976
08:53:46 ASG2502I              R12 - 0B41EE20     188870176
08:53:46 ASG2502I              R13 - 0B2BD028     187420712
08:53:46 ASG2502I              R14 - 00084980       543104
08:53:46 ASG2502I              R15 - 80084990   -2146940528
08:53:46 ASG2502I
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
08:53:46 ASG2502I
```

**Figure 131 • Access Register Information**

```
08:53:46 ASG2502I             ASG-SmartTest            User Requested Symptom DUMP            Page   3
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
08:53:46 ASG2502I Access Register information
08:53:46 ASG2502I
08:53:46 ASG2502I
----------------------------------------------------------------------------------------------------------
08:53:46 ASG2502I
08:53:46 ASG2502I              REG    Hex       Decimal
08:53:46 ASG2502I              ---   --------   -----------
08:53:46 ASG2502I              A0  - 007FB01F      8368159
08:53:46 ASG2502I              A1  - 00000000            0
08:53:46 ASG2502I              A2  - 00000000            0
08:53:46 ASG2502I              A3  - 00000000            0
08:53:46 ASG2502I              A4  - 00000000            0
08:53:46 ASG2502I              A5  - 00000000            0
08:53:46 ASG2502I              A6  - 00000000            0
08:53:46 ASG2502I              A7  - 00000000            0
08:53:46 ASG2502I              A8  - 00000000            0
08:53:46 ASG2502I              A9  - 00000000            0
08:53:46 ASG2502I              A10 - 00000000            0
08:53:46 ASG2502I              A11 - 00000000            0
08:53:46 ASG2502I              A12 - 00000000            0
08:53:46 ASG2502I              A13 - 00000000            0
08:53:46 ASG2502I              A14 - 807FA03C   -2139119556
08:53:46 ASG2502I              A15 - 00000000            0
08:53:46 ASG2502I
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
08:53:46 ASG2502I
```

**Figure 132 • Floating Point Register Information**

```
08:53:46 ASG2502I             ASG-SmartTest            User Requested Symptom DUMP            Page   4
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
08:53:46 ASG2502I Floating Point Register information
08:53:46 ASG2502I
08:53:46 ASG2502I
----------------------------------------------------------------------------------------------------------
08:53:46 ASG2502I
08:53:46 ASG2502I              REG       Hex            Mantissa       Exp
08:53:46 ASG2502I             ---   ----------------   -----------------  ---
08:53:46 ASG2502I              F0  - 0000000000000000             +0.0  E+00
08:53:46 ASG2502I              F2  - 0000000000000000             +0.0  E+00
08:53:46 ASG2502I              F4  - 0000000000000000             +0.0  E+00
08:53:46 ASG2502I              F6  - 0000000000000000             +0.0  E+00
08:53:46 ASG2502I
08:53:46 ASG2502I
==========================================================================================================
08:53:46 ASG2502I
```

**Figure 133 • Snap Dump Index**

```
JOB VIAUSR4         STEP VIA180         TIME 145939   DATE 01170                                PAGE 00000909

                           DUMP INDEX
                           ----------

 DATA AREAS                                              PAGE NUMBER
-----------                                              -----------

AREAS RELATED TO TCB AT: 00789060/....                    00000001
ENQ/DEQ CONTROL BLOCKS................                    00000030
DATA MANAGEMENT CONTROL BLOCKS........                    00000035
IOS CONTROL BLOCKS....................                    00000035
RTM CONTROL BLOCKS....................                    00000035
PCLINK STACKS / SAVE AREAS............                    00000035
INSTALLATION/SUBSYSTEM AREA...........                    00000035
LSQA..................................                    00000036
SWA AND SP229/230.....................                    00000129
REGISTERS.............................                    00000169
MODULES...............................                    00000169
  CEEBINIT............................                    00000213
  CEEPLPKA............................                    00000350
  IGZCEV5 ............................                    00000249
  IGZCLNK ............................                    00000241
  IGZCPAC ............................                    00000284
  IGZCPCO ............................                    00000256
  IGZCTCO ............................                    00000170
  IGZCULE ............................                    00000169
  IGZCXFR ............................                    00000212
  IGZEINI ............................                    00000207
  IGZEPLF ............................                    00000206
  IGZEQBL ............................                    00000244
  TESTCOBL............................                    00000347
  VIAPEMON............................                    00000170
TRACE TABLE...........................                    00000828

END OF DUMP
```

# END Command

```
END ──────────────────────────────────────────────────────────────►◄
```

## Function

The END command is used to terminate the current screen function and redisplay the prior screen.

## Operands

None.

## Usage Notes

The END command (PF3/15) terminates any SmartTest screen function. The current function ends and the previous function or screen is redisplayed.

When the END command is entered on the primary screen, SmartTest is exited and control is returned to ISPF. SAVE options specified on the Options - Product Parameters pop-up determine if pseudo code, Marks and Equates are automatically saved in the AKR when exiting SmartTest.

# ENVIRONMENT Command

ENVironment ──────────────────────────────────────────►◄

## Function

The ENVIRONMENT command is used to display the Environment Selection pop-up, which is used to select the SmartTest testing environment, specify the AKR to be used, and to specify the application load libraries and procedure libraries to be used for the test session.

## Operands

None.

## Usage Notes

The ENVIRONMENT command can be issued from any SmartTest screen.

Selecting File ▸ Setup test environment and then selecting Select execution environment on the File - Setup Test Environment pop-up also displays the Environment Selection pop-up.

See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for more information about the Environment Selection pop-up.

# EQUATE Command

```
EQuate ──name──┬──────────┬──────────────────────────────────────►◄
               └──string──┘
```

## Function

The EQUATE command is used to define a name for a character string.

## Operands

| Operand | Description |
|---|---|
| *name* | Specifies the equate name. Names must conform to these standards:<br>• A maximum of 10 alphanumeric characters<br>• First character must be alphabetic<br>• A hyphen is the only special character allowed |
| *string* | Specifies a character string to be substituted by the EQUATE command. A character string can be a long command, pattern, dataname, concatenated datanames, etc. Literals and blanks can be specified in the character string if desired. If the string operand is not entered, the equated name is deleted (if it exists). |

## Usage Notes

Equated names can be used during a test session to reduce the number of keystrokes required.

If an equated name is used in another EQUATE command, it must first be defined. For example:

```
EQ CC TOT-COST
EQ BB CC + PROD-COST
```

BB now contains TOT-COST + PROD-COST.

Equates can be saved in the AKR, if desired. See "SAVE Command" on page 399 for more information.

To delete an equated name, type `EQUATE` plus the name without the string.

Selecting Options ▸ Scratchpad and then selecting Equate on the Options - Scratchpad pop-up also allows you to define a name for a character string.

## Examples

To equate the name A to END-OF-FILE-FLAG, type this command:

```
EQ A 'END-OF-FILE-FLAG'
```

You can then enter this command to find END-OF-FILE-FLAG:

```
FIND A
```

To equate BB to the FIND END-OF-FILE-FLAG command, type this command:

```
EQ BB FIND 'END-OF-FILE-FLAG'
```

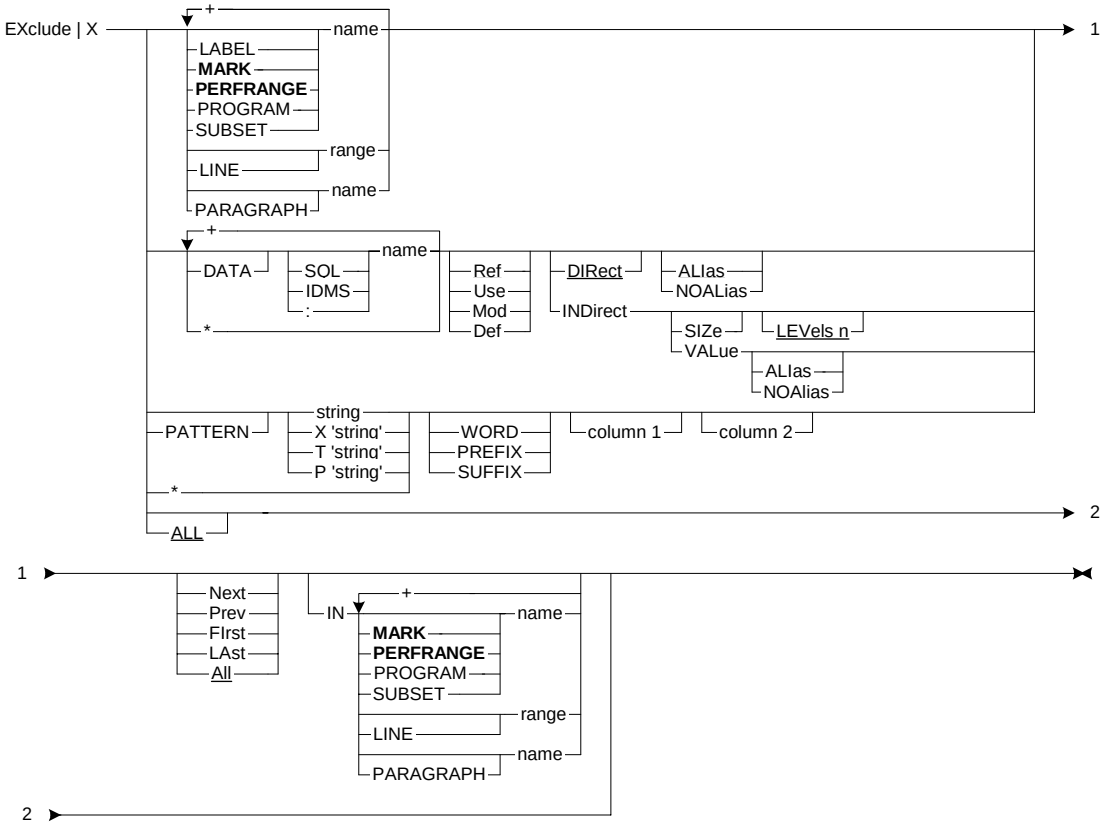
You can then enter this command to find END-OF-FILE-FLAG:

```
BB
```

To delete the BB equate, type this command:

```
EQ BB
```

# EXCLUDE Command




Bold operands are available only with ASG-Insight

## Function

The EXCLUDE command performs a FINDXTND command on the specified target, then excludes the resulting lines from the display. This command is similar in function to the ISPF EXCLUDE command.

## Operands

| Operand | Description |
|---|---|
| Blank | Excludes all lines from the display when no operands are used. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies the name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK, and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement or the name of any section contained in the Declaratives. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight, or scroll action or command. |

| Operand | Description |
|---|---|
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the EXCLUDE command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |
| SQL *name* | Excludes datanames that are DB2/SQL variables only. |
| IDMS *name* | Excludes datanames that are IDMS variables only. |
| : *name* | Excludes datanames that are COBOL variables only. |
| Ref | Excludes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Excludes occurrences of the dataname where its value is being tested or used. |
| Mod | Excludes occurrences of the dataname where its value is being set or modified. |
| Def | Excludes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |

| Operand | Description |
|---|---|
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |
| ALIas | Excludes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. |
| *string* | Specifies a string of alphanumeric characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*string*' | Specifies a hexadecimal string option. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string option. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |

| Operand | Description |
|---|---|
| P'*string*' | Specifies a picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Specifies a specified pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| `Column`1 | Specifies the column number where the search is to begin. |
| `Column`2 | Specifies the column number where the search is to end. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| ALL | Excludes all lines from the display. The ALL operand cannot be entered with any other operands. The default is ALL if EXCLUDE is entered with no operands. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |

| Operand | Description |
|---|---|
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Searches for all occurrences of the requested target and displays the number found in the short/long message field. This is the default value for the EXCLUDE command. |
| IN | Restricts the EXCLUDE command to the specified target type. |

## Usage Notes

The EXCLUDE command can be used to remove specific lines from the screen, resulting from a previous command.

Excluded lines are represented by a line of dashes and text stating n LINE(S) NOT DISPLAYED. The dashed line can be suppressed with the SET SHADOW OFF command.

See the SET command in online help or "SET Command" on page 409 for more information.

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

- The asterisk (*) operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

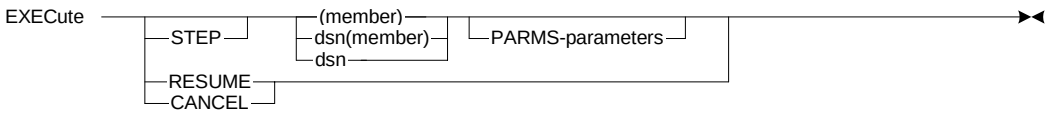
You can also exclude lines using the X or XX line commands or by selecting View ▸ Exclude (which displays the View - Exclude Request pop-up), or by using one of the Search pop-ups available on the Search pull-down.

## Example

To remove all non-highlighted lines from the screen, type this command:

```
EXCLUDE NHI
```

# EXECUTE Command




## Function

The EXECUTE command is used to read and execute the commands in a SmartTest script file.

Script files can contain EXECUTE commands that execute lower level (nested) script files. The STEP operand allows stepping through each script command.

## Operands

| Operand | Description |
|---|---|
| STEP | Steps through the commands in the script file. Each command in the script file displays in the command input area. The displayed command can be changed or erased from the command input area. Press Enter to execute the displayed command. Processing of the script file continues in this manner until all commands have been displayed and/or executed. |
| RESUME | Executes the remaining script file commands without stepping through each. EXECUTE RESUME is entered after a script file has been executed with the STEP operand specified. |
| CANCEL | Cancels the remaining script file commands. EXECUTE CANCEL can be entered after a script file has been executed with the STEP operand specified. |
| (*member*) | Specifies a script member in a default script file defined during product installation. You can modify the default script files, or their concatenation sequence, for your profile by selecting Options ▸ Script file allocation. The parentheses are required. |

| Operand | Description |
|---|---|
| *dsn* | Specifies the dataset name of a sequential file that contains the script commands to be executed. This dataset must be in card image format (LRECL=80). The specified dataset name must be entered using TSO usage conventions as shown:<br>`'sequential.dataset.name'`<br>If the quotes are omitted, your TSO prefix or user ID is the high-level qualifier. |
| *dsn*(*member*) | Specifies the dataset and member name of a partitioned file that contains the script commands to be executed. This member must be in card image format (LRECL=80). The specified dataset member must be entered using TSO usage conventions in this format:<br>`'pds.dsn(member)'`<br>If the quotes are omitted, your TSO prefix or user ID is the high-level qualifier. |
| PARMS *parameters* | Specifies the parameters to be passed to the script being executed.<br>Commands within a script file may contain substitution variables numbered &1 through &9. Parameter values are passed at execution time to the substitution variables using the PARMS operand of the EXECUTE command. Enter the parameters values in the numerical order 1 through 9. If more than 9 parameter values are entered, the first nine are used and the remainder are ignored. If substitution variables are included in the script and parameters are not included in the EXECUTE command, the substitution variable values are passed as null.<br>Substitution variables can be used anywhere in the executable script commands, but are not substituted if they occur in comment lines.<br>Parameter values may contain any non-quoted literal without embedded blanks. |

## Usage Notes

The EXECUTE command reads the specified dataset and executes the script contained in the file. The script can contain EXECUTE commands that execute lower level (nested) script files. Script files that create a loop are recognized and an error message displays.

The STEP operand provides a means of stepping through each script command. This allows the opportunity to modify or skip commands as desired.

Scripts must be sequential datasets or members of a PDS. In both cases, scripts must be in card image format (LRECL=80).

Scripts can be used to perform these repetitive tasks:

- Initialize variables for a test session
- Set default values
- Set up a test session
- Execute a test
- Reexecute a test
- Execute a predefined command sequence

Comments can be included in script files by entering an asterisk (*) in column one followed by the comment text.

You can execute script files by selecting File ▸ Execute, which displays the File - Execute Script pop-up.

## Examples

To have the EXECUTE command cause the first command in the script file to be displayed in the command input area, type this command:

```
EXECUTE STEP 'USERID.STT00015.VIASCRIP'
```

You can modify the displayed script command if desired and then press Enter. The command is executed and the next command in the script file displays in the command input area. Any command can be changed or bypassed (erased) when it displays in the command input area.

To execute all script commands in member SCRIPT01 in the default script datasets sequentially, type this command. These commands can consist of any valid SmartTest primary command (i.e., BREAK, BRANCH, FX, LPRINT).

```
EXECUTE (SCRIPT01)
```

To pass the program name MYPROG and the dataname HIRE-DATE as parameters to the script in member SCRIPT02 in the default script files, type this command:

```
EXECUTE (SCRIPT02) PARMS MYPROG HIRE-DATE
```

If the script contains these commands, the parameters are substituted in the commands in place of the substitution variables to allow flexibility in using the script:

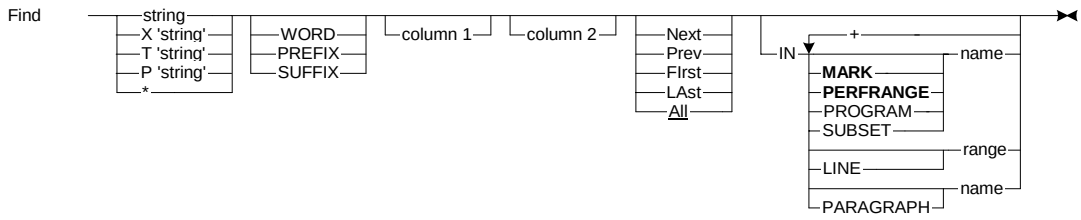
```
QUALIFY &1
FX &2
```

These are the actual commands resulting from the parameter substitution:

```
QUALIFY MYPROG
FX HIRE-DATE
```

See "PROCESS Command" on page 364 for additional uses of EXECUTE parameters.

# FIND Command




Bold operands are available only with ASG-Insight

## Function

The FIND command searches for one or all occurrences of the specified character string. The syntax is similar to the ISPF/PDF FIND command.

## Operands

| Operand | Description |
|---|---|
| *string* | Specifies a string of characters. If the string contains blanks, it must be enclosed in single or double quotes. The string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These operands describe how the string is to be used. |
| X'*string*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |

| Operand | Description |
|---|---|
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight, or scroll action or command. |
| WORD | Specifies a string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified string. |
| SUFFIX | Specifies a word that ends with the specified string. |
| Column1 | Specifies the column number where the search is to begin. |
| Column2 | Specifies the column number where the search is to end. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. This is the default value for the FIND command. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |

| Operand | Description |
|---|---|
| All | Searches for all occurrences of the requested target and displays the number found in the short/long message field. |
| IN | Restricts the FIND command to the specified target type. |
| MARK *name* | Specifies a name (1 through 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide*. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives.<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program including all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | A single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | A PROCEDURE DIVISION paragraph name or section name. The PROCEDURE and PROC literals can also be specified. The entire paragraph or section is included. |

## Usage Notes

The search begins from the current line when the NEXT or PREV operands are specified. All lines are searched regardless of the current line or direction when the ALL operand is specified. Any excluded lines containing FIND targets are included and displayed. The IN operand restricts the FIND command to only the specified targets.

To concatenate targets, place a plus sign (+) between the target names. A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | |
|---|---|---|
| PARAGRAPH | MARK | SUBSET |
| LINE | PERFRANGE | PROGRAM |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

You can also conduct a search by selecting a Search pop-up on the Search pull-down.

## Examples

To highlight all occurrences of XYZ, type this command:

```
FIND 'XYZ' ALL
```

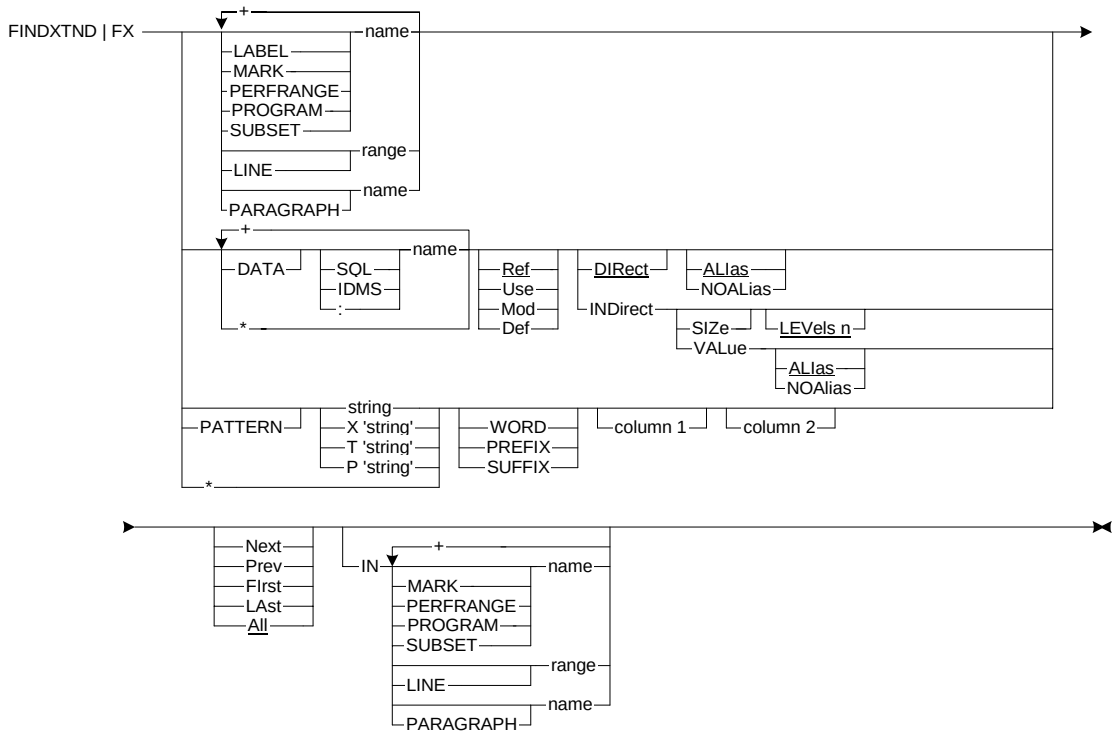
To search for the DATA NAME string without regard to case, type this command. The DATA NAME string can be uppercase, lowercase, or mixed case:

```
FIND T'DATA NAME'
```

To search for two non-blank characters separated by a blank, type this command:

```
FIND P'¬ ¬'
```

# FINDXTND Command




Bold operands are available only with ASG-Insight

## Function

The FINDXTND command performs a COBOL intelligent search of the source code for one or all occurrences of the specified target.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. This operand is available only if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies on of these predefined COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight or scroll action or command. |

| Operand | Description |
|---|---|
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the EXCLUDE command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT and INDIRECT. The defaults are REF, ALIAS and DIRECT. |
| SQL *name* | Includes datanames that are DB2/SQL variables only. |
| IDMS *name* | Includes datanames that are IDMS variables only. |
| : *name* | Includes datanames that are COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Def | Includes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |

| Operand | Description |
|---|---|
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels n | Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. |
| *`string`* | Specifies a string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*`string`*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*`string`*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |

| Operand | Description |
|---|---|
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Specifies the specified pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| Column1 | Specifies the column number where the search is to begin. |
| Column2 | Specifies the column number where the search is to end. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Searches for all occurrences of the requested target and displays the number found in the short/long message field. This is the default value for the FINDXTND command. |
| IN | Restricts the FINDXTND command to the specified target type. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | PATTERN |

- The asterisk (*) operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

Highlighting is used to indicate the occurrences found. If lines containing targets have been excluded, they are redisplayed on the screen. Targets highlighted previously are reset so that only the results of the current command are highlighted. Tags are placed on the source code lines indicating the type of target found.

## TAGS

These tags are placed in columns 73 through 80:

| Tags |
|---|
| MARK *name*<br>TRACK<br>NETWORK<br>MARK<br>SUBNET |
| PERFRANGE *name*<br>PERF RNG |

| Tags |
|---|
| PROGRAM *name* |
| PROGRAM |
| PGM |

LABEL *name*

LABELREF

SUBSET *name* - COBOL

| | | |
|---|---|---|
| ASsignment | DEBug | IO |
| CAll | DEFinition | LABel |
| CIcs | DIRective | MAINline |
| COBOLII | DIVision | MATH |
| COBOL/370 | DL/I \| DL/1 | Output |
| COMment | DML | PARagraph |
| CONditional | ENtry | PERform |
| COPy | EXIt \| PGMExit | RETurn |
| DB2/SQL | FALLthrough | SECtion |
| DDL | GOto | SORTMerge |
| DEAD | IDMS | STructure |
| DEADCode | INClude | TAG |
| DEADData | Input | TESted \| UNTested |

**Note:**

The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.

SUBSET *name* - SCREEN

HIGH

NONHIGH

EXCLUDE

NONX

| Tags |
| --- |
| SUBSET *name* - TRACE tags<br>DECISION<br>OPTION<br>START<br>TARGET |
| LINE *range*<br>LINE RNG |
| PARAGRAPH *name*<br>PARAGRAP |
| DATA *name*<br>DATA REF<br>DATA USE<br>DATA MOD<br>DATA DEF |
| PATTERN *string*<br>PATTERN |

The NEXT and PREV operands start the search from the current position and locate the closest occurrence. If ALL is specified, all lines are searched regardless of the current line or direction. A message displays indicating the number of targets found.

When an unqualified dataname is entered and more than one occurrence of the specified dataname exists, all occurrences are found. This provides a means of seeing all like datanames but under different group level names, then selecting the appropriate dataname. A message displays if an invalid qualification is entered.

When a dataname is entered with the REF operand, the screen is positioned to show the first occurrence of the dataname in the PROCEDURE DIVISION. DEF information in the DATA DIVISION is also highlighted.

**Note:**

You can also initiate a search by selecting a Search pop-up on the Search pull-down.

## FX INDIRECT

A dataname can impact other datanames in source statements. These datanames can be subsequently propagated indirectly through other statements, and referenced at each level. For example, consider:

```
MOVE A TO B
MOVE B TO C
MOVE C TO D
```

When the dataname of interest is A, the first statement contains a direct reference to A. The direct reference to a dataname is level 1. The last two statements contain indirect references to A, based on the propagation of the dataname A by moving its value to B, then to C, then to D. The second statement is a level 2 reference to dataname A, and the last statement is a level 3 reference to dataname A.

All levels of the indirect impact to a dataname displays, or you can view one level at a time using the LEVELS operand. For example, to locate all datanames that could be indirectly affected by a change in the size of the specified dataname, type `FX A SIZE`. Because the LEVELS operand is not specified, the default of all levels is used. This results in a complete list of all datanames that must be reviewed (for a size change of the specified dataname). If the LEVELS operand is specified (e.g., FX A SIZE LEVELS 2), only the results of that many levels are highlighted. The ripple effect can be more clearly identified by starting with LEVELS 1, then increasing the number of levels to be searched.

When using INDIRECT SIZE, any occurrence of an affected dataname in the USING clause of a CALL statement is tagged as a MOD. The called subprogram must be examined for other datanames that are affected as a result of the CALL. IN target can be used to restrict the search to the specified IN target.

When the REF operand is specified or used by default, all appropriate references are highlighted. If the reference is a DEFinition, USE, or MODification, the line is tagged accordingly. If the line contains a USE and a MODification, it is tagged as a REFerence.

### *Examples*

To search backward for the previous occurrence of a use of DATE-CODE or its aliases, type this command:

```
FINDXTND DATE-CODE USE PREV
```

To highlight the definition of DATE-CODE in the DATA DIVISION, type this command:

```
FX DATE-CODE DEF NOALIAS ALL
```

To highlight all occurrences of the SWITCH field used in conditional statements, type this command:

```
FX SWITCH IN COND ALL
```

In this example, if you enter a FINDXTND command for C with ALIAS specified, these pertain:

- A is the parent.
- D and E are the children.
- Z is a redefine/rename.
- Z1 is a redefine/rename.
- L is a rename.

```
01  A.
     05  B        PIC X.
     05  C.
         10  D    PIC X.
         10  E    PIC X.
 01  Z REDEFINES A.
     05  Z1       PIC XXX.
 66  L RENAMES Z.
```

These statements are used in the following FINDXTND examples:

```
000010  READ INFILE INTO A.
000020  MOVE A TO B.
000030  MOVE B TO C.
000040  WRITE OUTFILE FROM C.
```

This FINDXTND command shows the possible origin of the data value:

```
FX B MOD INDIRECT
```

The MOD operand is used to locate only those places where B is directly or indirectly set or modified. The fields in lines 10 and 20 are highlighted since B is modified by A on line 20, and A is modified by the READ statement on line 10.

This FINDXTND command shows all possible destinations of a data field:

```
FX B USE INDIRECT VALUE
```

The USE operand is used to locate every place B is directly or indirectly used. The fields in lines 30 and 40 are highlighted because both statements use B. The above FINDXTND command shows that the value of B is used to modify C on line 30. On line 40, the value of C that was received from B is used.

In this example, the FINDXTND command is used to locate all groups whose definitions might have to be changed every time the definition for B changes:

```
FX B REF INDIRECT

 000010  01  A.
 000020      05  B.       PIC XX.
           2
 000070  01  C.
 000080      05  D        PIC XX.
           2
 001010      MOVE B TO X.
 001020      MOVE C TO A.
 001030      MOVE SPACES TO D.
```

The results are lines 10, 20, 70, 80, 1010, and 1020. Any change in the size of B is a change in the size of A as well. This means the definition for C should be changed since A now has a different record layout, and the statement on line 1020 would not execute as intended. D is also shown as a result because it is at the same offset within C as B is within A, and is the same size as B.

**Note:**

The number of results from FX INDIRECT REF depends on the target name. For best results, select a target that is the smallest group item whose subfields are to be modified.

This example uses this code to show the LEVELS, VALUE, and SIZE operands:

```
000010  MOVE A TO B.
000020  MOVE X TO B.
000030  MOVE Y TO X.
000100  MOVE Y TO A.
```

To identify the datanames in the program directly affected by a change in A, type this command. This list consists of only dataname A and its related aliases.

```
FX A INDIRECT LEVELS 0
```

To identify the statement (move of size from A to B) that shows B might be affected by this move, type this command. This statement (line 10) and the definitions of A and B are highlighted.

```
FX A INDIRECT LEVELS 1
```

To highlight the definitions of A, B, and X, plus the statements on lines 10 and 20, type this command:

```
FX A INDIRECT LEVELS 2
```

To identify all direct and indirect uses of the value of A, type this command. The statement on line 10 is highlighted.

```
FX A USE INDIRECT VALUE
```

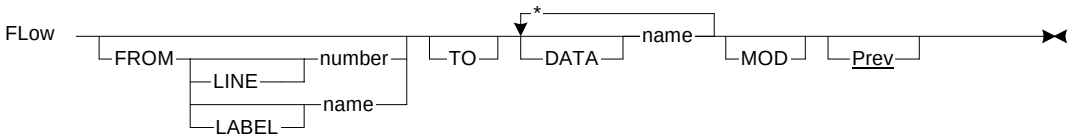
To identify all direct and indirect modifications to the value of A, type this command. The statement on line 100 is highlighted.

```
FX A MOD INDIRECT VALUE
```

To identify where A is affected by a change in the size of the target dataname, type this command. The statements on lines 20 and 100 are highlighted.

```
FX A USE INDIRECT SIZE
```

# FLOW Command (without Insight)




Extended Analysis only

## Function

Identifies all possible execution paths from a given point in a program, backwards through the execution flow to the modification point of the specified target(s).

This is a limited version of the Insight FLOW command available when Insight is not installed and an Extended Analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| FROM | Specifies an optional keyword entered with a line number or label to indicate where the FLOW command is to begin. |
| LINE *number* | Specifies the line number where the FLOW command is to begin. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| TO | Specifies an optional keyword followed by a target for the FLOW command. |
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. All occurrences of the dataname where its value is being set or modified are included. Flow is backward from the current cursor position to the previous occurrence of the requested target. |

| Operand | Description |
|---|---|
| Mod | Specifies an optional keyword that follows the dataname indicating that all occurrences of the dataname, where its value is being set or modified, are included. The default is Mod. |
| Prev | Specifies an optional keyword following the dataname indicating that flow is backward from the current cursor position to the previous occurrence of the requested target. The default is Prev. |

## Usage Notes

The FLOW command is used to determine the execution flow of a program as it specifically relates to the previous modification of a dataname. For example, if the program under test abends with invalid data in a variable, FLOW can be used to highlight the statement(s) that may have modified the variable. Since the FLOW command finds all possible paths to the current statement, multiple targets can be highlighted.

To concatenate targets, place a plus sign (+) between the target names. A concatenated dataname can be used wherever a dataname is valid. All occurrences of the dataname where its value is being set or modified are included.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the dataname ZIP-CODE appears in data structures MASTER-RECORD and WORK-RECORD, it can be qualified with OF (e.g., ZIP-CODE OF MASTER-RECORD).

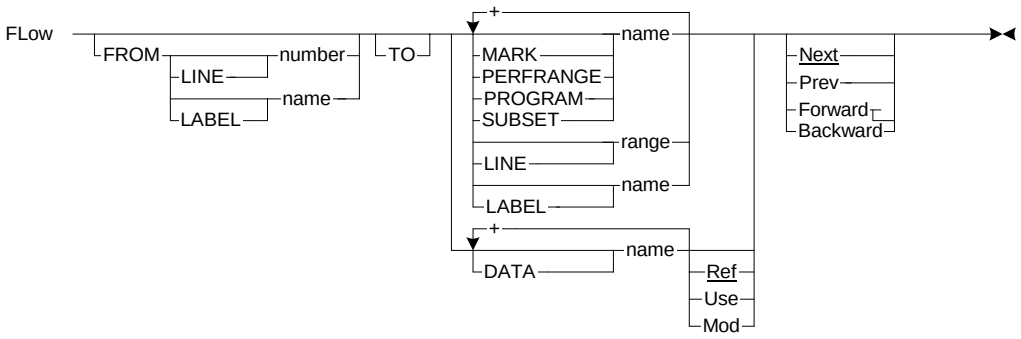
The FROM operand is used as the starting point for the FLOW command. If a FROM operand is not specified, the cursor location is used as the starting point. The cursor positions result in these starting points:

| Cursor Position | Description |
|---|---|
| Command input area | Starts the search in the first column of the source line shown at the top of the screen. |
| Line command area | Starts the search from the first column on that line. |
| Program source | Starts the search from the cursor location. |

If the cursor is positioned on an executable COBOL statement, and the cursor is after the last character of the statement, the statement is included in the search.

When Insight is installed, the full power of the Insight FLOW command (and also the TRACE command) is available within SmartTest. See the "FLOW Command (Insight Only)" on page 294.

# FLOW Command (Insight Only)




ASG-Insight only

## Function

The FLOW command follows all possible execution paths from a given point in a program searching for the specified target(s). The identified paths are stored with Mark names of NETWORK and SUBNET*n*.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| FROM | Specifies an optional keyword entered with a line number or label to indicate where the FLOW command is to begin. |
| LINE *number* | Specifies the line number where the FLOW command is to begin. |
| LABEL *name* | Specifies the paragraph or section name of the PROCEDURE DIVISION or the literals PROCEDURE or PROC where the FLOW command is to begin. |
| TO | Specifies an optional keyword that is followed by a target for the FLOW command. |

| Operand | Description |
|---|---|
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>The TRACK, NETWORK, and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information. |

| Operand | Description |
|---|---|
| | A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the FLOW command: REF, USE, and MOD. The default is REF. |
| Ref | Includes all occurrences of the dataname where its value is tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes all occurrences of the dataname where its value is being tested or used. |
| Mod | Includes all occurrences of the dataname where its value is being set or modified. |
| Next | Flows forward from the current cursor position to the next occurrence of the requested target. This is the default value for the FLOW command. |
| Prev | Flows backward from the current cursor position to the previous occurrence of the requested target. |
| Forward | Flows forward from the current cursor position to all occurrences. |
| Backward | Flows backward from the current cursor position to all occurrences. |

## Usage Notes

The FLOW command is used to determine the execution flow of a program and indicates if the specified targets can be reached from a given point. A path of the executable statements that are followed is captured and called NETWORK. The NETWORK can be saved and used for additional analysis. A NETWORK can also be displayed using the FINDXTND command.

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, USE, MOD) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | |
|---|---|---|
| LABEL | MARK | SUBSET |
| LINE | PERFRANGE | PROGRAM |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

The FROM operand is used as the starting point for the FLOW command. If a FROM operand is not specified, the cursor location is used as the starting point. The cursor positions result in these starting points:

| Cursor Position | Description |
|---|---|
| Command input area | Starts the search in the first column of the source line shown at the top of the screen. |
| Line command area | Starts the search from the first column on that line. |
| Program source | Starts the search from the cursor location. If the cursor is positioned on an executable COBOL statement, the starting point of the search is based on the location of the cursor as well as the target and direction specified in the FLOW command. |
| FLOW data-name forward | If the cursor is before the statement, the statement is included. |
| FLOW data-name backward | If the cursor is after the last character of the statement, the statement is included. |

If the starting point is an unexecutable COBOL statement, these general rules apply:

- Comment lines and compiler directives occurring at the end of an executable code block are considered part of the paragraph or section that follows. The search begins at the paragraph or section name that follows.
- A line followed by one or more executable statements prior to the succeeding section, or a line that is part of the last section, is considered part of the preceding section. The search begins at the end of the statement before the comment or directive.
- Blank lines that occur at the end of an executable code block, and that are not preceded by a comment or compiler directive, are considered part of the preceding paragraph or section. The search begins at the end of the last executable statement in the preceding paragraph.
- A blank line that follows all executable statements of the preceding paragraph, and that is preceded by a comment or compiler directive, is considered part of the following paragraph. The search begins at the paragraph or section name in the following paragraph.
- A blank line that is followed by one or more executable statements, or that is part of the last paragraph or section of the program, is considered part of the preceding paragraph. The search begins at the end of the preceding paragraph.

## Special Conditions

If you specify a dataname with USE NEXT or USE PREV, special conditions are considered. When the FLOW command searches for the next use, the value currently in the dataname is the target of the search. If a modification of that variable is encountered along a path, that portion of the execution path is not reflected in the result. The same is true for PREV. FLOW locates all uses occurring before a modification of the dataname. For example:

```
MOVE ZIP-CODE TO OUT-ZIP-CODE   (ZIP-CODE = 46201)
MOVE HOLD-ZIP TO ZIP-CODE       (HOLD-ZIP = 93721)
MOVE ZIP-CODE TO OUT-ZIP-CODE   (ZIP-CODE = 93721)
```

If the FLOW USE PREV command is entered on the last line, the first line would not be shown as a result because the value of ZIP-CODE is modified between the starting and ending points.

You can identify execution paths by selecting an action on the Logic pull-down.

## Example

This FLOW command flows to the previous uses of LOAN-AMT where the same value was used.

```
FLOW LOAN-AMT USE PREV
```

# FORCE Command

```
FORCE ──────────────────────────────────────────────────────────────────────►◄
```

## Function

The FORCE command is used to force an update from the Memory Display screen to be made after a software protected error message displays. You must have proper authorization to use this command.

**Note:**

This command is available only in the CICS environment.

## Operands

None.

## Usage Notes

The SmartTest-CICS FORCE command protects all storage areas that do not belong to the transaction. If FORCE is used to override this protection, then care should be taken to ensure that CICS is not corrupted during this override process.

# GO Command

```
GO ──┬─TO─┬─┬─label──────┬──────────────────────────────────►◄
     │    │ ├─pslabel────┤
     │    │ └─line───────┘
```

## Function

The GO command transfers control to the statement containing the specified COBOL or Assembler label name, PL/I label or procedure name, pseudo code label, or line.

## Operands

| Operand | Description |
|---|---|
| TO | Specifies an optional keyword used as a connective for the GO command. |
| *label* | Specifies a standard COBOL paragraph or section name, Assembler label, or PL/I label or procedure name. |
| *pslabel* | Specifies a pseudo code label name. The COBOL names are entered in Area A and Assembler labels are entered in column 1 and can be used to identify blocks of pseudo code. |
| *line* | Specifies a COBOL, Assembler, or PL/I statement's source line number. |

## Usage Notes

Transferring control to a label outside of the paragraph being executed may cause unpredictable results. Use of the GO Command on a program compiled with the OPTIMIZE compiler option can also cause unpredictable results. See the optimized Assembler output from the compiler as a navigational aid for testing optimized programs.

You can also transfer control by selecting Test ▸ Go.

# HELP Command

```
Help | ?  ──┬──────────────────────────────┬──────────────────────────────────────────►◄
            ├─MESSAGE | MSG─┬─────────────┤
            ├               └─VIAmsg# | msg#─┤
            ├─COMMANDs──────────────────────┤
            ├─cmd───────────────────────────┤
            ├─ABENDs────────────────────────┤
            └─SCREEN────────────────────────┘
```

## Function

The HELP command is used to display information about the current SmartTest screen, commands, messages, or abend codes.

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Help Tutorial for the current screen if no long message displays on the screen, which describes all fields on the screen and any special processing considerations. If a long message displays on the screen, typing `HELP` with no operands displays the Help Explanation and Action Panel for the current message. |
| MESsage \| MSG | Displays the Help Explanation and Action Panel, which shows the current short and long message, an explanation of the current message, and any actions to be performed. |
| ASGmsg# \| msg# | Specifies the ASG message number for which the Help Explanation and Action Panel is to be displayed. This number consists of 1 to 4 digits. It is not necessary to enter leading zeros. |
| COMMANDs | Displays a list of all SmartTest primary commands. Information for a particular command can then be displayed by selecting the appropriate number. |
| *cmd* | Specifies a SmartTest primary command. Typing `HELP` followed by the command name displays the command level Help Tutorial screen that shows the command syntax and operand descriptions. |

| Operand | Description |
|---|---|
| ABENDs | Displays the ABENDS screen. Selecting Topic 2 on this screen displays the Abend Codes screen, which lists all the ASG user abend messages, and explanations for each message. |
| SCREEN | Displays help for the current screen or pop-up, which describes all fields on the screen or pop-up and any special processing considerations. |

## Usage Notes

Help can also be viewed using these methods:

- To see help for a command, enter a command or keyword, then press PF1/13.
- To see screen information, press PF1/13 while the screen displays.
- To see an explanation of a message and actions (if any), enter the message number, then press PF1/13.
- Select Help from the action bar.

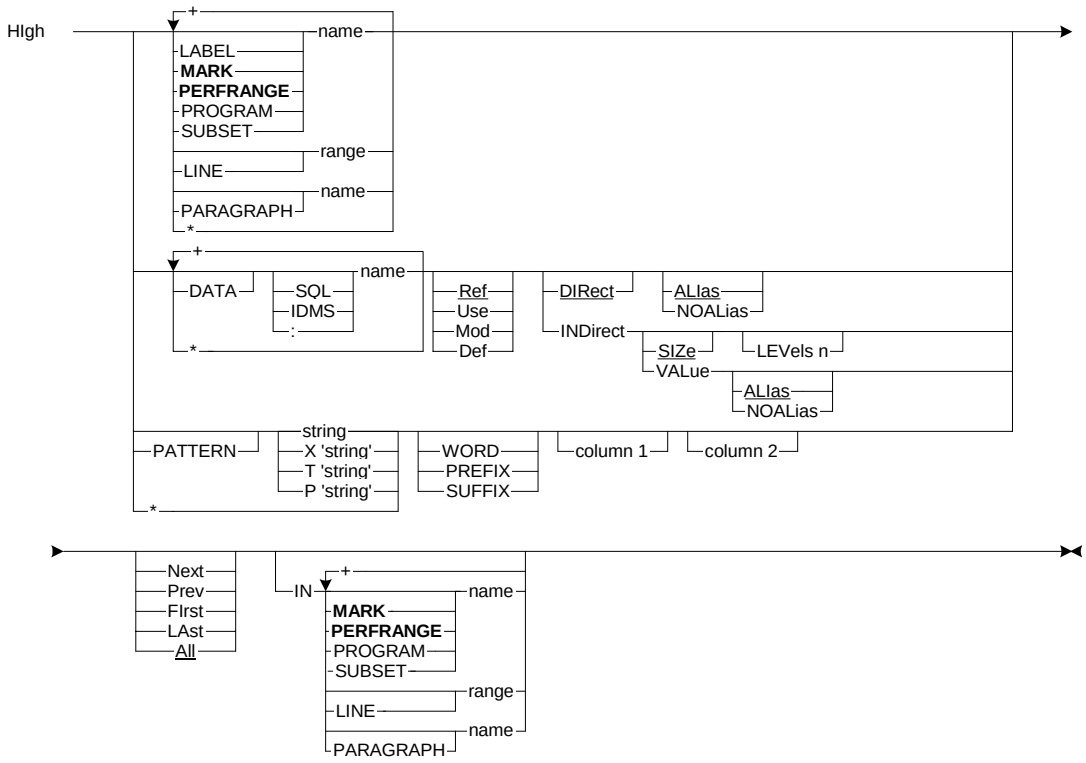
See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for additional information about SmartTest online help as accessed through the action bar.

## Example

This command displays the Help Explanation and Action screen for message 1229. The explanation for the message describes why the message displays, and the action portion of the screen describes any required or suggested user actions.

```
HELP ASG1229
```

# HIGH Command




Bold operands are available only with ASG-Insight

## Function

The HIGH command highlights source code lines containing the specified targets. Lines that are already highlighted are not reset.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. This operand is only available if Insight is installed and an Insight analysis has been run on the program. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6 of the Program View screen. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight or scroll action or command. |

| Operand | Description |
|---|---|
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the HIGH command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |
| SQL *name* | Includes datanames that are DB2/SQL variables only. |
| IDMS *name* | Includes datanames that are IDMS variables only. |
| : *name* | Includes datanames that are COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Def | Includes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |

| Operand | Description |
|---|---|
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified number of indirect levels. The VALUE subordinate operand is not used with LEVELS. The default is All levels for the LEVELS operand. LEVELS is only valid with the INDIRECT operand. |
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. |
| *string* | Specifies a string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*string*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |

| Operand | Description |
|---|---|
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Identifies the specified pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| *column*1 | Specifies the column number where the search is to begin. |
| *column*2 | Specifies the column number where the search is to end. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Searches for all occurrences of the requested target and displays the number found in the short/long message field. This is the default value for the HIGH command. |
| IN | Restricts the HIGH command to the specified target type. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (e.g., REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | PATTERN |

- The asterisk (*) operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

Lines can be added to a highlighted set. For example, to see all PERFORM statements, then to add the conditional statements related to one of the PERFORM statements, use FINDXTND PERFORM, then use HIGH CONDITIONAL to add the additional lines.

To step through all IO statements, enter HIGH IO NEXT, then keep repeating the command until the end of the program is located.

You can also highlight selected lines by selecting a Search pop-up on the Search pull-down.

## Example

To highlight all lines containing IO statements, type this command:

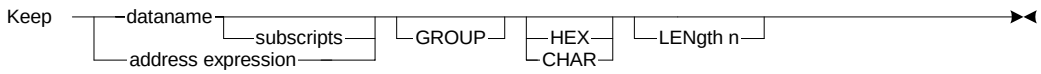
```
HIGH IO
```

Existing highlighted lines are not reset when this command is issued:

```
FX LAB IN HI
```

Only labels within the highlighted lines are highlighted. All other highlighted lines are reset.

# KEEP Command




## Function

The KEEP command is used to keep the value and address of the specified data item displayed at the top of the Program View screen. The screen can be scrolled, and the kept lines remain displayed at the top of the screen.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a data item for which the value and address is to be displayed at the top of the screen. |
| *subscripts* | Specifies subscript information using the standard COBOL syntax:<br>(*subscript1, subscript2, ... subscriptn*)<br>Left and right parentheses and at least one blank between subscripts are required. Commas between subscripts are optional. A subscript may be a numeric literal, an index data item, a numeric data item, or a numeric data item plus or minus a numeric literal. |
| *address expression* | Specifies an address expression consists of either an address or a register number followed by a maximum of 32 indirection indicators and/or offsets.<br>These are definitions of the possible components of an address expression. The address expression must not contain spaces. |
| Address | X'*nnnn*' An absolute address specified in hexadecimal notation. |
| Registers | R*n* A register 0 through 15.<br>*n*R A register 0 through 15. |
| Indirection indicators | % Indicates a 24-bit indirection.<br>? Indicates a 31-bit indirection. |

| Operand | Description | |
|---|---|---|
| Offsets: | + | Indicates that the following offset is to be added. |
| | - | Indicates that the following offset is to be subtracted. |
| | X'*nn*' | An offset specified in hexadecimal notation. |
| | *nn* | An offset specified in decimal notation. |
| | R*n* | A register 0 through 15 that contains the offset. |
| | *n*R | A register 0 through 15 that contains the offset. |
| GROUP | Displays the levels, values, and addresses of the group items associated with the specified dataname. | |
| HEX | Displays the value and address of the specified dataname in hexadecimal format. | |
| CHAR | Displays the value for the specified data item in character format. | |
| LENgth *n* | Displays the value of the data item specified for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. | |

## Usage Notes

All Keep windows are sizable and scrollable. Use the SET KEEP command to control the number of lines used by the Keep window. When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. Place the cursor within a Keep window and use the UP command or the DOWN command to scroll. The address commands, L and LX, can be used within Keep windows. See the Memory Display screen in online help for more information about the L and LX commands.

You can enter the KEEP command using these abbreviations:

| | |
|---|---|
| KG `dataname` | Keep Group |
| KH `dataname` | Keep Hex |
| KGH `dataname` | Keep Group Hex |

The maximum number of lines that can be specified in the Keep line area is 60. Lines can be deleted from the Keep line area using the D (Delete) line command. Any display and/or input fields in lines being kept are available for editing in the kept lines.

The VALUE area contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

An effective picture clause displays on the dataname line. For example:

- A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
- An alphanumeric field PIC XXXX displays as X(4).
- A field defined as COMP displays as COMP-4.
- A field with a long dataname that is defined as COMP displays as C3 or C4.

These notes apply to using the KEEP command with Assembler programs.

- The KEEP command can be used in Assembler source programs with CSECT relative named data items and literal data items. CSECT relative named data items are data items that occupy storage locations within the supported CSECT of the source module. Literal data items are those data items defined in the supported CSECT that begin with an equal (=) sign. Unnamed data items are shown with the name UNNAMED ITEM.
- Data items defined outside of the supported CSECT, are not supported.

You can keep a value at the top of the Program View by selecting Test ▸ Keep.

## Examples

To display the value and address of the ZIP-CODE data item at the top of the screen, type this command:

```
KEEP ZIP-CODE
```

To display the levels, values, and addresses of the COMPARISON-KEYS data item and its associated group items, type this command:

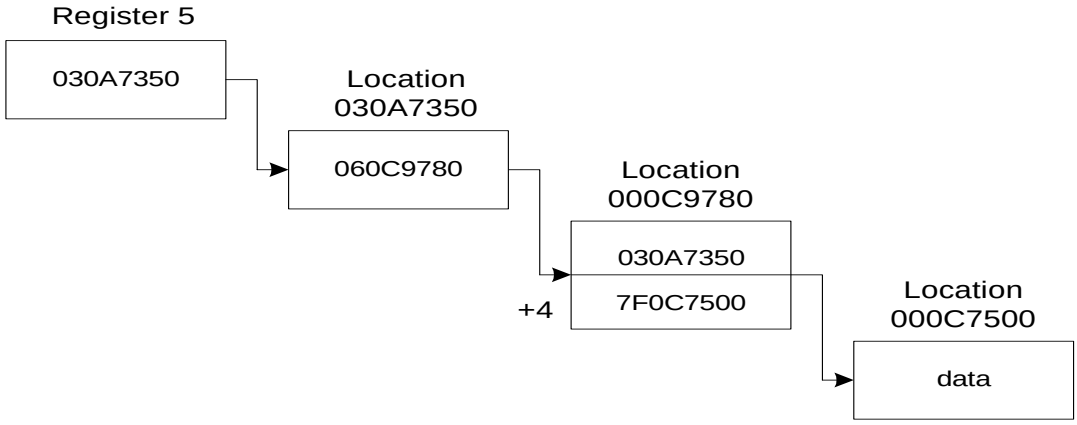
```
KEEP COMPARISON-KEYS GROUP
```

To display data at the address expression specified, type this command:

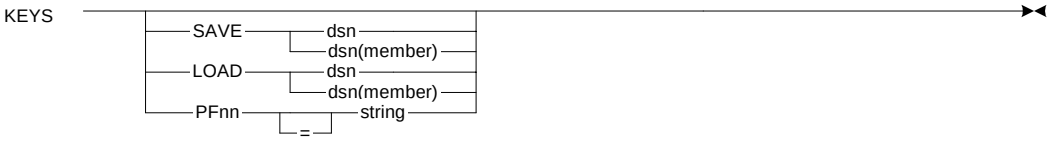
```
KEEP R5?%+4% CHAR LEN 50
```

Figure 134 shows this address expression graphically:

**Figure 134 • Example of Address Expression**



# KEYS Command




## Function

The KEYS command, with no operands, displays the Options - PF Key Definition pop-up. This screen is used to display and/or modify the current SmartTest PF key assignments.

The KEYS command may also save, restore, or modify the PF key assignments without displaying the Options - PF Key Definition pop-up.

## Operands.

| Operand | Description |
|---|---|
| SAVE | Saves the current PF Key definitions in a sequential or partitioned dataset. If you do not specify a dataset name, the default name is:<br>`<TSO UserID>.PPPnnnnn.VIAPFKEY`<br>where *nnnnn* is a generated unique number. See the description of *dsn* and *dsn(member)* for additional dataset naming rules |
| LOAD | Restores the PF Key definitions from the specified sequential or PDS. A dataset name must be specified for the LOAD operand. |
| *dsn* | Specifies a sequential dataset name. If quotes are included, the dataset name is used as is. If there are no quotes, your TSO User ID is appended to the front of the dataset name and VIAPFKEY is appended to the end of the dataset name.<br>For example:<br>`SAVE TEST1`<br>generates this dataset name:<br>`<TSO UserID>.TEST1.VIAPFKEY` |

| Operand | Description |
|---|---|
| *dsn*(*member*) | Specifies a partitioned dataset and member name. If quotes are included, the dataset name and member name are used as is. If there are no quotes, your TSO prefix or user ID is appended to the front of the dataset name. If the PDS member already exists, it is overwritten. |
| PF*nn* | Provides a command line update to the PF Key definition. For example:<br>`KEYS PF21 RESET EXCLUDED`<br>assigns the text RESET EXCLUDED to PF Key 21.<br>If the text to be assigned begins with an equal (=) symbol, you must enter two equal symbols separated by a space.<br>If a null string is entered for the PF*nn* operand, the PF Key assignment reverts to the default value for the current ESW product.<br>The script facility verifies that the PF Key number is valid. |

## Usage Notes

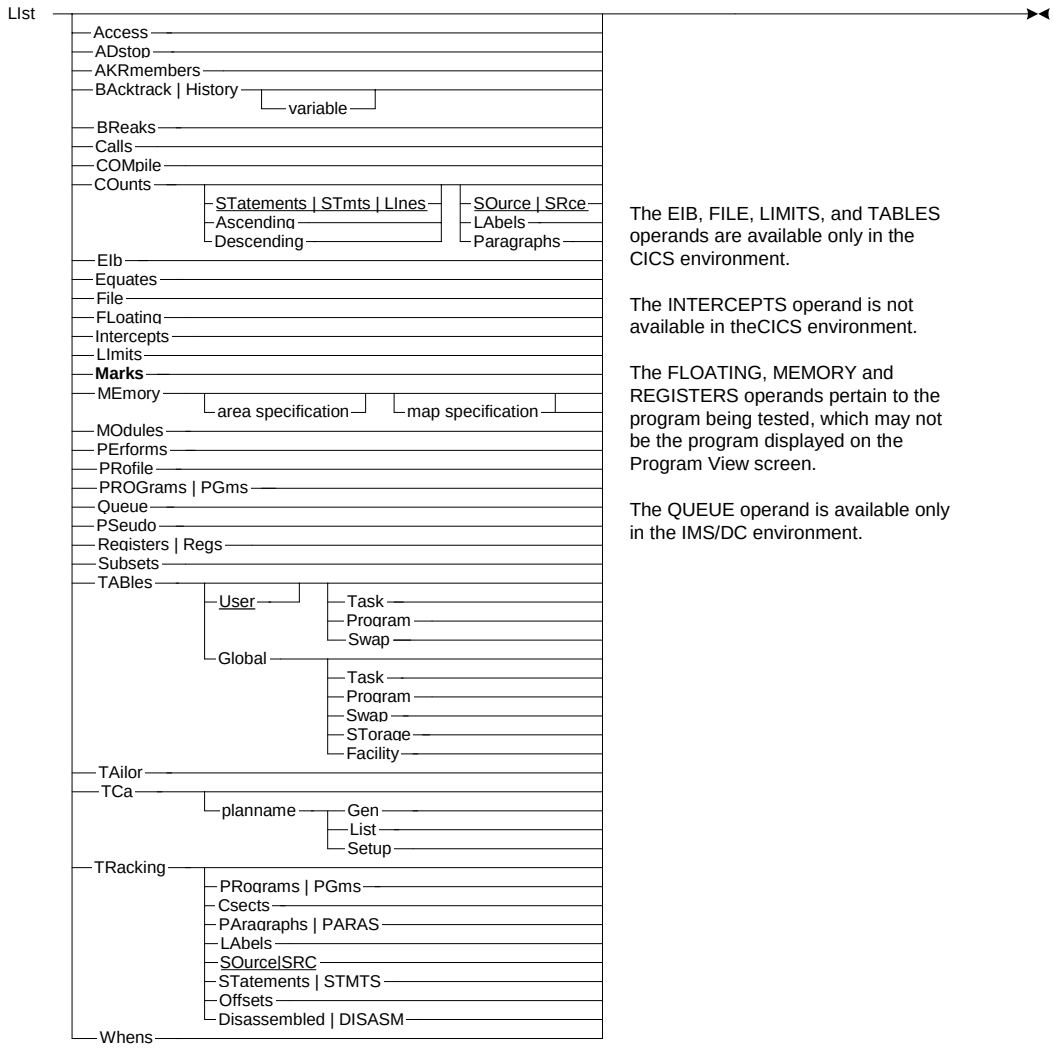
The KEYS command can be entered without operands on any SmartTest screen to display and/or modify the current PF key assignments. Values assigned to the SmartTest PF keys have no effect on other ISPF applications.

The SAVE and LOAD operands can be used to save or restore PF Key settings. This feature allows the KEYS command to be used in scripts to set PF Key assignments as desired. This also provides a mechanism for multiple users to share a set of PF Key definitions.

The PF*nn* operand allows the KEYS command to be used in a script file to modify PF Key settings during script file execution.

You can display the Options - PF Key Definition pop-up by selecting Options ▸ PF Keys.

# LIST Command




The EIB, FILE, LIMITS, and TABLES operands are available only in the CICS environment.

The INTERCEPTS operand is not available in theCICS environment.

The FLOATING, MEMORY and REGISTERS operands pertain to the program being tested, which may not be the program displayed on the Program View screen.

The QUEUE operand is available only in the IMS/DC environment.

`Bold operands are available only with ASG-Insight`

## *Function*

The LIST command is used to display the specified screen. List screens are available for each operand shown above.

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Test Facilities List screen that is used to select any List screen for display. |
| AREGisters | Displays the Access Registers screen that lists the contents of the access registers for the program being tested. Register contents are displayed in hexadecimal and decimal formats and can be modified if desired. |
| ADstop | Displays the Address Stop Entry screen that is used to specify storage areas to be monitored during a test session. Address stops can be activated or deactivated on the Address Stop Entry screen. When activated, test execution is interrupted before the specified storage area is modified. Address stops may also be set using the STOP command. See the online help for more information about the Address Stop Entry screen. |
| AKRmembers | Displays the Environment - AKR Directory pop-up that is used to list all members in the current AKR, including concatenated AKRs. |
| BAcktrack | If no variable is entered with the command, displays the List - BackTrack Variable History pop-up that is used to select a variable that has been modified while in BackTrack recording mode. If a variable is specified with the command, the List - BackTrack Variable History screen displays. This screen shows all statements that have modified the specified variable. If desired, you can select a statement on the List - BackTrack Variable History screen to view the statement in the context of the execution history. |
| BReaks | Displays the Breakpoints List screen that is used to show all breakpoints in the program being tested or in the qualified program. Breakpoints can be activated or deactivated on the Breakpoints List screen. |
| Calls | Displays the List - CALL Statements pop-up that shows programs that are CALLed by the current program. The LPRINT * command can be entered on the List - CALL Statements pop-up to copy the screen contents to the List file. |
| COMpile | Displays the Compiler Options screen that lists compiler and optimizer information for the current program. |

| Operand | Description |
|---|---|
| COunts | Displays the Execution Counts screen that shows the current execution count and a histogram indicating the relative count for each executable statement. The LPRINT * command can be entered on the Execution Counts screen to copy the screen contents to the List file.<br><br>For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*. |
| STatements \| STmts \| Lines | Shows the current execution counts, sorting source by line. This is the default if COUNTS ASCENDING or COUNTS DESCENDING has not been specified during the current test session. This operand is used with the COUNTS operand. |
| Ascending | Shows the current execution counts, sorting source by ascending counts. Once specified this becomes the default until another COUNTS operand is specified. This operand is used with the COUNTS operand. |
| Descending | Shows the current execution counts, sorting source by descending counts. Once specified this becomes the default until another COUNTS operand is specified. This operand is used with the COUNTS operand. |
| SOurce \| SRce | Shows the current execution counts by source statement, sorting statements according to the sort option selected. This is the default if COUNTS PARAGRAPHS or COUNTS LABELS has not been specified during the current test session. This operand is used with the COUNTS operand. |
| LAbels | Shows the current execution counts by LABEL point, sorting labels according to the sort option specified. This operand is intended for use by Assembler programmers and is synonymous with the PARAGRAPHS operand. This operand is used with the COUNTS operand. |
| Paragraphs | Shows the current execution counts by PARAGRAPH name, sorting labels according to the sort option specified. This operand is synonymous with the LABELS operand. This operand is used with the COUNTS operand. |
| EIb (CICS only) | Displays the Exec Interface Block (EIB) screen. This screen is used to present a formatted display of the command level EIB for the purpose of review and updating. This operand is available only in the CICS environment. |

| Operand | Description |
|---|---|
| Equates | Displays the List - Equates pop-up that shows all equates for the program being tested or in the qualified program. The LPRINT * command can be entered on the List - Equates pop-up to copy the screen contents to the List file. |
| FIle (CICS only) | Displays the CICS File Services screen that is used to select the type of CICS data to be processed. File Services provide browse, update, add, and delete capability for CICS files including VSAM and BDAM (FCT), Temporary Storage (TS), Transient Data (DCT), DB2, and DL/I. This operand is available only in the CICS environment. |
| FLoating | Displays the Floating Point Registers screen that lists the floating point register values for the program being tested. Values on the Floating Point Registers screen can be changed if desired. |
| Intercepts (non-CICS only) | Displays the Load Module Intercept List screen. A load module that is linked, attached, or called may be entered on the Load Module Intercept List screen. Entries must be made before the start of the test. |
| LImits (CICS only) | Displays the Transaction Limits and Options screen. CICS transaction limits and options may be set and viewed on this screen. This operand is available only in the CICS environment. |
| Marks | Displays the List - User Marks pop-up that shows the marks, sets and paths for the program being tested. The LPRINT * command can be entered to copy the screen contents to the List file. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| MEmory | Displays the Memory Display screen that shows areas of storage in a hexadecimal/character format. Specific memory locations can be modified using the Memory Display screen if desired. |
| area specification | Specifies an optional address specification that can be entered with the MEMORY operand to specify the area of memory to be displayed. You can use absolute addresses (24-bit and 31-bit), general registers, or SmartTest storage related keywords. |

| Operand | Description |
|---|---|
| map specification | Specifies the high-level dataname for the structure and/or the PDS name containing the structure to be used for mapping. The supported map structures are external COBOL copybooks and PL/I Includes, as well as standard in-source structures. Map structures must also be in the native source of the program you are testing, or have been analyzed separately.<br>This dataname must be specified in this format:<br>`MAPDN=`*`dataname`*<br>**Note:**<br>The memory mapping function is memory and CPU intensive. Only basic structures are supported, with limited support for arrays or occurs. |
| MOdules | Displays the Module Directory screen that lists the modules residing in a load library. Information such as module size, entry point address, alias name, authorization code, and link edit attributes displays next to the module name. |
| PErforms | Displays the List - Perform Range Names pop-up that shows all COBOL PERFORM ranges in the program being tested or in the qualified program. The LPRINT * command can be entered on the List - Perform Range Names pop-up to copy the screen contents to the List file. |
| PRofile | Displays the Profile Data Set Member List screen which lists the profiles and may be used to copy from the dataset of another user, delete a member from the profile, restore an environment, or save the current environment ISPF profile information. |
| PROGrams \| PGms | Displays the internal subprograms defined with a COBOL II or later program. The internal subprograms are indented in relation to their hierarchy on the List - Program/ Subprogram Names pop-up. |
| PSeudo | Displays the Pseudo Code List screen that lists all pseudo code in the program being tested or in the qualified program. Pseudo code can be activated or deactivated using the Pseudo Code List screen. |
| Queue (IMS/DC only) | Displays the Processing Queue List screen that lists categories of IMS/DC processing queues. |

| Operand | Description |
|---|---|
| Registers \| Regs | Displays the General Registers screen that lists the contents of the general registers for the program being tested. Register contents are displayed in hexadecimal and decimal formats and can be modified if desired. |
| Subsets | Displays the List - COBOL Subsets Names pop-up that shows all subsets along with a brief description of each. The LPRINT * command can be entered on the List - COBOL Subsets Names pop-up to copy the screen contents to the List file. |
| TABles (CICS only) | Provides access to the user and global protection screens. Use this command without operands to display the User Protection Menu. Use the subordinate operands MENU, TASK, PROGRAM, SWAP, GLOBAL, STORAGE, and FACILITY operands to display other protection screens. This operand and its subordinate operands are available only in the CICS environment. |
| User | Displays the User Protection Menu when used with the TABLES operand without other operands. Use this operand with the TASK, PROGRAM, or SWAP operand to display screens that may be accessed from the User Protection Menu. |
| Task | Displays the user Task Specification screen when used with the TABLES operand or with the TABLES USER operands. Use this operand with the TABLES GLOBAL operands to display the global Task Specification screen. |
| Program | Displays the user Program Specification screen when used with the TABLES operand or with the TABLES USER operands. Use this operand with the TABLES GLOBAL operands to display the global Program Specification screen. |
| Swap | Displays the user Swap Specification screen when used with the TABLES operand or with the TABLES USER operands. Use this operand with the TABLES GLOBAL operands to display the global Swap Specification screen. |
| Global | Displays the Global Protection Menu when used with the TABLES operand without other operands. Use this operand with the TASK, PROGRAM, SWAP, STORAGE, or FACILITY operand to display screens that may be accessed from the Global Protection Menu. |
| STorage | Displays the Storage Specification screen when used with the TABLES GLOBAL operands. |

| Operand | Description |
|---|---|
| Facility | Displays the Facility Specification screen when used with TABLES GLOBAL operands. |
| TAilor | Displays the Test Session Tailoring screen that is used to selectively turn SmartTest features on or off for a set of modules and/or programs. |
| TCa | Displays the File - Test Plan Selection pop-up that is used to select a TCA option for setting up or modifying a plan, setting up a TCA test, or generating reports about the test results. |
| *planname* | Select from a list of TCA plans in the specified AKR. For a pattern, only a trailing asterisk (*) is allowed as a wild card character. |
| Gen | Displays the T.C.A. - Report Selection pop-up that is used to select the desired test results report. |
| List | Displays the T.C.A. - Test Results pop-up that is used to select the desired test results for review or export. |
| Setup | Displays the Session Setup screen that is used to run test coverage in the environment associated with this plan. |
| TRacking | Displays the Execution Tracking screen that lists the modules, programs, paragraphs, statements, and offsets that were last executed for the program being tested. The LPRINT * command can be entered on the Execution Tracking screen to copy the screen contents to the List file. |
| PRograms \| PGms | Displays the Execution Tracking screen listing programs that were last executed for the program being tested. |
| Csects | Displays the Execution Tracking screen listing CSECTs that were last executed for the program being tested. |
| PAragraphs \| PAras | Displays the Execution Tracking screen listing programs, CSECTs, and paragraphs that were last executed for the program being tested. |
| LAbels | Displays the Execution Tracking screen with listing programs, CSECTs, and labels that were last executed for the program being tested. This operand is for use by Assembler and PL/I programmers and is synonymous with the PARAGRAPHS operand. |

| Operand | Description |
|---|---|
| SOurce \| SRC | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, and source statements that were last executed for the program being tested. This is the default LIST TRACKING operand. |
| STatements \| STmts | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, and line numbers of statements that were last executed for the program being tested. |
| Offsets | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, and offsets that were last executed for the program being tested. |
| Disassembled \| Disasm | Displays the Execution Tracking screen listing programs, CSECTs, paragraphs, and disassembled source statements that were last executed for the program being tested. |
| Whens | Displays the When Conditions List screen that shows the When conditions in the program being tested or in the qualified program. When conditions can be made active or inactive using the When Conditions List screen. |

## Usage Notes

The LIST command is used to display the specified screen that shows the related information in a consolidated list. When the LIST command is entered without an operand, the Test Facilities List screen displays. From this screen, any LIST command category can be selected.

The FLOATING, MEMORY, and REGISTERS operands pertain to the program being tested, not necessarily the program being viewed on the Program View screen. For example, when the QUALIFY command is used during a test session, the program specified with the QUALIFY command replaces the program being displayed on the Program View screen. The test session however, remains active. If LIST REGISTERS is entered at this time, the general registers for the program being tested are displayed, not those for the qualified program.

You can also access List screens by selecting the associated action from the List pull-down.

## Examples

To display the List - COBOL Subsets Names pop-up that shows all available COBOL subsets and a brief description of each, type this command:

```
LIST SUBSETS
```

To display the Execution Counts screen, sorting source by descending counts, type this command:

```
LIST COUNTS DESCENDING
```

Figure 135 is an example of the COBOL Basic Memory Display.

**Figure 135 • COBOL Basic Memory Display**

```
Memory Display             TESTCOBL.TESTCOBL -A
 Command ===> ____________________________________________ Scroll ===> CSR

 Area 001293F0                              Offset 000000      Length 004000
 Member          Data Name
 001293F0  000000  F0F0F0F0 F0F1F0F0 C1E2D4C9 E3C84040  * 00000100ASMITH   *
 00129400  000010  40404040 40404040 C2D6C240 40404040  *         BOB      *
 00129410  000020  4040F1F2 F340C1D5 E840E2E3 D9C5C5E3  *   123 ANY STREET *
 00129420  000030  40404040 40404040 404040D7 C8D6C5D5  *            PHOEN *
 00129430  000040  C9E74040 40C1E9F8 F5F0F1F8 F1F2F3F4  * IX   AZ850181234 *
 00129440  000050  F140C1E2 C760E3C5 E2E3C3D6 C2D340C5  * 1 ASG-TESTCOBL E *
 00129450  000060  E7C1D4D7 D3C540D9 C5D7D6D9 E3404040  * XAMPLE REPORT    *
 00129460  000070  40404040 40404040 40404040 40404040  *                  *
 00129470  000080  40404040 40404040 40404040 40404040  *                  *
 00129480  000090  40404040 40404040 40404040 40404040  *                  *
 00129490  0000A0  40404040 40404040 40404040 40404040  *                  *
 001294A0  0000B0  40404040 40404040 40404040 40404040  *                  *
 001294B0  0000C0  40404040 40404040 40404040 40404040  *                  *
 001294C0  0000D0  40404040 40404040 40404040 40404040  *                  *
 001294D0  0000E0  40404040 40404040 40404040 40404040  *                  *
 001294E0  0000F0  40404040 40404040 40404040 40404040  *                  *
 001294F0  000100  40404040 40404040 40404040 40404040  *                  *
 00129500  000110  40404040 40404040 40404040 40404040  *                  *
 00129510  000120  40404040 40404040 40404040 40404040  *                  *
 00129520  000130  40404040 40404040 40404040 40404040  *                  *
 00129530  000140  40404040 40404040 40404040 40404040  *                  *
 00129540  000150  40404040 40404040 40404040 40C9C440  *               ID *
 00129550  000160  4040C1D9 40C140D3 C1E2E340 D5C1D4C5  *   AR A LAST NAME *
 00129560  000170  40404040 404040C6 C9D9E2E3 40D5C1D4  *        FIRST NAM *
 00129570  000180  C540E2E3 D9C5C5E3 40C1C4C4 D9C5E2E2  * E STREET ADDRESS *
 00129580  000190  40404040 40404040 40404040 C3C9E3E8  *               CITY *
 00129590  0001A0  40404040 404040E2 E340E9C9 D7404040  *         ST ZIP    *
 001295A0  0001B0  4EF44040 40404040 40404040 40404040  * +4                *
 001295B0  0001C0  40404040 40404040 40404040 40404040  *                  *
 001295C0  0001D0  40404040 40404040 40404040 40404040  *                  *
 001295D0  0001E0  7E7E7E7E 7E7E407E 7E407E40 7E7E7E7E  * ====== == = ==== *
 +-------------------------------------------------------------------------+
 |STATUS: STOPPED AFTER BREAK                PROGRAM: TESTCOBL  DATE: 29MAY2002 |
 |  STMT: 000125  OFF: 0004A8  AMODE: 31     MODULE: TESTCOBL  TIME: 10:27:46  |
 |SOURCE: PERFORM 2000-PROCESS-TESTDATA                                        |
 |R0-7 8B39DE0E 0B474F40 0B39D242 000DA7FC 00129120 8B39D424 00000000 0B474F40 |
 |R8-F 00130EE0 001293C0 0B39CE28 0B39D140 0B39CE20 0B4C9078 8012C2BE 00BF8830 |
 +-----------------------------------------------------------------------------
```

Figure 136 is an example of a COBOL List Memory Map.

**Figure 136 • COBOL List Memory Map**

```
                                Memory Display             TESTCOBL.TESTCOBL -A
Command ===> ____________________________________________ Scroll ===> CSR

Area 001293F0                            Offset 000000      Length 004000
Member          Data Name TESTDATA-RECORD
000046  01  TESTDATA-RECORD.
000047      05  TESTDATA-KEY.
000048          10  TESTDATA-KEY-CUST-NO                PIC 9(06).
001293F0  000000  F0F0F0F0 F0F1                        > 000001           <
000049          10  TESTDATA-KEY-CUST-AREA              PIC 9(02).
001293F6  000006  F0F0                                 > 00               <
000050      05  TESTDATA-DATA.
000051          10  TESTDATA-DATA-CUST-STATUS           PIC X(01).
001293F8  000008  C1                                   > A                <
000054          10  TESTDATA-DATA-CUST-NAME.
000055              15  TESTDATA-DATA-CUST-LNAME        PIC X(15).
001293F9  000009  E2D4C9E3 C8404040 40404040 404040    > SMITH            <
000056              15  TESTDATA-DATA-CUST-FNAME        PIC X(10).
00129408  000018  C2D6C240 40404040 4040               > BOB              <
000057          10  TESTDATA-DATA-CUST-ADDRESS.
000058              15  TESTDATA-DATA-CUST-STREET       PIC X(25).
00129412  000022  F1F2F340 C1D5E840 E2E3D9C5 C5E34040  > 123 ANY STREET   <
00129412  000022  40404040 40404040 40                 >                  <
000059              15  TESTDATA-DATA-CUST-CITY         PIC X(10).
0012942B  00003B  D7C8D6C5 D5C9E740 4040               > PHOENIX          <
000060              15  TESTDATA-DATA-CUST-STATE        PIC X(02).
00129435  000045  C1E9                                 > AZ               <
000061              15  TESTDATA-DATA-CUST-ZIP          PIC 9(05).
00129437  000047  F8F5F0F1 F8                          > 85018            <
000062              15  TESTDATA-DATA-CUST-ZIP-4        PIC 9(04).
0012943C  00004C  F1F2F3F4                             > 1234             <


+-------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK                PROGRAM: TESTCOBL  DATE: 29MAY2002 |
|  STMT: 000125  OFF: 0004A8  AMODE: 31     MODULE: TESTCOBL  TIME: 10:27:46  |
|SOURCE: PERFORM 2000-PROCESS-TESTDATA                                       |
|R0-7 8B39DE0E 0B474F40 0B39D242 000DA7FC 00129120 8B39D424 00000000 0B474F40 |
|R8-F 00130EE0 001293C0 0B39CE28 0B39D140 0B39CE20 0B4C9078 8012C2BE 00BF8830 |
+-------------------------------------------------------------------------
```

Figure 137 is an example of a PL/I Basic Memory Display.

**Figure 137 • PL/I Basic Memory Display**

```
                                Memory Display              TESTPL1.TESTPL11 -A
Command ===> ____________________________________________ Scroll ===> CSR

Area 0006BA35                              Offset 000000      Length 004000
Member          Data Name
0006BA35  000000  F0F0F0F0 F0F1F0F0 C1E2D4C9 E3C84040  * 00000100ASMITH   *
0006BA45  000010  40404040 40404040 C2D6C240 40404040  *         BOB      *
0006BA55  000020  4040F1F2 F340C1D5 E840E2E3 D9C5C5E3  *   123 ANY STREET *
0006BA65  000030  40404040 40404040 404040D7 C8D6C5D5  *            PHOEN *
0006BA75  000040  C9E74040 40C1E9F8 F5F0F1F8 F1F2F3F4  * IX   AZ850181234 *
0006BA85  000050  00000000 00000000 00000000 00000000  * ................ *
0006BA95  000060  05CFE013 36000000 00021000 00000000  * .õ\............. *
0006BAA5  000070  00000000 05CF0000 06B8A000 06BAA800  * .....õ...½µ..[y. *
0006BAB5  000080  00000000 05CFE000 05CCC000 06B95800  * .....õ\..ö{..¾ì. *
0006BAC5  000090  00000000 00000080 00000080 25000000  * .......Ø...Ø.... *
0006BAD5  0000A0  06B8A000 0000005E 04534000 05D0A800  * .½µ....;.ë ..}y. *
0006BAE5  0000B0  06BDD800 06BDC05E 05CAC400 05CC8800  * .¨Q..¨{;.D..öh. *
0006BAF5  0000C0  05CFE000 06B8A000 06BBA800 06BD3700  * .õ\..½µ..]y..¨.. *
0006BB05  0000D0  05CFE000 06BC2D00 06BA3500 06BBA000  * .õ\..¯...[...]µ. *
0006BB15  0000E0  00000000 06B62000 06BDD800 06BDD891  * .....¶...¨Q..¨Qj *
0006BB25  0000F0  E091E000 06B8A000 00000000 00000000  * \j\..½µ......... *
0006BB35  000100  00000000 00000000 00000000 00000000  * ................ *
0006BB45  000110  00020088 00000000 06BAD000 0000004E  * ...h.....[}....+ *
0006BB55  000120  0332EE00 41588800 41588800 06BAD000  * ..Ó. ìh. ìh..[}. *
0006BB65  000130  06BAB84E 03308000 05D04000 00000000  * .[½+..Ø..} ..... *
0006BB75  000140  05D4BC00 00000100 06BBC000 05D06000  * .M¯.....]{..}-. *
0006BB85  000150  06BBDC00 06BD3700 00008500 06BCB200  * .]ü..¨....e..¯¥. *
0006BB95  000160  00008500 06BC2D00 00008500 06BBA800  * ..e..¯....e..]y. *
0006BBA5  000170  00008540 7E7E7E7E 7E7E407E 7E407E40  * ..e ====== == =  *
0006BBB5  000180  7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E40  * ===============  *
0006BBC5  000190  7E7E7E7E 7E7E7E7E 7E7E407E 7E7E7E7E  * ========== ===== *
0006BBD5  0001A0  7E7E7E7E 7E7E7E7E 7E7E7E7E 7E7E7E7E  * ================ *
0006BBE5  0001B0  7E7E7E7E 407E7E7E 7E7E7E7E 7E7E7E40  * ==== ==========  *
0006BBF5  0001C0  7E7E407E 7E7E7E7E 407E7E7E 7E404040  * == ===== ====    *
0006BC05  0001D0  40404040 40404040 40404040 40404040  *                  *
0006BC15  0001E0  40404040 40404040 40404040 40404040  *                  *
+-------------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK                PROGRAM: TESTPL11  DATE: 29MAY2002 |
|  STMT: 000034  OFF: 00012E  AMODE: 24     MODULE: TESTPL1   TIME: 15:43:10  |
|SOURCE: CALL PRINT_DETAIL;                                                   |
|R0-7 0006BAD0 0006BAB8 4E05C8DC 0005CC88 0005CFE0 0006B8A0 0005C7BC 0006B958 |
|R8-F 0005CFE0 00000000 0006BA35 0006B2A0 0006B010 0006B8A0 0E05C93E 0005D0A8 |
+-----------------------------------------------------------------------------
```

Figure 138 is an example of a PL/I List Memory Map.

**Figure 138 • PL/I List Memory Map**

```
                                Memory Display              TESTPL1.TESTPL11 -A
Command ===> ____________________________________________ Scroll ===> CSR

Area 0006BA35                            Offset 000000      Length 004000
Member          Data Name TESTDATA_RECORD
000007   DCL 1 TESTDATA_RECORD,
000008         2  TESTDATA_KEY,
000009            3  TESTDATA_KEY_CUST_NO                     CHAR(6),
0006BA35  000000  F0F0F0F0 F0F1                        > 000001          <
000010            3  TESTDATA_KEY_CUST_AREA                 CHAR(2),
0006BA3B  000006  F0F0                                 > 00              <
000011         2  TESTDATA_DATA,
000012            3  TESTDATA_DATA_CUST_STATUS              CHAR(01),
0006BA3D  000008  C1                                   > A               <
000013            3  TESTDATA_DATA_CUST_NAME,
000014               4  TESTDATA_DATA_CUST_LNAME            CHAR(15),
0006BA3E  000009  E2D4C9E3 C8404040 40404040 404040    > SMITH           <
000015               4  TESTDATA_DATA_CUST_FNAME            CHAR(10),
0006BA4D  000018  C2D6C240 40404040 4040               > BOB             <
000016            3  TESTDATA_DATA_CUST_ADDRESS,
000017               4  TESTDATA_DATA_CUST_STREET           CHAR(25),
0006BA57  000022  F1F2F340 C1D5E840 E2E3D9C5 C5E34040  > 123 ANY STREET   <
0006BA57  000022  40404040 40404040 40                 >                 <
000018               4  TESTDATA_DATA_CUST_CITY             CHAR(10),
0006BA70  00003B  D7C8D6C5 D5C9E740 4040               > PHOENIX         <
000019               4  TESTDATA_DATA_CUST_STATE            CHAR(02),
0006BA7A  000045  C1E9                                 > AZ              <
000020               4  TESTDATA_DATA_CUST_ZIP              CHAR(5),
0006BA7C  000047  F8F5F0F1 F8                          > 85018           <
000021               4  TESTDATA_DATA_CUST_ZIP_4            CHAR(4);
0006BA81  00004C  F1F2F3F4                             > 1234            <

+-----------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK                PROGRAM: TESTPL11  DATE: 29MAY2002 |
|  STMT: 000034  OFF: 00012E  AMODE: 24      MODULE: TESTPL1   TIME: 15:43:10  |
|SOURCE: CALL PRINT_DETAIL;                                                    |
|R0-7 0006BAD0 0006BAB8 4E05C8DC 0005CC88 0005CFE0 0006B8A0 0005C7BC 0006B958 |
|R8-F 0005CFE0 00000000 0006BA35 0006B2A0 0006B010 0006B8A0 0E05C93E 0005D0A8 |
+-----------------------------------------------------------------------------
```

Figure 139 is an example of an Assembler Basic Memory Display

**Figure 139 • Assembler Basic Memory Display**

```
          Memory Display               TESTASM.TESTASM -A
Command ===> ____________________________________________ Scroll ===> CSR

Area 000C4984                            Offset 000000      Length 004000
Member          Data Name
000C4984  000000  F0F0F0F0 F0F1F0F0 C1E2D4C9 E3C84040  * 00000100ASMITH   *
000C4994  000010  40404040 40404040 C2D6C240 40404040  *         BOB      *
000C49A4  000020  4040F1F2 F340C1D5 E840E2E3 D9C5C5E3  *   123 ANY STREET *
000C49B4  000030  40404040 40404040 404040D7 C8D6C5D5  *            PHOEN *
000C49C4  000040  C9E74040 40C1E9F8 F5F0F1F8 F1F2F3F4  * IX   AZ850181234 *
000C49D4  000050  00000000 00000000 00000000 00000000  * ................ *
000C49E4  000060  00000000 00000000 00000000 00000000  * ................ *
000C49F4  000070  00000000 00000000 00000000 00000000  * ................ *
000C4A04  000080  00000000 00000000 00000000 00000000  * ................ *
000C4A14  000090  00000000 00000000 00000000 00000000  * ................ *
000C4A24  0000A0  00000000 00000000 00000000 00000000  * ................ *
000C4A34  0000B0  00000000 00000000 00000000 00000000  * ................ *
000C4A44  0000C0  00000000 00000000 00000000 00000000  * ................ *
000C4A54  0000D0  00000000 00F140C1 E2C760E3 C5E2E3C1  * .....1 ASG-TESTA *
000C4A64  0000E0  E2D44040 C5E7C1D4 D7D3C540 D9C5D7D6  * SM  EXAMPLE REPO *
000C4A74  0000F0  D9E34040 40404040 40404040 40404040  * RT               *
000C4A84  000100  40404040 40404040 40404040 40404040  *                   *
000C4A94  000110  40404040 40404040 40404040 40404040  *                   *
000C4AA4  000120  40404040 40404040 40404040 40404040  *                   *
000C4AB4  000130  40404040 40404040 40404040 40404040  *                   *
000C4AC4  000140  40404040 40404040 40404040 40404040  *                   *
000C4AD4  000150  40404040 40404040 40404040 40404040  *                   *
000C4AE4  000160  40404040 40404040 40404040 40404040  *                   *
000C4AF4  000170  40404040 40404040 40404040 40404040  *                   *
000C4B04  000180  40404040 40404040 40404040 40404040  *                   *
000C4B14  000190  40404040 40404040 40404040 40404040  *                   *
000C4B24  0001A0  40404040 40404040 40404040 40404040  *                   *
000C4B34  0001B0  40404040 40404040 40404040 40404040  *                   *
000C4B44  0001C0  40404040 40404040 40404040 40404040  *                   *
000C4B54  0001D0  40404040 40404040 40404040 40404040  *                   *
000C4B64  0001E0  4040C9C4 404040C1 D940C140 D3C1E2E3  *   ID   AR A LAST *
+-------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK              PROGRAM: TESTASM   DATE: 29MAY2002 |
|  STMT: 000081  OFF: 00003E  AMODE: 31     MODULE: TESTASM   TIME: 15:48:55  |
|SOURCE: MVC   DTL_CUST_NO,TESTDATA_CUST_NO                                   |
|R0-7 000C4984 000C4984 00000000 00000000 00000000 00000000 00000000 00000000 |
|R8-F 00000000 00000000 000C4984 00000000 800C47C6 000C4CF8 800C47FA 00BF8830 |
+-------------------------------------------------------------------------
```

Figure 140 is an example of an Assembler List Memory Display.

**Figure 140 • Assembler List Memory Display**

```
                Memory Display                TESTASM.TESTASM -A
Command ===> ____________________________________________ Scroll ===> CSR

Area 000C4984                              Offset 000000      Length 004000
Member          Data Name TESTDATA
000300 TESTDATA_CUST_NO      DS  CL6
000C4984  000000  F0F0F0F0 F0F1                        > 000001           <
000301 TESTDATA_CUST_AREA    DS  CL2
000C498A  000006  F0F0                                 > 00               <
000302 TESTDATA_CUST_STATUS  DS  CL1
000C498C  000008  C1                                   > A                <
000303 TESTDATA_CUST_LNAME   DS  CL15
000C498D  000009  E2D4C9E3 C8404040 40404040 404040    > SMITH            <
000304 TESTDATA_CUST_FNAME   DS  CL10
000C499C  000018  C2D6C240 40404040 4040               > BOB              <
000305 TESTDATA_CUST_STREET  DS  CL25
000C49A6  000022  F1F2F340 C1D5E840 E2E3D9C5 C5E34040  > 123 ANY STREET   <
000C49A6  000022  40404040 40404040 40                 >                  <
000306 TESTDATA_CUST_CITY    DS  CL10
000C49BF  00003B  D7C8D6C5 D5C9E740 4040               > PHOENIX          <
000307 TESTDATA_CUST_STATE   DS  CL2
000C49C9  000045  C1E9                                 > AZ               <
000308 TESTDATA_CUST_ZIP     DS  CL5
000C49CB  000047  F8F5F0F1 F8                          > 85018            <
000309 TESTDATA_CUST_ZIP4    DS  CL4
000C49D0  00004C  F1F2F3F4                             > 1234             <


+-------------------------------------------------------------------------+
|STATUS: STOPPED AFTER BREAK              PROGRAM: TESTASM   DATE: 29MAY2002 |
|  STMT: 000081  OFF: 00003E  AMODE: 31     MODULE: TESTASM   TIME: 15:48:55  |
|SOURCE: MVC   DTL_CUST_NO,TESTDATA_CUST_NO                                  |
|R0-7 000C4984 000C4984 00000000 00000000 00000000 00000000 00000000 00000000 |
|R8-F 00000000 00000000 000C4984 00000000 800C47C6 000C4CF8 800C47FA 00BF8830 |
+-------------------------------------------------------------------------+
```

# LOCATE Command

```
Locate ──┬─PROCedure──────────────────────────────────────────────────►◄
         ├─WHEN──────┤
         ├─line──────┤
         ├─hex offset┤
         ├─.label────┤
         ├─*─────────┤
         ├─string────┤
         └─&CURRent──┘
```

## Function

On the Program View screen, the LOCATE command is used to display the PROCEDURE DIVISION statement, the beginning of the WHEN command statements, a specific line or label, or a hexadecimal offset within the program. From any SmartTest directory or list screen, the LOCATE command can be entered with a string to display an item that matches the specified string. During a test session, the LOCATE command can be entered with the asterisk (*) operand to display the statement that executes next.

## Operands

| Operand | Description |
|---|---|
| PROCedure | Displays the line that contains the PROCEDURE DIVISION label. |
| WHEN | Displays the WHEN statements that are produced by using the WHEN command. WHEN statements are automatically placed after the last PROCEDURE DIVISION statement. |
| *line* | Displays a particular source statement. Optionally, a pseudo code line can be located by entering the line number that precedes the pseudo code line, a period, then the pseudo code line number. For example:<br>2367.2<br>would locate the second pseudo code line that follows line number 2367. The specified line number must be the line that immediately precedes the pseudo code lines. |
| hex offset | Displays the statement that corresponds to the specified hexadecimal offset. An offset can be specified as follows:<br>`L X'offset'`<br>where `X` is required to indicate the value is hexadecimal, and offset is an offset value such as 3C5E. |
| *.label* | Locates a line label entered in the line command area (columns 1 through 6) on the Program View screen. |

| Operand | Description |
|---|---|
| asterisk (*) | Displays the line at which the program is stopped. This is the next statement to be executed. |
| *`string`* | Locates a particular item on a list or directory type screen. |
| &CURRent | Positions the Program View screen to the last location marked by the CURRENT command. If no CURRENT command has preceded the LOCATE &CURRENT command, the message `&CURRENT NOT AVAIL` displays and the screen is not repositioned.<br>The &CURRENT operand is only valid on the Program View screen. |

## Usage Notes

A character string can be specified when an exact program name or list item is not known. The LOCATE command displays the item that most closely matches the specified character string. Matching is done alphabetically. Note that a character string is only valid on directory or list screens.

The STRING operand of the LOCATE command can only be used when a list such as a directory list displays.

The asterisk (*) operand of the LOCATE command can only be used on the Program View screen.

You can also locate specified lines by selecting Search ▸ Line.

## Examples

To display statement number 4697, type this command:

```
L 4697
```

To display the line that contains the PROCEDURE DIVISION label. In COBOL II and later versions this is the PROCEDURE DIVISION label of the main program, type this command:

```
L PROC
```

To display the line containing an item that matches or follows S alphabetically, type this command:

```
LOCATE S
```

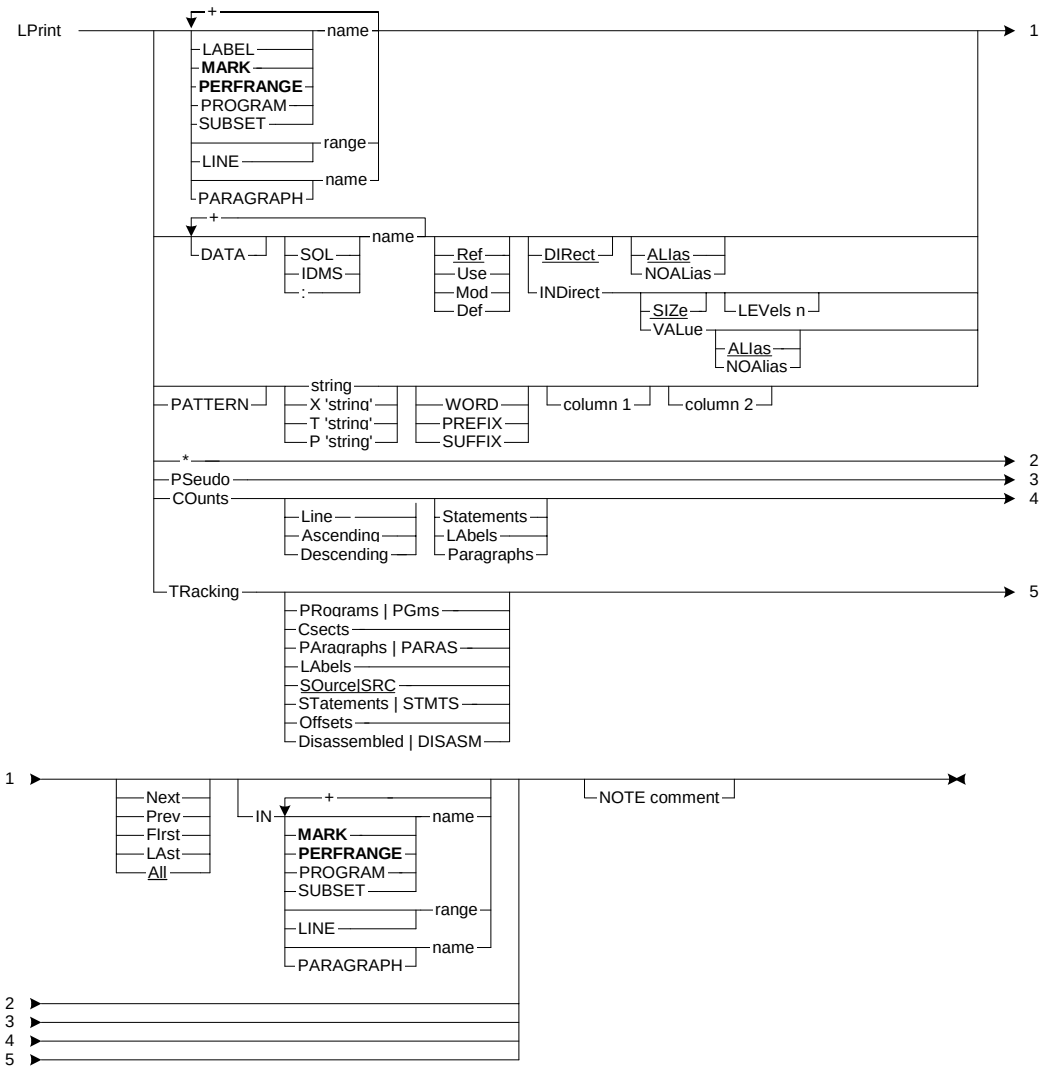
To display the line at which program execution has stopped. type this command. Program execution stops as the result of a Breakpoint or an error condition. For example, if a MOVE statement is executed during a test session, and the data to be moved is not in the same format as the receiving field, an error condition occurs (S0C7). The LOCATE * command can be used to display the line on which the interrupt occurred.

```
LOCATE *
```

To position the display to the location marked by the most recent CURRENT command, type this command:

```
LOCATE &CURRENT
```

# LPRINT Command




Bold operands are available only with ASG-Insight

## Function

The LPRINT command is used to copy lines containing the requested target to the List file. The List file is processed on from the Options - Log/List/Punch Definition pop-up.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set of lines or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide*. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement, including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. This operand is only available if Insight is installed and an Insight analysis has been run on the program. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the EXCLUDE command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |
| SQL *name* | Includes datanames that are DB2/SQL variables only. |

| Operand | Description |
|---|---|
| IDMS *name* | Includes datanames that are IDMS variables only. |
| : *name* | Includes datanames that are COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Def | Includes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |

| Operand | Description |
|---|---|
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. |
| *string* | Specifies a string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*string*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |

| Operand | Description |
|---|---|
| WORD | Specifies the specified pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| *column*1 | Specifies the column number where the search is to begin. |
| *column*2 | Specifies the column number where the search is to end. |
| asterisk (*) | Copies the entire virtual screen (all data that can be viewed by scrolling up and down) to the List file. All excluded lines are copied to the List file as excluded lines (as they appear on the screen at the time the LPRINT * command is entered). |
| PSeudo | Copies all pseudo code within the program currently being tested to the List file. |
| COunts | Copies the COBOL counts from the program currently being tested to the List file. COBOL counts must be activated for the program currently being tested. |
| Line | Displays the current execution counts, sorting source by line. This is the default if COUNTS ASCENDING or COUNTS DESCENDING has not been specified during the current test session. |
| Ascending | Displays the current execution counts, sorting source by ascending counts. Once specified this becomes the default until another COUNTS operand is specified. |
| Descending | Displays the current execution counts, sorting source by descending counts. Once specified this becomes the default until another COUNTS operand is specified. |
| Statements | Displays the current execution counts by source statement, sorting statements according to the sort option selected. This is the default if COUNTS PARAGRAPHS or COUNTS LABELS has not been specified during the current test session. |
| LAbels | Displays the current execution counts by LABEL point, sorting labels according to the sort option specified. This operand is intended for use by Assembler programmers and is synonymous with the PARAGRAPHS operand. |

| Operand | Description |
|---|---|
| Paragraphs | Displays the current execution counts by PARAGRAPH name, sorting labels according to the sort option specified. This operand is synonymous with the LABELS operand. |
| TRacking | Copies tracking information into the Execution Tracking list file. |
| PRograms \| PGMS | Prints the Execution Tracking screen listing programs that were last executed for the program being tested. |
| Csects | Prints the Execution Tracking screen listing CSECTs that were last executed for the program being tested. |
| PAragraphs \| PAras | Prints the Execution Tracking screen listing programs, CSECTs, and paragraphs that were last executed for the program being tested. |
| SOurce \| SRC | Prints the Execution Tracking screen listing programs, CSECTs, paragraphs, and source statements that were last executed for the program being tested. This is the default LIST TRACKING operand. |
| STatements \| STmts | Prints the Execution Tracking screen listing programs, CSECTs, paragraphs, and line numbers of statements that were last executed for the program being tested. |
| Offsets | Prints the Execution Tracking screen listing programs, CSECTs, paragraphs, and offsets that were last executed for the program being tested. |
| Disassembled \| Disasm | Prints the Execution Tracking screen listing programs, CSECTs, paragraphs, and disassembled source statements that were last executed for the program being tested. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Searches for all occurrences of the requested target and displays the number found in the short message field. This is the default value for the LPRINT command. |

| Operand | Description |
|---|---|
| IN | Restricts the LPRINT command to the specified target type. |
| NOTE comment | Places descriptive comment lines in the List file. These comments are included in the printed output from the List file. Comment text can be a maximum of 50 alphanumeric characters. |

## Usage Notes

LPRINT copies the specified target to the List file for subsequent printing.

You can enter LPRINT * on these List screens to copy them to the List file:

- BackTrack Variable History screen
- Breakpoints List screen
- Counts List screen
- Execution Counts screen
- Execution Tracking screen
- List - CALL Statements pop-up
- List - COBOL Subsets Names pop-up
- List - Equates pop-up
- List - Perform Range Names pop-up
- List - Program/Subprogram Names pop-up
- List - User Marks pop-up
- Pseudo Code List screen
- When Conditions List screen

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

You can also copy selected lines to the List file by selecting a search pop-up on the Search pull-down.

## Examples

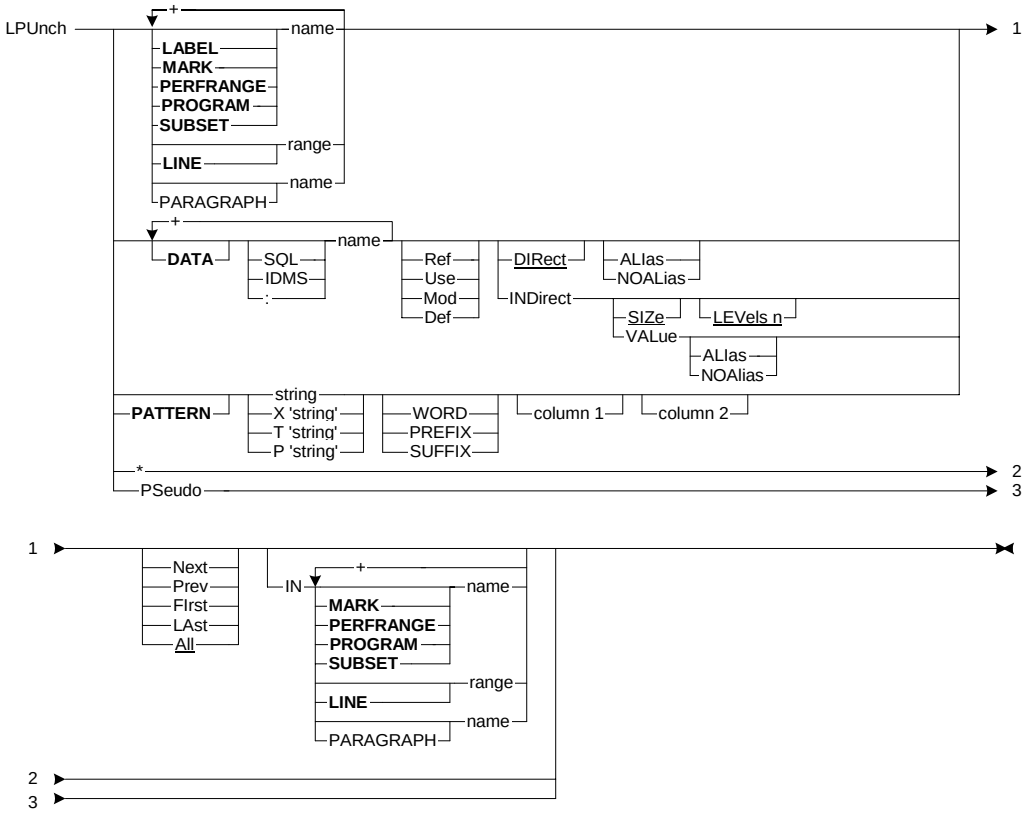
To copy all lines containing the characters ZIP to the List file for subsequent printing, type this command:

```
LPRINT ZIP
```

To copy the entire virtual screen (including any pseudo code, zoomed data, disassembled object code, etc.) to the List file, when Program View screen displays, type this command:

```
LPRINT *
```

# LPUNCH Command




Bold operands are available only with ASG-Insight

## Function

The LPUNCH command is used to copy lines containing the specified target to the Punch file for subsequent processing. The Punch file can be de-allocated from the Options - Log/List/Punch Definition pop-up. If Insight is not installed, the operands MARK, PERFRANGE, PROGRAM, SUBSET, LINE, LABEL, DATA, and PATTERN are not available.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK\| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK, and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement or the name of any section contained in the Declaratives. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *AASG-SmartTest PLI User's Guide* for a description of each subset.<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |

| Operand | Description |
|---|---|
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the LPUNCH command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |
| SQL *name* | Includes datanames that are DB2/SQL variables only. |
| IDMS *name* | Includes datanames that are IDMS variables only. |
| : *name* | Includes datanames that are COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Def | Includes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT. If the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |

| Operand | Description |
|---|---|
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates that the characters that follow are part of a string. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| *string* | Specifies a string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*string*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |

| Operand | Description |
|---|---|
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Specifies the specified pattern string preceded and followed by any non-alphanumeric character (except hyphen). |
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| *column*1 | Specifies the column number where the search is to begin. |
| *column*2 | Specifies the column number where the search is to end. |
| asterisk (*) | Copies the entire virtual screen (all data that can be viewed by scrolling forward and backward) to the Punch file. All excluded lines are copied to the Punch file as excluded lines (as they appear on the screen at the time the LPUNCH * command is entered). |
| PSeudo | Copies all pseudo code within the program currently being tested to the Punch file. |
| Next | Searches forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Searches backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Searches from the top of the source file to the first occurrence of the requested target. |

| Operand | Description |
|---|---|
| LAst | Searches backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Searches for all occurrences of the requested target and displays the number found in the short message field. This is the default value for the LPUNCH command. |
| IN | Restricts the LPUNCH command to the specified target type. The IN target type PERFRANGE is only available with Extended Analysis. |

## Usage Notes

LPUNCH copies the specified targets to the Punch file for subsequent processing.

Any pseudo code that is incompatible with COBOL syntax rules is automatically changed to a comment line when copied to the Punch file. These pseudo code statements remain intact but contain an asterisk (*) in column seven.

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

You can also copy selected lines to the Punch file by selecting a search option on the Search pull-down.

## Examples

To copy the entire source file to the Punch file, type this command:

```
LPUNCH *
```

To copy all pseudo code in the current program to the Punch file for subsequent processing, type this command:

```
LPUNCH PSEUDO
```

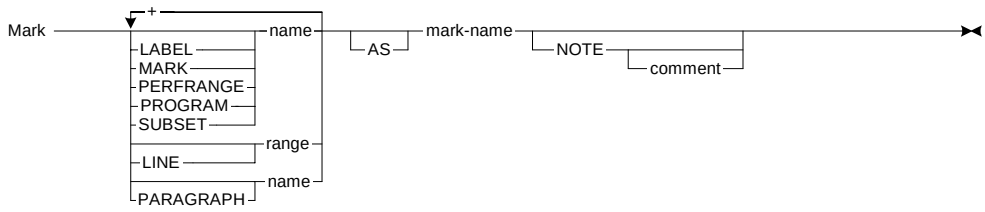
Figure 141 shows the Punch file after the command above is entered.

**Figure 141 • Punch File Example**

```
      *   ASG-SMARTTEST-OS(XA) Rx.x LVLnnn             DDMMMYYYY  HH:MM

      *   LPUNCH PSEUDO CODE FOR PRDEMO2

      *   SET PSEUDO ===> ON   PSEUDO ACTIVE:    2   INACTIVE:   0
      *   SET BREAKS ===> ON   BREAKS ACTIVE:    2   INACTIVE:   0
      *   SET  WHENS ===> ON    WHENS ACTIVE:    0   INACTIVE:   0

      *   START OF PSEUDO CODE

017900                      OUTRPT-WORK-AREA.
018000                      OUTFILE-WORK-AREA.
.....1*    BREAK.

042300 3100-READ-INFILE1.
042400     MOVE ZERO TO READ-INFILE1-SWITCH.
.....1*    BREAK.

043500 3200-READ-INFILE2.
043600     MOVE ZERO TO READ-INFILE2-SWITCH.
.....1*    BREAK.


      *   END OF PSEUDO CODE
```

The same information that displays on the Pseudo Code List screen is copied to the top of the Punch file. The START OF PSEUDO CODE statement indicates the following lines are part of the program for which the LPUNCH PSEUDO command was entered. The two source statements preceding each block of pseudo code are also copied to the Punch file to indicated the context in which the pseudo code was entered. Source statements are shown as they actually appear in the program. Pseudo code statements are shown as entered; however, if they are incompatible with standard COBOL syntax, an asterisk (*) is automatically placed in column seven, making them comment lines.

# MARK Command




ASG-Insight only

## Function

The MARK command is used to save the requested target as a Mark set or path. An optional description can be included if desired.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |

| Operand | Description |
|---|---|
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| AS | Specifies an optional keyword indicating the specified target is to be referenced as the mark-name that follows. |

| Operand | Description |
|---|---|
| *mark-name* | Specifies a name assigned to a path or set of lines using the MARK command or the Options - Scratchpad Mark pop-up. The specified name must meet these requirements:<br>• May be a maximum of 10 alphanumeric characters and can include hyphens.<br>• Names longer than 10 characters are truncated.<br>• Name must begin with an alphabetic character. |
| NOTE *comment* | Includes an optional description about the Mark name. Comment text can be a maximum of 50 alphanumeric characters. If the NOTE comment operand is not entered, the existing comment is copied to the new mark name. If NOTE is entered without comment text, the existing comment is not copied to the new mark name. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

The target of a MARK command can be saved multiple times under different names if desired.

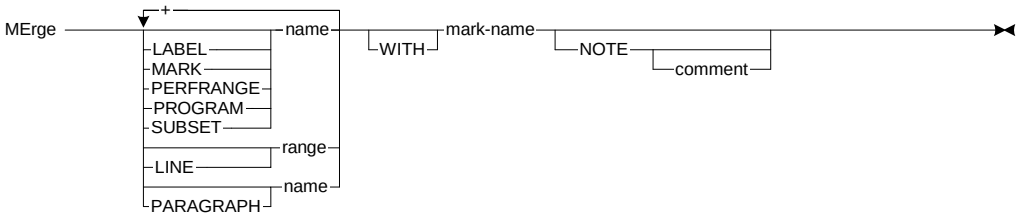
Selecting Options ▸ Scratchpad, then selecting Mark on the Options - Scratchpad pop-up also allows you to save a mark set or path.

## Example

In this command, the system-generated path called TRACK is captured and named FICA (both paths exist):

```
MARK TRACK AS FICA
```

# MERGE Command




ASG-Insight only

## Function

The MERGE command adds the lines from the specified target to the specified Mark name. If the specified Mark name does not exist, it is created. A comment included with the MERGE command replaces an existing one.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |

| Operand | Description |
|---|---|
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| WITH | Specifies an optional keyword indicating the Mark name that follows is to be merged with the specified target. |

| Operand | Description |
|---|---|
| *mark-name* | Specifies a Mark name assigned to the target of the MERGE command. If the Mark name does not exist, it is created. The specified name must meet these requirements:<br>• May be a maximum of 10 alphanumeric characters and can include hyphens<br>• Names longer than 10 characters are truncated<br>• Name must begin with an alphabetic character |
| NOTE *comment* | Includes an optional description about the Mark name. Comment text can be a maximum of 50 alphanumeric characters. If the NOTE comment operand is not entered, the existing comment is copied to the new mark name. If NOTE is entered without comment text, the existing comment is not copied to the new mark name. |

## Usage Notes

A set of lines can be merged with another set of lines, or a path can be merged to a set of lines.

To concatenate targets, place a plus sign (+) between the target names. A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

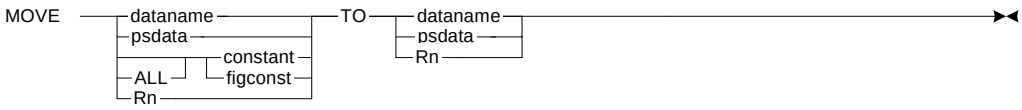
Selecting Options ▸ Scratchpad, then selecting Merge on the Options - Scratchpad pop-up also allows you to add lines to a specified Mark name.

## Example

This command adds the contents of TRACK to the set of lines called FICA.

```
MERGE TRACK TO FICA
```

# MOVE Command




## Function

The MOVE command assigns the value contained in or represented by the first operand to the specified data item. The value is converted to the proper format for the data item if possible. If the value cannot be converted to the proper format, an error message displays.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname or qualified COBOL dataname, Assembler label, or PL/I variable name. Dataname refers to any valid COBOL reference for a data element. |
| *psdata* | Specifies aA level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| ALL | Replaces all characters of the dataname with the constant. Used with a figurative constant for readability. |
| *constant* | Specifies a numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about coding pseudo code. |
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to move specific values. These figurative constants are supported:<br>HIGH-VALUE LOW-VALUESZERO<br>HIGH-VALUES SPACE ZEROS<br>LOW-VALUE SPACES ZEROES |
| R*n* | Specifies a register 0 through 15. |
| TO | Specifies a required keyword used as a connective for the MOVE command operands.<br>**Note:**<br>You can also move a value to a data item by selecting Test ▸ Move. |

# NEWCOPY Command

```
NEWcopy ──module-name──────────────────────────────────────────────────►◄
```

## Function

The NEWCOPY command loads a new copy of the PPT module specified into the CICS environment.

**Note:**

This command is available in the CICS environment only.

## Operands

**module-name.** The PPT module name to be loaded.

## Usage Notes

The new copy of the module must meet these normal CICS requirements:

- The module must have a valid PPT module name.
- The module must not be currently in use.
- The module must not have been disabled by CICS.
- Generic names are not supported.

You can also load a new copy of a specified PPT module into the CICS environment by selecting Test ▸ CICS Newcopy.

# PARMDEF Command

PARMDEF | PDEF

## Function

The PARMDEF command is used to display the Options - Product Parameters pop-up. The Options - Product Parameters pop-up is used to set parameters that affect the online operation of SmartTest.

## Operands

None.

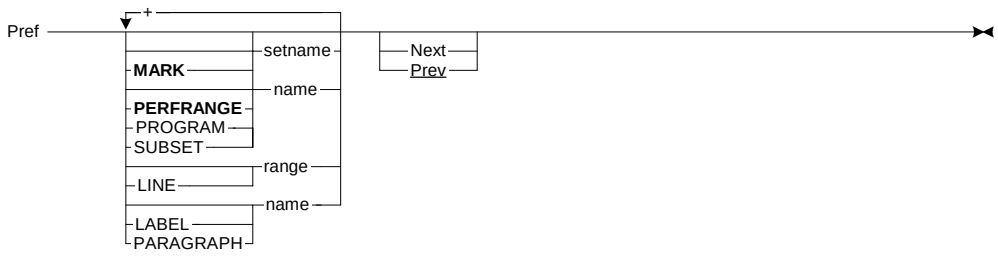
## Usage Notes

The PARMDEF command displays the Options - Product Parameters pop-up.

Use the END command to return to the previous screen from where the PARMDEF command was issued.

You can also display the Options - Product Parameters pop-up by selecting Options ▸ Product parameters.

# PREF Command




Bold operands are available only with ASG-Insight

## Function

The PREF command is used to display the View - Paragraph Cross Reference pop-up for the requested target. The View - Paragraph Cross Reference pop-up shows how control is transferred to or from the target paragraphs.

## Operands

| Operand | Description |
|---|---|
| blank | If no operand is entered, the target paragraph set is determined by the cursor location when the PREF command is issued. |
| MARK *setname* | Specifies a name (1 to 10 alphanumeric characters) given to a set using the COPY, MARK, MERGE, or RENAME commands. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.The TRACK, NETWORK, and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the Logic chapter in the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Indicates the first statement within the specified PERFORM range. This operand is only available if Insight is installed and an Insight analysis has been run on the program. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program including all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| PARAGRAPH *name* | Specifies a PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included. |
| Next | Generates a list of paragraphs to which the target paragraph transfers control. |
| Prev | Generates a list of paragraphs that transfer control to the target paragraph. This is the default for the PREF command. |

## Usage Notes

The PREF command includes all paragraphs containing the target lines. For example:

```
PREF IO
```

displays every paragraph containing IO statements.

To concatenate targets, place a plus sign (+) between the target names. A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

When there is a blank target with the cursor located on excluded lines, these items apply:

- The first line of the excluded block is implicitly selected.
- If the selected line is not a label, then a backward search is done to locate the previous label. This label then becomes the target.

You can also display the View - Paragraph Cross Reference pop-up by selecting View ▸ Paragraph X-Ref.

## Example

This command displays the View - Paragraph Cross Reference pop-up with all paragraphs that transfer control to the WRITE-ROUTINE paragraph.

```
PREF WRITE-ROUTINE
```

# PRINTLOG Command

PRINTLOG | PLOg

## Function

The PRINTLOG command is used to display the Options - Log/List/Punch Definition pop-up. On this pop-up the Log, List and Punch files can be printed.

## Operands

None.

## Usage Notes

The PRINTLOG command can be issued on any SmartTest screen.

You can also display the Options - Log/List/Punch Definition pop-up by selecting Options ▸ Log/List/Punch.

# PRINTLST Command

PRINTLST | PLIst

## Function

The PRINTLST command is used to display the Options - Log/List/Punch Definition pop-up. On this screen the Log, List, and Punch files can be printed.

## Operands

None.

## Usage Notes

The PRINTLST command can be issued on any SmartTest screen.

You can also display the Options - Log/List/Punch Definition pop-up by selecting Options ▸ Log/List/Punch.

# PROCESS Command

```
PROCess ──┬─────────┬──┬─(member)──────┬──model command──────────────────────►◄
          └─STEP────┘  ├─dsn(member)───┤
                       ├─'dsn'─────────┤
                       └─'dsn(member)'─┘
```

## Function

The PROCESS command defines an action to be performed repetitively on a file of input items.

This file may contain the results list from an ESW product, or the file may be nested by the user. The input items, which are called tokens, may be any text without embedded blanks or any text enclosed in quotes. Tokens must be separated by one or more spaces.

The PROCESS primary command processes the first nine tokens in a file. Any tokens beyond nine are ignored. Any tokens that are referenced in the PROCESS command but not present in the token file are passed as null.

Tokens enclosed in quotes retain their quotes after substitution in the model statement.

This example shows a possible format for a record in the token file:

```
token1 'token 2' token3 token4 token5 'token 6' token7 token8 token9
```

Insight, Encore, SmartTest, and Alliance can produce an output file using LPRINT, LPUNCH, or the Query Facility. This file can be edited to produce a file of tokens that can be read by the PROCESS primary command.

## Operands

| Operand | Description |
|---|---|
| STEP | Steps through the file. Each command in the file displays in the command input area. The displayed command can be changed or erased from the command input area, or you can press Enter to execute the displayed command. Processing continues in this manner until all of the commands have been displayed. |
| (*member*) | Specifies a member in the default script library (defined during product installation) that contains tokens to be processed. You can modify the default script libraries for your profile by selecting Options ▸ Script file allocation. Parentheses are required if only the member is entered. |

| Operand | Description |
|---|---|
| *dsn*(*member*) | Specifies a partitioned dataset member that contains tokens to be processed. This dataset must be in card image format (LRECL=80). The specified dataset name and member must be entered using TSO usage conventions in this format:<br>`'pds.dataset.name(member)'`<br>Quotes are required unless the high-level qualifier is your TSO user ID. |
| *dsn* | Specifies a sequential dataset that contains tokens to be processed. This dataset must be in card image format (LRECL=80). The specified dataset must be entered using TSO usage conventions in this format:<br>`'sequential.dataset.name'`<br>Quotes are required unless the high-level qualifier is your TSO prefix or user ID. |
| *model command* | Specifies a model command. The model command may be any ESW product command that is valid in a Script file and that can be defined using substitution variables. Tokens from the file are always substituted in the model command in numerical order from 1 through 9 for the substitution variables &1 through &9.<br>When the PROCESS command is included in a script, the model command may optionally reference substitution variables that were passed to the script. This is done by specifying any of the override substitution variables &&1 through &&9. |

## Usage Notes

The PROCESS command allows automated processing of the results from any ESW product that can produce a text file of tokens.

For example, if Alliance creates a file of records named *userid*.PROGDATA.TEXT in which each record contained a program name and a dataname, the file can be processed with the command:

```
PROCESS PROGDATA.TEXT EXECUTE SCANPROG.SCRIPT PARMS &1 &2
```

This command reads each record of the file PROGDATA.TEXT in turn and substitutes program name from the record into substitution variable &1 and dataname from the record into substitution variable &2. It then executes the named script (*userid*.SCANPROG.SCRIPT), passing the substituted variables &1 and &2.

The script file SCANPROG.SCRIPT might contain these commands to qualify a program for viewing and execute another script file for the program.

```
Q &1
EXECUTE PROGPROC.SCRIPT PARMS &2
```

This allows the processing of multiple programs with a single command sequence.

To accomplish this task, the PROCESS command verifies and opens the specified file. The logic is then repeated until an end-of-file occurs for the specified file.

### *To process multiple programs with a single command sequence*

1. Read a record. If the record is null or contains an asterisk (*) in column 1, ignore the record and begin again.
2. Substitute tokens from the record into the substitution variables &1 through &9.
3. Construct a command using the model statement with the variable substitution.
4. Execute the constructed command as if it were a script command.
5. Begin again at step 1 and continue until an end-of-file.

PROCESS commands may occur within scripts and may be nested. If PROCESS commands are nested, they must refer to different datasets.

## Examples

These examples show how you might use the PROCESS command in its most basic form to process a dataname list and then how this basic concept could be extended to automate the same action for a list of programs.

To cause the HIGHLIGHT command (HI) to be executed repetitively for each data item in the file *`userid`*.DATANAME.LIST, type this command from the Program View screen:

```
PROCESS DATANAME.LIST HI &1
```

Given a file of program names called *userid*.PROGRAM.LIST, you could execute this PROCESS command:

```
PROCESS PROGRAM.LIST EXEC MYSCRIPT.SCRIPT PARMS &1
```

Assume that *userid*.MYSCRIPT.SCRIPT contains these commands:

```
* STEP 1: OPEN THE PROGRAM
QUALIFY &1
* STEP 2: POSITION TO THE PROGRAM VIEW SCREEN (ASG-SmartTest
only)
FX 1
* STEP 3: PROCESS THE DATA NAME LIST
PROCESS DATANAME.LIST HI &1
* STEP 4: SAVE THE RESULTS OF THE PROCESS COMMAND
SAVE MARKS AS &1MK
* STEP 5: CLOSE THE PROGRAM
QUALIFY CANCEL
```

The set of commands in *userid*.MYSCRIPT.SCRIPT executes one time for each program in *userid*.PROGRAM.LIST. *userid*.MYSCRIPT.SCRIPT contains a PROCESS command that executes the model command, HIGHLIGHT, one time for each record in *userid*.DATANAME.LIST.

The result is a MARK set, named *pgmname*MK, for each program in *userid*.PROGRAM.LIST. Each MARK set contains highlighted references to uses in the program of all datanames in *userid*.DATANAME.LIST.

To pass override substitution variables *&&1* and *&&2* to the PROCESS command contained in the script, type this command:

```
EXEC (MYSCRIPT) PARMS ARG1 ARG2 ARG3
```

Assume the script in member MYSCRIPT in the default script libraries contains this PROCESS command:

```
PROCESS 'PROGRAM.LIST' EXEC (PGSCRIPT) PARMS &&2 &1 &2 &&1
```

The parameters from the initial EXECUTE statement and the tokens from the file are substituted in the PROCESS statement model command as follows:

```
&&2  is passed the value ARG2
&1   is passed the first token of each record read from the file 'PROGRAM.LIST'
&2   is passed the second token of each record read from the file 'PROGRAM.LIST'
&&1  is passed the value ARG1
```

# PRODLVL Command

```
PRODLVL ──────────────────────────────────────────────────────────────►◄
```

## Function

The PRODLVL command is used to display the current SmartTest and Center product level.

## Operands

None.

## Usage Notes

The PRODLVL command displays the product name, operating system, product release number, and release level on the message line, in this format for the CICS environment:

```
ASG1554I ASG-SMARTTEST-CICS-OS(XA) Rn.n AT Lnnn, ASG-CENTER Rn.n AT Lnnn
```

or in this format for other environments:

```
ASG1554I ASG-SMARTTEST-OS(XA) Rn.n AT Lnnn, ASG-CENTER Rn.n AT Lnnn
```

where R*n.n* is the release number and L*nnn* is the release level. This information is requested when you contact ASG Customer Support for assistance.

You can also display the Center and SmartTest product levels by selecting Help ▸ About.

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for additional information about SmartTest online help as accessed through the action bar.

# QUALIFY Command

```
Qualify ──┬─pgm──────────┬──────────────────────────────────────────────►◄
          ├─module.pgm───┤
          ├─*────────────┤
          └─CANcel──┬────┤
                    └─ALL┘
```

## Function

The QUALIFY command is used to display a different program on the Program View screen. The program being tested remains active; that is, if a STEP or RUN command is entered, it is executed for the active program (the program being tested), NOT the qualified program. All other SmartTest commands are performed for the qualified program.

The status box changes when a STEP or RUN command is entered to indicate the status of the active program; not the qualified program.

## Operands

| Operand | Description |
|---|---|
| *pgm* | Specifies the program that is to be displayed on the Program View screen if it resides in the current load module. |
| *module.pgm* | Specifies the module.program operand used to access a program that is not in memory. If the desired module resides in the current load library, a program or CSECT within the load module can be specified with the QUALIFY command. The specified module and program or CSECT is then displayed on the Program View screen. |
| asterisk (*) | Returns to (requalify) the active program. |
| CANcel | The CANCEL operand is used to release a qualified program so it becomes available for other processing such as compile and batch analysis. Pseudo code and equates entered into the program while it is qualified are saved/disregarded based on the values specified on the Options - Product Parameters pop-up. |
| ALL | Releases all programs that may have been qualified during the test session when used with the CANCEL command. |

## Usage Notes

The QUALIFY command is useful for viewing applications that have multiple programs.

When a test has been executed after a program has been recompiled and a message indicating that the AKR is in use displays, use the QUALIFY CANCEL ALL command to release all programs.

The CANCEL operand must not be used for the program currently being executed during a test session.

You can also qualify a program for viewing by selecting View ▸ Qualify.

# RECALL Command

```
RECall ──┬─COMmand | CMD─┬──┬─NEXT─┬──────────────────────────►◄
         ├─MESsage | MSG─┤  └─PREV─┤
         └─POPup─────────┴─────────┴
```

## Function

The RECALL command is used to display the previous primary command, message, or pop-up. The last 20 commands that have been executed, and the last 20 messages that have been displayed, are stacked. These commands or messages can be redisplayed using the RECALL command.

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the last primary command that was stacked. Once the RECALL command has been entered with operands, subsequently entering RECALL with no operands reuses the same operands that were last entered. |
| COMmand \| CMD | Displays a stacked primary command. |
| MESsage \| MSG | Displays a stacked message. |
| NEXT | Displays the next command or message in the stack. |
| PREV | Displays the previous command or message in the stack. This is the default value. |
| POPup | Displays the pop-up that was most recently requested from a pull-down. |

## Usage Notes

The RECALL command can be entered repeatedly to display any of the 20 stacked commands or messages. The NEXT and PREV operands can be used to move forward or backward through the stacked commands or messages. Once the desired command displays, it can be executed again by pressing Enter. Any command that is recalled can be changed prior to executing it. To re-execute the last stacked primary command again without modification, the REPEAT command can be used.

The operands specified for the RECALL command remain in effect until one of these conditions occur:

- A different operand is specified.
- A different primary command is executed. When this occurs, the RECALL command default operands are automatically set. A message displays that indicates all stacked commands or messages have been shown and the stack is being redisplayed.

These commands are not stacked for use by the RECALL command:

- ALLOCDEF
- ANALYZE
- CURRENT
- DUMP
- END
- FLOW
- FORCE
- HELP
- KEYS
- PRINTLOG
- PRINTLST
- PROCESS
- PRODLVL
- RECALL
- REDO
- REPEAT
- RETURN
- RFIND
- RHIGH
- RSCROLL
- RTRACE
- TOGGLE
- UPDATE

## Example

This example assumes these commands were entered for the program currently being monitored, and that seven messages were displayed:

**1** X ALL

**2** ZOOMIN

**3** PREF

**4** ZOOMOUT

**5** FI

**6** SCROLL

**7** FI

This command displays the previous message (number 7):

```
RECALL MSG
```

The MSG operand remains in effect until the COMMAND or CMD operand is specified. Therefore, entering RECALL without changing the operands displays the previous message (number 6):

```
RECALL
```

To display the previous command (number 7), type this command:

```
RECALL CMD
```

The CMD operand remains in effect until the MESSAGE or MSG operand is specified. Therefore, typing `RECALL` without changing the operands displays the previous command (number 6).

This command displays the next command (number 7):

```
RECALL NEXT
```

# REDO Command

```
REDo ──┬──────────────┬──────────────────────────────────────►◄
       ├─trace-dec-opt─┤
       └─BACKup────────┘
```

Bold operands are available only with ASG-Insight

## Function

The REDO command executes the corresponding REPEAT command from the cursor position as shown in this table:

| Last Command | Command Executed |
|---|---|
| FIND | RFIND |
| FINDXTND | RFIND |
| HIGH | RHIGH |
| PREF | RPREF |
| SCROLL | RSCROLL |
| TRACE | RTRACE |

Only the commands listed in this table are re-executed when the REDO command is entered.

## Operands

| Operand | Description |
|---|---|
| Blank | Executes the corresponding REPEAT command from the cursor position as shown in the table above.<br>These operands are valid only if the last command was TRACE. These operands are ignored if the last command was not TRACE. |
| *trace-dec-opt* | Specifies a TRACE command decision option. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| BACKup | Returns to the decision point where the last decision option was entered. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |

## Usage Notes

REDO automatically executes the corresponding REPEAT command, depending on the last command entered. For example, if the last command entered was FIND, RFIND is executed.

# REFRESH Command

REFresh

## Function

The REFRESH command is used to bring in fresh versions of program summaries or copy members. This command need only be used when one or more of the programs called by the active program is analyzed while the active program is being viewed.

## Operands

None.

## Usage Notes

The most recent CALL summaries are retrieved from the AKR when you use the REFRESH command.

You can also update CALL summaries by selecting Options ▸ Refresh.

# RENAME Command

```
REName ──mark-name──┬──────┬──mark-name──┬──────┬──────┬──────┬──────────────────────►◄
                    └──TO──┘             └NOTE──┬──────┘
                                                └comment──┘
```

ASG-Insight only

## Function

The RENAME command is used to change the name of a Mark path or set of lines.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| *mark-name* | Specifies a name assigned to a path or set of names using the MARK command or the Options - Scratchpad Mark pop-up. The specified name must meet these requirements:<br>• May be a maximum of 10 alphanumeric characters and can include hyphens.<br>• Names longer than 10 characters are truncated.<br>• Name must begin with an alphabetic character. |
| TO | Specifies an optional keyword indicating the name following is the new Mark name. |
| NOTE comment | Includes an optional description about the Mark name. Comment text can be a maximum of 50 alphanumeric characters. If the NOTE comment operand is not entered, the existing comment is copied to the new mark name. If NOTE is entered without comment text, the existing comment is not copied to the new mark name. |

## Usage Notes

An error message displays if the Mark name to be renamed does not exist or the resulting Mark name already exists.

Selecting Options ▸ Scratchpad, then selecting Rename on the Options - Scratchpad pop-up also allows you to rename a Mark path or set of lines.

## Example

In this command, the Mark name SETA is changed to SETB. SETA is no longer available as a Mark name:

```
RENAME SETA TO SETB
```

# REPEAT Command

```
REPeat ──────────────────────────────────────────────────────────────►◄
```

## Function

Executes the last stacked primary command again.

## Operands

None.

## Usage Notes

Type `REPEAT` in the command input area to re-execute the last saved primary command. The command is executed from the current cursor location.

To modify the command before it is re-executed, type `RECALL`. The & (Retain) command may be used to repeat and modify a specified command.

## Example

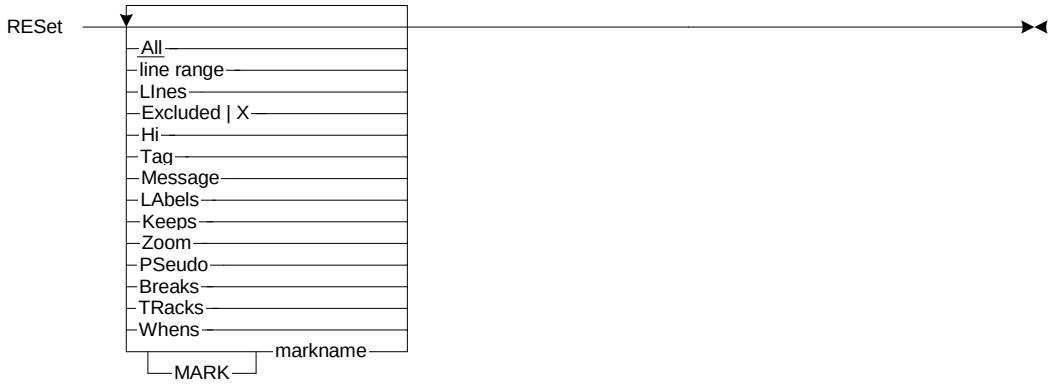
To display the next occurrence of the REC-CNT data item, type this command:

```
FIND REC-CNT NEXT
```

Instead of re-entering the same command, type this command:

```
REPEAT
```

# RESET Command




## Function

The RESET command is used to turn off highlighting, erase tags, redisplay excluded lines, cancel pending line commands, terminate message line displays, delete pseudo code statements, delete lines created as a result of one of the SmartTest Zoom line commands (ZA, ZD, or ZH), delete WHEN statements, erase line labels, and delete kept lines. All ISPF reset options are supported.

## Operands

| Operand | Description |
|---|---|
| Blank | Defaults the RESET command to the ALL value. |
| All | Resets all conditions indicated by these operands:<br>HI LINES EXCLUDED<br>TAG MESSAGE<br>The ALL operand does not reset conditions indicated by these operands:<br>PSEUDO BREAKS LABELS<br>ZOOM WHENS KEEPS |
| *`line range`* | Resets the HI, TAG, EXCLUDED, LINES, and MESSAGE conditions only for the specified line or range of lines. |
| LInes | Clears any pending line commands. |
| Excluded \| X | Displays any lines that have been excluded. |

| Operand | Description |
|---|---|
| Hi | Removes the highlighting from any line that has been highlighted as a result of another SmartTest command. |
| Tag | Erases tags in columns 73 through 80 that have been set as a result of another SmartTest command. |
| Message | Erases short or long messages that are displayed. |
| LAbels | Erases all line labels entered in the prefix area (columns 1 through 6). A line label can also be erased by typing over it with blanks on the Program View screen. |
| Keeps | Deletes all lines kept at the top of the screen as the result of a KEEP command. |
| Zoom | Deletes all lines created by using one of the SmartTest Zoom line commands such as ZA, ZD, ZG, ZGH, or ZH. |
| PSeudo | Deletes all pseudo code statements within the current active or qualified program. |
| TRacks | Deletes all Execution Tracking entries that have been recorded in the active test session. Deletion actually occurs when execution resumes. |
| Breaks | Deletes all BREAK statements within the current program if they are the first pseudo code verbs in the block of pseudo code. |
| Whens | Deletes all WHEN statements within the current active or qualified program. |
| MARK *markname* | Resets the HI, TAG, EXCLUDED, LINES, and MESSAGE conditions only for statements belonging to the mark set. The keyword MARK is not needed if there is no ambiguity. |

## Usage Notes

You can specify any combination of operands for the RESET command and reset attributes by selecting View ▸ Reset.

# RETURN Command

```
RETURn ──────────────────────────────────────────────────────────────►◄
```

## Function

The RETURN command is used to terminate the current screen function and return to the primary screen.

## Operands

None.

## Usage Notes

Using the RETURN command is a quick way of ending a SmartTest function. RETURN is similar to the ISPF RETURN command that simulates multiple END commands and returns directly to the invoking screen.

# RFIND Command

RFind

## Function

Use the RFIND command to repeat the last FIND or FINDXTND command from the current cursor location.

## Operands

None.

## Usage Notes

The RFIND command is a convenient means of locating the next occurrence of the specified target of a FIND or FINDXTND command. The search is performed in the direction indicated in the FIND or FINDXTND command. RFIND is assigned to the PF5/17 key as a default.

The REDO command can also be used to execute the RFIND command. See "REDO Command" on page 374 for more information.

# RHIGH Command

RHIgh

## Function

The RHIGH command is used to repeat the last HIGH command from the current cursor location.

## Operands

None.

## Usage Notes

Enter RHIGH in the command input area to repeat the last HIGH command from the current cursor location.

The REDO command can also be used to execute the RHIGH command. See "REDO Command" on page 374 for more information.

# RPREF Command

RPref

## Function

When on the Program View screen, the RPREF command is used to redisplay the last View - Paragraph Cross Reference pop-up. When on the View - Paragraph Cross Reference pop-up, the RPREF command is used to redisplay the last Program View screen.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

None.

## Usage Notes

The View - Paragraph Cross Reference pop-up is redisplayed with the exact information as previously shown. RPREF can be used to go back and forth between the Program View screen and the View - Paragraph Cross Reference pop-up.

When RPREF is entered and no PREF command was entered, RPREF functions as a PREF command with the current cursor location as the target and default direction.

The REDO command can also be used to execute the RPREF command. See the REDO command for more information.

# RSCROLL Command

RSCroll

## Function

The RSCROLL command is used to repeat the last SCROLL command from the current cursor location.

## Operands

None.

## Usage Notes

Enter RSCROLL in the command input area to repeat the last SCROLL command from the current cursor location.

The REDO command can also be used to execute the RSCROLL command. See "REDO Command" on page 374 for more information.

# RTRACE Command

```
RTrace ──┬──────────────┬──────────────────────────────────────►◄
         ├─trace-dec-opt─┤
         └─BACKup────────┘
```

ASG-Insight only

## Function

The RTRACE command is used to continue the last TRACE command from where it stopped. This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Trace Decision Options pop-up when RTRACE is entered without an operand at a decision point. When RTRACE is entered without an operand and not at a decision point, the trace function is resumed from the stopping point. |
| *trace-dec-opt* | Specifies a TRACE command decision option. |
| BACKup | Returns to the decision point where the last decision option was entered. The TRACK is reset to include only those lines in it when that decision point was reached. Only one BACKUP is allowed following an RTRACE command. A message displays if there is no previous decision point or if BACKUP has already been done. |

## Usage Notes

The RTRACE command can be used to continue a trace function without exiting the Program View screen. Enter the desired option number in the command input area and press Enter. This process can be continued until the end of the trace function is reached. If a different path is to be followed, type BACKUP in the command input area and press Enter, then select the desired option.

If a trace function is interrupted and another primary command is entered, the messages displayed by the trace function are cleared. To continue the trace function, enter the RTRACE command. The screen is then scrolled to the last trace location and the messages are redisplayed.

If a decision point prompt is unclear, the Trace Decision Options pop-up can be displayed that explains the decision point. Enter the RTRACE command without an operand to display the Trace Decision Options pop-up. The possible options from the decision point, the type of branch, the line number, and the first statement of the destination are shown on the Trace Decision Options pop-up. An asterisk (*) next to an option indicates that option has already been traced.

The REDO command can also be used to execute the RTRACE command. See "REDO Command" on page 374 for more information.

## Trace Decision Options Pop-up

Figure 142 displays when the RTRACE command is entered without an operand. This pop-up is used to list all possible options from the decision point, the type of branch, the line number and the first statement of the destination. An asterisk displayed next to an option shows that you have already TRACEd that option.

**Figure 142 • Trace Decision Options Pop-up**

```
                              Trace Decision Options
Command ===> ____________________________________________ Scroll ===> CSR

Trace of: VIAPCOB
Direction: FORWARD
Paragraph: 000242  PROCEDURE DIVISION USING PARM-AREA.
Decision: 000243  PERFORM PROGRAM-INIT.

Option Info indicates TARGET's and '*' for code already TRACE'd

Select one of the following decision options:                    Option
S    Option                                                       Info
- ------------------------------------------------------------- -------
_  1. BYPASS CODE (000258  PERFORM MAIN-ROUTINE THRU MAIN-ROUTIN
_  2. ENTER PERFORMED CODE (000294  PROGRAM-INIT.)
****************************** BOTTOM OF DATA ****************************
```

## Fields

| Field | Description |
|---|---|
| Trace of | Specifies the name of the target you are tracing. For a dataname, it also indicates the usage (USE, MOD, REF). If an IN clause was specified, it is also displayed. |
| Direction | Specifies the direction requested for the trace (NEXT, PREVIOUS, FORWARD, BACKWARD). |
| Paragraph | Specifies the line number and name of the paragraph that contains the decision point. |
| Decision | Specifies the line number and statement of the decision point. |
| S | Selects the option. |
| Option | Specifies all feasible options from the decision point. Each option lists the type of branch, line number and first statement of the destination. |
| Option Info | Indicates whether the option is the requested target or has already been traced. |

# RUN Command

```
RUN ──┬─MONitor──┬──┬─TO─┬─CURsor | CSR─┬──────────────────────────►◄
      └─Nomonitor─┘  │    └─line number──┘
                     └──────────────────┘
```

## Function

**Note:**

The RUN command operates differently if SET BACKTRACK is ON or if testing in the CICS environment. See "RUN Command (BACKTRACK ON)" on page 396 for information about the RUN command with BACKTRACK ON. See "RUN Command (CICS Only)" on page 393 for information about the RUN command in the CICS environment.

The RUN command begins testing of a program or resumes testing after an interrupt. Interrupts result from Breakpoints or error conditions.

The SET MONITOR command determines the default operand.

The test session executes until one of these conditions occur:

- A Breakpoint is reached.
- An abend occurs.
- The test session is completed.
- The target of the TO operand is reached.

## Operands

| Operand | Description |
|---|---|
| Blank | Specifies the operand indicated with the SET MONITOR command is used. |
| MONitor | Indicates all features are available for testing, including Address Stop Entry, Statement Counts, Execution Tracking, and pseudo code monitoring. The default is MONITOR. |
| Nomonitor | Reduces the CPU overhead during a test session. This is particularly useful when testing large programs or programs that read thousands of records. |

| Operand | Description |
|---|---|
| TO | Indicates that execution is completed to the current cursor position or the specified line number. |
| CURsor \| CSR | Indicates that execution is completed at the current cursor position when used with the TO target operand. The default is CURSOR. |
| *line number* | Indicates that execution is completed at the specified line number when used with the TO target command. |

## Usage Notes

The RUN command is used to begin TSO execution of a program or to resume execution after an interrupt occurs. If there are no interrupts, the program executes to completion. The STEP command can be used instead of the RUN command to execute the program statement-by-statement or paragraph-by-paragraph.

See "STEP Command (BACKTRACK OFF)" on page 421 for detailed information.

The NOMONITOR operand is useful for executing to the nth program in a series without incurring the CPU overhead of monitoring the other programs (see Session Tailoring). Unpredictable results can occur when an unmonitored program issues an operating system service such as SPIE/STAE or ATTACH. When the NOMONITOR operand is specified, various SmartTest facilities are unavailable, including Backtrack, Address Stop Entry, Statement Counts, When processing, and Execution Tracking.

The TO operand is useful for executing to a certain location in the program without inserting a Breakpoint. If no interrupts occur, the program executes to the target of the TO operand. If the target is reached without an interrupt, the status indicates that the program execution was stopped by the RUN TO request. The target of the TO operand is active only for the RUN command for which it was specified. If the target is not reached, an interrupt has occurred prior to the target or the program did not execute to the target based on program logic.

If pseudo code is entered in a batch program prior to submitting the program for use in a batch test session, the SAVE PSEUDO command must be entered before the batch submit. The SAVE command makes the pseudo code available to the batch test session.

The ATTENTION key functions differently during a TSO test session when the RUN command is executed. If the ATTENTION key is pressed after a test session has been started with the RUN command, the RUN command is changed to a STEP command and a break occurs at the next COBOL verb or Assembler source instruction. If the ATTENTION key is pressed after a test session has been started with the STEP *nnn* command, the remaining steps are not executed and a break occurs at the next COBOL verb or Assembler source instruction.

When a test is running in unmonitored mode and the ATTENTION key is pressed, a break does not occur until a return to a monitored program. If the ATTENTION key is pressed again, an abend is issued and the test session terminates.

During a connected batch session, if the ATTENTION key is pressed after the RUN command is executed, the batch session is disconnected. The batch session test continues to run until completion or until a program interrupt occurs. The batch session must be reconnected to see the status of the batch session test run.

When using the DYNAM Compile option and the RUN NOMONITOR command, these steps must be performed to allow a break on entry to a subroutine:

- The Link Edit must have the REUS option (specified on the List - Analyze Submit pop-up).
- The name of the subroutine must be specified on the Test Session Tailoring screen. (Set BREAK ON ENTRY to YES.) This screen displays by issuing the LIST TAILOR command.
- Prior to starting the test by entering the RUN command, the subroutine must be qualified (i.e., loaded). For example, type `Q VIASUB.VIASUB`. Alternatively, you can enter the load module name on the Load Module Intercept List screen.

## Examples

These examples assume the VIAPCOB program was selected for testing.

To pause program execution at the first executable statement following the PROCEDURE DIVISION label, type this command. Continue execution to the end of the program by repeating this command.

```
RUN
```

In these examples, RUN NOMONITOR processing is combined with STEP processing.

To execute the test session unmonitored until a S0C7 abend occurs, type this command:

```
RUN NOMONITOR
```

Correct the invalid data by typing valid data over the “* INVALID *” in the Zoom Data window.

To use the STEP command to ensure the S0C7 error condition has been corrected, type this command:

```
STEP
```

To continue the test session until the end of the job is reached, type this command:

```
RUN NOMONITOR
```

# RUN Command (CICS Only)

```
RUN ──┬──────────────────────────────┬──────────────────────────────►◄
      ├─FORCE──────────────────────┤
      ├─tran-id──┬──────────────┤
      │          └─operands──────┤
      └─TO───┬─CURSOR | CSR─┤
             └─line number──┘
```

## Function

**Note:**

The RUN command operates differently if SET BACKTRACK is ON. See "RUN Command (BACKTRACK ON)" on page 396 for information.

The RUN command is used to initiate a transaction or resume execution of an active test session after an interrupt occurs. Interrupts can result from a Breakpoint or an error condition. If there are no interrupts, the program executes through completion. The STEP command can be used instead of the RUN command to execute the program statement by statement.

See "STEP Command (BACKTRACK OFF)" on page 421 for more information.

If RUN is entered when not in an active test session, the effect is the same as entering TOGGLE, which toggles the user into the CICS environment and allows an active test to be initiated. See"TOGGLE Command" on page 439 for more information.

## Operands

| Operand | Description |
|---|---|
| Blank | Initiates a test session or resumes execution when entered with no operands. |
| FORCE | **1** Causes the monitor to ignore the current storage violation warning allowing the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. You must have proper authorization to use the FORCE operand.<br><br>**Note:**<br>Careful consideration should be given to which storage violations are to be overridden with this feature. This feature is intended to temporarily override a Storage Violation message for non-standard coding practices.<br><br>**2** Causes processing to continue on an invalid FREEMAIN error if the INVREQ condition has been handled or ignored. |
| *tran-id* | Performs an implied TOGGLE command and invokes the specified transaction. This option is not available during an active test session. |
| *operands* | Specifies any valid CICS transaction operands. |
| TO | Indicates that execution is completed to the current cursor position or the specified line number. |
| CURSOR \| CSR | Indicates that execution is completed at the cursor position when used with the TO target operand. The default is CURSOR. |
| *line number* | Indicates that execution is completed at the specified line number when used with the TO target operand. |

## Usage Notes

When a storage violation occurs, SmartTest stops the test and displays a status box at the bottom of the screen. Do not use the FORCE operand to continue execution without understanding the significance of the Storage Violation Warning.

The TO target operand is useful for executing to a specific location in the program without inserting a Breakpoint. If no interrupts occur, the program executes to the specified target (cursor position or line number). If the target is reached without an interrupt, the status indicates that the program execution was stopped by the RUN TO request. The target of the RUN TO request is active only for the RUN command for which it was specified. If the target is not reached, either an interrupt has occurred prior to the target, or program logic prevents execution to the target.

You can also initiate or resume testing by selecting Test ▸ Run.

## Examples

When an active test session is in effect for VIACCOB and execution is stopped at a Breakpoint or an error condition, type this command:

```
RUN
```

Program execution continues until another Breakpoint or error condition, or to the end of the program.

To initiate the SmartTest-CICS demonstration program using the VCOB CICS transaction, type this command:

```
RUN VCOB
```

To initiate the CEMT transaction to inquire on the program VIACCOB, type this command:

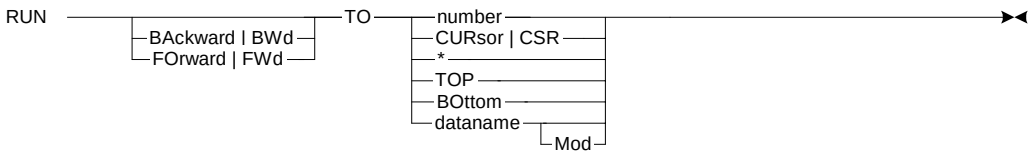
```
RUN CEMT I PROGRAM(VIACCOB)
```

In this example, RUN TO processing is used with cursor positioning. After the VIACCOB program is stopped at the first executable statement following the PROCEDURE DIVISION label, locate line 807. This should be the third IF statement in the PARM-EDIT paragraph. Type this command:

```
RUN TO
```

Position the cursor on the IF statement on line 807 and press Enter. The test session should stop with a status of stopped by 'RUN TO' request. A ZD on the IF statement displays the PARM passed at execution.

# RUN Command (BACKTRACK ON)




## Function

The RUN command may be used to enter, exit and simulate execution backward or forward in the execution history. The Backtrack Recording facility must be active for this command to be valid (see the BACKTRACK option of the SET command). When BACKTRACK is ON, testing can be directed backward or forward to a line number, the cursor position, or the Top or Bottom of the execution history. The RUN command can also be used to terminate the Backtrack Review facility and return to normal execution mode.

**Note:**

While executing in Backtrack Review mode, the Backtrack arrow (BKTR =>) points to the current position in the execution history.

## Operands

| Operand | Description |
|---|---|
| BAckward \| BWd | Directs the RUN in the backward direction to the specified target. If the Backtrack Review facility is not active, the BACKWARD operand automatically invokes it. If currently positioned at the top of the statement execution history, the BACKWARD operand generates an error. |
| FOrward \| FWd | Directs the RUN in the forward direction to the specified target. FORWARD is valid only in BackTrack Review mode, that is, after a RUN BACKWARD or STEP BACKWARD command. |
| *number* | Defines the target line number in the execution history. If the line number cannot be found in the execution history (in the current direction), SmartTest displays an error message. If the target number exists more than once in the execution history, the target is the first occurrence of the line number in the direction in which you are going (e.g. if your current direction is backward, positioning is at the most recent execution of the statement at the target line prior to the current point). |

| Operand | Description |
|---|---|
| CURsor \| CSR | Indicates that the target line in the execution history is to be determined by the current cursor position. If the line on which the cursor is positioned cannot be found in the execution history (in the current direction), SmartTest displays an error message. If the target number exists more than once in the execution history, you are positioned at the first occurrence of the line number in the direction in which you are going (e.g. if your current direction is Backward, you are positioned at the prior (most recent) execution of the statement at the target line prior to the current point in time. |
| asterisk (*) | Terminates the Backtrack Review facility and returns to the next instruction to be executed in normal execution mode. |
| TOP | Positions the Backtrack Review facility to the top (oldest entry) of the execution history. The Backtrack arrow (BKTR=>) points to the first available statement to have been executed in Backtrack Recording mode. The current direction is automatically set to FORWARD. |
| BOttom | Positions the Backtrack Review facility at the bottom (most current entry) of the execution history. The Backtrack arrow (BKTR=>) points to the last statement executed in the Backtrack Recording mode (i.e.: the most recently executed statement). The current direction is automatically set to BACKWARD. |
| *dataname* | Defines a target dataname in the execution history. This dataname must have been modified by a statement in the execution history. If the dataname cannot be found, SmartTest displays an error message. If the dataname exists more than once, the target is the first occurrence of the dataname in the current direction (e.g. if your current direction is Backward, you are positioned at the most recent statement that modified the dataname). |
| Mod | Indicates that the specified dataname is found in the execution history only if it has been modified. The default is Mod. |

## Usage Notes

The BACKWARD or TO TOP operands of the RUN command issued during normal testing initiates the Backtrack Review facility, if it is not already active. SmartTest adds a status indicator to the Execution Status Box indicating that the Backtrack Review facility is active and shows your current execution direction. When the Backtrack Review facility is entered or exited, a message displays.

BACKTRACK restores data items to their historical values based on its position within the execution history. Any Zoomdata or Keep windows displayed data or operands data updates with these historical values automatically. Zoomdata or Keep commands issued during Backtrack Review mode reflect the values at the current position in the execution history.

The amount of storage used to record Backtrack execution history is determined by the BACKTRACK option of the SET command. If this storage should fill up before normal program execution stops, Backtrack Recording begins overwriting the oldest entries in the table (see the BACKTRACK option of the SET command).

You can initiate or resume testing by selecting Test ▸ Run.

See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for detailed information about the BACKTRACK option of the RUN command.

# SAVE Command

```
SAve ──┬──────────┬──────────────────────────────────────────────►◄
       ├─PSeudo───┤
       ├─Marks────┤
       ├─Equates──┤
       └─All──────┘
```

Bold operands are available only with ASG-Insight

## Function

The SAVE command is used to save pseudo code, Marks, and/or Equates in the AKR. When the SAVE command is entered with no operands, the Save Options pop-up displays.

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Save Options pop-up. From this screen, pseudo code, Marks, and Equates can be saved. A member name can be specified if the program/CSECT being viewed does not exist in the AKR. Figure 143 on page 400 shows the Save Options pop-up. |
| PSeudo | Saves all pseudo code entered for the program currently being viewed on the Program View screen. |
| Marks | Saves all Marks created or changed for the program currently being viewed on the Program View screen. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested. |
| Equates | Saves all Equates entered for the program currently being viewed on the Program View screen. |
| All | Saves pseudo code, Marks, and Equates entered for the program currently being viewed on the Program View screen. |

## Usage Notes

If pseudo code is entered for a batch test before submitting the test, the SAVE PSEUDO command must be entered to make the pseudo code available to the batch test.

You can also save pseudo code, Marks and Equates by selecting File ▸ Save.

## Examples

To save all pseudo code entered for the program currently being viewed on the Program View screen with the program/CSECT in the AKR, type this command:

```
SAVE PSEUDO
```

To display the Save Options pop-up, (Figure 143) type SAVE.

**Figure 143 • Save Options Pop-up**

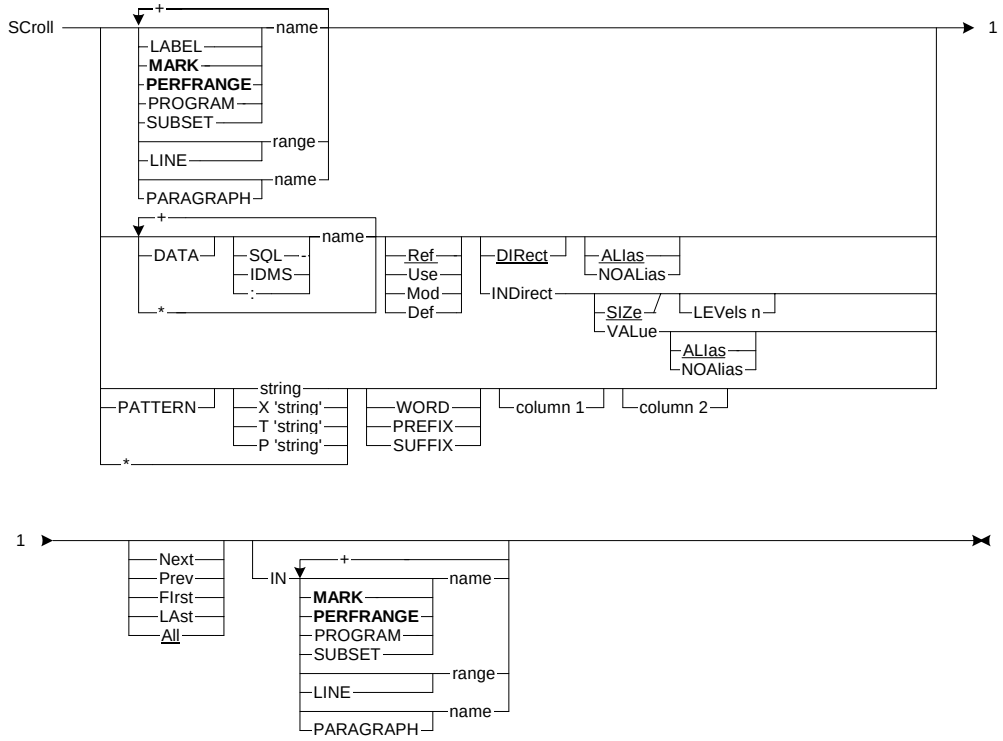
```
  File  View  Test  Search  List  Options  Help
-------------------------------------------------------------------------------
                                Save Options
 Command ===> ____________________________________________________________ _

 Type the desired save options.  Then press Enter.  Press PF3/15 to
 discontinue the save request.

 Save pseudo code  NO       (pseudo code, breakpoints and when conditions)
 Save equates  . . NO
 Save marks  . . . NO

 Program/CSECT  . VIAPCOB       AKR member  __________  (if different)

 Application Knowledge Repository (AKR):
    Data set name 'SRENTED.VIACENXX.AKR'

+-----------------------------------------------------------------------------+
001133         MOVE +0 TO CURRENT-PARM-NUMBER
+-----------------------------------------------------------------------------+
|STATUS: STOPPED BY STEP REQUEST           PROGRAM: VIAPCOB   DATE: DDMMMYYYY
|  STMT: 001140  OFF: 001E00  AMODE: 24     MODULE: VIAPCOB   TIME: 17:23:25
|SOURCE: MOVE SPACES TO PADDED-PARM.
+-----------------------------------------------------------------------------
```

Default values for saving pseudo code, Marks, and Equates are those entered on the Options -Parameter Definition pop-up. If the default values on the Save Options pop-up are changed, the values apply only to this execution of the SAVE command. The next time the Save Options pop-up displays, the default values are restored.

When the program/CSECT being viewed exists in the AKR but is currently being accessed by another user, this message displays:

```
The program currently exists in the AKR but you do not
currently have write access to that program. You CANNOT
save PSEUDO CODE, EQUATES, or MARKS for this program.
```

# SCROLL Command




Bold operands are available only with ASG-Insight

## Function

The SCROLL command is used to position the screen to the first line containing the specified target. Highlighted lines remain unchanged.

## Operands

| Operand | Description |
|---|---|
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. This operand is only available if Insight is installed and an Insight analysis has been run on the program. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, DB2 \| SQL, DDL, DEBug, DEFinition, DIRective, DIVision, DL/I \| DL/1, DML, ENtry, EXIt \| PGMExit, GOto, IDMS, Input, IO, LABel, MATH, Output, PARagraph, PERform, SECtion, SORTMerge, STructure, TESted \| UNTested<br>**Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. When a range is specified, data is scrolled to the first line in the range. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| asterisk (*) | Reuses the target of the previous search, exclude, find, highlight or scroll action or command. |
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the EXCLUDE command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS, and DIRECT. |

| Operand | Description |
|---|---|
| SQL *name* | Includes datanames that are DB2/SQL variables only. |
| IDMS *name* | Includes datanames that are IDMS variables only. |
| : *name* | Includes datanames that are COBOL variables only. |
| Ref | Includes occurrences of the dataname where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Def | Includes definitions of the dataname in the DATA DIVISION. |
| DIRect | Includes occurrences of the dataname (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT; however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed. |
| INDirect | Includes occurrences of any dataname indirectly affected by the specified dataname (and aliases if specified). This can be very useful when working with datanames. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. |
| SIZe | Includes occurrences of datanames that could be directly or indirectly affected if the size of the specified dataname were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. |
| VALue | Includes occurrences of datanames that could be directly or indirectly affected if the specified dataname were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand. |
| LEVels *n* | Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. The default is All levels for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. |

| Operand | Description |
|---|---|
| ALIas | Includes aliases for the dataname. These are the valid aliases:<br>• Parent - higher level group item<br>• Child - lower level item<br>• Rename/Redefinition - renamed, redefined, or 88 level items<br>This is the default value for the DATA name operand. |
| NOAlias | Ignores aliases for the specified dataname. |
| PATTERN | Indicates the characters that follow are part of a string. |
| *string* | Specifies a string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used. |
| X'*string*' | Specifies a hexadecimal string. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes. |
| T'*string*' | Specifies a text string. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes. |
| P'*string*' | Specifies a picture string. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. Nine special characters are defined by SmartTest for use in picture strings. These special characters can be combined with other characters. These are the special characters:<br>P'=' Any character<br>P'¬' Any nonblank character<br>P'.' Any nondisplay character<br>P'#' Any numeric character<br>P'-' Any non-numeric character<br>P'@' Any alphabetic character (upper or lowercase)<br>P'<' Any lowercase alphabetic character<br>P'>' Any uppercase alphabetic character<br>P'$' Any special character (not alphabetic or numeric) |
| WORD | Specifies a pattern string preceded and followed by any non-alphanumeric character (except hyphen). |

| Operand | Description |
|---|---|
| PREFIX | Specifies a word that begins with the specified pattern string. |
| SUFFIX | Specifies a word that ends with the specified pattern string. |
| Next | Scrolls forward from the current cursor position to the next occurrence of the requested target. |
| Prev | Scrolls backward from the current cursor position to the previous occurrence of the requested target. |
| FIrst | Scrolls from the top of the source file to the first occurrence of the requested target. |
| LAst | Scrolls backward from the bottom of the source file to the first occurrence of the requested target. |
| All | Scrolls to the first occurrence of the requested target. This is the default value for the SCROLL command. |
| IN | Restricts the SCROLL command to the specified target type. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, ALIAS, etc.) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

| | | | |
|---|---|---|---|
| LABEL | MARK | SUBSET | PARAGRAPH |
| LINE | PERFRANGE | PROGRAM | |

- The * operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

After an FX command has been entered, SCROLL PARA PREV can be entered with the cursor positioned on an FX command result to see the paragraph name containing the result.

You can also scroll to selected lines by selecting a search option on the Search pull-down.

## Examples

Using this command, the first occurrence of a line containing the characters ZIP displays:

```
SCROLL ZIP
```

# SELECT Command

```
Select ──┬─trace-dec-opt─┬──────────────────────────────────────────────────►◄
         └─string────────┘
```

Bold operands are available only with ASG-Insight

## Function

The SELECT command is used to select an option on the Trace Decision Options pop-up. SELECT followed by the desired option can be entered in the command input area. To select a particular item, type S on any screen with a selection field to the left of the displayed items.

## Operands

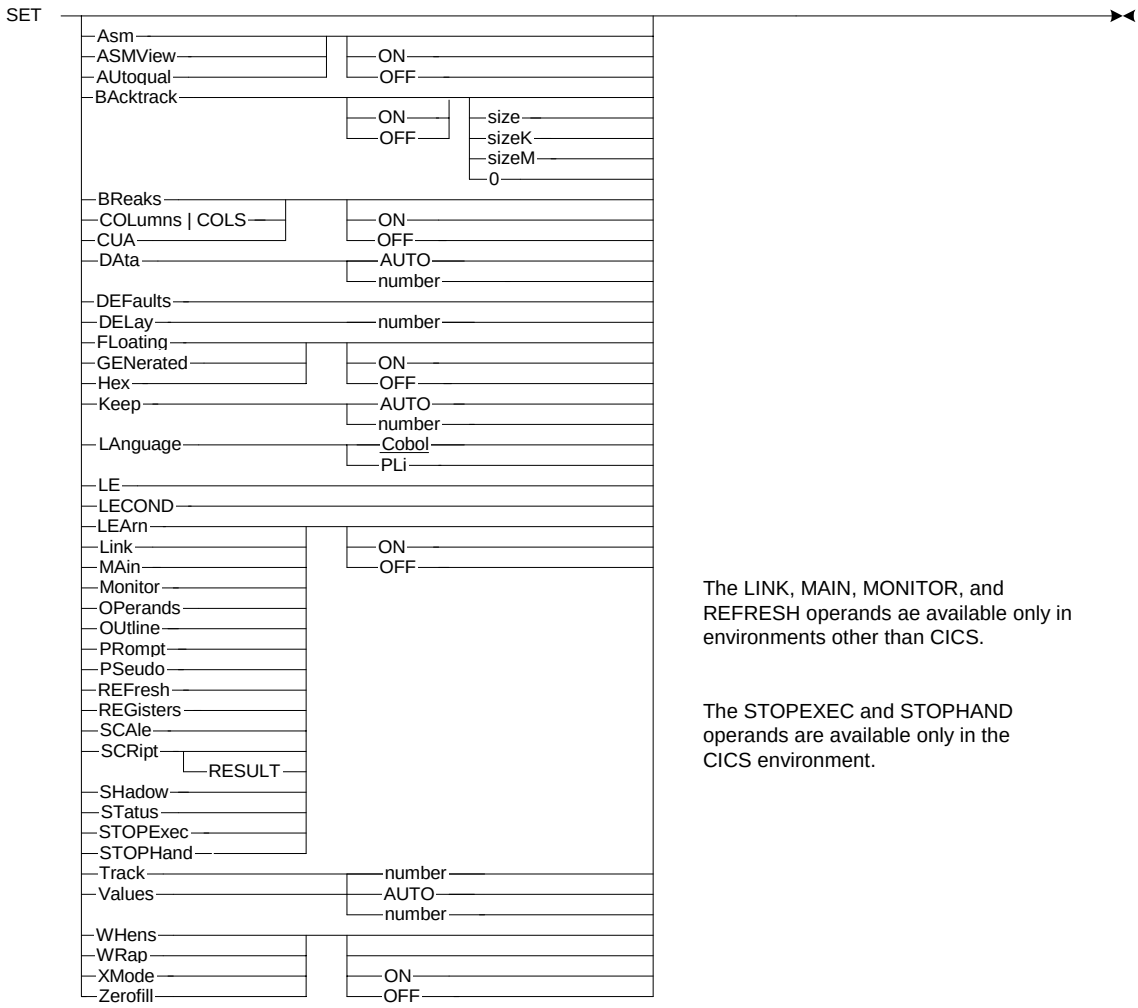
**trace-dec-opt.** Specifies a TRACE command decision option. This operand is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

**string.** Specifies a string of characters.

## Usage Notes

The SELECT command can also select a particular item from a list of items, such as a directory list.

# SET Command




The LINK, MAIN, MONITOR, and REFRESH operands ae available only in environments other than CICS.

The STOPEXEC and STOPHAND operands are available only in the CICS environment.

## Function

The SET command is used to enable or disable the mode indicated by the specified operand. Entering the SET command with only a mode operand functions as a toggle switch (e.g. SET ASM sets the mode to ON if the current value is OFF).

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Options - Modes screen that shows the current setting for each SET command mode. The setting for any mode can be changed on the Options - Modes screen. |
| Asm | Indicates if the increment for the STEP command is at the source statement level (OFF) or the Assembler instruction level (ON). Changing the value of this mode affects the status box, the Program View Screen, and the Execution Tracking screen. Assembler information shown in the status box includes the offset, addressing mode, Assembler instruction, hexadecimal value, and the general register contents.<br><br>When this mode is ON, all Assembler instructions generated by a source statement are displayed on the Program View Screen with their corresponding source statement.<br><br>For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*. |
| ASMView | Specifies whether a program that has not been analyzed may be stepped into at the Assembler instruction level. When this mode is ON, Assembler code for a program that is not in the AKR displays for a program being tested or displayed on the Program View screen. When this mode is OFF, a message displays that indicates the source is unavailable in the AKR. The default is OFF. |
| AUtoqual | Specifies whether the program being tested is automatically displayed when a STEP or RUN command is entered after qualifying a program. When this mode is ON, the calling program displays if an error condition occurs in a CALLed program. When this mode is OFF, the qualified program remains displayed and the status box is updated for the program being tested, not the qualified program. The default is ON. |

| Operand | Description |
|---|---|
| BAcktrack | Controls the Backtrack Recording facility and the Backtrack Review facility. SET BACKTRACK sets the Backtrack Recording mode ON or OFF, and optionally sets the size of the Backtrack Recording buffer to other than the default size of 1 megabyte (100 KB for CICS). The buffer size affects how much statement execution history is available to the Backtrack Review facility. When the buffer fills, SmartTest begins overwriting the oldest information, thus saving the more recent execution history. Typing `SET BACKTRACK` with no other operands toggles the current mode ON and OFF.<br><br>**Note:**<br>Backtrack Recording mode can be turned ON only during an active test.Toggling or turning the BACKTRACK Recording mode OFF causes the recording buffer to be discarded/destroyed. |
| *size* | Defines the Backtrack Recording buffer's size in absolute bytes (e.g., 750000). The maximum allowable value for size is 16,777,216. The minimum allowable value is 32,768. If a size in absolute bytes greater than 1 MB (1,048,576) is entered, SmartTest rounds it down to the next lower megabyte. If a size in absolute bytes less than 1 MB and greater than the minimum allowable size is entered, SmartTest rounds it down to the next lower kilobyte.<br><br>**Note:**<br>The default values (maximum allowable and default size) are user definable with the Backtrack Control Table (VIAPBTTB). See your systems programmer for the defaults established for your installation.<br>BACKTRACK must be OFF for the size operand to take effect. If BACKTRACK is ON, the size operand updates, and a warning message issued. The new value does not take effect until a new Backtrack Recording session is started. |
| *sizeK* | Defines the Backtrack Recording buffer's size in kilobytes (e.g., 750K). The maximum allowable value for size is 16,384 KB. The minimum allowable value is 32 KB. |
| *sizeM* | Defines the Backtrack Recording buffer's size in megabytes (e.g., 3 MB). The maximum allowable value for size is 16 MB. |

| Operand | Description |
|---|---|
| 0 (zero) | Specifies a size of 0 (zero) restores the default buffer size.<br><br>**Note:**<br>The default values (maximum allowable and default size) are user definable with the Backtrack Control Table (VIAPBTTB). See your systems programmer for the defaults established for your installation. |
| BReaks | Enables or disables the Breakpoint facility for a test session. When this mode is OFF, all Breakpoints are ignored. When this mode is ON, all Breakpoints are active unless deactivated on the Breakpoints List screen or the Session Tailoring screen. All Breakpoints are ignored regardless of the BREAK mode setting when the SET PSEUDO OFF command is used. The default is ON. |
| COLumns \| COLS | Displays a ruler across the top of the Program View screen indicating column positions. |
| CUA | Toggles the CUA action bar on and off so that those users who prefer to use the command interface can gain two more lines on the screen. |
| DAta | Controls the number of lines used by all Zoom Data windows (excluding the lines used to display the outline around the window). When the number of lines required to display the data items within the window exceeds the number of lines in the window, the window becomes vertically scrollable. The default is AUTO. |
| DEFaults | Sets all modes to their default settings. |
| DElay number | Controls the number of seconds to delay between steps when using the STEP AUTO command. The default is 1. |
| Floating | Displays the floating point registers in the program status box when this mode is ON. The STATUS operand must also be ON or this operand is ignored. The default is OFF. |
| GENerated | Displays generated statements for Assembler macros and Sage APS Painter code (when the Sage APS option is installed) when this mode is ON. The default is OFF. |
| Hex | Displays data values in hexadecimal format when this mode is ON. Changing the value of this mode affects the Execution Tracking screen and data area displays. The default is OFF. |

| Operand | Description |
|---|---|
| Keep | Controls the number of lines used by the Keep window (excluding the separator line at the bottom). When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. The default is AUTO. |
| LAnguage | Identifies the source language so that SmartTest can make available access to debugging facilities that apply to the language of the active program being tested. You can specify COBOL or PLI as additional operands. Application programmers who maintain either COBOL, PL/I, or a mixture of both, find the user interface improves productivity because it is more intuitive for doing the debugging task at hand. |
| LE | Specifies whether the Language Environment (LE) processes all errors without interpretation by SmartTest. If this mode is set to ON, LE processes all errors. If this mode is set to OFF, SmartTest processes program checks before the Language Environment does. Setting LE to ON causes SmartTest to bypass monitoring of many CEE modules and gives the Language Environment control. |
| LECOND | This option is used in conjunction with LE. It does not appear on the Options - Modes screen unless LE is enabled. If LECOND is set to ON, then SmartTest passes all errors to an error handler (registered through CEEHDLR callable service) upon notification of the error from LE. When this option is set to OFF, SmartTest stops on the error condition. |
| LEArn | Displays all primary commands that are generated internally when actions from the pull-downs are executed. The default for this operand is OFF. |
| Link | Monitors linked programs for the test session when this mode is ON. The default value is OFF. This operand is ignored in the CICS environment. |

| Operand | Description |
|---|---|
| MAin | Identifies the program to be tested as a mainline routine executed by coding JCL EXEC PGM=, or as a subroutine that is called by another program. SET MAIN ON, the default, identifies the program as a mainline routine. SET MAIN OFF identifies the program to be tested as a subroutine. SET MAIN can only be turned OFF for COBOL programs with source code on the AKR. Entry to the COBOL program may be either a PROCEDURE DIVISION USING statement or an ENTRY USING statement.<br><br>When SET MAIN is OFF, data used by the subroutine is stored and initialized via the linkage section. Storage is initialized to zero for numeric items and to spaces for nonnumeric items. Data in the subroutine may be user modified using the ZOOMDATA command after stepping through the procedure division initialization code.<br><br>This operand is valid only in the TSO (Standard) environment. |
| Monitor | Specifies whether the MONITOR operand of the RUN command is the default. When this mode is ON, it is the default. When this mode is OFF, the NOMONITOR operand of the RUN command is the default. This operand is ignored in the CICS environment. |
| OPerands | Displays the current values for the data items on the current statement in the long message area after each STEP or RUN command when this mode is set to ON. If the current statement has only one data item, its name and as much of the current value as possible are displayed. If the current statement contains more than one data item, a maximum of 16 characters of the name and current value for each data item displays. The default is OFF. |
| OUtline | Specifies whether command and/or function results are enclosed in a box (outline) on the Program View screen. The default is ON. |
| PROMPT | Enables or disables the prompt to save the current Environment profile if there have been changes. |
| PSeudo | Enables or disables the pseudo code facility for a test session. When this mode is ON, pseudo code is executed unless deactivated on the Test Session Tailoring screen or the Breakpoints List screen. If this mode is OFF, any attempt to execute pseudo code, including Breakpoints and When Conditions, is ignored, regardless of the values of the BREAKS and WHENS operands. When this mode is set to OFF from within a block of pseudo code, the remaining pseudo code is ignored (including Breakpoints and When Conditions). The default is ON. |

| Operand | Description |
|---|---|
| REFresh | Refreshes the full ISPF screen after each STEP or RUN command when this mode is ON. The default is ON.<br><br>**Note:**<br>Remote users using terminal emulators should turn the REFRESH mode and the STATUS mode OFF to avoid repainting the screen between each step. |
| REGisters | Displays the general registers in the program status box when this mode is ON. The STATUS operand must also be ON or this operand is ignored. The default is OFF. |
| SCAle | When this mode is ON, a scale line displays above alphanumeric and hexadecimal format data items within Zoomdata and Keep windows. The default is OFF. |
| SCRipt | Determines whether to generate a script file automatically. When this mode is ON, all SmartTest primary commands entered during a test session are saved to the script file. Using this mode allows execution of a predefined command sequence in any test session. Scripts support all SmartTest primary commands.<br><br>The SET SCRIPT ON command allocates and opens a dataset, and saves all SmartTest primary commands in this dataset. The dataset name is in the format: `TSOUSERID.STT`*nnnnn*`.VIASCRIP`, where *nnnnn* is a number assigned by SmartTest, beginning with 00001 and incremented by 1 for each new script file. The name of the dataset displays when it is opened and closed.<br><br>The script file is closed (ignored) when you type `SET SCRIPT OFF`. Once a script file is closed, it can be used as the dataset name in the EXECUTE command. The default is OFF. |
| RESULT | Saves the results of FIND commands as comments in the script file when used with the SCRIPT operand. |
| SHadow | Specifies whether excluded lines are shown as a line of dashes with the number of excluded lines indicated at the right side of the dashed line. When this mode is OFF, there is no indication of excluded lines other than the sequence numbers shown in the line command area. The default is ON. |

| Operand | Description |
|---|---|
| STatus | Displays the current status box at the bottom of all Program View screens when this mode is ON. The content of the status box varies depending on the status of the ASM, FLOATING and REGISTERS operands. A status box is automatically displayed when an error condition occurs even if this mode is OFF. The STATUS operand must be ON to enable the FLOATING and REGISTERS operands. The default is ON.<br><br>**Note:**<br>Remote users using terminal emulators should turn the STATUS mode and the REFRESH mode OFF to avoid repainting the screen between each step. |
| STOPExec | Stops execution before each EXEC CICS request. When execution is stopped, the Status Box description indicates the type of EXEC CICS request that is about to be made (e.g., STOPPED BEFORE SEND MAP). Issuing the STEP or RUN command continues execution. SET STOPEXEC affects all monitored programs for the current user including those that are not analyzed into the AKR. This operand is only available in the CICS environment. |
| STOPHand | Stops execution after a CICS Handle condition has been raised. When execution is stopped, the Status Box description indicates the type of Handle condition that is being processed (e.g., `STOPPED AFTER HANDLE MAPFAIL`). Issuing the STEP or RUN command continues execution. SET STOPHAND affects all monitored programs for the current user including those that are not analyzed into the AKR. This operand is only available in the CICS environment. |
| Track number | Sets the number of execution tracking entries. The number of execution entries can be set from 0 to 9999 or 0K to 64K. The default is 1024. See the Execution Tracking section in online help.<br><br>**Note:**<br>The maximum allowable entries for the CICS environment is 9999. |
| Values | Controls the number of lines used to display each alphanumeric data item and each data item displayed in hexadecimal format within a Zoom Data or the Keep window. When the number of lines required to display a data item exceeds the number of lines specified by the VALUE operand, the window becomes vertically scrollable. The default is AUTO. |

| Operand | Description |
|---|---|
| WHens | Enables or disables the When Conditions facility for a test session. When this mode is ON, all When Conditions are active and checked unless deactivated on the When Conditions List screen, the Breakpoints List screen, or the Test Session Tailoring screen. When this mode is OFF, all When Conditions are ignored. All When Conditions are ignored regardless of the WHENS mode setting when the SET PSEUDO OFF command is used. The default is ON. |
| WRAP | Enables or disables a prompt when the tracking table is full. This allows you to save the tracking table before it is overlaid. |
| XMode | Specifies whether all lines are excluded from the screen before a primary command is executed. This mode affects all primary commands with these exceptions:<br>BRANCH RFIND WHEN<br>LOCATE SCROLL<br>The default is OFF. |
| Zerofill | Specifies whether leading zeros display on numeric data displays. This operand affects numeric data items displayed using the ZOOMDATA, KEEP, and DISPLAY commands. |
| ON | Enables the corresponding mode. |
| OFF | Disables the corresponding mode. |
| AUTO | Specifies that the number of window lines displayed automatically adjusts to the number of lines necessary to display data items within the window. This keyword is used with the SET DATA, SET KEEP, and SET VALUES operands. |
| *number* | Specifies the number of lines specified in the SET DATA, SET KEEP, or SET VALUES operands. The number of lines can be set from 1 to 99. |

## Usage Notes

The Options - Modes screen displays when the SET command is entered without operands. This screen shows the current setting for each SET command mode. The Options - Modes screen can be used to change the setting for any mode by entering the desired value in the SET field. Modes that have been changed from the default setting are highlighted on the Options - Modes screen.

The current setting for each Test Session option is saved between sessions with the exception of BREAK, PSEUDO, SCRIPT, and WHEN. Default values for these four options are restored when a new session is initiated.

Operands that accept ON or OFF are toggles. Specifying one of these operands changes its status from ON to OFF or from OFF to ON.

You can enable or disable Modes by selecting Options ▸ Modes.

### Examples

In this command, the current status box is not displayed unless an interrupt occurs:

```
SET STATUS OFF
```

To reset the current status box mode so the current status box displays, type this command:

```
SET STATUS
```

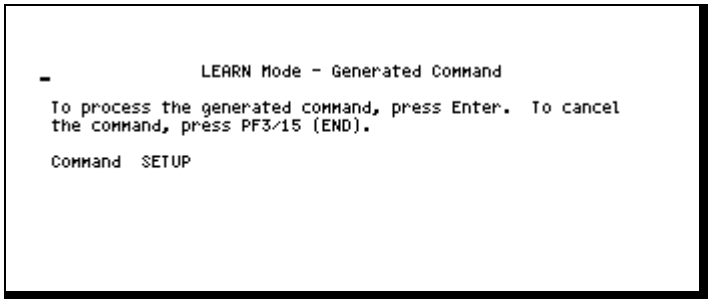
To alter the default value of the RUN command to NOMONITOR, type this command. This reduces CPU utilization for the test session, but disables other SmartTest features, such as COUNTS, ADSTOP, and TRACKING.

```
SET MONITOR OFF
```

## LEARN Mode - Generated Command Pop-up

The LEARN Mode - Generated Command pop-up (Figure 144) displays the internally-generated primary command when actions are requested on pop-ups, if the LEARN mode has been set to ON with the SET command or on the Options - Processing Modes pop-up.

**Figure 144 • LEARN Mode - Generated Command Pop-up**




Completing the instructions on the pop-up processes or cancels the command.

# SETUP Command

```
SETUP ──┬──────┬──────────────────────────────────────────────►◄
        └NOTCA─┘
```

## Function

The SETUP command is used to display the Session Setup screen for the currently selected environment.

## Operands

**NOTCA.** The NOTCA operand disables the TCA recorder, resets the TCA active indicator, and resets the session setup to non-TCA using the current execution environment setup.

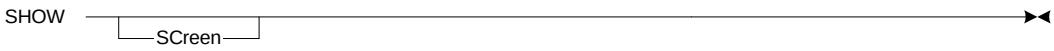
## Usage Notes

The SETUP command can be issued from any SmartTest screen. The Session Setup screen displayed corresponds to the environment that is currently selected. To change environments, use the ENVIRONMENT primary command (ENV).

You can also display the Session Setup screen by selecting File ▸ Setup Test Environment.

See the *ASG-SmartTest for COBOL and Assembler User's Guide* or the *ASG-SmartTest PLI User's Guide* for more information about the Session Setup screens.

# SHOW Command (CICS Only)




## Function

The SHOW command is used to redisplay the last user application screen.

## Operands

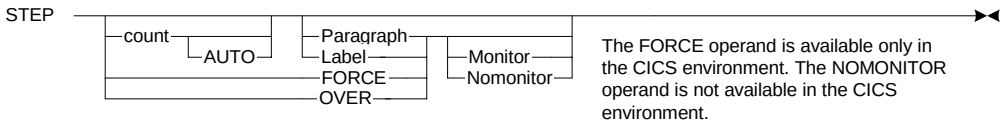
| Operand | Description |
|---|---|
| Blank | Causes the last user application screen to be redisplayed. |
| SCreen | Causes the last user application screen to be redisplayed. |

## Usage Notes

The SHOW command redisplays the last display-only screen. This command is only available during an active test session.

You can also view the last user application screen by selecting View ▸ Show.

# STEP Command (BACKTRACK OFF)




## Function

**Note:**

The STEP command operates differently if SET BACKTRACK is ON. See "STEP Command (BACKTRACK ON)" on page 425 for information about the STEP command with SET BACKTRACK ON.

The STEP command is used to begin a test of a program, to resume testing after an interrupt occurs, or to continue stepping through program execution after a STEP command has been entered. Interrupts can result from a Breakpoint or an error condition. To set the increment for the STEP command to the COBOL statement level, Assembler instruction level, or PL/I statement level, use the SET command. See the ASM operand of the SET command for more information.

**Note:**

If STEP is entered in the CICS environment when not in an active test session, the effect is the same as entering TOGGLE, which toggles the user into the CICS environment and allows an active test to be initiated.

## Operands

| Operand | Description |
|---|---|
| Blank | Causes the next COBOL statement, Assembler instruction, or PL/I statement to be executed. Execution pauses after the statement or instruction is executed. |
| *count* | Causes the specified number of COBOL or PL/I statements or Assembler instructions to be executed. Execution pauses after the specified number of statements or instructions are executed. The default is 1. |
| AUTO | Forces a display of each source statement before it is executed. The message `STEP` *n* `OF` *n* `EXECUTED` displays, indicating the current step number as the STEP AUTO command executes. |

| Operand | Description |
|---|---|
| Paragraph | Causes SmartTest to step at the paragraph level rather than at the statement level. STEP PARAGRAPH steps to the first statement of the next paragraph in the execution path. PARAGRAPH is synonymous with LABEL. |
| Label | Causes SmartTest to step at the LABEL. LABEL is synonymous with PARAGRAPH and is intended for use by PL/I and Assembler programmers. STEP LABEL steps to the first instruction following a LABEL point in the execution path. |
| FORCE | **1** Instructs the monitor to ignore the current storage violation warning and allow the current machine instruction to be executed. The Storage Protection screen should be used to permanently allow this instruction to execute. This operand is available only in the CICS environment.<br><br>**Note:**<br>Careful consideration should be given to which storage violations are to be overridden with this feature. This feature is intended to temporarily override a Storage Violation message for non-standard coding practices.<br><br>**2** Causes processing to continue on an invalid FREEMAIN error if the INVREQ condition has been handled or ignored. |

| Operand | Description |
|---|---|
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. The code contained within the performed or called routine is executed and the execution stops at the statement following the PERFORM or CALL. The status box message indicates `STOPPED BY STEP OVER REQUEST`. If a Breakpoint or program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER command is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement that follows the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test is encountered.<br><br>For COBOL source programs, the OVER operand functions if the current statement is a PERFORM statement or a CALL statement. If the current statement is not one of these statements, the operand is ignored.<br><br>For Assembler source programs and disassembled object code, the OVER operand functions if the machine instruction that corresponds to the current statement contains a BAL, BALR, BAS, BASR, BASSM, BCT, BCTR, BXH, or BXLE instruction. If the current statement does not contain one of these instructions, the operand is ignored.<br><br>For PL/I source programs, the OVER operand functions only if the current statement is a CALL statement. |
| Monitor | Indicates that all features are available for testing, including Address Stop Entry, Statement Counts, and Execution Tracking. STEP MONITOR can be used to override the SET MONITOR OFF mode when issuing the STEP command with the OVER, PARAGRAPH, or LABEL operands. |
| Nomonitor | Reduces the CPU overhead during a test session by disabling Address Stop Entry, Statement Counts and Execution Tracking. STEP NOMONITOR can be used to override the SET MONITOR ON mode when issuing the STEP command with the OVER, PARAGRAPH, or LABEL operands. This operand is not available in the CICS environment. |

## Usage Notes

The STEP command is used to begin execution of a program or to resume execution after an interrupt occurs. If there are no interrupts, the program executes the next statement or the next number of statements indicated by the COUNT operand. The RUN command can be used to execute the program to the next interrupt or to completion, whichever occurs first.

See the "RUN Command" on page 390 for more information.

One is the minimum value for the STEP command and is the default (if blank). 65,535 is the maximum value for the STEP command when the AUTO operand is not specified.

The maximum count value that can be specified with the AUTO operand is 999. The AUTO operand can impact performance if the count value is large.

In environments other than CICS, if the ATTENTION key is pressed after a test session has been started with the STEP *nnn* command, the remaining steps are not executed and a break occurs at the next COBOL verb.

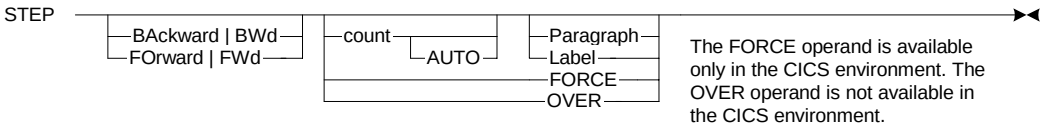
The OVER, PARAGRAPH, and LABEL operands set temporary breakpoints, then issue the RUN command.

When a storage violation occurs in the CICS environment, SmartTest stops the test and displays a status box at the bottom of the screen. Do not use the FORCE operand to continue execution without understanding the significance of the Storage Violation Warning.

For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*.

You can also step through a program by selecting Test ▸ Step.

# STEP Command (BACKTRACK ON)




## Function

The STEP command with SET BACKTRACK ON may be used to enter, exit and simulate execution backward and forward in the statement execution history. The Backtrack Recording facility must be active for this command to be valid (see the BACKTRACK option of the SET command). Stepping can be directed backward or forward one statement at a time or a maximum of 999 statements at a time. The AUTO operand allows viewing of each logical step as it occurs. Stepping at the paragraph or label level is also supported.

**Note:**

While executing in Backtrack Review mode, the Backtrack arrow (BKTR=>) points to the current position in the execution history.

## Operands

| Operand | Description |
|---|---|
| blank | Causes the next COBOL statement, Assembler instruction, or PL/I statement to be executed. Execution pauses after the statement or instruction is executed. |
| BAckward \| BWd | Sets the direction of execution for the Backtrack Review facility. Once set, the direction remains in effect until changed by a subsequent RUN or STEP command. If the Backtrack Review facility is not active, the BACKWARD operand automatically invokes it. If currently positioned at the top of the statement execution history, the BACKWARD operand generates an error. |
| Forward \| FWd | Sets the direction of execution for the Backtrack Review facility. Once set, the direction remains in effect until changed by a subsequent RUN or STEP command. If currently positioned at the Bottom of the statement execution history, the FORWARD operand causes you to exit the Backtrack Review facility, and positions the next instruction to be executed in normal test mode.<br><br>FORWARD is valid only in Backtrack Review mode, that is, after a RUN BACKWARD or STEP BACKWARD command. |

| Operand | Description |
|---|---|
| *count* | Causes the number of COBOL statements, Assembler instructions or PL/I statements to be executed (or paragraphs or labels, if the PARAGRAPH or LABEL operand is specified). Execution pauses after the specified number of statements or instructions are executed. The default is 1.<br><br>If the COUNT operand of the STEP command causes Backtrack to execute past the most current entry in the execution history, SmartTest pauses at the next instruction to be executed in normal test mode, and displays a message indicating the termination of BACKTRACK Review mode. |
| Auto | Forces a display of each source statement (or the first statement of each paragraph or label point in the execution path, if PARAGRAPH or LABEL is specified) before it is executed. The message `STEP` *`nn`* `OF` *`nn`* `EXECUTED` displays indicating the current step number as the STEP AUTO command executes. |
| Paragraph | Steps to the first statement to be executed after a Paragraph label in the execution path in the current direction. |

| Operand | Description |
|---|---|
| Label | Steps to the first statement to be executed after a label in the execution path in the current direction. LABEL is synonymous with PARAGRAPH, and is intended for use by PL/I or Assembler programmers. STEP LABEL steps to the first instruction to be executed after a label point in the current direction. |
| OVER | Steps through a program bypassing the display of any PERFORMed or CALLed routines that are not of interest. The code contained within the PERFORMed or CALLed routine is executed and the execution stops at the statement following the PERFORM or CALL. The status box message indicates `STOPPED BY STEP OVER REQUEST`. If a Breakpoint or program interruption occurs within the PERFORMed or CALLed code, the execution stops at that point and the STEP OVER command is terminated. If the PERFORMed or CALLed code terminates the program or does not return to the statement that follows the PERFORM or CALL, the program execution continues until a break pseudo code statement is reached, a program interruption occurs, or the end of the test is encountered.<br><br>For COBOL source programs, the OVER operand functions if the current statement is a PERFORM statement or a CALL statement. If the current statement is not one of these statements, the operand is ignored.<br><br>For Assembler source programs and disassembled object code, the OVER operand functions if the machine instruction that corresponds to the current statement contains a BAL, BALR, BAS, BASR, BASSM, BCT, BCTR, BXH, or BXLE instruction. If the current statement does not contain one of these instructions, the operand is ignored.<br><br>For PL/I source programs, the OVER operand functions only if the current statement is a CALL statement.<br><br>**Note:**<br>The NOMONITOR operand is not available when Backtrack Review mode or Backtrack Recording Mode is active. |

## Usage Notes

The BACKWARD operand of the STEP command issued during normal testing initiates the Backtrack Review facility, if it is not already active. SmartTest adds a status indicator to the Execution Status Box indicating that the Backtrack Review facility is active and showing your current execution direction. When the Backtrack Review facility is entered or exited, a message displays.

BACKTRACK restores data items to their historical values based on its position within the execution history. Any Zoomdata or Keep windows displayed data and operands data updates with these historical values automatically. Zoomdata, Keep, or Display commands issued during Backtrack Review mode reflect the values at the current position in the execution history.

The STEP command can be used to initiate or exit the Backtrack Review mode. The program simulates execution of the next statement or the next number of statements indicated by the COUNT operand. The RUN command can be used to simulate execution of the program to a specific line number or dataname modification.

See the "RUN Command" on page 390 for more information.

One is the minimum value for the STEP command and is the default (if blank). 65,535 is the maximum value for the STEP command when the AUTO operand is not specified.

The maximum count value that can be specified with the AUTO operand is 999. The AUTO operand can impact performance if the count value is large.

In environments other than CICS, if you press the ATTENTION key after a test session has been started with the STEP *nnn* command, the remaining steps are not executed and a break occurs at the next COBOL verb.

For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*.

You can also step through a program by selecting Test ▸ Step.

# STOP Command

```
STOP ──┬─dataname──┬──────────┬──┬──────────┬──────────────────────────────►◄
       │           └─subscripts─┘  └─LENgth n─┘
       └─address expression──────┘
```

## Function

The STOP command is used to set an address stop for a specified data item. Test execution is interrupted before each modification of the specified data item (when running MONITORed).

## Operands

| Operand | Description |
|---|---|
| Blank | Displays the Address Stop Entry screen. |
| *dataname* | Specifies a data item for which an address stop is set. |
| *subscripts* | Specifies subscript information using the standard COBOL syntax:<br>(*subscript1, subscript2, ... subscriptn*)<br>Left and right parentheses and at least one blank between subscripts are required. Commas between subscripts are optional. A subscript may be a numeric literal, an index data item, a numeric data item, or a numeric data item plus or minus a numeric literal. |
| *address expression* | Specifies that an address expression consists of an address or a register number followed by a maximum of 32 indirection indicators and/or offsets.<br>These are definitions of the possible components of an address expression. The address expression must not contain spaces. |
| Address | X'*nnnn*' An absolute address specified in hexadecimal notation. |
| Registers | R*n* A register 0 through 15.<br>*n*R A register 0 through 15. |
| Indirection indicators | % Indicates a 24-bit indirection.<br>? Indicates a 31-bit indirection. |

| Operand | Description | |
|---|---|---|
| Offsets | + | Indicates that the following offset is to be added. |
| | - | Indicates that the following offset is to be subtracted. |
| | X'*nn*' | An offset specified in hexadecimal notation. |
| | *nn* | An offset specified in decimal notation. |
| | R*n* | A register 0 through 15 that contains the offset. |
| | *n*R | A register 0 through 15 that contains the offset. |
| LENgth *n* | Sets an address stop for the data item specified to the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. | |

## Usage Notes

See the discussion on the Address Stop Entry screen in online help for more information about address stops.

When subscripts are entered for a dataname, the address of the data item is saved in the Address Stop Table. If the subscripts are subsequently changed, the Address Stop Table still contains the original address for the data item.

You can also set address stops by selecting Test ▸ Stop.

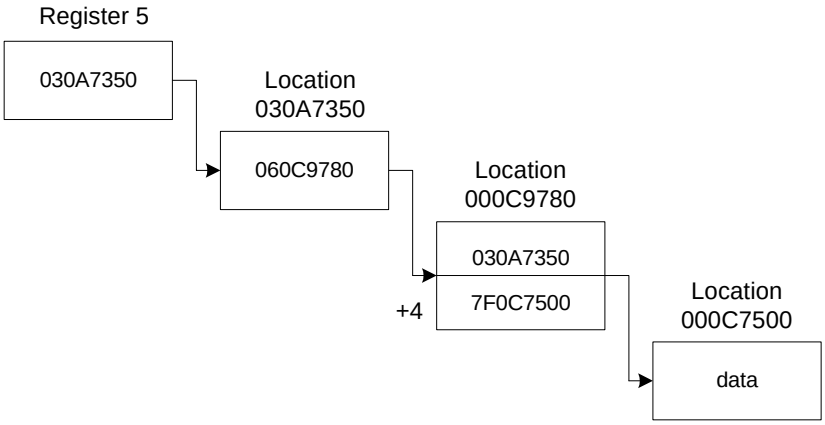
## Example

This command sets an address stop at the address expression specified:

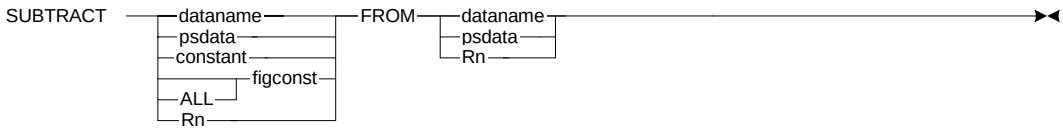
```
STOP R5?%+4% LEN 50
```

Figure 145 shows the above address expression graphically:

**Figure 145 • Graphic Illustration of Address Expression**



# SUBTRACT Command



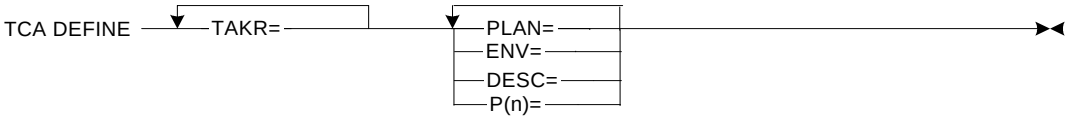

## Function

The SUBTRACT command subtracts the value contained in or represented by the first operand from the specified data item. The value is converted to the proper format for the data item.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, Assembler numeric data area, PL/I dataname, qualified PL/I dataname, or, PL/I variable. Dataname refers to any valid COBOL, PL/I, or Assembler reference for a data element. |
| *psdata* | Specifies a level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| *constant* | Specifies a numeric value. See "Using Pseudo Code" on page 512 for more information about coding pseudo code. |
| ALL | Specifies readability when used with a figurative constant. |
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to subtract specific values. The figurative constants ZERO, ZEROS, and ZEROES are supported and may be further qualified with the ALL figurative constant. |
| R*n* | Specifies a register 0 through 15. |
| FROM | Specifies a required keyword used as a connective for the SUBTRACT command operands.<br><br>**Note:**<br>You can also subtract values from a specific data item by selecting Test ▸ Subtract. |

# TCA DEFINE Command




## Function

The TCA DEFINE command is used to specify an existing AKR to be used as the TCA AKR, and to define a plan in a defined TCA-specific AKR.

## Operands

| Operand | Description |
|---|---|
| TAKR= | Defines the name of the TCA AKR. |
| PLAN= | Defines the name of the TCA plan. |
| ENV= | The test session environment. The valid values are 1 through 12 as shown:<br>1 - TSO<br>2 - CICS<br>3 - ISPF Dialog<br>4 - IMS/DC<br>5 - IMS/DB<br>6 - BTS<br>7 - DB2<br>8 - DB2 Procedure<br>9 - MVS Batch<br>10 - IMS Batch<br>11 - BTS Batch<br>12 - DB2 Batch |
| DESC= | Specifies the AKR description. If you do not specify a description, a default is used. |
| P{*n*}= | Specifies the load module CSECT name, where *n*= 1 through 50. |

## Usage Notes

You can enter a single command on multiple lines since continuation occurs when a command line ends in a comma. However, parameters cannot span lines.

# TCA LIST Command

## Function

The TCA LIST command opens a TCA plan. You can also export or delete the results in the plan.

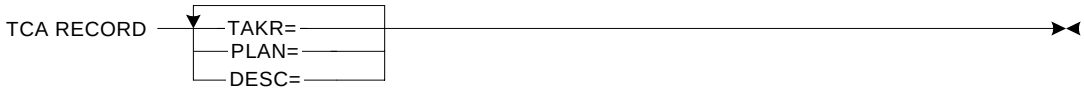
## Operands

| Operand | Description |
|---|---|
| PLAN= | Specifies the name of the TCA plan. |
| EXPORT= | Specifies the number of the TCA test result you want to export. SmartTest-TCA exports the results to a VIAPUNCH file. Select Options ▸ Log/List/Punch to begin the process of exporting a test result. |
| DELETE= | Specifies the number of the TCA test result you want to delete. |

## Usage Notes

You can enter a single command on multiple lines since continuation occurs when a command line ends in a comma. However, parameters cannot span lines.

# TCA RECORD Command




## Function

The TCA RECORD command records the execution coverage information into a TCA plan.

## Operands

| Operand | Description |
|---|---|
| TAKR= | Specifies the name of the TCA AKR. |
| PLAN= | Specifies the name of the TCA plan. |
| DESC= | Specifies the description of the TCA plan. |

## Usage Notes

You can enter a single command on multiple lines since continuation occurs when a command line ends in a comma. However, parameters cannot span lines.

# TCA REPORT Command

## Function

The TCA REPORT command is used to generate reports from the results in a TCA plan.

## Operands

| Operand | Description |
|---|---|
| PLAN= | Specifies the name of the TCA plan. |
| TYPE= | The type of report. The valid values are 1 through 5 as described blow. The default value is 1.<br>1. Detail Count and Program Summary Report<br>2. Detail Execution Report<br>3. Program/Paragraph Label Count Report<br>4. Program/Paragraph Label Execution Report<br>5. Executed/Not Executed Summary Report |
| RESULT= | Specifies the number of the results. You can specify 1-x or ALL. The default is ALL. |
| PROG= | Specifies the name of the program to be run. The default is ALL. |
| HARDCPY= | Generates a hardcopy report. The valid values are Y and N. |
| APPLY | Applies the tested and untested results to the AKR. |

## Usage Notes

You can enter a single command on multiple lines since continuation occurs when a command line ends in a comma. However, parameters cannot span lines.

# TCA RUN Command

## Function

The TCA RUN command executes a test session using the information stored in the TCA plan.

## Operands

| Operand | Description |
|---|---|
| PLAN= | Specifies the name of the TCA plan. |
| TYPE= | Specifies the result type. The valid values are COVERAGE and POINT. COVERAGE tests record the number of times each executable line of code is executed for all programs executed during the test session. POINT tests record whether each statement following a breakpoint is executed. The default is COVERAGE. |
| DESC= | Specifies the test description. If you do not specify a description, a default is used. |

## Usage Notes

You can enter a single command on multiple lines since continuation occurs when a command line ends in a comma. However, parameters cannot span lines.

# TEST Command

TEST ──────────────────────────────────────────────────►◄

## Function

The TEST command is used to display the Test Menu.

## Operands

None.

## Usage Notes

The TEST command can be issued from any SmartTest screen.

# TESTPOINT Command

```
TESTPOINT ──────────────────────────────────────────────────────────────────►◄
```

## Function

The TESTPOINT command activates the Test - SmartTest Testpoint Generation screen, which is used to specify necessary criteria when setting breakpoints with an impact dataset.

## Operands

None.

## Usage Notes

Impact datasets can be created by users or imported from Alliance, AutoChange, and Estimate. Breakpoints can be automatically set wherever specific data items are modified or referenced.

# TOGGLE Command

```
TOGgle ──────────────────────────────────────────────────────────►◄
```

## Function

The TOGGLE command is used to toggle from the SmartTest ISPF environment to the connected CICS or IMS environment.

## Operands

None.

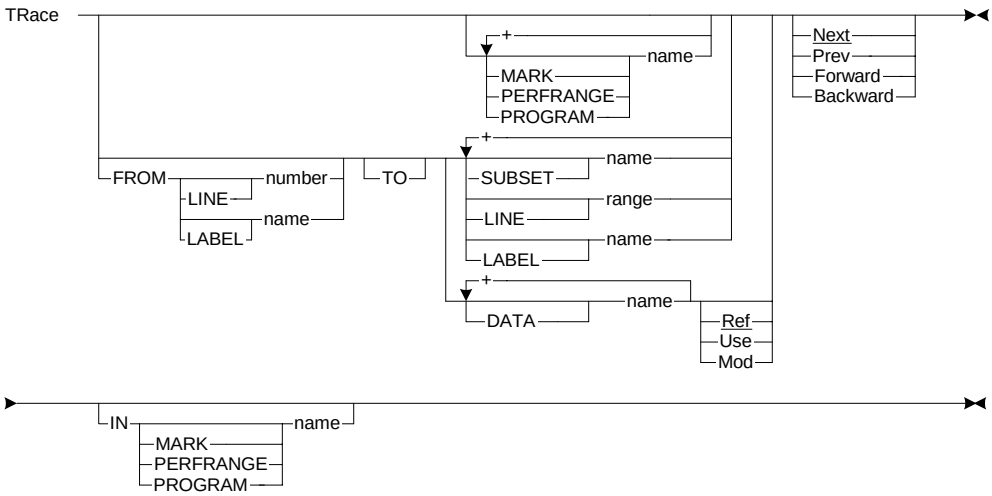
## Usage Notes

For the IMS environment, TOGGLE is valid between SmartTest and IMS at any time. If an IMS/DC test session has not been previously established, the TOGGLE command initiates one. If an IMS/DC test session is established, the TOGGLE command restores the previous IMS/DC screen and awaits user input. To return to ISPF, press the predefined toggle key (PF12) or enter the /TOGGLE command on the IMS screen.

For the CICS environment, TOGGLE is valid only between SmartTest test sessions when CICS is connected. Once the test session has been successfully toggled to CICS, any transaction may be initiated whether it is to be monitored by SmartTest or not. To return to ISPF, press the predefined toggle key (PF12) or enter the TOGGLE command on the CICS screen.

You can also toggle from the ISPF environment by selecting View ▸ Toggle.

# TRACE Command




ASG-Insight only

## Function

The TRACE command is used to follow the execution of a program, searching for the specified target. The MARK and PERFRANGE operands trace the path represented by the MARK or PERFRANGE name. Tracing to a SUBSET name, LINE range, LABEL name, or DATA name traces from the starting point to the lines represented by these targets.

This command is only available if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

## Operands

| Operand | Description |
|---|---|
| Blank | Specifies the direction of the trace. When TRACE is entered without an operand, the default direction of NEXT is used. The trace begins at the current cursor location and continues to the program exit. |
| FROM | Specifies an optional keyword entered with a line number or label to indicate where the TRACE command is to begin. The MARK name and PERFRANGE name operands cannot be used with this operand. |

| Operand | Description |
|---|---|
| LINE *number* | Specifies the line number where the TRACE command is to begin when the target is a SUBSET name, LINE range, LABEL name, or DATA name. |
| LABEL *name* | Specifies the paragraph, section, or division name where the TRACE command is to begin when the target is a SUBSET name, LINE range, LABEL name, or DATA name. |
| TO | Specifies an optional keyword that is followed by a target to indicate where the TRACE command is to end. |
| MARK *name* | Specifies a name (1 to 10 alphanumeric characters) given to a set or path using the COPY, MARK, MERGE, or RENAME commands, or one of these system-generated paths:<br>TRACK \| TRK — Created by a TRACE command.<br>NETWORK \| NET — Created by a FLOW command.<br>SUBNET*n* \| SUB*n* — A path created by the FLOW command that reaches from the beginning of the NETWORK to one of the results (nth result).<br>The TRACK, NETWORK and SUBNET paths can also be created using the Logic Order Search pop-ups. See the online help or the *ASG-Insight User's Guide* for more information. |
| PERFRANGE *name* | Specifies the name in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the Declaratives. |
| PROGRAM *name* | Specifies the name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. |

| Operand | Description |
|---|---|
| SUBSET *name* | Specifies one of these predefined COBOL language subsets:<br>ASsignment, DEFinition, IO<br>CAll, DIRective, LABel<br>CIcs, DIVision, MATH<br>COBOLII, DL/I \| DL/1, Output<br>COBOL/370, DML, PARagraph<br>COMment, ENtry, PERform<br>CONditional, EXIt \| PGMExit, SECtion<br>DB2 \| SQL, GOto, SORTMerge<br>DDL, IDMS, STructure<br>DEBug, Input, TESted \| UNTested<br>**Note:**<br>The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest TCA User's Guide* for more information.<br>A screen subset:<br>Highlighted \| HI<br>NONHighlighted \| NHI<br>Excluded \| X<br>NONExcluded \| NX<br>A tagged lines subset of tags appearing in columns 73 through 80.<br>See the online help, the *ASG-SmartTest for COBOL and Assembler User's Guide*, or the *ASG-SmartTest PLI User's Guide* for a description of each subset. |
| LINE *range* | Specifies a single line number or range of lines. Line numbers are displayed in columns 1 through 6. |
| LABEL *name* | Specifies a PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included. |
| DATA *name* | Specifies a COBOL dataname or qualified COBOL dataname. Dataname refers to any valid COBOL reference for a data element. These subordinate operands apply to the dataname and can be used to further qualify the TRACE command: REF, USE, and MOD. The default is REF. |

| Operand | Description |
|---|---|
| Ref | Includes occurrences of the dataname where its value is tested, used, set, or modified. This is the default for the DATA name operand. |
| Use | Includes occurrences of the dataname where its value is being tested or used. |
| Mod | Includes occurrences of the dataname where its value is being set or modified. |
| Next | Traces forward from the current cursor position to the next occurrence of the specified target. This is the default value for the TRACE command. |
| Prev | Traces backward from the current cursor position to the previous occurrence of the specified target. |
| Forward | Traces forward from the current cursor position to all occurrences. |
| Backward | Traces backward from the current cursor position to all occurrences. |
| IN | Restricts the TRACE command to the specified target type. To trace a path within a path, there must be a common starting point and at least one common ending point. |

## Usage Notes

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated dataname can be used wherever a dataname is valid. Dataname subordinate operands (REF, USE, MOD) pertain to all datanames in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. LABEL, LINE, and SUBSET can be concatenated; and MARK and PERFRANGE can be concatenated.

For COBOL II and later programs, a dataname, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

When TRACE is entered with only a direction operand:

- Next — Traces to the program exit statements.
- Previous — Traces to the PROCEDURE DIVISION statement or the program ENTRY points.
- Forward — Traces to the program exit statements.
- Backward — Traces to the PROCEDURE DIVISION statement or the program ENTRY points.

The trace function can be performed on an existing path (e.g., NETWORK, SUBn), or from a specific starting point to a target. The execution path followed during the trace function is automatically saved into a MARK path called TRACK. This path has a single entry and single exit, beginning with the line where a trace function starts and ending where the trace function is discontinued. TRACK is available until a new trace function is initiated.

Unconditional PERFORMs can be included in TRACK without following a path into the PERFORMed unit by using the BYPASS PERFORMED CODE decision option. This method shortens the trace function without a detailed trace of the PERFORMed unit.

Using the trace function is a convenient way to view paths in logical execution order. All executable statements on the path from the starting point to where the trace function stops are highlighted. A long message indicates the reason for stopping. For a decision point, the line number where the trace function stopped and the line number to continue with are shown. Columns 73 through 80 contain these information tags on a traced line:

| | |
|---|---|
| DECISION | for a decision point |
| STOP PNT | for a stopping point |
| START | for the beginning of the trace function |
| OPTION n | for an option choice |
| TARGET | for a destination of the trace function |

Multiple information tags appearing on a line are separated with a slash (/).

The trace function stops and redisplays the Program View screen when:

- A target has been reached.
- A decision point has been reached. All available options are presented in columns 73 through 80 and line three of the Program View screen.
- Before entering or exiting a PERFORMed unit (DECISION).
- At a GO TO statement that transfers to code not displayed on the screen (STOP PNT).
- At a GO TO or EXIT from PERFORMs in the forward direction, or a LABEL in the backward direction (STOP PNT).
- The bottom of the screen has been reached in the NEXT or FORWARD direction, or the top of the screen has been reached in the PREVIOUS or BACKWARD direction (STOP PNT).
- An apparent decision point has been reached but only one choice exists. This is shown as a stopping point rather than a decision point.

To continue the trace function, enter the desired option in the command input area and press Enter.

You can also trace the execution of a program by selecting an action on the Logic pull-down.

See the online help or the *ASG-Insight User's Guide* for more information about the Logic pull-down.

## Examples

To trace forward to the first EXIT in the program, type this command:

```
TRACE EXITS
```

To highlight the resulting MARK path, type this command:

```
FX TRACK
```

To trace forward through SUBNET1, which is a part of the NETWORK created by the FLOW command, type this command:

```
TRACE SUBNET1
```

# UPDATE Command

```
UPDate ──┬─CHANGE─┬─┬─Next─┬──────────────────────────────────►◄
         ├─FIND───┤ ├─Prev─┤
         └─RESET──┘ ├─FIrst─┤
                    ├─LAst─┤
                    └─All──┘
```

## Function

The UPDATE command is used to change pseudo code lines to actual COBOL source lines, making them part of the program. This command can only be entered while in the EDIT mode of the Update facility.

## Operands

| Operand | Description |
|---|---|
| blank | Specifies that the default operands of CHANGE and FIRST are used. |
| CHANGE | Updates all pseudo code lines without requesting a confirmation. This is the default for the UPDATE command. |
| FIND | Displays pseudo code as note lines in the program being edited, providing a means of viewing changes before actually making them. Note lines include three context lines indicating where the Update facility recommends the lines be placed. The lines to be changed follow the context lines. Press the RCHANGE PF key or execute the UPDATE command again without the FIND operand to insert the new code into the program. |
| RESET | Displays the Pseudo Code Update Facility screen that can be used to select a program in an AKR that contains changes to be updated. The AKR and program name(s) can be entered on the Pseudo Code Update Facility screen or just the AKR can be specified. When only the AKR is specified, the Pseudo Code Program Selection screen displays for selection of a program.<br><br>The first time the UPDATE command is executed, the Pseudo Code Update Facility screen displays. |
| Next | Displays or inserts the next block of pseudo code to be updated. |
| Prev | Displays or inserts the previous block of pseudo code to be updated. |
| FIrst | Displays or inserts the first block of pseudo code found in the selected program. This is the default for the UPDATE command. |

| Operand | Description |
|---|---|
| LAst | Displays or inserts the last block of pseudo code found in the selected program. |
| All | Displays or inserts all pseudo code in the selected program. |

## Usage Notes

The UPDATE command is used to update a source program with temporary COBOL statements created in a test session and then stored in an AKR. One use of the UPDATE command is to enter UPDATE ALL, select a program, and then press Enter (if sure of all changes). The Update facility opens the AKR, reads all update records, then applies the changes to the source program being edited. All inserted lines are highlighted.

UPDATE FIND FIRST can be entered to view the changes before they are actually made to the source program. The first block of pseudo code displays as note lines. RCHANGE can then be used to insert these lines as actual code, or RFIND can be used to locate the next block of pseudo code. Note lines are not saved as part of the source file when the Update facility is exited and can be removed from the screen by entering the ISPF RESET SPECIAL command.

You can also update a source program by selecting File ▸ Edit pseudo. Before you can use the UPDATE command, you must have saved your pseudo code in an AKR.

## Examples

To display the first block of pseudo code to be updated as note lines, type this command:

```
UPDATE FIND FIRST
```

To read all update records and updates the source with the changes, type this command:

```
UPDATE ALL
```

# USING Command

## Function

The USING command specifies which Register to use as a base for determining the address of fields within Assembler DSECTs. The ZOOMDATA command uses this information to display data fields that are DSECT relative.

**Note:**

Full-screen Assembler support is available only with the ASG-SmartTest-ASM option.

## Operands

| Operand | Description |
|---|---|
| *dsectname* | Specifies the DSECT name. |
| *n* | Specifies a register (0 through 15) for which the address is to be set. |
| R*n* | Specifies a register R0 through R15. |
| *n*X | Indicates the register address is 31-bit. |
| R*n*X | Specifies a 31-bit register R0X through R15X. |
| *dsectname,n* | Specifies the DSECT name and Register may be entered with a comma separating the two operands, rather than a space. |
| *dsectname*,R*n* | Specifies a register R0 through R15. |
| *dsectname*,*n*X | Indicates the register address is 31-bit. |
| *dsectname*,R*n*X | Specifies a 31-bit register R0X through R15X. |

## Usage Notes

USING sets addressability to an Assembler DSECT of dsectname. After this command is issued, all ZOOMDATA commands and pseudo code references to a field name in the DSECT specified uses Register *n* as a base for determining the field name's address.

If the optional *n*X or R*n*X is specified, the value in Register *n* is treated as a 31-bit address. If the X is omitted, SmartTest uses the current addressing mode of the active test session. The current AMODE displays in the test status box for Assembler programs.

The DROP command ends addressability to any Assembler DSECT currently being addressed.

You can also specify a register to be used as a base for fields within Assembler DSECTs by selecting View ▸ Using.

# UTILITY Command

UTILity | AKR

## Function

The UTILITY command is used to display the File - AKR Utility pop-up.

## Operands

None.

# VIEW Command

```
VIEW ──┬──────────────┬──────────────────────────────────────────►◄
       ├─pgm──────────┤
       ├─module.pgm───┤
       └─*────────────┘
```

## Function

The VIEW command is used to display the Program View screen for the currently qualified program.

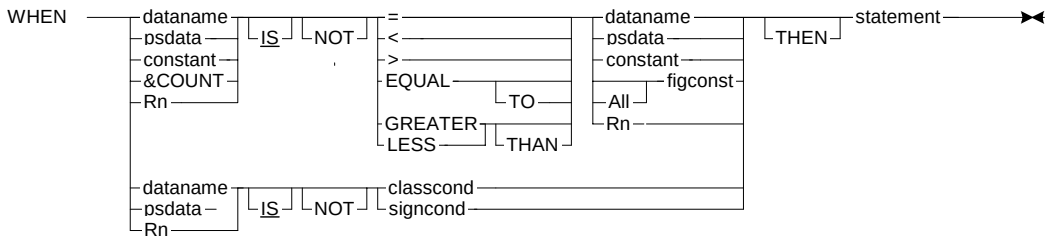
## Operands

| Operand | Description |
|---|---|
| *pgm* | Specifies the program that is to be displayed on the Program View screen if it resides in the current load module. |
| *module.pgm* | Specifies the *module.program* operand used to access a program that is not in memory. If the desired module resides in the current load library, a program or CSECT within the load module can be specified with the QUALIFY command. The specified module and program or CSECT is then displayed on the Program View screen. |
| asterisk (*) | Returns to (requalify) the active program. |

## Usage Notes

The VIEW command can be issued from any SmartTest screen.

# WHEN Command




## Function

The WHEN command is used to insert a WHEN pseudo code statement into the source program immediately preceding the bottom-of-data line. WHEN tests conditional expressions and if the condition is true, the imperative statement following THEN is executed. Testing of conditional expressions is performed the same as in COBOL conditional expressions. Use of the WHEN statement causes the test to be performed after the execution of every COBOL verb that modifies data, after every Assembler instruction for an Assembler source program, and after every PL/I statement for a PL/I program.

The WHEN command can be entered with or without operands. If the WHEN command is entered without operands, the WHEN statement is started in the command input area, with the cursor positioned to allow input of the operands.

These are the supported condition tests:

| Condition Test | Description |
|---|---|
| RELATION | Compares two operands to determine if they are equal, not equal, greater than, not greater than, less than, or not less than each other. |
| CLASS | Determines if a data item is alphabetic, not alphabetic, numeric, or not numeric. |
| SIGN | Determines if the value of a data item is negative, not negative, positive, or not positive. |

## Operands

| Operand | Description |
|---|---|
| blank | Places the user in input mode. |
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable, qualified PL/I variable, or Assembler data item. |
| *psdata* | Specifies a pseudo code dataname. |
| *constant* | Specifies a COBOL numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about coding pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. This operand can be used to execute a statement only once as in WHEN &COUNT=1 THEN BREAK. |
| R*n* | Specifies a register 0 through 15. |
| IS | Specifies the COBOL reserved word IS. This operand is included in the WHEN command by default. |
| NOT | The COBOL reserved word NOT. This operand is used to test a not condition. |
| Equals (=) | Makes an equal or not equal comparison. |
| Less than (<) | Makes a less than or not less than comparison. |
| Greater than (>) | Makes a greater than or not greater than comparison. |
| EQUAL | Makes an equal or not equal comparison. |
| TO | Specifies the COBOL reserved word used with the EQUAL operand to perform an equal to or not equal to comparison. |
| GREATER | Makes a greater than or not greater than comparison. |
| LESS | Makes a less than or not less than comparison. |
| THAN | Specifies the COBOL reserved word used with the GREATER and LESS operands to perform greater than, not greater than, less than, or not less than comparisons. |
| ALL | Specifies readability when used with a figurative constant. |

| Operand | Description |
|---|---|
| *figconst* | Specifies the COBOL reserved words (figurative constants) that are used to test specific values. These figurative constants, which may be further qualified with the ALL figurative constant, are supported:<br>HIGH-VALUE LOW-VALUES ZERO<br>HIGH-VALUES SPACE ZEROS<br>LOW-VALUE SPACES ZEROES |
| *classcond* | Specifies the COBOL reserved words that are used to test class conditions. The class conditions that are supported are ALPHABETIC and NUMERIC. |
| *signcond* | Specifies the COBOL reserved words that are used to test sign conditions. The sign conditions that are supported are NEGATIVE and POSITIVE. |
| THEN statement | Specifies a pseudo code statement to be executed if the conditional test is true. |

## Usage Notes

The WHEN command evaluates the conditional expression to determine whether the condition is true or false. If the conditional expression is true, the imperative statement that follows (THEN statement) is executed.

See the "STOP Command" on page 429 for a more efficient way to determine when a field is modified.

These commands and pseudo code statements cannot be specified as operands in a WHEN command:

77 level data definition
Procedure name
GO TO
WHEN

## Examples

In this command, an interrupt occurs when the value of ZIP-CODE equals 85018:

```
WHEN ZIP-CODE IS EQUAL '85018' THEN
BREAK.
```

In this command, an interrupt occurs when the value of ZIP-CODE does not equal 85018:

```
WHEN ZIP-CODE NOT EQUAL '85018' THEN
BREAK.
```

In this command, the value in SUB-TOT is moved to the TOTAL field when the &COUNT system variable is equal to one:

```
WHEN &COUNT=1 THEN
MOVE SUB-TOT TO TOTAL.
```

In this command, an interrupt occurs when the value in INPUT-REC is alphabetic:

```
WHEN INPUT-REC ALPHABETIC THEN
BREAK.
```

# WHERE Command

```
WHERe ──────── address expression ────────────────────────────────►◄
```

## Function

The WHERE command is used to identify a storage location. A message displays that indicates the offset and the area of the program where the address is located.

## Operands

| Operand | Description | |
|---|---|---|
| address expression | Consists of an address or a register number followed by a maximum of 32 indirection indicators and/or offsets.<br>These are definitions of the possible components of an address expression. The address expression must not contain spaces. | |
| Address: | X'*nnnn*' | An absolute address specified in hexadecimal notation. |
| Registers: | R*n* | A register 0 through 15. |
| | *n*R | A register 0 through 15. |
| Indirection indicators: | % | Indicates a 24-bit indirection. |
| | ? | Indicates a 31-bit indirection. |
| Offsets: | + | Indicates that the following offset is to be added. |
| | - | Indicates that the following offset is to be subtracted. |
| | X'*nn*' | An offset specified in hexadecimal notation. |
| | *nn* | An offset specified in decimal notation. |
| | R*n* | A register 0 through 15 that contains the offset. |
| | *n*R | A register 0 through 15 that contains the offset. |

## Usage Notes

The WHERE command locates an offset in the program being viewed and/or tested on the Program View screen.

You can also identify a storage location by selecting Test ▸ Where.

## Examples

To display a message indicating the area of the program and the offset of the address 00094898, type this command:

```
WHERE 94898
```

A message similar to this displays:

```
ADDRESS '00094898' IS AREA 'WKS' PLUS OFFSET 'D0'
```

The above message indicates the address is located at offset D0 within WORKING-STORAGE.

To display a message indicating the address, area of the program, and the offset of the register R5, type this command:

```
WHERE R5
```

A message similar to this displays:

```
ADDRESS '000B4004' IS AREA 'VIAMERGE.VIAMERGE' PLUS OFFSET '2A24'
```

# WIZARD Command

WIZard

## Function

Executes the Setup Wizard that displays the Test - Setup Wizard pop-up to assist in setting up a test session in programs in TSO foreground or using batch connect, IMS, or CICS environments.

## Operands

None.

## Usage Notes

You can use the WIZARD command from any SmartTest screen, or by selecting Test ▸ Setup Wizard. The Test - Setup Wizards (Environment Selection) screen, shown in Figure 146, displays. The instructions on each screen will guide you through the setup process.

**Figure 146 • Test - Setup-Wizards Screen**

```
              Test - Setup Wizards (Environment Selection)

     Please select the environment you wish to test in.

__   1.  CICS  - Set up test to run an online CICS program
     2.  IMS   - Set up test to run an online IMS program
     3.  BATCH - Set up test to run a Batch program

     Use the CICS Environment option to interactively test and debug
     online CICS programs.

     Use the IMS Environment option to interactively test and debug
     online IMS/DC programs.

     Use the Batch Environment option to interactively test and debug
     batch programs in TSO Foreground or using Batch Connect.  Batch
     programs can be: MVS Batch, DB2 Batch, IMS Batch, BTS Batch
     or a combination thereof.

      Press ENTER to proceed, Press PF3 to terminate the Wizard.
```

# ZOOMDATA Command

## Function

The ZOOMDATA command is used to scroll to the definition of the specified dataname, and display the value and address of the dataname.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a data item for which the value and address is to be displayed. |
| *subscripts* | Specifies subscript information using the standard COBOL syntax:<br>(*subscript1, subscript2, ... subscriptn*)<br>Left and right parentheses and at least one blank between subscripts are required. Commas between subscripts are optional. A subscript may be a numeric literal, an index data item, a numeric data item, or a numeric data item plus or minus a numeric literal. |
| address expression | Consists of an address or a register number followed by a maximum of 32 indirection indicators and/or offsets.<br>These are definitions of the possible components of an address expression. The address expression must not contain spaces. |
| Address: | X'*nnnn*' An absolute address specified in hexadecimal notation. |
| Registers: | R*n* A register 0 through 15.<br>*n*R A register 0 through 15. |
| Indirection indicators: | % Indicates a 24-bit indirection.<br>? Indicates a 31-bit indirection. |

| Operand | Description | |
|---|---|---|
| Offsets: | + | Indicates that the following offset is to be added. |
| | - | Indicates that the following offset is to be subtracted. |
| | X'*nn*' | An offset specified in hexadecimal notation. |
| | *nn* | An offset specified in decimal notation. |
| | R*n* | A register 0 through 15 that contains the offset. |
| | *n*R | A register 0 through 15 that contains the offset. |
| GROUP | Displays the levels, values, and addresses of the group items associated with the specified dataname. | |
| HEX | Displays the value and address of the specified dataname in hexadecimal format. | |
| CHAR | Displays the value for the specified data item in character format. This parameter is ignored for operands other than dataname. | |
| LENgth *n* | Displays the value of the data item specified for the length specified. The default length for the symbolic data items is the defined length of the data item. For address expressions, the default length is 4. | |

## Usage Notes

The results of the ZOOMDATA command are displayed in a scrollable window on the screen. See the DATA operand of the SET command for information about adjusting the size of the scrollable windows. Zoom windows can be removed from the screen using the ZO (Zoom Out) or X (Exclude) line commands. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

The ZOOMDATA command can be entered using these abbreviations:

| | |
|---|---|
| ZG `dataname` | Zoomdata Group |
| ZH `dataname` | Zoomdata Hex |
| ZGH `dataname` | Zoomdata Group Hex |

The value area of a ZOOMDATA line contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed unless the SET ZEROFILL mode is ON. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

An effective picture clause displays on the dataname line. For example:

A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
An alphanumeric field with PIC XXXX displays as X(4).
A field defined as COMP displays as COMP-4.
A field with a long dataname that is defined as COMP displays as C3 or C4.

For Assembler programs, the USING command can be used to specify which Register to use as a base for determining the address of fields that are DSECT relative.

See the "USING Command" on page 448 for more information.

When the ZOOMDATA command is used to display a numeric data item that contains invalid numeric data, the contents of the field are displayed as character data [with non-displayable characters displayed as periods (.)], and the highlighted text *INVALID NUMERIC* displays to the right of the field.

You can also display the value and address of a data item by selecting View ▸ ZoomData.

## Examples

To display the definition of COMPARISON-KEYS in the DATA DIVISION along with the value and address, type this command:

```
ZOOMDATA COMPARISON-KEYS
```

To display the definition of COMPARISON-KEYS in the DATA DIVISION, including the group levels, values, and addresses in hexadecimal format, type this command:

```
ZGH COMPARISON-KEYS
```

To display data at the address expression specified, type this command:

```
ZOOMDATA R5?%+4% CHAR LEN 50
```

# ZOOMIN/ZOOMOUT Commands

```
ZOOMIn | ZI   ──────────────────────────────────────────────────────────────────────────────►◄
ZOOMOut | ZO  ──────────────────────────────────────────────────────────────────────────────►◄
```

## Function

The ZOOMIN and ZOOMOUT commands are used to display or exclude source lines according to the hierarchical levels of the program.

## Operands

None. These commands are entered in the command input area with no operands and are dependent on the cursor location.

## Usage Notes

The ZOOMIN and ZOOMOUT commands show the structure of a program and provide a means of stepping through each level or going directly into or out of a particular section of source code.

PROCEDURE DIVISION hierarchy consists of section labels, paragraph labels, and paragraph code. DATA DIVISION hierarchy consists of sections, FDs, 01 or 77 levels, and all definitions within an 01 level.

For example, begin by excluding all lines from the screen, then type `ZOOMIN`. The DIVISION headings are shown. Type `ZOOMIN` in the command input area and place the cursor on the PROCEDURE DIVISION statement and press Enter. (Or, place the cursor on the PROCEDURE DIVISION statement and press PF7/19.) All PROCEDURE DIVISION headings are displayed. To see the source within a particular heading, scroll to it and enter another ZOOMIN command. The paragraph names are displayed. Continue this process to see the code and copy statements within several paragraphs. Enter ZOOMOUT on a paragraph name to exclude it from the screen once it has been viewed. All source lines under the paragraph name are excluded from the screen.

For COBOL II with nested programs, excluding all lines from the screen and entering ZOOMIN shows the IDENTIFICATION DIVISION, PROGRAM-ID, and END-PROGRAM headings for the main program and all subprograms.

These are the general guidelines for ZOOMIN:

- If the cursor is on an excluded block of lines, ZOOMIN displays the highest structure level within that block of lines. If there are multiple statements at that level, they are also displayed.
- If ZOOMIN is issued repeatedly from the same cursor position, successive heading levels are displayed.

These are the general guides for ZOOMOUT:

- The cursor cannot be positioned on an excluded line when ZOOMOUT is entered.
- ZOOMOUT excludes the levels in a lowest to highest order. If ZOOMOUT is issued repeatedly from the same cursor location, successive heading levels are excluded.

Zoom in and Zoom out are also available on the View pull-down.

# SmartTest Line Commands

Line commands are entered over the line numbers in the prefix area on the screen. Line commands are either single format or block format. Single format refers to a line command being entered on an individual line. Figure 147 shows the line command to exclude line 332.

**Figure 147 • Single Format Line Command Example**

```
000331       MOVE ZIP-CODE TO HLD-ZIP.
X00332      IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333           NEXT SENTENCE
```

A number can be included with some single format commands to indicate a specified number of lines that are to be processed using the same command. Processing begins with the line on which the command is entered and includes the current line and subsequent lines as indicated by the number specified. If the specified number of lines exceeds the available source lines, all remaining lines are processed. 99999 is the maximum that can be entered. Figure 148 shows the single format X (Exclude) line command followed by a number to exclude lines 332-336.

**Figure 148 • Single Format Line Command With a Number Specified**

```
000331      MOVE ZIP-CODE TO HLD-ZIP.
X50332     IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333          NEXT SENTENCE
000334        ELSE
000335          PERFORM P150-SUBTOT
000336              THRU P169-EXIT
000337        MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

Block format refers to double character line commands that are entered on multiple lines. Block line commands are processed for all lines between (and including) the lines containing the commands. Figure 149 shows the block format XX (Exclude Block) command being used to exclude lines 332-336.

**Figure 149 • Block Format Line Command Example**

```
000331      MOVE ZIP-CODE TO HLD-ZIP.
XX0332     IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333          NEXT SENTENCE
000334        ELSE
000335          PERFORM P150-SUBTOT
XX0336              THRU P169-EXIT
000337        MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

Multiple line commands can be entered and are processed based on this hierarchy:

### Group 1

. (Label)
F (First)
H (Highlight)
HH (Highlight Block)
L (Last)
S (Show)
SS (Show Block)

### Group 2

BR (Break)
I (Insert)
X (Exclude)
XX (Exclude Block)

### Group 3

GO
KH (Keep Hexadecimal)
KG (Keep Group)
KGH (Keep Group Hexadecimal)
ZA (Zoom Assembler)
ZD (Zoom Data)
ZG (Zoom Group)
ZGH (Zoom Group Hexadecimal)
ZH (Zoom Hexadecimal)
ZI (Zoom In)
ZO (Zoom Out)

### Group 4

D (Delete)
DD (Delete Block)
R (Repeat)
RR (Repeat Block)

### Group 5

C (Copy)
CC (Copy Block)
M (Move)
MM (Move Block)

When multiple line commands are processed, those in Group 1 are processed first (sequentially within the group), followed by Group 2 commands, and so on. An invalid line command stops the processing of all remaining commands. If an error is encountered in a line command, a message displays in the short message area of the screen. The line command in error is highlighted and remains displayed along with any other unprocessed commands.

Line commands can be removed by typing over them with spaces, pressing ERASE EOF with the cursor at the beginning of the command, or entering the RESET primary command in the command input area.

Line commands can be ambiguous when entered. Figure 150 shows a common ambiguity.

**Figure 150 • Ambiguous Line Command Example**

```
024500       MOVE ZIP-CODE TO HLD-ZIP.
X24600      IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
024700           NEXT SENTENCE
```

It is unclear whether two lines or 246 lines are to be excluded. If two lines are to be excluded, one or more spaces should be entered following the X2. The ERASE EOF key could be used to erase the numbers following the X2.

# . (Label) Line Command

. label

## Function

The . (Label) line command is used to assign a name to a particular source line.

This command is not valid on pseudo code lines.

## Operands

**.label.** A period followed by 1 to 5 alphabetic characters to be used as the name for the line on which it is entered.

## Usage Notes

The LOCATE primary command followed by a label name can be used to scroll to the requested line.

The . (Label) line command assigns an EQUATE to the line number. The RESET LABEL command resets all .label definitions.

## Examples

Figure 151 shows the Label line command being used to label line 241.

**Figure 151 • .(Label) Line Command Example**

```
.b0241     MOVE ZIP-CODE TO HLD-ZIP.
000242    IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000243         NEXT SENTENCE
```

Line 241 can now be referenced as .b and as line 241.

# A (After) Line Command

A

## Function

The A (After) line command is used to indicate the line after which the results of an operation such as copy or move are to be placed.

## Operands

None.

## Usage Notes

The A (After) line command is used to indicate the line after which results are to be placed. Results can be from a copy or move operation.

## Examples

Figure 152 and Figure 153 show the A (After) line command being used with the C (Copy) line command. Pseudo code line 2 is being copied after line 286.

**Figure 152 • A (After) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
C....2         BREAK.
000284
000285    PERFORM P100-PRINT
A00286           THRU P119-EXIT.
000287
```

**Figure 153 • A (After) Line Command Results Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2         BREAK.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
.....1         BREAK.
000287
```

# B (Before) Line Command

B

## Function

The B (Before) line command is used to indicate the line before which the results of an operation such as copy or move are to be placed.

## Operands

None.

## Usage Notes

The B (Before) line command is used to indicate the line before which results are to be placed. Results can be from a copy or move operation.

## Examples

Figure 154 and Figure 155 show the B (Before) line command being used with the C (Copy) line command. Pseudo code line 2 is being copied before line 287.

**Figure 154 • B (Before) Line Command Example**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                   MASTER-END-OF-FILE
000283                   MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
C....2        BREAK.
000284
000285    PERFORM P100-PRINT
000286          THRU P119-EXIT.
B00287
```

**Figure 155 • B (Before) Line Command Results**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                   MASTER-END-OF-FILE
000283                   MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2        BREAK.
000284
000285    PERFORM P100-PRINT
000286          THRU P119-EXIT.
.....1        BREAK.
000287
```

# BR (Break) Line Command

BR

## Function

The BR (Break) line command is used to enter a Breakpoint before a specific statement. Program execution stops at the Breakpoint during a test session.

## Operands

None.

## Usage Notes

The BR (Break) line command is used to insert a Breakpoint in a program during a test session. An unconditional interrupt occurs when the Break is encountered during program execution.

When the BR (Break) line command is entered on a line that contains a paragraph name, the BREAK statement is inserted after the paragraph name, before the first executable statement in the paragraph.

When the BR (Break) line command is entered on a line that is not executable, the BREAK statement is inserted before the next executable statement in the program. The screen is scrolled to the inserted BREAK statement and a warning message displays.

## Examples

In Figure 156 and Figure 157, the BR (Break) line command is used to enter a Breakpoint before line 284.

**Figure 156 • BR (Break) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
BR0284
000285     PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

**Figure 157 • BR (Break) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1     BREAK.
000284
000285     PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

# C (Copy) Line Command

C*n*

## Function

The C (Copy) line command is used to copy a pseudo code line or group of pseudo code lines. This command is used with the A (After) and B (Before) line commands to indicate where to copy the line(s).

## Operands

**n.** The number of consecutive lines to copy. The default is 1. If the number specified is greater than the number of available lines, all remaining lines are copied.

## Usage Notes

COBOL source statements that are copied become pseudo code lines.

This command cannot be used on Assembler or disassembled object statements.

## Examples

Figure 158 and Figure 159 show pseudo code line 2 being copied after line 286.

**Figure 158 • C (Copy) Line Command Example**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                    MASTER-END-OF-FILE
000283                    MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
C....2        BREAK.
000284
000285    PERFORM P100-PRINT
A00286          THRU P119-EXIT.
000287
```

**Figure 159 • C (Copy) Line Command Results**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                    MASTER-END-OF-FILE
000283                    MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2        BREAK.
000284
000285    PERFORM P100-PRINT
000286          THRU P119-EXIT.
.....1        BREAK.
000287
```

# CC (Copy Block) Line Command

CC

## Function

The CC (Copy Block) line command is used to copy a block of pseudo code lines. This command is used with the A (After) and B (Before) line commands to indicate where to copy the block of lines.

## Operands

None.

## Usage Notes

The CC (Copy Block) line command is entered in the prefix area on the first and last line of the block to be copied. COBOL source statements can be included in the range and become pseudo code lines.

This command cannot be used on Assembler or disassembled object statements.

## Examples

Figure 160 and Figure 161 show pseudo code lines 1 and 2 being copied after line 286. The CLIENT-ID number was changed after the copy operation to be valid for that conditional test.

**Figure 160 • CC (Copy Block) Line Command Example**

```
000281     CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
CC...1     IF CLIENT-ID = '387962' THEN
CC...2          BREAK.
000284
000285     PERFORM P100-PRINT
A00286               THRU P119-EXIT.
000287
```

**Figure 161 • CC (Copy Block) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2         BREAK.
000284
000285    PERFORM P100-PRINT
000286          THRU P119-EXIT.                .....1
IF CLIENT-ID = '395298' THEN
.....2         BREAK.
000287
```

# D (Delete) Line Command

D*n*

## Function

The D (Delete) line command is used to delete a pseudo code or Keep line or group of pseudo code or Keep lines.

## Operands

**n.** The number of consecutive lines to delete. The default is 1. If the number specified is greater than the number of available lines, all remaining lines are deleted.

## Usage Notes

The range of the command must be contained within a contiguous block of pseudo code lines. There cannot be any source lines within the range of the D (Delete) line command.

## Examples

Figure 162 and Figure 163 show the D (Delete) line command being used to delete pseudo code lines 1 and 2.

**Figure 162 • D (Delete) Line Command Example**

```
000281     CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
D2...1     IF CLIENT-ID = '387962' THEN
.....2          BREAK.
000284
000285     PERFORM P100-PRINT
000286             THRU P119-EXIT.
000287
```

**Figure 163 • D (Delete) Line Command Results**

```
000281     CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
000284
000285     PERFORM P100-PRINT
000286              THRU P119-EXIT.
000287
```

# DD (Delete Block) Line Command

DD

## Function

The DD (Delete Block) line command is used to delete a block of pseudo code lines or a block of Keep lines.

## Operands

None.

## Usage Notes

The DD (Delete Block) line command is entered in the prefix area on the first and last line of the block to be deleted. The range of the command must be contained within a contiguous block of pseudo code lines. There cannot be any source lines within the range of the DD (Delete Block) line command.

## Examples

Figure 164 and Figure 165 show the DD (Delete Block) line command being used to delete pseudo code lines 1 and 2.

**Figure 164 • DD (Delete Block) Line Command Example**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                       MASTER-END-OF-FILE
000283                       MASTER-REPORT-DATE.
DD...1    IF CLIENT-ID = '387962' THEN
DD...2        BREAK.
000284
000285    PERFORM P100-PRINT
000286            THRU P119-EXIT.
000287
```

**Figure 165 • DD (Delete Block) Line Command Results**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                        MASTER-END-OF-FILE
000283                        MASTER-REPORT-DATE.
000284
000285    PERFORM P100-PRINT
000286            THRU P119-EXIT.
000287
```

# F (First) Line Command

F*n*

## Function

The F (First) line command redisplays the specified number of excluded lines. These lines are redisplayed starting with the first line in the block of excluded lines.

## Operands

**n.** The number of excluded lines to redisplay. The default is 1. If the number specified is greater than the number of excluded lines, all lines in the excluded block are redisplayed.

## Usage Notes

The F (First) line command redisplays lines excluded as the result of any primary or line command that excludes lines from the screen display.

## Examples

Figure 166 and Figure 167 show the first two lines in the excluded block being redisplayed.

**Figure 166 • F (First) Line Command Example**

```
000002   PROGRAM-ID. PRDEMO.
F2      -  -  -  -  -  -  -  -  -  -  5 LINE(S) NOT DISPLAYED
000008   INPUT-OUTPUT SECTION.
```

**Figure 167 • F (First) Line Command Results**

```
000002   PROGRAM-ID. PRDEMO.
000003   AUTHOR. WRITTEN BY ASG AT LANGLVL 2.
000004   ENVIRONMENT DIVISION.
-  -  -  -  -  -  -  -  -  -  -  -  -  3 LINE(S) NOT DISPLAYED
000008   INPUT-OUTPUT SECTION.
```

# GO Line Command

GO

## Function

The GO line command is used to make the specified line the next line to be executed during a test session.

## Operands

None.

## Usage Notes

The GO line command is only valid during a SmartTest test session. ASG recommends that the GO line command be used only within the paragraph currently being executed. Specifying the GO line command on a statement that is not within the current paragraph can cause unpredictable results.

The GO line command can only be used on source lines that are executable. If the GO line command is entered on a non-executable line, an error message displays.

If a GO line command is placed on a source statement that has pseudo code, then the pseudo code associated with that source statement re-executes before the source statement unless the GO line command was entered during a break within that pseudo code block.

The GO line command cannot be entered on pseudo code lines.

## Examples

Figure 168 shows the GO line command being used to specify that line 196 is to be executed next. An interrupt has occurred on statement 1122. The status box indicates the interrupt was caused by use of the STEP command. The GO line command is entered on line 1127.

**Figure 168 • GO Line Command Example**

```
  File  View  Test  Search  List  Options  Help
-----------------------------------------------------------------------------
                                Program View                VIAPCOB.VIAPCOB -A
Command===>_____________________________________________Scroll===>CSR
001121  PARM-EDIT.
>>>>>>      IF CURRENT-PARM-NUMBER = 0 THEN
001123         PERFORM BUILD-PADDED-PARM
001124         IF PADDED-PARM = 'ALL' THEN
001125            MOVE PADDED-PARM TO FIRST-PARM-TEXT
001126         ELSE
G01127            PERFORM FIND-FIRST-PARAMETER
001128                    VARYING SUB1 FROM 1 BY 1
001129                    UNTIL SUB1 > VALID-PARMS-COUNT.
001130
001131      IF FIRST-PARM-TEXT = 'ALL' THEN                           RETURN
001132         MOVE CURRENT-PARM-NUMBER TO SUB1
001133         MOVE +0 TO CURRENT-PARM-NUMBER
001134         PERFORM FIND-NEXT-PARAMETER
+---------------------------------------------------------------------------+
|STATUS: STOPPED BY STEP REQUEST      PROGRAM: VIAPCOB   DATE: DDMMMYYYY |
|  STMT: 001122 OFF: 001D0C AMODE:24   MODULE: VIAPCOB   TIME: 09:17:17  |
|SOURCE: IF CURRENT-PARM-NUMBER = 0 THEN                                |
+---------------------------------------------------------------------------+
```

Figure 169 shows the result of the GO command. The status box is updated and indicates that statement 1127, PERFORM FINE-FIRST-PARAMETER, is to be executed next. The status box indicates the interrupt was caused by use of the GO command. Statement 1127 executes when the RUN or STEP command is entered to continue program execution.

**Figure 169 • GO Line Command Results**

```
  File  View  Test  Search  List  Options  Help
-----------------------------------------------------------------------------
                                Program View                VIAPCOB.VIAPCOB -A
Command ===> ___________________________________________ Scroll ===> CSR

001121  PARM-EDIT.
001122      IF CURRENT-PARM-NUMBER = 0 THEN
001123         PERFORM BUILD-PADDED-PARM
001124         IF PADDED-PARM = 'ALL' THEN
001125            MOVE PADDED-PARM TO FIRST-PARM-TEXT
001126         ELSE
>>>>>>            PERFORM FIND-FIRST-PARAMETER
001128                    VARYING SUB1 FROM 1 BY 1
001129                    UNTIL SUB1 > VALID-PARMS-COUNT.
001130
001131      IF FIRST-PARM-TEXT = 'ALL' THEN                           RETURN
001132         MOVE CURRENT-PARM-NUMBER TO SUB1
001133         MOVE +0 TO CURRENT-PARM-NUMBER
001134         PERFORM FIND-NEXT-PARAMETER
+---------------------------------------------------------------------------+
|STATUS: STOPPED AT "GOTO" LOCATION    PROGRAM: VIAPCOB   DATE: DDMMMYYYY|
|  STMT: 001127 OFF: 001D5E AMODE: 24    MODULE: VIAPCOB   TIME: 09:40:58|
|SOURCE: PERFORM FIND-FIRST-PARAMETER                                   |
+---------------------------------------------------------------------------+
```

# H (Highlight) Line Command

H*n*

## Function

The H (Highlight) line command is used to highlight a line or group of lines.

## Operands

**n.** The number of consecutive lines to highlight. The default is 1. If the number specified is greater than the number of available lines, all remaining lines are highlighted.

## Usage Notes

The H (Highlight) line command is used to highlight a contiguous group of lines.

## Examples

Figure 170 and Figure 171 show the H (Highlight) line command being used to highlight three source code lines.

**Figure 170 • H (Highlight) Line Command Example**

```
H3 281   CALL 'VIAPDEM1' USING MASTER-IN
000282                     MASTER-END-OF-FILE
000283                     MASTER-REPORT-DATE.
000284
000285   PERFORM P100-PRINT
000286         THRU P119-EXIT.
```

**Figure 171 • H (Highlight) Line Command Results**

```
000281   CALL 'VIAPDEM1' USING MASTER-IN
000282                     MASTER-END-OF-FILE
000283                     MASTER-REPORT-DATE.
000284
000285   PERFORM P100-PRINT
000286         THRU P119-EXIT.
```

# HH (Highlight Block) Line Command

HH

## Function

The HH (Highlight Block) line command is used to highlight a block of lines.

## Operands

None.

## Usage Notes

The HH (Highlight Block) line command is entered in the prefix area on the first and last line of the block to be highlighted.

## Examples

Figure 172 and Figure 173 show the HH (Highlight Block) command being used to highlight three source code lines.

**Figure 172 • HH (Highlight Block) Line Command Example**

```
HH0281  CALL 'VIAPDEM1' USING MASTER-IN
000282                  MASTER-END-OF-FILE
HH0283                  MASTER-REPORT-DATE.
000284
000285  PERFORM P100-PRINT
000286        THRU P119-EXIT.
```

**Figure 173 • HH (Highlight Block) Line Command Results**

```
000281  CALL 'VIAPDEM1' USING MASTER-IN
000282                  MASTER-END-OF-FILE
000283                  MASTER-REPORT-DATE.
000284
000285  PERFORM P100-PRINT
000286        THRU P119-EXIT.
```

# I (Insert) Line Command

I*n*

## Function

The I (Insert) line command is used to insert one or more blank lines into the file.

## Operands

**n.** The number of consecutive blank lines to insert into the file. The default is 1.

## Usage Notes

The I (Insert) line command is used when entering pseudo code into the file during a test session.

The I (Insert) line command adjusts the location of the inserted line(s) to be located between source statements. A line cannot be inserted between two parts of one COBOL statement such as a PERFORM statement and its VARYING clause. Inserted lines are also placed after any Zoom Data or Zoom Assembler lines that may exist for the line where the I (Insert) line command was entered.

When the I (Insert) line command is entered on a line that is not followed by an executable statement, the inserted line(s) is placed before the next executable statement in the program. If this statement is not on the current screen, the screen is scrolled to the first inserted line and a long warning message displays.

## Examples

Figure 174 and Figure 175 show the I (Insert) line command being used to insert 2 blank lines into the source code to allow entering of pseudo code.

**Figure 174 • I (Insert) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
I2 283                          MASTER-REPORT-DATE.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

**Figure 175 • I (Insert) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1
.....2
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

# K (Keep) Line Command

K*n*

## Function

The K (Keep) line command is used to display the value and address of data items at the top of the screen. When the screen is scrolled, the kept lines remain displayed at the top of the screen.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the K (Keep) line command are displayed in a window at the top of the screen. This data can be removed from the screen using the D (Delete) line command. All Keep windows are sizable and scrollable. Use the SET KEEP command to control the number of lines used by the Keep window. When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. Place the cursor within a Keep window and use the UP command or the DOWN command to scroll. The address commands, L and LX, can be used within Keep windows. See the Memory Display screen in online help for more information about the L and LX commands.

When the K (Keep) line command is entered on a data item that is subscripted or indexed, the occurrence line displays to allow the desired entry to be displayed.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The VALUE area of a kept line contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

An effective picture clause displays on the dataname line. For example:

- A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
- An alphanumeric field PIC XXXX displays as X(4).
- A field defined as COMP displays as COMP-4.
- A field with a long dataname that is defined as COMP displays as C3 or C4.

These notes apply to using the K (Keep) line command with Assembler programs:

- The K (Keep) line command can be used in Assembler source programs with CSECT relative named data items and literal data items. CSECT relative named data items are data items that occupy storage locations within the supported CSECT of the source module. Literal data items are those data items defined in the supported CSECT that begin with an equal (=) sign. Unnamed data items are shown with the name UNNAMED ITEM.
- Data items defined outside of the supported CSECT are not supported.

## Examples

Figure 176 and Figure 177 show the K (Keep) line command being used to keep two data items displayed at the top of the screen.

**Figure 176 • K (Keep) Line Command DATA DIVISION Example**

```
000047      05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
K00048          10  END-FILE-COUNT              PIC 9(8).
K00049          10  END-MESSAGE-TEXT            PIC X(10).
000050          10  FILLER                      PIC X(2).
000051  01  INFILE1-WORK-REC.
000052      05  INFILE1-WORK-KEY                PIC 9(10).
```

Figure 177 shows the Program View screen after the K (Keep) line command was used to keep two items displayed on the screen. The line number prefix area shows the number of the source statement from which the lines were kept.

**Figure 177 • K (Keep) Line Command DATA DIVISION Results**

```
''''''   10 END-MESSAGE-TEXT              PIC X(10)       ADDR 000C0F38
''''''       VALUE > ......9009 <
''''''   10 END-FILE-COUNT                PIC 9(8)        ADDR 000C0F30
''''''       VALUE > * INVALID *             <
-------------------------------------------------------------------
000047      05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
000048          10  END-FILE-COUNT              PIC 9(8).
000049          10  END-MESSAGE-TEXT             PIC X(10).
```

Both kept data items remain displayed at the top of the screen even if the ZO (Zoom Out) line command is used to delete the zoom data lines. The D (Delete) line command must be entered on a kept data item to delete it.

# KG (Keep Group) Line Command

KG*n*

## Function

The KG (Keep Group) line command is used to keep the display of the levels, values, and addresses of group items at the top of the screen.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the KG (Keep Group) line command are displayed in a window on the screen. This window can be removed from the screen using the D (Delete) line command. All Keep windows are sizable and scrollable. Use the SET KEEP command to control the number of lines used by the Keep window. When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. Place the cursor within a Keep window and use the UP command or the DOWN command to scroll. The address commands, L and LX, can be used within Keep windows. See the Memory Display screen in online help for more information about the L and LX commands.

A Keep command is executed when the KG (Keep Group) line command is entered for a data item with no subordinate data items.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The VALUE area of a Keep Group line contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and the less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

## Examples

Figure 178 and Figure 179 show the KG (Keep Group) line command used to display subordinate items for the EOF-SWITCHES group item.

**Figure 178 • KG (Keep Group) Line Command Example**

```
KG0047     05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
000048         10  END-FILE-COUNT             PIC 9(8).
000049         10  END-MESSAGE-TEXT           PIC X(10).
000050         10  FILLER                     PIC X(2).
```

**Figure 179 • KG (Keep Group) Line Command Results**

```
''''''   05 END-MESSAGE                    PIC X(20)         ADDR 000C0F30
''''''   10 END-FILE-COUNT                 PIC 9(8)          ADDR 000C0F30
''''''          VALUE > * INVALID *             <
''''''   10 END-MESSAGE-TEXT               PIC X(10)         ADDR 000C0F38
''''''          VALUE > ......9009 <
''''''   10 FILLER                         PIC XX            ADDR 000C0F42
''''''           VALUE > 04 <
-----------------------------------------------------------------------------
000047      05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
000048          10  END-FILE-COUNT              PIC 9(8).
000049          10  END-MESSAGE-TEXT            PIC X(10).
000050          10  FILLER                      PIC X(2).
```

# KGH (Keep Group Hexadecimal) Line Command

KGH*n*

## Function

The KGH (Keep Group Hexadecimal) line command is used to display the levels, values, and addresses of group items in hexadecimal format.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

All Keep windows are sizable and scrollable. Use the SET KEEP command to control the number of lines used by the Keep window. When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. Place the cursor within a Keep window and use the UP command or the DOWN command to scroll. The results of the KGH (Keep Group Hexadecimal) line command are displayed in a window on the screen. The window can be removed using the D (Delete) line command. The address commands, L and LX, can be used within Keep windows. See the Memory Display screen in online help for more information about the L and LX commands.

A KH (Keep Hexadecimal) line command is executed when the KGH (Keep Group Hexadecimal) line command is entered for a data item with no subordinate data items.

When the KGH (Keep Group Hexadecimal) line command is entered on a data item that is subscripted or indexed, the occurrence line displays.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

## Examples

Figure 180 and Figure 181 show the KGH (Keep Group Hexadecimal) line command being used to display the subordinate data items for the EOF-SWITCHES group item.

**Figure 180 • KGH (Keep Group Hexadecimal) Line Command Results**

```
KH0047      05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
000048          10  END-FILE-COUNT              PIC 9(8).
000049          10  END-MESSAGE-TEXT            PIC X(10).
000050          10  FILLER                      PIC X(2).
```

**Figure 181 • KGH (Keep Group Hexadecimal) Line Command Results**

```
''''''   05 END-MESSAGE                    PIC X(20)         ADDR 0009DA38
''''''    0009DA38  00000000 00000000 00000000 00000000 * ................*
''''''    0009DA48  00000000                              * ....           *
-----------------------------------------------------------------------------
000043      05  BEGIN-MESSAGE.
000044          10  BEGIN-PROGRAM-NAME           PIC X(8).
000045          10  FILLER                       PIC X(6).
000046          10  BEGIN-DATE                   PIC 9(6).
```

# KH (Keep Hexadecimal) Line Command

KH*n*

## Function

The KH (Keep Hexadecimal) line command is used to keep a display of the value and address of a data item in hexadecimal format at the top of the screen.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the KH (Keep Hexadecimal) line command are displayed in a window at the top of the screen. This window can be removed from the screen using the D (Delete) or E (Exclude) line command. All Keep windows are sizable and scrollable. Use the SET KEEP command to control the number of lines used by the Keep window. When the number of lines required to display the data items within the window exceeds the number of lines specified for the window, the window becomes vertically scrollable. Place the cursor within a Keep window and use the UP command or the DOWN command to scroll. The address commands, L and LX, can be used within Keep windows. See the Memory Display screen in online help for more information about the L and LX commands.

When the KH (Keep Hexadecimal) line command is entered on a data item that is subscripted or indexed, the occurrence line displays.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The Keep Hexadecimal lines are displayed after the last line that contains the COBOL statement for which the KH (Keep Hexadecimal) line command was entered.

## Examples

Figure 182 and Figure 183 show the KH (Keep Hexadecimal) line command being used to display the value and address of the SUB2 data item in the DATA DIVISION and the SUB1 and BINARY data items in the PROCEDURE DIVISION.

**Figure 182 • KH (Keep Hexadecimal) Line Command DATA DIVISION Example**

```
KH0047      05  END-MESSAGE REDEFINES BEGIN-MESSAGE.
000048          10  END-FILE-COUNT                PIC 9(8).
000049          10  END-MESSAGE-TEXT              PIC X(10).
000050          10  FILLER                        PIC X(2).
```

**Figure 183 • KH (Keep Hexadecimal) Line Command DATA DIVISION Results**

```
''''''   05 END-MESSAGE                    PIC X(20)        ADDR 0009DA38
''''''    0009DA38  00000000 00000000 00000000 00000000 * ................*
''''''    0009DA48  00000000                              * ....            *
-----------------------------------------------------------------------------
000043      05  BEGIN-MESSAGE.
000044          10  BEGIN-PROGRAM-NAME            PIC X(8).
000045          10  FILLER                        PIC X(6).
000046          10  BEGIN-DATE                    PIC 9(6).
```

# L (Last) Line Command

L*n*

## Function

The L (Last) line command redisplays the specified number of excluded lines. These lines are redisplayed starting with the last line in the block of excluded lines.

## Operands

**n.** The number of excluded lines to redisplay. The default is 1. If the number specified is greater than the number of excluded lines, all lines in the excluded block are redisplayed.

## Usage Notes

The L (Last) line command redisplays lines excluded as the result of any primary or line command that excludes lines from the screen display.

## Examples

Figure 184 and Figure 185 show the L (Last) line command being used to redisplay the last two lines in the excluded block.

**Figure 184 • L (Last) Line Command Example**

```
000002   PROGRAM-ID. PRDEMO.
L2       -  -  -  -  -  -  -  -  -  -  5 LINE(S) NOT DISPLAYED
000008   INPUT-OUTPUT SECTION.
```

**Figure 185 • L (Last) Line Command Results**

```
000002   PROGRAM-ID. PRDEMO.
-  -  -  -  -  -  -  -  -  -  -  -  3 LINE(S) NOT DISPLAYED
000006   SOURCE-COMPUTER.  IBM-370.
000007   OBJECT-COMPUTER.  IBM-370.
000008   INPUT-OUTPUT SECTION.
```

# M (Move) Line Command

M*n*

## Function

The M (Move) line command is used to move a pseudo code line or group of pseudo code lines. This command is used with the A (After) and B (Before) line commands to indicate where to move the line(s).

## Operands

**n.** The number of consecutive lines to move. The default is 1. If the number specified is greater than the number of available lines, all remaining lines are moved.

## Usage Notes

The M (Move) line command pertains to pseudo code lines only.

## Examples

Figure 186 and Figure 187 show pseudo code lines 1 and 2 being moved after line 287.

**Figure 186 • M (Move) Line Command Example**

```
000281     CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
M2...1     IF CLIENT-ID = '387962' THEN
.....2          BREAK.
000284
000285     PERFORM P100-PRINT
000286             THRU P119-EXIT.
A00287
```

**Figure 187 • M (Move) Line Command Results**

```
000281     CALL 'VIAPDEM1' USING MASTER-IN
000282                           MASTER-END-OF-FILE
000283                           MASTER-REPORT-DATE.
000284
000285     PERFORM P100-PRINT
000286             THRU P119-EXIT.
000287
.....1     IF CLIENT-ID = '387962' THEN
.....2          BREAK.
```

# MM (Move Block) Line Command

MM

## Function

The MM (Move Block) line command is used to move a block of pseudo code lines. This command is used with the A (After) and B (Before) line commands to indicate where to move the block of lines.

## Operands

None.

## Usage Notes

The MM (Move Block) line command is entered in the prefix area on the first and last line of the block to be moved. The range must be contained within a contiguous block of pseudo code lines. There cannot be any source lines within the range of the MM (Move Block) command.

## Examples

Figure 188 and Figure 189 show pseudo code lines 1 and 2 being moved after line 287.

**Figure 188 • MM (Move Block) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
MM...1    IF CLIENT-ID = '387962' THEN
MM...2         BREAK.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
A00287
```

**Figure 189 • MM (Move Block) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
.....1    IF CLIENT-ID = '387962' THEN
.....2         BREAK.
```

# R (Repeat) Line Command

R*n*

## Function

The R (Repeat) line command is used to replicate a COBOL source or pseudo code line one or more times.

## Operands

**n.** The number of times to replicate the line on which the R (Repeat) line command is entered. The default is 1.

## Usage Notes

The replicated lines are placed immediately after the line on which the R (Repeat) line command is entered. Repeated COBOL source statements become pseudo code lines.

This command cannot be used on Assembler, disassembled object, or PL/I statements.

## Examples

Figure 190 and Figure 191 show a pseudo code line being replicated.

**Figure 190 • R (Repeat) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
R....2         BREAK.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

**Figure 191 • R (Repeat Block) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2         BREAK.
.....3         BREAK.
000284
000285    PERFORM P100-PRINT
000286           THRU P119-EXIT.
000287
```

# RR (Repeat Block) Line Command

RR

## Function

The RR (Repeat Block) line command is used to replicate a block of COBOL source or pseudo code lines.

## Operands

None.

## Usage Notes

The RR (Repeat Block) line command is entered in the prefix area on the first and last line of the block to be replicated. The range of the command must be contained within a contiguous block of lines. COBOL source statements included in the range become pseudo code lines.

This command cannot be used on Assembler, disassembled object, or PL/I statements.

## Examples

Figure 192 and Figure 193 show pseudo code lines 1 and 2 being repeated. The CLIENT-ID number was changed after the lines were repeated to be valid for that conditional test.

**Figure 192 • RR (Repeat Block) Line Command Example**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
RR...1    IF CLIENT-ID = '387962' THEN
RR...2         BREAK.
000284
000285    PERFORM P100-PRINT
000286            THRU P119-EXIT.
000287
```

**Figure 193 • RR (Repeat Block) Line Command Results**

```
000281    CALL 'VIAPDEM1' USING MASTER-IN
000282                          MASTER-END-OF-FILE
000283                          MASTER-REPORT-DATE.
.....1    IF CLIENT-ID = '387962' THEN
.....2         BREAK.
.....3    IF CLIENT-ID = '395298' THEN
.....4         BREAK.
000284
000285    PERFORM P100-PRINT
000286            THRU P119-EXIT.
000287
```

# S (Show) Line Command

*Sn*

## Function

The S (Show) line command is used to redisplay the specified number of excluded lines. These lines. These lines are redisplayed starting with the first line in the excluded block.

## Operands

**n.** The number of consecutive excluded lines to show. The default is 1. If the number specified is greater than the number of excluded lines, all lines in the excluded block are redisplayed.

## Usage Notes

The S (Show) line command redisplays lines excluded as the result of any primary or line command that excludes lines from the screen display.

## Examples

Figure 194 and Figure 195 show the S (Show) line command being used to redisplay two excluded source lines.

**Figure 194 • S (Show) Line Command Example**

```
000002   PROGRAM-ID. PRDEMO.
S2       -  -  -  -  -  -  -  -  -  -  5 LINE(S) NOT DISPLAYED
000008   INPUT-OUTPUT SECTION.
```

**Figure 195 • S (Show) Line Command Results**

```
000002   PROGRAM-ID. PRDEMO.
000003   AUTHOR. WRITTEN BY ASG AT LANGLVL 2.
000004   ENVIRONMENT DIVISION.
-  -  -  -  -  -  -  -  -  -  -  -  3 LINE(S) NOT DISPLAYED
000008   INPUT-OUTPUT SECTION.
```

## SS (Show Block) Line Command

SS

### Function

The SS (Show Block) line command is used to redisplay a block of excluded lines.

### Operands

None.

### Usage Notes

The SS (Show Block) line command is entered in the prefix area on the first and last line of the excluded blocks to be redisplayed.

### Examples

Figure 196 and Figure 197 show the SS (Show Block) line command being used to redisplay two blocks of excluded lines.

**Figure 196 • SS (Show Block) Line Command Example**

```
000401  C4H-TO-NUM.
SS0402 * MOVE COMP-4 (HALFWD) TO NUMERIC (UNSIGNED)
- - -  -  -  -  -  -  -  -  -  -  -  -  - 4 LINES NOT DISPLAYED
000407 * MOVE COMP-4 (HALFWD) TO NUMERIC (SIGNED)
- - -  -  -  -  -  -  -  -  -  -  -  -  - 4 LINES NOT DISPLAYED
SS0412  C4H-TO-C1.
```

**Figure 197 • SS (Show Block) Line Command Results**

```
000401  C4H-TO-NUM.
000402 * MOVE COMP-4 (HALFWD) TO NUMERIC (UNSIGNED)
000403      MOVE N-C4HALF   TO 77-NU.
000404      MOVE N-C4HALFS  TO 77-NU.
000405      MOVE X-C4HALF   TO 77-NU.
000406      MOVE X-C4HALFS  TO 77-NU.
000407 * MOVE COMP-4 (HALFWD) TO NUMERIC (SIGNED)
000408      MOVE N-C4HALF   TO 77-NUS.
000409      MOVE N-C4HALFS  TO 77-NUS.
000410      MOVE X-C4HALF   TO 77-NUS.
000411      MOVE X-C4HALFS  TO 77-NUS.
000412  C4H-TO-C1.
```

# X (Exclude) Line Command

X*n*

## Function

The X (Exclude) line command is used to exclude a line or group of lines from being displayed.

## Operands

**n.** The number of consecutive lines to exclude. If the number specified is greater than the number of available lines, all remaining lines are excluded. The default is 1.

## Usage Notes

Excluded lines are shown as a row of dashes with the number of excluded lines indicated. The dashed line can be suppressed by using the SET SHADOW OFF command.

See the SET primary command in chapter "SET Command" on page 409 for more information.

The X (Exclude) line command can also be used to remove a window from the screen that was displayed as the result of a Z (Zoom) line command.

## Examples

Figure 198 and Figure 199 show the X (Exclude) line command being used to exclude lines 332-336.

**Figure 198 • X (Exclude) Line Command Example**

```
000331          MOVE ZIP-CODE TO HLD-ZIP.
X50332          IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333              NEXT SENTENCE
000334          ELSE
000335              PERFORM P150-SUBTOT
000336                  THRU P169-EXIT
000337             MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

**Figure 199 • X (Exclude) Line Command Results**

```
000331          MOVE ZIP-CODE TO HLD-ZIP.
 - - -  -  -  -  -  -  -  -  -  -  -  - 5 LINES NOT DISPLAYED
000337             MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

# XX (Exclude Block) Line Command

XX

## Function

The XX (Exclude Block) line command is used to exclude a block of lines from being displayed.

## Operands

None.

## Usage Notes

Excluded lines are shown as a row of dashes with the number of excluded lines indicated. The dashed line can be suppressed by using the SET SHADOW OFF command.

See "SET Command" on page 409 for more information.

The XX (Exclude Block) line command is entered in the line command area on the first and last line of the block of lines to be excluded from being displayed.

## Examples

Figure 200 and Figure 201 show the XX (Exclude Block) line command being used to exclude a block of five lines.

**Figure 200 • XX (Exclude Block) Line Command Example**

```
000331          MOVE ZIP-CODE TO HLD-ZIP.
XX0332          IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333              NEXT SENTENCE
000334          ELSE
000335              PERFORM P150-SUBTOT
XX0336                  THRU P169-EXIT
000337              MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

**Figure 201 • XX (Exclude Block) Line Command Results**

```
000331          MOVE ZIP-CODE TO HLD-ZIP.
 - - -  -  -  -  -  -  -  -  -  -  -  - 5 LINES NOT DISPLAYED
000337              MOVE HLD-ZIP-PREFIX TO CUR-PREFIX
```

# ZA (Zoom Assembler) Line Command

ZA

## Function

The ZA (Zoom Assembler) line command is used to display the Assembler instructions that correspond to a COBOL or PL/I source statement. When the ZA (Zoom Assembler) line command is entered on an Assembler source code line, the object code and pseudo Assembler statements for any source lines are displayed.

## Operands

None.

## Usage Notes

The results of the ZA (Zoom Assembler) line command are displayed in a window on the screen. See the DATA operand of the SET command for more information about adjusting the size of the scrollable windows. Zoom windows can be removed from the screen using the ZO (Zoom Out) or X (Exclude) line commands. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

The ZA (Zoom Assembler) line command displays the corresponding Assembler instructions for the COBOL or PL/I statement on which the command is entered. This command can only be entered on executable statements within the PROCEDURE DIVISION of a COBOL program or within a PROC of a PL/I program. If no executable code exists for the statement, a message displays indicating this condition. Zoom Assembler lines are displayed after the COBOL or PL/I statement for which the ZA (Zoom Assembler) line command was entered. The cursor is positioned to the first input field for data items. The EBCDIC character translation of the object code displays at the right side of the window.

SET ASM ON automatically displays the Assembler lines for every COBOL verb or PL/I statement in the program.

## Examples

Figure 202 and Figure 203 show the ZA (Zoom Assembler) line command being used to display the Assembler code for line 191.

**Figure 202 • ZA (Zoom Assembler) Line Command Example**

```
000190      MOVE 'VMERGE  ' TO BEGIN-PROGRAM-NAME.
ZA0191      ACCEPT BEGIN-DATE  FROM DATE.
>>>>>>      WRITE OUTRPT-REC FROM BEGIN-MESSAGE.
```

**Figure 203 • ZA (Zoom Assembler) Line Command Results**

```
000191      ACCEPT BEGIN-DATE  FROM DATE.
''''''   001660 4100 D228      LA    R0,X'228'(R13)              * ..K.   *
''''''   001664 58F0 C018      L     R15,X'018'(R12)             * .0{.   *
''''''   001668 05EF           BALR  R14,R15                     * ..     *
''''''   00166A D205 6036 D228 MVC   X'036'(6,R6),X'228'(R13)     * K.-.K. *
''''''   001670 96F0 603B      OI    X'03B'(R6),X'F0'            * o0-.   *
>>>>>>      WRITE OUTRPT-REC FROM BEGIN-MESSAGE.                    FALLTHRU
```

# ZD (Zoom Data) Line Command

ZD*n*

## Function

The ZD (Zoom Data) line command is used to display the value and address of data items.

## Operand

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the ZD (Zoom Data) line command are displayed in a window on the screen. See the DATA operand of the SET command for information about adjusting the size of the scrollable windows. Zoom windows can be removed from the screen using the ZO (Zoom Out) or X (Exclude) line commands. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

When the ZD (Zoom Data) line command is entered on a data item that is subscripted or indexed, the occurrence line displays to allow the desired entry to be displayed.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The Zoom Data lines are displayed after the last line that contains the COBOL statement for which the ZD (Zoom Data) line command was entered. The cursor is positioned at the first input field for data items.

The VALUE area of a Zoom Data line contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

An effective picture clause displays on the dataname line. For example:

A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
An alphanumeric field PIC XXXX displays as X(4).
A field defined as COMP displays as COMP-4.
A field with a long dataname that is defined as COMP displays as C3 or C4.

Entering a ZD (Zoom Data) line command removes any existing Zoom Data lines for this statement.

These notes apply to using the ZD (Zoom Data) line command with Assembler programs.

- The ZD (Zoom Data) line command can be used in Assembler source programs with CSECT relative named data items and literal data items. CSECT relative named data items are data items that occupy storage locations within the supported CSECT of the source module. Literal data items are those data items defined in the supported CSECT that begin with an equal sign (=). Unnamed data items within the supported CSECT are shown with the name UNNAMED ITEM.
- Data items defined outside of the supported CSECT, are not supported.

When the ZD (Zoom Data) line command is used to display a numeric data item that contains invalid numeric data, the contents of the field are displayed as character data [with non-displayable characters displayed as periods (.)], and the highlighted text `*INVALID NUMERIC*` displays to the right of the field.

## Examples

Figure 204 and Figure 205 show the ZD (Zoom Data) line command being used to display the BEGIN-PROGRAM-NAME and BEGIN-DATA data items.

**Figure 204 • ZD (Zoom Data) Line Command DATA DIVISION Example**

```
000043  05  BEGIN-MESSAGE.
ZD0044      10  BEGIN-PROGRAM-NAME        PIC X(8).
000045      10  FILLER                    PIC X(6).
000046      10  BEGIN-DATE                PIC 9(6).
```

**Figure 205 • ZD (Zoom Data) Line Command DATA DIVISION Results**

```
000043      05  BEGIN-MESSAGE.
000044          10  BEGIN-PROGRAM-NAME          PIC X(8).
'''''' +-------------------------------------------------------------------+
'''''' | 10 BEGIN-PROGRAM-NAME            PIC X(8)           ADDR 000907F0 |
'''''' |      VALUE > VMERGE   <                                           |
'''''' +-------------------------------------------------------------------+
000045          10  FILLER                      PIC X(6).
000046          10  BEGIN-DATE                  PIC 9(6).
```

# ZG (Zoom Group) Line Command

ZG*n*

## Function

The ZG (Zoom Group) line command is used to display the levels, values, and addresses of group items.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the ZG (Zoom Group) line command are displayed in a window on the screen. See the DATA operand of the SET command for information about adjusting the size of the scrollable windows. Zoom windows can be removed from the screen using the ZO (Zoom Out) or X (Exclude) line command. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

A Zoom Data command is executed when the ZG (Zoom Group) line command is entered for a data item with no subordinate data items.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The Zoom Group lines are displayed after the last line that contains the COBOL statement for which the ZG (Zoom Group) line command was entered.

The VALUE area of a Zoom Group line contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. The maximum length of the data item is indicated by the number of bytes between the greater than (>) and the less than (<) symbols. If the length of the data item is greater than fifty bytes, additional lines (of fifty bytes each) are displayed with +50, +100, +150 and so on preceding them.

## Examples

Figure 206 and Figure 207 show the ZG (Zoom Group) line command used to display subordinate items for the EOF-SWITCHES group item.

**Figure 206 • ZG (Zoom Group) Line Command Example**

```
ZG0068  01  EOF-SWITCHES.
000069      05  INFILE1-EOF                      PIC 9.
000070          88  END-INFILE1                  VALUE 1.
000071          88  NOT-END-INFILE1              VALUE 0, 2 THRU 9.
000072      05  INFILE2-EOF                      PIC 9.
000073          88  END-INFILE2                  VALUE 1.
000074          88  NOT-END-INFILE2              VALUE 0, 2 THRU 9.
000075      05  INFILE3-EOF                      PIC 9.
000076          88  END-INFILE3                  VALUE 1.
000077          88  NOT-END-INFILE3              VALUE 0, 2 THRU 9.
000078      05  MASTER-EOF-SWITCH                PIC 9.
000079          88 FINISHED-READING-ALL-FILES    VALUE 3.
```

**Figure 207 • ZG (Zoom Group) Line Command Results**

```
000068  01  EOF-SWITCHES.
'''''' +-------------------------------------------------------------------+
'''''' | 01 EOF-SWITCHES                                        ADDR 00081910 |
'''''' | 05 INFILE1-EOF                     PIC 9               ADDR 00081910 |
'''''' |      VALUE > 0                        <                              |
'''''' | 05 INFILE2-EOF                     PIC 9               ADDR 00081911 |
'''''' |      VALUE > 0                        <                              |
'''''' | 05 INFILE3-EOF                     PIC 9               ADDR 00081912 |
'''''' |      VALUE > 0                        <                              |
'''''' | 05 MASTER-EOF-SWITCH               PIC 9               ADDR 00081913 |
'''''' |      VALUE > 0                        <                              |
'''''' +-------------------------------------------------------------------+
000069      05  INFILE1-EOF                      PIC 9.
```

# ZGH (Zoom Group Hexadecimal) Line Command

ZGH*n*

## Function

The ZGH (Zoom Group Hexadecimal) line command is used to display the levels, values, and addresses of group items in hexadecimal format.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the ZGH (Zoom Group Hexadecimal) line command are displayed in a window on the screen. See the DATA operand of the SET command for information about adjusting the size of the scrollable windows. Zoom windows can be removed using the ZO (Zoom Out) or X (Exclude) line commands. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

A ZH (Zoom Hexadecimal) line command is executed when the ZGH (Zoom Group Hexadecimal) line command is entered for a data item with no subordinate data items.

When the ZGH (Zoom Group Hexadecimal) line command is entered on a data item that is subscripted or indexed, the occurrence line displays.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The Zoom Group Hexadecimal lines are displayed after the last line that contains the COBOL statement for which the ZGH (Zoom Group Hexadecimal) line command was entered.

Entering a ZGH (Zoom Group Hexadecimal) line command removes any existing Zoom Group Hexadecimal lines for this statement.

## Examples

Figure 208 and Figure 209 show the ZGH (Zoom Group Hexadecimal) line command being used to display the subordinate data items for the EOF-SWITCHES group item.

**Figure 208 • ZGH (Zoom Group Hexadecimal) Line Command Example**

```
ZGH068  01  EOF-SWITCHES.
000069      05  INFILE1-EOF                      PIC 9.
000070          88  END-INFILE1                  VALUE 1.
000071          88  NOT-END-INFILE1              VALUE 0, 2 THRU 9.
000072      05  INFILE2-EOF                      PIC 9.
000073          88  END-INFILE2                  VALUE 1.
000074          88  NOT-END-INFILE2              VALUE 0, 2 THRU 9.
000075      05  INFILE3-EOF                      PIC 9.
000076          88  END-INFILE3                  VALUE 1.
000077          88  NOT-END-INFILE3              VALUE 0, 2 THRU 9.
000078      05  MASTER-EOF-SWITCH                PIC 9.
000079          88 FINISHED-READING-ALL-FILES    VALUE 3.
```

**Figure 209 • ZGH (Zoom Group Hexadecimal) Line Command Results**

```
000068  01  EOF-SWITCHES.
'''''' +-------------------------------------------------------------------+
'''''' | 01 EOF-SWITCHES                                        ADDR 00081910 |
'''''' | 05 INFILE1-EOF                     PIC 9               ADDR 00081910 |
'''''' |  00081910  F0                                   * 0                * |
'''''' | 05 INFILE2-EOF                     PIC 9               ADDR 00081911 |
'''''' |  00081911  F0                                   * 0                * |
'''''' | 05 INFILE3-EOF                     PIC 9               ADDR 00081912 |
'''''' |  00081912  F0                                   * 0                * |
'''''' | 05 MASTER-EOF-SWITCH               PIC 9               ADDR 00081913 |
'''''' |  00081913  F0                                   * 0                * |
'''''' +-------------------------------------------------------------------+
000069      05  INFILE1-EOF                      PIC 9.
```

# ZH (Zoom Hexadecimal) Line Command

ZH*n*

## Function

The ZH (Zoom Hexadecimal) line command is used to display the value and address of a data item in hexadecimal format.

## Operands

**n.** The relative position of the data item on the line to be displayed.

## Usage Notes

The results of the ZH (Zoom Hexadecimal) line command are displayed in a window on the screen. See the DATA operand of the SET command for information about adjusting the size of the scrollable windows. Zoom windows can be removed from the screen using the ZO (Zoom Out) or X (Exclude) line commands. The address commands, L and LX, can be used within Zoom windows. See the Memory Display screen in online help for more information about the L and LX commands.

When the ZH (Zoom Hexadecimal) line command is entered on a data item that is subscripted or indexed, the occurrence line displays.

The n (number) operand can only be used to display one data item for a statement that contains multiple data items. The number entered should be the relative number of the desired data item in the statement. Data items are assigned relative numbers from left to right.

The Zoom Hexadecimal lines are displayed after the last line that contains the COBOL statement for which the ZH (Zoom Hexadecimal) line command was entered.

Entering a ZH (Zoom Hexadecimal) line command removes any existing Zoom Data lines for this statement.

## Examples

Figure 210 through Figure 213 show the ZH (Zoom Hexadecimal) line command being used to display the value and address of the SUB2 data item in the DATA DIVISION and the SUB1 and BINARY data items in the PROCEDURE DIVISION.

**Figure 210 • ZH (Zoom Hexadecimal) Line Command DATA DIVISION Example**

```
000019 77 SUB1        PIC S9(4) COMP.
ZH0020 77 SUB2       PIC S9(4) COMP VALUE +5.
000021 77 SUBMAX      PIC S9(4) COMP VALUE +2.
```

**Figure 211 • ZH (Zoom Hexadecimal) Line Command DATA DIVISION Results**

```
000019  77  SUB1              PIC S9(4) COMP.
000020  77  SUB2              PIC S9(4) COMP VALUE +5.
...... +-------------------------------------------------------------------+
...... |  77  SUB2                                          ADDRESS 00302FF2  |
...... |  00302FF2  0005                                  * ..              * |
...... +-------------------------------------------------------------------+
000021  77  SUBMAX            PIC S9(4) COMP VALUE +2.
```

**Figure 212 • ZH (Zoom Hexadecimal) PROCEDURE DIVISION Example**

```
000045 INIT-ENTRY.
000046   MOVE SUB1 TO DISPNUM (SUB1).
ZH0047   MOVE SUB1 TO BINARY  (SUB1).
000048   MOVE SUB1 TO PACKED  (SUB1).
```

**Figure 213 • ZH (Zoom Hexadecimal) PROCEDURE DIVISION Results**

```
000045  INIT-ENTRY.
000046      MOVE SUB1 TO DISPNUM (SUB1).
000047      MOVE SUB1 TO BINARY  (SUB1).
...... +-------------------------------------------------------------------+
...... |  77  SUB1                                          ADDRESS 00302FF0  |
...... |  00302FF0  0000                                  * ..              * |
...... |  03  BINARY                                        ADDRESS 00302FFC  |
...... |  OCCURRENCE ( 1 )                                                    |
...... |  00302FF0  0000                                  * ..              * |
...... +-------------------------------------------------------------------+
000048      MOVE SUB1 TO PACKED  (SUB1).
```

# ZI (Zoom In) Line Command

ZI

## Function

The ZI (Zoom In) line command is used to redisplay excluded source lines according to the hierarchical levels in the program. The ZI (Zoom In) line command is used in conjunction with the ZO (Zoom Out) line command to show the structure of a program and provide a means of stepping through each level or going directly into or out of a particular section of source code.

**Note:**

This command is available for COBOL programs only.

## Operands

None.

## Usage Notes

The ZI (Zoom In) line command is used to show the structure of a program. PROCEDURE DIVISION hierarchy consists of section labels, paragraph labels, and paragraph code. DATA DIVISION hierarchy consists of sections, FDs, 01 or 77 levels, and all definitions within an 01 level.

If the ZI (Zoom In) line command is entered on an excluded block of lines, the highest structure level within that block of lines displays. If there are multiple statements at that level, they are also displayed.

See the "ZOOMIN/ZOOMOUT Commands" on page 462 for more information about displaying and excluding source lines according to the hierarchical levels of the program.

# ZO (Zoom Out) Line Command

ZO

## Function

The ZO (Zoom Out) line command is used to redisplay a source line in its original format after a Zoom line command has been entered. The ZO (Zoom Out) line command is used in conjunction with the ZI (Zoom In) line command to show the structure of a program and provide a means of stepping through each level or going directly into or out of a particular section of source code. The ZI (Zoom In) and ZO (Zoom Out) line commands are used to display or exclude source lines according to the hierarchical levels of the program.

**Note:**

This command is available for COBOL programs only.

## Operands

None.

## Usage Notes

The ZO (Zoom Out) line command can only be entered on a line that has been displayed using one of the Zoom line commands.

See "ZOOMIN/ZOOMOUT Commands" on page 462 for more information about displaying and excluding source lines according to the hierarchical levels of the program.

The ZO (Zoom Out) line command can also be used to remove lines that were displayed in a window on the screen, as the result of another Zoom line command.

## Example

Figure 214 and Figure 215 show the ZO (Zoom Out) line command being used to remove the window displayed on the screen as the result of a ZD (Zoom Data) line command.

**Figure 214 • ZO (Zoom Out) Line Command Example**

```
000045  INIT-ENTRY.
000046      MOVE SUB1 TO DISPNUM (SUB1).
000047      MOVE SUB1 TO BINARY  (SUB1).
...... +-------------------------------------------------------------------+
...... |  77  SUB1                                          ADDRESS 00302FF0  |
ZO.... |  00302FF0  0000                                      * ..          * |
...... |  03  BINARY                                        ADDRESS 00302FFC  |
...... |  OCCURRENCE ( 1 )                                                    |
...... |  00302FF0  0000                                  * ..              * |
...... +-------------------------------------------------------------------+
000048      MOVE SUB1 TO PACKED  (SUB1).
```

**Figure 215 • ZO (Zoom Out) Line Command Results**

```
000045 INIT-ENTRY.
000046  MOVE SUB1 TO DISPNUM (SUB1).
000047  MOVE SUB1 TO BINARY (SUB1).
000048  MOVE SUB1 TO PACKED (SUB1).
```

# 9 Pseudo Code

This chapter describes how to use pseudo code and contains these sections:


| Section | Page |
|---|---|
| Using Pseudo Code | 512 |
| Entering Pseudo Code into Source Code | 513 |
| Execution of Pseudo Code | 515 |
| Saving Pseudo Code | 516 |
| Pseudo Code Commands and Statements | 516 |
| COBOL Reserved Words | 531 |
| Pseudo Code Examples | 532 |


SmartTest provides COBOL compatible statements and commands called pseudo code. These statements and commands are entered in line with existing COBOL, PL/I or Assembler source code and are used to insert temporary logic to branch around sections of code during a test session and to add new code to a program.

Pseudo code statements are entered into a program while it displays on the Program View screen. Pseudo code can also be entered when a program interrupt is encountered during a test session. Pseudo code supports DBCS strings. Each pseudo code statement is associated with the executable source statement that follows it. In other words, a pseudo code statement is executed before the next verb when the program is tested. Multiple pseudo code statements that are associated with an executable source statement form a pseudo code block. Only one pseudo code block is associated with an executable source statement. Pseudo code statements are COBOL compatible and follow standard COBOL syntax rules. These rules also apply to the syntax:

- An asterisk (*) in the first position (column 7 for COBOL programs or column 1 for Assembler or PL/I programs) indicates a comment line.
- Multiple verbs can be entered on a line.

- Statements can extend across multiple lines.
- Each conditional statement must end with a period.

# Using Pseudo Code

SmartTest provides various screens, options, and information that control the use of pseudo code. This table describes these screens and options:

| Screen | Description |
|---|---|
| Pseudo Code List | Lists all pseudo code statements in the current program and indicates if they are active or inactive. This screen can be used to specify whether pseudo code can be executed during a test session or to activate/deactivate specific pseudo code statements. A COUNT field is provided that indicates the number of times a pseudo code statement has been executed during the current test session. Blocks of pseudo code are shown with the source code lines that immediately precede and follow them. |
| When Conditions List | Lists all When Conditions in the current program and indicates if they are active or inactive. This screen can be used to specify whether When Conditions can be executed in a test session, or to activate/deactivate specific When Condition statements. The COUNT field indicates the number of times a When Condition statement has been executed during the current test session. |
| Test Session Tailoring | Includes a Pseudo Code field used to specify whether pseudo code is active or inactive during a test session. If this field contains NO, the indicated program does not execute pseudo code statements. |
| Options - Modes | Displays the current setting for the SET PSEUDO command mode. When this mode is OFF, the pseudo code does not execute when the program is executed. |

See the online help for detailed information the Pseudo Code List screen, When Conditions List screen and Test Session Tailoring screen. See "SET Command" on page 409 for information about the SET command. See the online help for information about the Options - Modes screen. For COBOL II and later programs compiled with the Optimize Compiler option, see the *ASG-SmartTest for COBOL and Assembler User's Guide*.

# Entering Pseudo Code into Source Code

Pseudo code statements are entered in line with existing COBOL, PL/I, or Assembler source code on the Program View screen. During a test session, pseudo code statements can be entered whenever a program interrupt occurs.

For COBOL programs, all statements are entered in the PROCEDURE DIVISION with the exception of data definition statements (77 level). Data definitions can be entered in the DATA DIVISION or PROCEDURE DIVISION, but must manually be moved to the DATA DIVISION before pseudo code is used to update the source code. (Updating source code with pseudo code is described later in this chapter.) For Assembler programs, pseudo code can be entered after any statement within a CSECT. For PL/I programs, pseudo code can be entered anywhere in a procedure block.

Pseudo code statements can contain numeric and non-numeric literals. If the data element being tested is a character field (non-numeric), the second operand in the pseudo code statement must be in quotes. For example:

```
IF OUT_CNT_4 = '001' THEN BREAK.
```

If the data element being tested is a decimal field (numeric), then the second operand in a pseudo code statement does not need to be in quotes or zero padded. For example:

```
IF OUT_CNT_3 = 1 THEN BREAK.
```

Pseudo code statements are written in standard COBOL format and are entered in these areas:

| Pseudo Code Area | Description |
|---|---|
| Sequence Numbers | Contains the sequence numbers in columns 1 through 6. SmartTest automatically numbers lines sequentially within a block of pseudo code beginning with 1. These numbers are left justified with apostrophes. Numbering is reset to 1 within each block of pseudo code entered. |
| Comments | Contains comments. Column 7 is the first position on the input line for COBOL programs and on column 1 for PL/I and Assembler programs. An asterisk (*) in this column indicates the line is a comment. |
| Area A | Contains labels or data definition level numbers for COBOL programs (77) in columns 8 through 11, and column 1 is used to enter labels for PL/I and Assembler programs. |
| Area B | Contains pseudo code statements, sentences, clauses, and so on for COBOL programs in columns 12 through 72. Columns 2 through 72 are used for PL/I and Assembler programs. |

Various SmartTest primary and line commands are available for entering pseudo code statements into a program. These commands are described this table:

| Use the... | To... |
|---|---|
| BREAK primary command | Enter a pseudo code BREAK statement before or after a source statement that contains a specified target. Program execution stops when the BREAK statement is encountered during a test session. |
| WHEN primary command | Test conditional expressions and execute subordinate statements based on the truth value of the test. This command can be entered at any time during a test session and from any location in the program. Every WHEN command and the operands entered with it cause the corresponding pseudo code statements to be generated. These statements are automatically placed at the end of the program and can be viewed by scrolling to them on the Program View screen, or by accessing the When Conditions List screen. |
| BR (Break) line command | Insert a pseudo code BREAK statement before the desired source code statement. |
| C (Copy) and CC (Copy Block) line commands | Copy pseudo code statements from one location to another within the source code. Any source statements in the block of pseudo code lines to be copied are also copied to the new location as pseudo code statements. |
| D (Delete) and DD (Delete Block) line commands | Delete pseudo code statements. |
| I (Insert) line command | Insert one or more blank lines into the source code. Pseudo code statements can then be entered on these blank lines. |
| M (Move) and MM (Move Block) line commands | Move one or more pseudo code statements to another location within the source code. |
| R (Repeat) and RR (Repeat Block) line commands | Replicate one or more pseudo code statements. Any source statements in the block of lines to be repeated are also repeated as pseudo code statements. |

# Execution of Pseudo Code

Pseudo code is executed as part of the program logic when the STEP or RUN commands are entered. Each statement is associated with the source statement (COBOL verb) that follows it, which means it gets executed before the next verb. For example, when a Breakpoint occurs, processing stops and the arrows are positioned in the sequence number area of the next statement to be executed. If a block of pseudo code is inserted immediately before this statement, the arrows are repositioned to the first pseudo code statement in the block, making it the next statement to be executed when the STEP or RUN command is entered.

Pseudo code statements generated as a result of a WHEN primary command are automatically positioned after the last source line in the program. When condition tests are performed after the execution of every COBOL verb that may alter data, and if the conditional statement is true, the subordinate pseudo code logic is performed.

See "STOP Command" on page 429 for information about a more resource efficient method for determining where a field is modified.

Pseudo code is parsed (compiled) whenever a command is entered that references a pseudo code statement or pseudo code variable, and when saving the pseudo code to the AKR. These commands can reference pseudo code:

| | |
|---|---|
| ADD | RUN |
| KEEP | SAVE PSEUDO |
| LIST BREAKS | STEP |
| LIST WHENS | STOP |
| LIST PSEUDO | SUBTRACT |
| MOVE | ZD (Zoom Data) |
| QUALIFY | |

Pseudo code statements that are syntactically correct (based on COBOL syntax rules) are executed as part of the program logic. SmartTest does not validate these statements to determine if they are logically correct. When a pseudo code statement is encountered that causes a program execution error, the current test is interrupted and an error message displays.

# Saving Pseudo Code

When a test session ends, pseudo code statements remain intact and all BREAK statements remain active. Entering the SAVE command with the PSEUDO operand saves the pseudo code with the program in the AKR, so that the pseudo code can be saved across multiple test sessions.

Pseudo code can also be copied to the Punch file by using the LPUNCH command with the PSEUDO operand. All pseudo code in the program displayed on the pseudo code List screen is copied to the Punch file when this command is used. The Punch file can then be included in a source file during an ISPF edit session. This feature is useful when testing changes to a program, then implementing the changes as part of the program. The UPDATE command can also be used to include pseudo code into a COBOL source file.

**Note:**

Each conditional pseudo code statement ends with a period when inserted into source code. These statements should be reviewed before automatically updating the source code with them.

# Pseudo Code Commands and Statements

These pseudo code commands are used in imperative and conditional statements to specify new or alternative logic for the program being tested:

| | |
|---|---|
| ADD | SUBTRACT |
| BREAK | WHEN |
| GO | [label] |
| IF | 77items |
| MOVE | |

The &COUNT system variable is provided for use in pseudo code statements. This variable is a counter used to indicate the number of times each pseudo code statement has been executed. &COUNT is useful in tests for specifying alternative logic based on its value. The value of &COUNT is automatically initialized to zero and is incremented before the corresponding verb is executed.

Figure 216 shows how &COUNT can be used to process every tenth record of a file.

**Figure 216 • &COUNT System Variable Example**

```
000433      ADD 1 TO END-FILE-COUNT.
'''''1  77 TEMP-COUNTER PIC S9(7) COMP.
'''''2  TEMP-LABEL.
000434      READ INFILE1 INTO INFILE1-WORK-REC
000435          AT END
000436              MOVE INFILE1-EOF-MSG TO OUTRPT-WORK-DATA
000437              MOVE ZERO TO OUTRPT-WORK-KEY
000438              WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA
000439              MOVE 1 TO INFILE1-EOF
000440              MOVE EOF-KEY TO INFILE1-WORK-KEY.
000441  3100-READ-INFILE1-X.
'''''1      IF INFILE1-EOF NOT = 1 THEN
'''''2         IF &COUNT < TEMP-COUNTER THEN
'''''3            GO TO TEMP-LABEL.
'''''4      ADD 10 TO TEMP-COUNTER.
000442      EXIT.
```

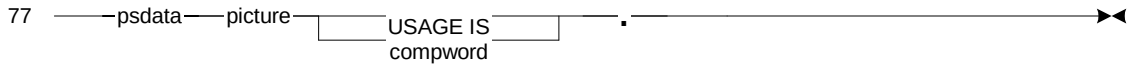
The &COUNT variable is initialized to zero at the start of a test. &COUNT can be referenced as required in any pseudo code statement but cannot be modified (e.g., MOVE 2 TO &COUNT is invalid).

The value of &COUNT can be displayed by entering the LIST PSEUDO command. The Pseudo Code List screen displays with &COUNT and its value.

To ensure that a pseudo code statement is executed only once per test session, use these with the syntax:

```
IF &COUNT = 1 THEN
         statement.
```

## 77 Statement




### Function

The 77 statement is used to define a pseudo code data item. These data items are entered in Area A within a block of pseudo code. Each 77 level dataname must be unique and cannot be qualified. 77 level data definitions can be entered in either the DATA DIVISION or the PROCEDURE DIVISION. If these entries are made in the PROCEDURE DIVISION, they must manually be moved to the DATA DIVISION when pseudo code is updated with source code. Pseudo code data items must be defined before being referenced and cannot be defined in WHEN commands.

## Operands

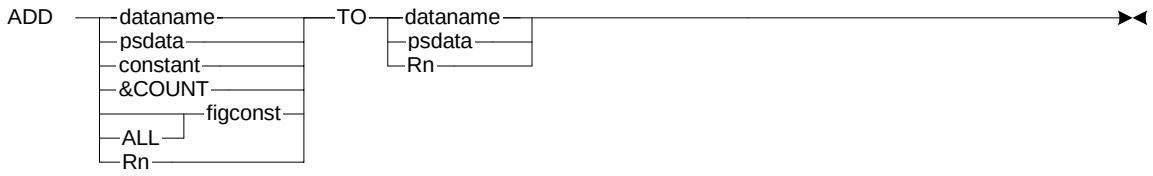
| Operand | Description |
|---|---|
| *psdata* | Specifies a unique name for the data item being defined. This name cannot be qualified. |
| *picture* | Specifies a standard COBOL picture clause that defines the data this data item contains. SmartTest supports these alphanumeric edit characters in the picture clause:<br>Alphanumeric - XXX or X(3)<br>Numeric - 999 or 9(3)<br>Signed numeric and/or implied decimal - S9V9(99) |
| USAGE IS *compword* | Specifies the standard COBOL computational reserved words used to describe data items used in arithmetic operations.<br>**Note:**<br>For COBOL II and later optimized nested programs, pseudo code data items should be defined only in the DATA DIVISION of the main program to avoid duplicate definitions caused by the optimization process. |

## Examples

These are examples of the 77 statement used to define pseudo code data items:

```
77 TEMP-COUNTER    PIC S9(7) COMP.
77 PSEUDO-DATA     PIC X(3).
```

# ADD Statement




## Function

The ADD statement adds the value contained in or represented by the first operand to the specified data item. The value is converted to the proper format for the data item.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, qualified PL/I variable name, or Assembler variable name. Dataname refers to any valid COBOL reference for a data element. |
| *psdata* | Specifies a level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| *constant* | Specifies a numeric literal. See "Using Pseudo Code" on page 512 for more information about entering pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. |
| ALL | Specifies readability and is used with a figurative constant (see figconst). |
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to test specific values. The figurative constants ZERO, ZEROS, and ZEROES, which may be further qualified with the ALL figurative constant, are supported. |
| R*n* | Specifies a register 0 through 15. |
| TO | Specifies a required keyword used as a connective for the ADD statement operands. |

### Examples

These are examples of the ADD statement used to add values to data items.

```
ADD 1 TO TEMP-COUNTER.
ADD TEMP-COUNTER TO RECORDS-READ-COUNT.
ADD &COUNT TO TEMP-COUNTER.
```

## BREAK Statement

```
BREAK ──────────────────────────────────────────────────────────────────────►◄
```

### Function

The BREAK statement forces a Breakpoint before a specific statement. A pseudo code BREAK statement causes an unconditional interrupt in the program execution.

### Operands

None.

### Usage Notes

You can add SmartTest commands to the Break command in pseudo code. If you know you want to execute a specific command or script at a certain breakpoint, you can type the command on the Break statement.

The commands must be enclosed in single quotes and can be any valid SmartTest command, included 'EXECUTE *script'*, for example:

```
000025      PERFORM VARYING D FROM 1 BY 1 UNTIL D > 4
000026          MOVE C (D) TO E
000027      END-PERFORM
'''''1      BREAK 'DISPLAY E'
```

To call a problem program with the monitor facility:

```
''''1      BREAK 'RUN NOMON'
000025      CALL program
'''''1      BREAK 'RUN MON'
```

You can also enter two commands on the Break statement. For example, if you want to execute a command without stopping the execution of the program, you could use the RUN command as the second command. The program would continue after the first command is performed as the example shows:

```
''''1       BREAK 'RESET TRACKING' 'RUN'
000025       PERFORM VARYING D FROM 1 BY 1 UNTIL D > 4
000026           MOVE C (D) TO E
000027       END-PERFORM
'''''1       BREAK 'LPRINT TRACKING' 'RUN'
```

## GO Statement

### Function

The GO statement transfers control to the statement containing the specified label, pseudo code label, or line number.

### Operands

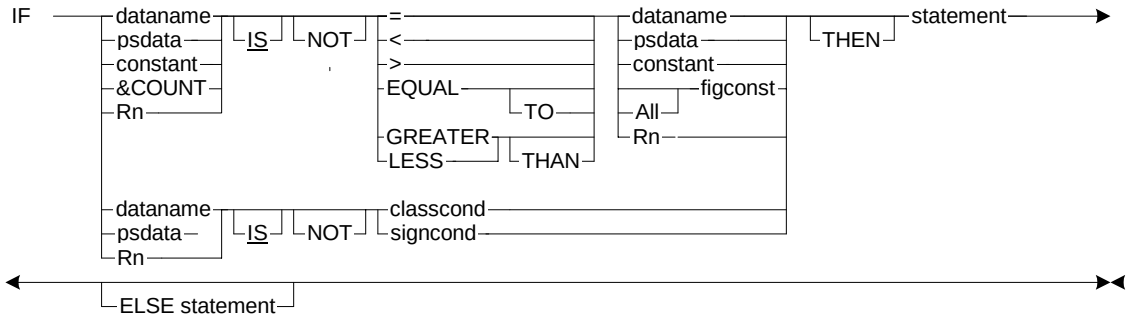
| Operand | Description |
|---|---|
| TO | Specifies an optional keyword used as a connective for the GO statement. |
| *label* | Specifies a standard COBOL paragraph or section name, Assembler label, or PL/I procedure or label. |
| pslabel | Specifies a pseudo code paragraph name. These names are entered in Area A and can be used to identify blocks of pseudo code. |
| *line* | Specifies a COBOL, PL/I, or Assembler source line number.<br><br>**Note:**<br>Transferring control to a label outside of the paragraph being performed may cause unpredictable results. Use of the GO statement on a program compiled with the OPTIMIZE compiler option can also cause unpredictable results. See the optimized Assembler output from the compiler as a navigational aid for testing optimized programs. |

## Examples

These are examples of the GO statement used to transfer control:

```
GO TO READ-INFILE.
GO TO PSEUDO-LABEL.
GO 434.
```

# IF Statement




## Function

The IF statement is used to test conditional expressions and if the condition is true, the imperative THEN statement is executed. If the condition is false, the imperative ELSE statement (if the ELSE statement has been specified) or the next sentence is executed. Nested IF statements are supported and each ELSE statement refers to the preceding IF statement.

These are the supported condition tests:

| Condition Test | Description |
|---|---|
| RELATION | Compares two operands to determine if they are equal, not equal, greater than, not greater than, less than, or not less than each other. |
| CLASS | Determines if a data item is alphabetic, not alphabetic, numeric, or not numeric. |
| SIGN | Determines if the value of a data item is negative, not negative, positive, or not positive. |

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, qualified PL/I variable name, or Assembler variable name. |
| *psdata* | Specifies a pseudo code dataname. |
| *constant* | Specifies a numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about entering pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. |
| R*n* | Specifies a register 0 through 15. |
| IS | Specifies the COBOL reserved word IS that is used as a connective for the IF statement. |
| NOT | Specifies the COBOL reserved word used to negate a condition. |
| Equals (=) | Specifies an equal or not equal comparison. |
| Less than (<) | Specifies a less than or not less than comparison. |
| Greater than (>) | Specifies a greater than or not greater than comparison. |
| EQUAL | Specifies an equal or not equal comparison. |
| TO | Performs an equal to or not equal to comparison when used with the EQUAL operand. |
| GREATER | Specifies a greater than or not greater than comparison. |
| LESS | Specifies a less than or not less than comparison. |
| THAN | Performs greater than, not greater than, less than, or not less than comparisons when used with the GREATER and LESS operands. |
| ALL | Specifies readability when used with a figurative constant (see "figconst" on page 524). |

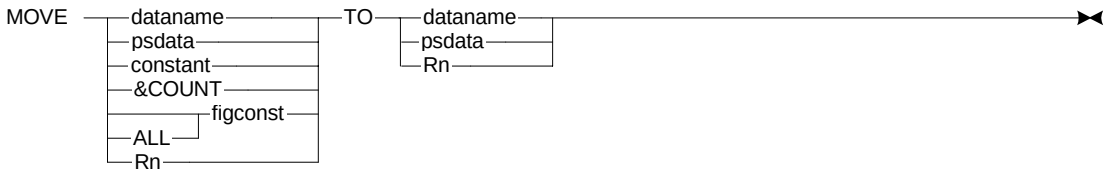
| Operand | Description |
|---|---|
| *figconst* | Tests specific values (figurative constants). These figurative constants, which can be further qualified with the ALL figurative constant, are supported:<br>HIGH-VALUE SPACES<br>HIGH-VALUES ZERO<br>LOW-VALUE ZEROS<br>LOW-VALUES ZEROES<br>SPACE |
| *classcond* | Specifies the COBOL reserved words used to test class conditions. The supported class conditions are ALPHABETIC and NUMERIC. |
| *signcond* | Specifies the COBOL reserved words used to test sign conditions. The sign conditions supported are NEGATIVE and POSITIVE. |
| THEN statement | Specifies that the statement is to be executed if the conditional test is true. |
| ELSE statement | Specifies that the statement following it is to be executed if the conditional test is false. If the ELSE statement is not specified, and the conditional test is false, the next sentence is executed. |
| ENDIF | Terminates an IF statement. |

## Examples

These are examples of the IF statement used to test conditional expressions:

```
IF TEMP-COUNTER IS LESS THAN 100 THEN
        GO TO PSEUDO-LABEL.
IF &COUNT=5 THEN
        MOVE 5 TO PSEUDO-DATA
ELSE
        GO TO PSEUDO-LABEL.
```

# MOVE Statement




## Function

The MOVE statement assigns the value contained in or represented by the first operand to the specified data item. The value is converted to the proper format for the data item if possible. If the value cannot be converted to the proper format, program execution stops and an error message displays.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, qualified PL/I variable name, or Assembler variable name. A COBOL dataname is any valid COBOL reference for a data element. |
| *psdata* | Specifies a level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| *constant* | Specifies a COBOL numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about entering pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. |
| ALL | Specifies readability when used with a figurative constant (see figconst). |

| Operand | Description |
|---|---|
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to move specific values. These figurative constants, which may be further qualified with the ALL figurative constant, are supported:<br>HIGH-VALUE SPACES<br>HIGH-VALUES ZERO<br>LOW-VALUE ZEROS<br>LOW-VALUES ZEROES<br>SPACE |
| R*n* | Specifies a register 0 through 15. |
| TO | Specifies a required keyword used as a connective for the MOVE statement operands. |

### Examples

These are examples of the MOVE statement used to move values to data items:

```
MOVE 20 TO TEMP-COUNTER.
MOVE ALL ZEROS TO RECORDS-READ-COUNT.
MOVE DATA-IN TO DATA-OUT.
```

## pslabel. Statement

pslabel. ──────────────────────────────────────────────────────────►◄

### Function

The pslabel. statement is used to define a pseudo code paragraph name. These names are entered in Area A (columns 8-11) and are referenced by a GO statement. A pseudo code label can consist of one to thirty alphanumeric characters, the first of which must be alphabetic. The specified name cannot be an existing COBOL data or label name. Pseudo code labels cannot be defined in WHEN command statements.

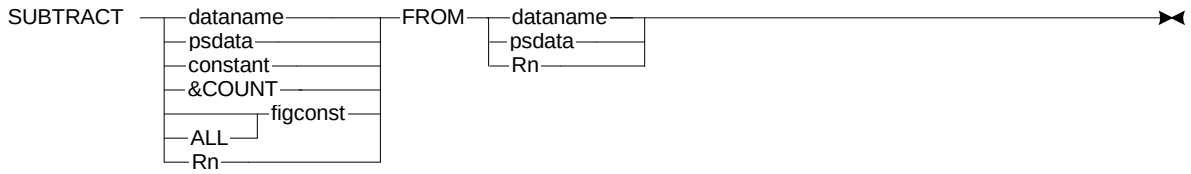
### Operands

None.

**Note:**

For COBOL II and later optimized nested programs, duplicate definitions may be caused by the compiler's optimization process.

# SUBTRACT Statement




## Function

The SUBTRACT statement subtracts the value contained in or represented by the first operand from the specified data item. The value is converted to the proper format for the data item.

## Operands

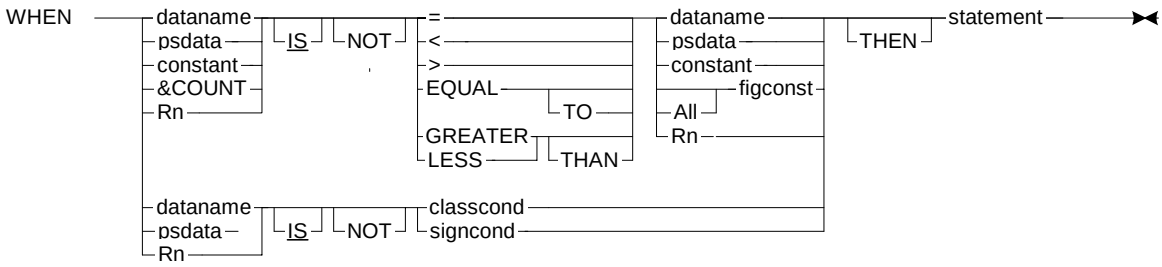
| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, or Assembler variable name. Dataname refers to any valid COBOL reference for a data element. |
| *psdata* | Specifies a level 77 data item that has been defined within a block of pseudo code. These data items must be unique and cannot be qualified. |
| *constant* | Specifies a numeric literal. See "Using Pseudo Code" on page 512 for more information about entering pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. |
| ALL | Specifies readability used with a figurative constant (see figconst). |
| *figconst* | Specifies the COBOL reserved words (figurative constants) used to test specific values. The figurative constants, ZERO, ZEROS, and ZEROES, which may be further qualified with the ALL figurative constant, are supported. |
| R*n* | Specifies a register 0 through 15. |
| FROM | Specifies a required keyword used as a connective for the SUBTRACT statement operands. |

## Examples

These are examples of the SUBTRACT statement used to subtract values from data items:

```
SUBTRACT 1 FROM RECORDS-READ-COUNT.
SUBTRACT &COUNT FROM TEMP-COUNTER.
SUBTRACT PSEUDO-DATA FROM DATA-IN.
```

# WHEN Statement




## Function

The WHEN statement is used to test conditional expressions and if the condition is true, the imperative statement following THEN is executed. Testing of conditional expressions is performed the same as in COBOL conditional expressions. Use of the WHEN statement causes the test to be performed after the execution of every COBOL verb. These are the supported condition tests:

| Condition Test | Description |
|---|---|
| RELATION | Compares two operands to determine if they are equal, not equal, greater than, not greater than, less than, or not less than each other. |
| CLASS | Determines if a data item is alphabetic, not alphabetic, numeric, or not numeric. |
| SIGN | Determines if the value of a data item is negative, not negative, positive, or not positive. |

**Note:**

WHEN conditions are ignored while in NOMONITOR mode.

## Operands

| Operand | Description |
|---|---|
| *dataname* | Specifies a COBOL dataname, qualified COBOL dataname, PL/I variable name, qualified PL/I variable name, or Assembler variable name. |
| *psdata* | Specifies a pseudo code dataname. |
| *constant* | Specifies a numeric or non-numeric literal. See "Using Pseudo Code" on page 512 for more information about entering pseudo code. |
| &COUNT | Specifies a SmartTest system variable used to track the number of times a pseudo code statement has been executed. |
| R*n* | Specifies a register 0 through 15. |
| IS | Specifies the COBOL reserved word IS. This operand is included in the WHEN command by default. |
| NOT | Specifies the COBOL reserved word NOT. This operand is used to test a not condition. |
| Equals (=) | Specifies an equal or not equal comparison. |
| Less than (<) | Performs a less than or not less than comparison. |
| Greater than (>) | Performs a greater than or not greater than comparison. |
| EQUAL | Performs an equal or not equal comparison. |
| TO | Performs an equal to or not equal to comparison when used with the EQUAL operand. |
| GREATER | Performs a greater than or not greater than comparison. |
| LESS | Performs a less than or not less than comparison. |
| THAN | Performs a greater than, not greater than, less than, or not less than comparison when used with the GREATER and LESS operands. |
| ALL | Specifies readability when used before a figurative constant. |

| Operand | Description |
|---|---|
| *figconst* | Specifies the COBOL reserved words (figurative constants) test specific values. These figurative constants are supported:<br>HIGH-VALUE SPACES<br>HIGH-VALUES ZERO<br>LOW-VALUE ZEROS<br>LOW-VALUES ZEROES<br>SPACE |
| *classcond* | Specifies the COBOL reserved words used to test class conditions. These class conditions are supported:<br>ALPHABETIC<br>NUMERIC |
| *signcond* | Specifies the COBOL reserved words used to test sign conditions. These sign conditions are supported:<br>NEGATIVE<br>POSITIVE |
| THEN statement | Specifies the THEN statement operand used to specify a statement to be executed if the conditional test is true. |

## Example

These are examples of the WHEN statement used to test conditional expressions:

```
WHEN &COUNT EQUAL TO 100 THEN
        MOVE DATA-IN TO DATA-OUT.
WHEN RECORDS-READ-COUNT = 50 THEN
        GO TO PSEUDO LABEL.
WHEN DATA-IN IS NOT ALPHABETIC
        BREAK.
```

# COBOL Reserved Words

Many standard COBOL reserved words can be used in pseudo code statements. These reserved words are used in pseudo code statements in the same manner in which they are used in standard COBOL programs. This table lists the COBOL reserved words that SmartTest supports.

| COBOL Reserved Words | | |
|---|---|---|
| ALL | ELSE | PIC |
| ALPHABETIC | EQUAL | PICTURE |
| COMP | FROM | POSITIVE |
| COMP-1 | GREATER | SPACE |
| COMP-2 | IS | SPACES |
| COMP-3 | LESS | THAN |
| COMP-4 | LOW-VALUE | THEN |
| COMPUTATIONAL | LOW-VALUES | TO |
| COIMPUTATIONAL-1 | NEGATIVE | USAGE |
| COMPUTATIONAL-2 | NEXT SENTENCE | ZERO |
| COMPUTATIONAL-3 | NOT | ZEROS |
| COMPUTATIONAL-4 | NUMERIC | ZEROES |

These symbols can be used in conditional pseudo code statements:

- = Equals
- < Less than
- > Greater than

# Pseudo Code Examples

Figure 217 shows an example of pseudo code statements.

**Figure 217 • Pseudo Code Examples**

```
146000     SUBTRACT 1.23      FROM 77-C4HV3.
146100     SUBTRACT 1.234     FROM 77-C4HV3.
'''''1
'''''2 * EXAMPLES OF PSEUDO CODE ALPHANUMERIC DATA DEFINITIONS.
'''''3 * ALPHANUMERIC FIELDS ARE INITIALIZED TO SPACES.
'''''4
'''''5  77  PSEUDO-CODE-DATA-X  PIC X(15).
'''''6  77  PSEUDO-CODE-DATA-Y  PIC X.
'''''7
'''''8 * EXAMPLES OF PSEUDO CODE LABEL DEFINITIONS.
'''''9
''''10  PSEUDO-CODE-LABEL.
''''11      IF S0C7 = 'N' THEN
''''12         GO TO PSEUDO-CODE-EXIT
''''13      ELSE
''''14         IF PSEUDO-CODE-DATA-X = SPACES THEN
''''15            MOVE HIGH-VALUES TO PSEUDO-CODE-DATA-X
''''16            GO TO PSEUDO-CODE-LABEL
''''17         ELSE
''''18            IF &COUNT = 1 THEN
''''19               GO TO PSEUDO-CODE-LABEL
''''20            ELSE
''''21               MOVE 'N' TO S0C7
''''22               GO TO PSEUDO-CODE-LABEL.
''''23  PSEUDO-CODE-EXIT.


146500     MOVE 0 TO 77-C3V2
146600     MOVE 0 TO 77-C3V3.
'''''1
'''''2 * EXAMPLES OF PSEUDO CODE NUMERIC DATA DEFINITIONS.
'''''3 * NUMERIC FIELDS ARE INITIALIZED TO ZERO.
'''''4  77  PC-FLOAT1            COMP-1.
'''''5  77  PC-FLOAT2            COMP-2.
'''''6  77  PC-PACKED  PIC S9(5) COMP-3.
'''''7  77  PC-BINARY  PIC S9(4) COMP.
'''''8
'''''9 * THESE ARE EXAMPLES OF PSEUDO CODE NUMERIC DATA OPERATIONS.
''''10
''''11      BREAK.
''''12      IF PC-PACKED IS LESS THAN 1 THEN
''''13         MOVE +12345 TO   PC-PACKED
''''14         ADD      +1 TO   PC-PACKED
''''15         SUBTRACT +1 FROM PC-PACKED
''''16         IF PC-PACKED NOT = +12345 THEN
''''17            BREAK.
```

# Glossary

### ACB
See application control block.

### action bar
The line of keywords at the top of a screen. Each keyword represents a category of actions that may be performed on that screen. An action is selected by moving the cursor to the desired keyword and pressing Enter.

### active program
A program that is being viewed and/or tested on the Program View screen. Also see qualified program.

### address command
A command that is entered in the hexadecimal address area within the status box or in any address or offset field displayed on a SmartTest screen. These commands are used to display specific areas of memory such as the current word, 24-bit address, or the 31-bit address. A message can also be displayed that indicates to where the 24-bit or 31-bit address points. See Figure 90 on page 145 for more information.

### address stop
An absolute address and length of storage that is to be monitored during a test session. A program interrupt occurs before the specified area is actually updated.

### AKR
See Application Knowledge Repository.

### alias name
The name of a program entry point. Alias names are shown on the AKR Directory and Module Directory screens.

### alias of
A field on a pop-up listing entries in the AKR. If the analyzed program contains and ENTRY point, Alias Of is the name of the program which contains the ENTRY point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias Of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.

### analyze
The process used by SmartTest to prepare a COBOL, PL/I, or Assembler program for testing. See analyzer.

### analyzer
The batch component of SmartTest that executes complex algorithmic formulas to analyze the complexities of a COBOL or Assembler program. It produces a detailed analysis of the elements and relationships for a program, then places this information in the Application Knowledge Repository (AKR).

### analyze options
Run-time options that control the Program Analyzer processing. Many of these options are similar to the COBOL compiler options. Default values are established at installation time and can be overridden by editing the Analyzer JCL or by using the Analyze screens. The Analyze Options appendix contains a complete description of each Analyze option. Analyze Options are not valid for the Assembler Analyzer.

### analyzer summary
A summary of the run-time statistics and diagnostic messages that are produced when an Analyze job completes.

### application control block
(ACB) - A control block for IMS/DC. This control block contains the PSB and DBD, and is output from the ACBGEN process.

### Application Knowledge Repository
The Application Knowledge Repository component of SmartTest. The AKR is a BDAM or VSAM file organization that contains all analysis information produced by the Program Analyzer or Assembler Analyzer. Multiple AKRs can be defined. SmartTest supports concatenated AKRs.

### application plan
See plan name.

### ASG-Insight
An ASG proprietary product that provides in-depth analysis of COBOL programs. SmartTest can be interfaced with Insight and can utilize the comprehensive analysis features it provides.

### Assembler analyzer
See analyzer.

### Batch terminal simulator
(BTS) - An IBM product that allows execution of IMS/VS DB/DC applications in a TSO or batch environment.

### Batch test session
A SmartTest test session that is established in an MVS batch region, to which an online test session can be connected. This feature allows batch program testing to be performed interactively.

### breakpoint
An interruption that occurs during the execution of a program being tested. Breakpoints result from a BREAK command or an error condition.

### BTS
See Batch terminal simulator.

### COBOL subset
COBOL verbs of a similar nature that have been grouped together. For example, READ, WRITE, OPEN, and CLOSE are grouped into the IO subset. The LIST SUBSETS command can be entered to display all subsets online. See the *ASG-SmartTest for COBOL and Assembler User's Guide* for a description of each COBOL subset.

### command input area
The field on SmartTest screens where primary commands are entered, indicated by ===> on the second line of the screen.

### current cursor location
Refers to the COBOL source statement where command processing begins. The current cursor location can be one of these based on the cursor position:

If the cursor is in the command input area, the current location is the first source code line on this screen.

If the cursor is in the line command input area, the current location is the beginning of that line.

If the cursor is in the source code area, the current location is the cursor position.

### cursor character
A substitution character that can be used in commands. The command and cursor character are typed in the command input area, the cursor is placed on the desired token, and Enter is pressed. SmartTest locates the cursor and reads the specified token as part of the command. The cursor character is set on the Options - Product Parameters pop-up. See "Options - Product Parameters Pop-up" on page 187 for more information.

### dataname
A standard term for data fields defined in the program. For example, variable names, files, groups, and array elements can be a dataname, depending on the type of program.

### data usage
Defines how a data item is used: DEF indicates the statements in the DATA DIVISION where the data item is defined; USE indicates the statements where the value is used or tested; MOD indicates the statements where the value is set or modified; REF indicates any of the above conditions.

### database description (DBD)
A control block that describes the physical structure of a database in an IMS environment.

### DBCS
See Double Byte Character Set (DBCS).

### DBD
See database description (DBD).

### DB2
An IBM relational database management system.

### DB2 Stored Procedure
A user-written program that resides on a DB2 server and is invoked by an EXEC SQL CALL statement. A stored procedure is a 3GL Language Environment compliant program written in COBOL, C/370, PL/I, or Assembler. The SmartTest DB2 Stored Procedure option enables programmers/analysts to use SmartTest features to interactively test DB2 Stored Procedures.

### decision option
Choices listed on the Trace Decision Menu. Decision options represent the branches in the path that the TRACE command is following.

### decision point
A statement that causes a branch in the execution path of the program. It could be a PERFORM, conditional statement, exception condition, input statement, GO TO or LABEL.

### debug
The process of locating and correcting program errors.

### diagnostic message
An informational or error message generated by the online and batch components. Online - A short message displays in the upper right corner of the screen (if available). A long message displays when you press PF1/13 or type HELP in the command input area. Batch - Messages are included in the Analyzer Summary.

### DL/I | DL/1
The database (DB) portion of the IMS system.

### Double Byte Character Set (DBCS)
A character set that uses two bytes to represent each character. Various Double Byte Character Sets are used with languages such as Chinese and Japanese which cannot be represented with single byte codes.

### equate
A substitution name for a character string. The substitution name is created using the EQUATE command. Long commands, patterns, datanames, etc. can be equated to a substitution name.

### IMS
An IBM hierarchical database management system.

### label name
Any PROCEDURE DIVISION paragraph or section name and the PROCEDURE and PROC literals.

### line command
An abbreviated keyword that is entered in the line command area (columns 1-6) on the screen.

### list file
A file that is allocated the first time the LPRINT command is executed. This file is used for LPRINT output.

### log file
A file that is allocated upon entry into SmartTest. This file is used for error messages and log commands.

### long message
A diagnostic or error message that displays near the bottom of SmartTest screens. Long messages are sometimes preceded by short messages that are displayed in the upper right corner of the screen. Pressing PF1/13 after receiving a short message displays the corresponding long message.

### member
A member in a PDS or source manager such as Panvalet or Librarian. This is the alias name found in the Application Knowledge Repository (AKR).

### module
A link edited member of a load library PDS. A module can contain several programs.

### monitored storage
The data monitored by specifying the absolute addresses and length of storage areas on the Address Stop Entry screen. All storage areas listed on this screen are monitored by SmartTest and a program interrupt occurs before a specified storage area is actually updated.

### network
The result of a FLOW command. It consists of all possible executable statements that reach from the starting point of the command to all possible targets. The next FLOW command will replace the previous network with the new results.

### path
Any group of executable statements that describe possible execution flows of the COBOL program. FLOW creates paths called NETWORK, and TRACE creates a path called TRACK.

### PCB
See Program Communication Block (PCB).

### perform range
A Perform range consists of the source code contained in a PERFORM statement, and includes all code that is executed as a result of GO TOs, PERFORMs, etc. within that PERFORM.

### plan name
A DB2 application plan that contains information about a program and the data used by the program. A DB2 plan is the output of the BIND process.

### pop-up
A window that displays as the result of selecting an item on a pull-down or pop-up, or as the result of entering certain commands. It is superimposed on the screen to allow entry of information for the requested action.

### primary command
A series of keywords that are entered in the command input area of the screen.

### program
Program source member name, the name specified in the IDENTIFICATION DIVISION of a COBOL program, or the CSECT name of a program that is not COBOL.

### program analyzer
See analyzer.

### Program Communication Block (PCB)
A definition of the database view used by an application program in an IMS environment.

### Program Specification Block (PSB)
All Program Communication Blocks (PCBs) are contained in a PSB. PCBs define the view of a database to an application program in an IMS environment.

### PSB
See Program Specification Block (PSB).

### pseudo code
Statements that are temporarily inserted into program source during a test session. The syntax of this code is compatible with COBOL.

### pull-down
The list that displays when an action is selected on the action bar. On a pull-down, actions that are followed by ... display a pop-up when selected. Actions not followed by ... will immediately activate internal commands.

### punch file
A file that is allocated the first time the LPUNCH command is executed. This file is used for LPUNCH output.

### qualified program
A program that displays on the Program View screen using the QUALIFY command, while another program is being viewed and/or tested. The program specified with the QUALIFY command is a qualified program; the program being viewed and/or tested is an active program.

### SBCS
See Single Byte Character Set (SBCS).

### screen subsets
Lines that have been acted upon by an interactive command that have caused them to be in one of these screen display set types:

| | |
|---|---|
| Highlighted \| HI | Excluded \| X |
| NONHighlighted \| NHI | NONExcluded \| NX |

### screen transfer feature
This feature can be used on any SmartTest screen that displays a screen number in the upper right corner. Users can immediately transfer to a numbered screen from any other SmartTest screen by entering a = followed by the desired screen number in the command input area.

### script file
A sequential file that contains a predefined sequence of commands that can be read and executed.

### short message
A diagnostic or error message that displays in the upper right corner of SmartTest screens. Pressing PF1 after receiving a short message displays the corresponding long message.

### Single Byte Character Set (SBCS)
A character set that uses one byte to represent each character. Single Byte Character Sets are used with languages such as English where the characters can be represented with a one-byte code.

### status box
A box that displays at the bottom of SmartTest screens during a test session. Current status information for the program being tested is shown. This information includes the statement number, offset, source statement, program name, module name, date, and time. When the Assembler mode is set ON, the status box also shows the AMODE value, Assembler instructions, hexadecimal value, and the general register contents. The status box can be removed from the screen using the SET STATUS OFF command.

### Storage Management Subsystem (SMS)
An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

### Stored Procedure
See DB2 Stored Procedure.

### subnet
A portion of the NETWORK that reaches from the starting point to one result. Created by the FLOW command as part of the NETWORK. Each portion of the NETWORK that reaches a target is assigned a number (SUB1...SUB*n*).

### subset
A COBOL subset, screen subset, or tagged lines subset. See the *ASG-SmartTest for COBOL and Assembler User's Guide* for detailed information about subsets.

### symbol extractor
A post-compile batch processor that extracts COBOL symbol and verb information from the COBOL compiler source listing for input into the Program Analyzer.

### tagged lines subsets
Information shown in columns 73 through 80 as the result of a FLOW or TRACE command. These tags can be used as subsets in commands that accept subsets as targets:

| | |
|---|---|
| TAGged \|TAGS | TARGet \| TGT |
| DECision | STArt |
| OPTions | |

### target
An object of a SmartTest primary command.

### TCA
See Test Coverage Analysis.

### Test Coverage Analysis
SmartTest-TCA, Test Coverage Analysis, is a testing option that measures test suite coverage for a set of programs. It fosters a systematic approach to assess the quality of testing performed in the internal operation of programs. Test Coverage Analysis is a tool for application systems testing and quality assurance testing--to help ensure that the program routines are covered by data execution (i.e., to what extent the data exercises the program). Test Coverage Analysis records the program's reaction to the data presented, without user intervention (in order to preserve the integrity of the results). For detailed information, see the *ASG-SmartTest TCA User's Guide*.

### token
A contiguous set of characters preceded by and followed by a blank, period, comma or parenthesis.

### TRACK
The result of a TRACE command. TRACK is the set of lines captured while moving through program logic selecting the branch options.

### TSO test session
A SmartTest test session that is executed in a TSO/ISPF environment. Batch JCL is converted to an allocation CLIST and is used to establish the TSO test session.

# Index


## Symbols

$$SEGMENTS program name 126
& command 224
&COUNT 516, 519, 523, 525, 527
. (Label) line command 467

## Numerics

77 statement 517

## A

A (After) line command 468
Access Registers screen
    example 121
    field descriptions 121
active program 369
ADD command 225
ADD statement 519
address
    absolute 145
    commands 151
    stop entries 429
Address Stop Entry screen
    example 122
    field descriptions 123
addressing mode 154, 162
AKR
    allocating 14, 17
    available space in 16
    expanding 14, 17
    list programs in 7, 15
    listing members 124
    metrics member 126
    profile member 9, 126
    renaming a program in 14
alias names 152–153
Alliance 48, 79–80, 226, 438
    accessing from ESW screen xvii
    description xiv
    linking xvii
ALLIANCE command 226
Alliance queries 81
allocating an AKR 14
ALLOCDEF command 226–227
AMODE 154
analysis types 8, 16, 125
analyze
    Insight 8, 17, 125
    Recap 9, 17, 125
    SmartDoc 8, 17, 125
    SmartDoc Extended 8, 17, 125
    SmartTest 9, 17, 126
    VIASUB edit macro 11
ANALYZE command 228
ASG-SmartDoc Options screen 11
ASG-SmartTest/ASG-Alliance Interface
    pop-up
    example 80
    field descriptions 80
Assembler
    instruction level 410
    program 8, 17, 125
    related keywords 149
authorization code 152–153
AutoChange 79, 438
    accessing from ESW screen xvii
    description xiv

## B

B (Before) line command 469
BACKTRACK
    operand 411, 425
    run 396
BackTrack Variable History pop-up
    example 127
    field descriptions 127
BackTrack Variable History screen
    example 128
    field descriptions 128
Batch test session 399
BR (Break) line command 470, 514
BRANCH command 229
Branch Previous Options pop-up
    example 111
    field descriptions 111

branching logic 230
Break - Data Name pop-up
example 64
field descriptions 65
Break - Label/Paragraph Name pop-up
example 67
field descriptions 67
Break - Line Number pop-up
example 66
field descriptions 66
Break - on Subset pop-up
example 68
field descriptions 68
Break - Pattern pop-up
example 71
field descriptions 71
Break - Program Name pop-up
example 69
field descriptions 70
BREAK command 167, 514
break on entry and return 167
BREAK statement 520
breakpoints 167, 232, 234, 238–239, 412, 421, 470, 520
Breakpoints List screen 238
example 130
field descriptions 130
Bridge
accessing from ESW screen xvii
description xiv

## C

C (Copy) line command 471, 514
CANCEL command 241
CC (Copy Block) line command 472, 514
Center, description xiv
character string 236, 260, 331
class condition 452, 454, 522, 524
COBOL
intelligent search 279
statement level 410
syntax rules 348
unexecutable statement 222, 298
verbs 515
COBOL II, related keywords 146
COBOL reserved word
ALPHABETIC 454, 524
HIGH-VALUE 454
HIGH-VALUES 454
IS 453, 523
LOW-VALUE 454
LOW-VALUES 454
NEGATIVE 454, 524
NOT 453, 523
NUMERIC 454, 524
POSITIVE 454, 524
THAN 453, 523
TO 453, 523
ZEROES 454
ZEROS 454
commands
& (Retain) 224
ADD 225
Address 151
ALLIANCE 226
ALLOCDEF 226–227
ANALYZE 228
BRANCH 229
BREAK 167, 381, 514
CANCEL 241
CONVERT 242
COPY 243
CURRENT 246
DELETE 247
diagram syntax 216
DISPLAY 250
DROP 253
DUMP 254
END 258
ENVIRONMENT 259
EQUATE 138, 260
EXCLUDE 262
EXECUTE 270, 415
FIND 274
FINDXTND 262, 279
FLOW 292, 294
FORCE 299
GO 300
HELP 301
HIGH 303, 384
input area 215, 221, 293, 297, 379
KEEP 310
KEYS 215, 314
line 215, 380, 464, 514
LIST 316
LIST AKRMEMBERS 124
LIST MEMORY 143
LIST PROFILE 157
LIST PSEUDO 160
LIST REGISTERS 162
LIST SUBSETS 164
LIST TRACKING 168
LOCATE 330
LPRINT 333
LPUNCH 342, 516
MARK 350
MERGE 353
MOVE 356

NEWCOPY 357
PARMDEF 358
PREF 359
primary 215, 224, 272, 514
PRINTLOG 362
PRINTLST 363
PROCESS 364
processing 216
PRODLVL 368
QUALIFY 228, 369
RECALL 216, 371
recalling 371
REDO 374
REFRESH 376
RENAME 377
REPEAT 379
repeating 219, 379, 383
RESET 380
RETURN 382
RFIND 383
RHIGH 384
RPREF 385
RSCROLL 386
RTRACE 387
RUN 369, 390, 515
RUN (BACKTRACK ON) 396
RUN (CICS) 393
SAVE 399
SCROLL 401
SELECT 408
sequence 216
SET 238, 409
SET MONITOR 390
SETUP 419
SHOW 420
starting point 222
STEP 369, 421, 515
STEP (BACKTRACK ON) 425
STOP 429
SUBTRACT 431
TCA DEFINE 432
TCA LIST 433
TCA RECORD 434
TCA REPORT 435
TCA RUN 436
TEST 437
TESTPOINT 438
TOGGLE 439
TRACE 375, 440
UPDATE 446
USING 448
UTILITY 450
VIEW 451
WHEN 330, 452, 514–515
WHERE 456
WIZARD 458
ZOOMDATA 459
ZOOMIN 462
ZOOMOUT 462
comment
line 511
statements 222, 298
compiler directives 222, 298
Compiler Options screen
example 134
field descriptions 134
conditional
expression 452, 454, 522
statements 516
test 239, 452, 454, 522, 524
conventions page xxi
CONVERT command 242
COPY command 243
copy members 376
CURRENT command 246
current location 379, 383
cursor
location 293
position 222
substitution character 222

## D

D (Delete) line command 238, 473, 514
data definition 513, 517
DATA DIVISION hierarchy 508
dataname
address 500
ALIAS 235
COBOL 234
concatenated 260
hexadecimal address 506
hexadecimal value 487, 506
INDIRECT 235, 265, 282, 306, 336, 345, 404
MOD 235
NOALIAS 235
REF 234
USE 235
value 500
DB2 Stored Procedure 536
DCA (dispatch control area) 147
DCP 149
DCT (destination control table 147
DD (Delete Block) line command 474, 514
decision options 375
decision point 375, 444
declaratives 295
DELETE command 247

direction
All 237
FIrst 237
LAst 237
next 237
prev 237
disassembled format 154
DISPLAY command 250
DROP command 253
dump area 151
DUMP command 254
DWE (deferred work area) 147

**E**

EIB (EXEC Interface Block) 147, 318
EIS (EXEC Interface Structure) 147
Encore 11
accessing from ESW screen xvii
description xv
END command 258
entry point address 152, 154
Environment - AKR Directory pop-up
example 124
field descriptions 125
ENVIRONMENT command 259
EQUATE command 138, 260
equates, saving 399
Estimate 79, 438
accessing from ESW screen xvii
description xv
ESW
description xiii
invoking products xvi
product integration xvii
EXCLUDE command 262
excluded lines 380, 415, 489, 496–497
executable code block 222, 298
EXECUTE command 270, 415
execution
flow 293, 297
path 292, 294
tracking 168
Execution Counts screen
example 135
field descriptions 136
Execution Tracking screen 169
example 169
field descriptions 169

**F**

F (First) line command 475
FCP 149
FCT (File Control Table) 147
figurative constants 454, 519, 524, 526
File - AKR Allocate/Expand pop-up
example 18
field descriptions 19
options 18
File - AKR Directory pop-up
example 7, 15
field descriptions 7, 15
File - AKR Utility pop-up
example 13
field descriptions 14
options 14
File - Analyze Submit pop-up
example 10
field descriptions 11
options 11
File - Execute Script pop-up
example 22
field descriptions 23
File - Open Program (Qualify to Program) pop-up
example 6
field descriptions 6
File - Setup Test Environment pop-up
example 3
options 4
File - TCA Test Plan Selection pop-up 2
example 5
field descriptions 5
File pull-down 2
FIND command 274
FINDXTND command 262, 279
FIO (file I/O area) 147
floating point registers 412
Floating Point Registers screen
example 139
field descriptions 140
FLOW command 292, 294
FORCE command 299
FWA (file work area) 147

**G**

general registers 145, 162, 415
General Registers screen
example 162
field descriptions 163
GO command 300
GO line command 476
GO statement 521, 526
GO TO target 229

**H**

H (Highlight) line command 478

Help - About pop-up
example 184
field descriptions 184
Help - Specific ASG Command pop-up
example 181
field descriptions 181
Help - Specific ASG Message pop-up
example 182
field descriptions 182
HELP command 301
Help Explanation and Action Panel 301
help messages 302
Help pull-down 180
Help Tutorial screen 301
hexadecimal offset 330
HH (Highlight Block) line command 479
HIGH command 303
highlighted set 309
highlighting 285, 478–479

**I**

I (Insert) line command 480, 514
ICE (interval control area) 147
ICP 149
IF statement 522
impact dataset 47, 79, 438
imperative statement 452, 516, 522
IN-Clause Option pop-up
example 112
field descriptions 113
information tags 444
Insight
accessing from ESW screen xvii
description xv
using analysis functions xvii
ISPF Edit session 516
ISPF/PDF reset options 380

**J**

JCA (journal control area) 147

**K**

K (Keep) line command 481
KCP 149
KEEP command 310
KEYS command 215, 314
keywords 146
KG (Keep Group) line command 483
KGH (Keep Group Hexadecimal) line command 485
KH (Keep Hexadecimal) line command 487

**L**

L (Last) line command 489
language types 154
LEARN Mode - Generated Command pop-up 418
line commands
. (Label) 467
A (After) 468
area 222, 293, 297
B (Before) 469
block format 464
BR (Break) 470, 514
C (Copy) 471, 514
CC (Copy Block) 472, 514
D (Delete) 238, 473, 514
DD (Delete Block) 474, 514
F (First) 475
GO 476
H (Highlight) 478
HH (Highlight Block) 479
I (Insert) 480, 514
K (Keep) 481
KG (Keep Group) 483
KGH (Keep Group Hexadecimal) 485
KH (Keep Hexadecimal) 487
L (Last) 489
M (Move) 490, 514
MM (Move Block) 491, 514
R (Repeat) 492, 514
RR (Repeat Block) 493, 514
S (Show) 494
single format 464
SS (Show Block) 495
X (Exclude) 496
XX (Exclude Block) 497
ZA (Zoom Assembler) 381, 498
ZD (Zoom Data) 381, 500
ZG (Zoom Group) 465, 502
ZGH (Zoom Group Hexadecimal) 465, 504
ZH (Zoom Hexadecimal) 381, 506
ZI (Zoom In) 508
ZO (Zoom Out) 509
link edit
attributes 152–154
AUTH 154
execution date 154
OVLY 154
RENT 154
REUS 154
RMODE 154
TEST 154

List - CALL Statements pop-up
example 132
field descriptions 133
List - CICS Features pop-up
example 172
field descriptions 172
List - COBOL Features pop-up
example 173
field descriptions 173
List - Equates pop-up
example 138
field descriptions 138
List - Execution Counts pop-up
example 174
field descriptions 174
List - Execution Tracking pop-up
example 175
field descriptions 176
List - Perform Range Names pop-up
example 155
field descriptions 156
List - Program/Subprogram Names screen
example 159
field descriptions 159
List - Registers pop-up
example 177
field descriptions 177
List - Subset Names screen
example 164
field descriptions 164
List - User Marks pop-up
example 142
field descriptions 142
LIST AKRMEMBERS command 124
LIST command 316
List file 89, 91–92, 95, 98, 100, 102, 104, 106, 108, 333
LIST MEMORY command 143
LIST PROFILE command 157
LIST PSEUDO command 160
List pull-down 117
LIST REGISTERS command 162
LIST SUBSETS command 164
LIST TRACKING command 168
LLA (load list area) 148
load libraries 154
Load Module Directory screen
example 152
field descriptions 153
Load Module Intercept List pop-up
example 140
field descriptions 141
LOCATE command 330
locate WHEN 330
log file 362
LPRINT command 333
LPUNCH command 342, 516

## M

M (Move) line command 490, 514
MARK command 350
Mark name
deleting 247
merging 353
operand 353
saving 399
memory
locations 145
modifications 145
Memory Display screen
example 145
field descriptions 145
MERGE command 353
messages, recalling 371
MM (Move Block) line command 491, 514
Module Map screen
example 153
field descriptions 154
module size 152–154
MOVE command 356
MOVE statement 525
multiple verbs 511

## N

nested IF 522
NEWCOPY command 357
NOT condition 523

## O

offsets 330
offsets and line numbers, displaying 169
OPF (optional feature list) 148
Options - COPY/Include Libraries screen
example 20
field descriptions 21
Options - Log/List/Punch Definition pop-up
example 190
field descriptions 191
options 190
Options - Modes screen
example 195
field descriptions 196
Options - PF Key Definition pop-up
example 193
field descriptions 194

Options - Product Allocations pop-up
displaying 227
example 188
field descriptions 189
Options - Scratchpad Copy pop-up
example 205
field descriptions 206
Options - Scratchpad Delete pop-up
example 207
field descriptions 207
Options - Scratchpad Equate pop-up
example 203
field descriptions 203
Options - Scratchpad Mark pop-up
example 204
field descriptions 204
Options - Scratchpad Merge pop-up
example 207
field descriptions 208
Options - Scratchpad pop-up
example 202
field descriptions 202
Options - Scratchpad Rename pop-up
example 209
field descriptions 209
Options - Script File Allocations pop-up
example 192
field descriptions 193
Options pull-down 186

## P

PAM (page allocation map) 148
PARMDEF command 358
pattern string 236, 260
PCP 149
PCT (program control table) 148
PERFORM statement 229
PF keys 215, 314
PIP 149
PLB (partition lower boundary address) 148
PPT (processing program table) 148
PREF command 359
PRINTLOG command 362
PRINTLST command 363
PROCEDURE DIVISION
hierarchy 508
label 229
PROCESS command 364
PRODLVL command 368
product integration xvii
product level 368
Product Parameters pop-up
example 187
field descriptions 188
Profile Data Set Member List screen
example 157
field descriptions 157
program
active 369
hierarchical levels 462
interrupt 167, 232, 238, 332, 421, 470, 511, 513, 515
qualified 369
structure 462, 508
summaries 376
Program Status Word (PSW) 162
Program View screen 2
pseudo code
77 statement 517
activate 512
active flag 167
ADD statement 519
area A 513, 517
area B 513
BREAK statement 520
COBOL reserved words 531
commands 516
comment line 513
copying to List file 338
data definition 513
deactivate 512
deleting 381
displaying 160
entering during interrupt 232
entering during test session 480
entering into source 513
examples 532
execution 515
facility 414
GO statement 521, 526
IF statement 522
incompatible with COBOL 348
label 526
locate line 330
MOVE statement 525
Pslabel statement 526
saving 516
sequence numbers 513
SUBTRACT statement 527
symbols 531
syntax 511
updating source 516
usage 512
verbs 381
WHEN statement 528
Pseudo Code List screen
example 160
field descriptions 160

pseudo hexadecimal data 151
Pslabel statement 526
PUB (CICS partition upper boundary address) 148
pull-down
File 2
Help 180
List 117
Options 186
Search 85
Test 46
View 26
Punch file 89, 91, 93, 95, 98, 100, 102, 104, 106, 108, 342, 516

**Q**

qualified program 369
QUALIFY command 228, 369
queries in Alliance 81

**R**

R (Repeat) line command 492, 514
RECALL command 216, 371
Recap
accessing from ESW screen xvii
description xv
REDO command 374
REFRESH command 376
registers, general 162
relation condition 452, 522
relative offset 154
release level 368
RENAME command 377
REPEAT command 379
RESET command 380
RETURN command 382
RFIND command 383
RHIGH command 384
RPREF command 385
RR (Repeat Block) line command 493, 514
RSA (register storage area) 148
RSCROLL command 386
RTRACE command 387
RUN (CICS) command 393
RUN command 369, 390, 396, 515

**S**

S (Show) line command 494
SAVE command 399
SCP 149
screen
Access Registers 121
Address Stop Entry 122
ASG-SmartDoc Options 11
ASG-SmartTest/ASG-Alliance Interface 80, 226
Branch Previous Options 111
Break - COBOL Subset Name 68
Break - Data Name 64
Break - Label Name 67
Break - Line Number 66
Break - Pattern 71
Break - Program Name 69
Breakpoints List 130, 238
Compiler Options 134
Environment - AKR Directory List 124
Environment Selection 259
Execution Counts 135
Execution Tracking 169
File - AKR Allocate/Expand 17
File - AKR Directory 7, 15
File - AKR Utility 13
File - Analyze Submit 10, 228
File - Execute Script 22
File - Open Program (Qualify to Program) 6
File - Setup Test Environment 3
File - TCA Test Plan Selection 2, 5
Floating Point Registers 139
General Registers 162
Help - Specific ASG Command 181
Help - Specific ASG Message 182
Help About 184
Help Explanation and Action Panel 301
Help Tutorial 301
IN-Clause Option 112
LEARN Mode - Generated Command 418
List - BackTrack Variable History 127–128
List - CALL Statements 132
List - CICS Features 172
List - COBOL Features 173
List - Equates 138
List - Execution Counts 174
List - Execution Tracking 175
List - Perform Range Names 155
List - Program/Subprogram Names 159
List - Registers 177
List - Subset Names 164
List - User Marks 142
Load Module Directory 152
Load Module Intercept List 140
Memory Display 145, 163

Module Map 153
Options - COPY/Include Libraries 20
Options - Log/List/Punch Definition 190
Options - Modes 195, 410
Options - Parameter Definition 258
Options - PF Key Definition 193
Options - Product Allocations 188, 227
Options - Scratchpad 202
Options - Scratchpad Copy 205
Options - Scratchpad Delete 207
Options - Scratchpad Equate 203
Options - Scratchpad Mark 204
Options - Scratchpad Merge 207
Options - Scratchpad Rename 209
Options - Script File Allocations 192
Product Parameters 187
Profile Data Set Member List 157
Program View 2, 238, 310, 511
Pseudo Code List 160, 512, 516
Save Options 238
Search - Any/Unknown Type 107
Search - Branch Request 109
Search - COBOL Subset 96
Search - Data Name 86
Search - IN-clause 100, 102, 104, 106
Search - Label Name 89
Search - Line Range 105
Search - Paragraph Name 91
Search - Pattern String pop-up 93
Search - Perform Range Name 99
Search - Program Name 101
Search - User Mark Name 103
Selection List 112
Session Tailoring 9, 126, 512
Test - Add Request 76
Test - ASG-SmartTest Testpoint Generation 79
Test - Break Request 63
Test - Go Request 74
Test - Move Request 75
Test - Newcopy Request 81
Test - Run Request 48
Test - Run Request (Backtrack Recording Mode Active) 49
Test - Run Request (Backtrack Review Mode Active) 51
Test - Run Request (CICS - Backtrack Recording Mode Active) 53
Test - Run Request (CICS Environment-Test active) 52
Test - Step Request 54
Test - Step Request (Backtrack Recording Mode) 56
Test - Step Request (Backtrack Review Mode) 58
Test - Step Request (CICS - BackTrack Recording Mode Active) 61
Test - Step Request (CICS Environment-Test active) 59
Test - Stop Request 73
Test - Subtract Request 77
Test - Where Request 78
Test Facilities List 120
Test Session Options 512
Test Session Tailoring 165
View - Display Request 35
View - Drop Request 44
View - Exclude Request 42
View - Execution Counts 28
View - Execution Tracking 29
View - Keep Request 34
View - Memory 30
View - Paragraph Cross Reference 38, 221
View - Paragraph Cross-Reference Request 36
View - Qualify Request 31
View - Reset Request 40
View - Using Request 43
View - Zoom Data Request 32
When Conditions List 170, 514
script file 272, 415
SCROLL command 401
scrolling 481
Search - Any/Unknown Type pop-up
example 107
field descriptions 107
Search - Branch Request pop-up
example 109
field descriptions 109
Search - COBOL Subset pop-up
example 96
field descriptions 97
Search - Data Name pop-up
example 86
field descriptions 87
Search - IN-clause pop-up 100, 102, 104, 106
Search - Label Name pop-up
example 89
field descriptions 90
Search - Line Range pop-up
example 105
field descriptions 105

Search - Paragraph Name pop-up
example 91
field descriptions 92
Search - Pattern String pop-up
example 93
field descriptions 93
Search - Perform Range Name pop-up
example 99
field descriptions 99
Search - Program Name pop-up
example 101
field descriptions 101
Search - User Mark Name pop-up
example 103
field descriptions 103
Search pull-down 85
SELECT command 408
Selection List pop-up 112
Session Tailoring screen 9, 126
SET command 238, 409
SET MONITOR command 390
SETUP command 419
SHOW command 420
sign condition 452, 454, 522, 524
SIT (system initialization table) 148
SmartDoc
accessing from ESW screen xvii
description xv
SmartEdit
accessing from ESW screen xvii
description xvi
SmartQuest
accessing from ESW screen xvii
description xvi
SmartTest
accessing from ESW screen xvii
description xvi
source manager 20
SPP 149
SRP 149
SS (Show Block) line command 495
SSP 149
statement/offset stepping 167
status box 169, 232, 416, 477
STEP command 369, 421, 425, 515
STOP command 429
stopping point 444
storage
protected 143
protection 151
related keywords 145
Stored Procedure 540
SUBNET 294
substitution character, cursor 222
SUBTRACT command 431
SUBTRACT statement 527
symbolic information 154
SYS (task control area/system) 148
system variable 519, 523, 525
system-related keywords 149

## T

tags 285
TCA (CICS Task Control Area) 148
TCA (Test Coverage Analysis) 541
TCA DEFINE command 432
TCA LIST command 433
TCA RECORD command 434
TCA REPORT command 435
TCA RUN command 436
TCE (CICS terminal control table terminal entry) 148
TCP 149
TCT name (CICS terminal control table entry) 148
TDP 149
temporary logic 511
terminal, specify type 194
Test - Add Request pop-up
example 76
field descriptions 76
Test - ASG-SmartTest Testpoint Generation pop-up
example 79
field descriptions 79
Test - Break Request pop-up
example 63
field descriptions 63
Test - Go Request pop-up
example 74
field descriptions 74
Test - Move Request pop-up
example 75
field descriptions 75
Test - Newcopy Request pop-up
example 81
field descriptions 81
Test - Run Request (Backtrack Recording Mode Active) pop-up
example 49
field descriptions 50
Test - Run Request (Backtrack Review Mode Active) pop-up
example 51
field descriptions 51

Test - Run Request (CICS - Backtrack Recording Mode Active) pop-up
example 53
field descriptions 53
Test - Run Request (CICS Environment-Test active) pop-up
example 52
field descriptions 52
Test - Run Request pop-up
example 48
field descriptions 48
Test - Step Request (Backtrack Recording Mode) pop-up
example 56
field descriptions 56
Test - Step Request (Backtrack Review Mode) pop-up
example 58
field descriptions 58
Test - Step Request (CICS - BackTrack Recording Mode Active)
example 61
field descriptions 61
Test - Step Request (CICS Environment-Test active) pop-up
example 59
field descriptions 59
Test - Step Request pop-up
example 54
field descriptions 55
Test - Stop Request pop-up
example 73
field descriptions 74
Test - Subtract Request pop-up
example 77
field descriptions 77
Test - Where Request pop-up
example 78
field descriptions 78
TEST command 437
Test Facilities List screen
example 120
field descriptions 120
Test pull-down 46
Test Session Tailoring screen
example 165
field descriptions 166
testing, resuming 390, 421
testpoint 47, 79
TESTPOINT command 438
TIA (current CICS terminal input/output area) 148
TIO-nn (nth CICS terminal input/output area) 148
TOGGLE command 439
TRACE command 440
TRP 149
TSA (CICS Temporary Storage Allocation Table) 148
TSI (nth Temporary Storage Area) 149
TSM (Temporary Storage Map) 149
TSP 149
TUA (terminal user area) 149
TWA (task work area) 149

## U

unconditional interrupt 520
UPDATE command 446
user-supplied impact datasets 79
USING command 448
UTILITY command 450

## V

VIASUB edit macro 11
View - Display Request pop-up
example 35
field descriptions 35
View - Drop Request pop-up
example 44
field descriptions 44
View - Exclude Request pop-up
example 42
field descriptions 42
View - Execution Counts pop-up
example 28
field descriptions 28
View - Execution Tracking pop-up
example 29
field descriptions 29
View - Keep Request pop-up
example 34
field descriptions 34
View - Memory pop-up
example 30
field descriptions 31
View - Paragraph Cross Reference pop-up
example 38
field descriptions 39
View - Paragraph Cross-Reference Request pop-up
example 36
field descriptions 37
View - Qualify Request pop-up
example 31
field descriptions 32

View - Reset Request pop-up
example 40
field descriptions 41
View - Using Request pop-up
example 43
field descriptions 43
View - Zoom Data Request pop-up
example 32
field descriptions 33
VIEW command 451
View pull-down 26

## W

WHEN command 330, 452, 514–515
When conditions 514–515
When Conditions facility 417
When Conditions List screen
example 170
field descriptions 171
WHEN statement 528
WHERE command 456
WIZARD command 458

## X

X (Exclude) line command 496
XX (Exclude Block) line command 497

## Z

ZA (Zoom Assembler) line command 498
ZCP 149
ZD (Zoom Data) line command 500
ZG (Zoom Group) line command 465, 502
ZGH (Zoom Group Hexadecimal) line
command 465, 504
ZH (Zoom Hexadecimal) line command 506
ZI (Zoom In) line command 508
ZO (Zoom Out) line command 509
ZOOMDATA command 459
ZOOMIN command 462
ZOOMOUT command 462