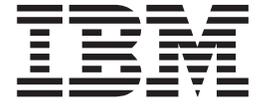


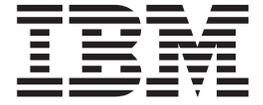
Interactive
System Productivity Facility (ISPF)



Edit and Edit Macros

OS/390 Version 2 Release 8.0

Interactive
System Productivity Facility (ISPF)



Edit and Edit Macros

OS/390 Version 2 Release 8.0

Note

Before using this document, read the general information under "Notices" on page xv.

Fourth Edition (September 1999)

This edition applies to ISPF for Version 2 Release 8 of the licensed program OS/390 (program number 5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications by phone or fax. IBM Software Manufacturing Solutions takes publication orders between 8:30 a.m. and 7:00 p.m. eastern standard time (EST). The phone number is (800) 879-2755. The fax number is (800) 284-4721.

You can also order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for comments appears at the back of this publication. If the form has been removed, and you have ISPF-specific comments, address your comments to:

International Business Machines Corporation
Attn: Information Development
Department T99 / Building 062
P.O. Box 12195
Research Triangle Park, NC 27709-2195

FAX (United States & Canada): 1+919+254-0206
FAX (Other Countries): Your International Access Code +1+919+254-0206

If you have OS/390 - specific comments, address them to:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries): Your International Access Code +1+914+432-9405

IBMLink (United States customers only): KGNVMC(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMAIL
Internet: mhvrdfs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

Title and order number of this book
Page number or topic related to your comment

The ISPF development team maintains a site on the World-Wide Web, as does the OS/390 development team. The URLs for the sites are:

<http://www.software.ibm.com/ad/ispf>
<http://www.ibm.com/s390/os390>

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

© Copyright International Business Machines Corporation 1984, 1999. All rights reserved.

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

Contents

Figures	xiii
--------------------------	-------------

Notices	xv
Programming Interface Information.	xvi
Trademarks.	xvi

Preface	xix
About This Book	xix
Who Should Use This Book	xix

Summary of Changes	xxi
ISPF Product Changes	xxi
ISPF DM Component Changes	xxi
ISPF PDF Component Changes	xxii
ISPF SCLM Component Changes	xxiii
ISPF Client/Server Component Changes	xxiii
ISPF User Interface Considerations	xxiv
ISPF Migration Considerations	xxiv
ISPF Profiles	xxiv
Year 2000 Support for ISPF	xxiv

Elements and Features in OS/390	xxvii
--	--------------

The ISPF User Interfacexxxii
Some Terms You Should Know	xxxii
How to Navigate in ISPF without Using Action Bars	xxxii
How to Navigate in ISPF Using the Action Bar Interface	xxxii
Action Bars.	xxxii
Action Bar Choices	xxxv
Point-and-Shoot Text Fields	xxxvi
Function Keys	xxxvii
Selection Fields	xxxvii
Command Nesting	xxxviii

Part 1. The ISPF Editor. 1

Chapter 1. Introducing the ISPF Editor	3
What is ISPF?	3
What the ISPF Editor Does	4
How to Use the ISPF Editor	4
Beginning an Edit Session	4
Using the ISPF Editor Basic Functions	14
Ending an Edit Session	15
Edit Commands	16
Line Commands	16
Primary Commands	17
Edit Commands and PF Key Processing	17
Edit Macros.	18
Editing Data in Controlled Libraries	19
Packing Data	19

Chapter 2. Controlling the Edit

Environment	21
What is an Edit Profile?.	21
Using Edit Profile Types	21
Displaying or Defining an Edit Profile	21
Modifying an Edit Profile	23
Locking an Edit Profile	23
Edit Modes	23
Edit Profile Modes	24
Edit Mode Defaults	25
Flagged Lines	27
Changed Lines.	27
Error Lines	27
Special Lines	27
Edit Boundaries	28
Initial Macros	29
Application-Wide Macros	30
Statistics for PDS Members	30
Effect of Stats Mode When Beginning an Edit Session	31
Effect of Stats Mode When Saving Data	31
Version and Modification Level Numbers.	31
Sequence Numbers	32
Sequence Number Format and Modification Level	32
Sequence Number Display.	33
Initialization of Number Mode	33
Enhanced and Language-sensitive Edit Coloring	34
Language Support	34
The HILITE Command/Dialog	38
Highlighting Status and the Edit Profile	46
Edit Recovery	47

Chapter 3. Managing Data 49

Creating and Replacing Data	49
Copying and Moving Data	50
Shifting Data	51
Column Shift	51
Data Shift	52
Finding, Seeking, Changing, and Excluding Data	53
Specifying the Search String	54
Effect of CHANGE Command on Column-Dependent Data	57
Using the CHANGE Command With EBCDIC and DBCS Data	57
Controlling the Search	57
Qualifying the Search String	59
Column Limitations	59
Split Screen Limitations.	59
Excluded Line Limitations.	60
Using the X (Exclude) Line Command with FIND and CHANGE.	60
Repeating the FIND, CHANGE, and EXCLUDE Commands	60
Examples	61

Excluding Lines	65
Redisplaying Excluded Lines	65
Redisplaying a Range of Lines	66
Labels and Line Ranges.	66
Editor-Assigned Labels	66
Specifying a Range	67
Using Labels and Line Ranges	68
Word Processing	68
Formatting Paragraphs	68
Splitting Lines	70
Entering Text (Power Typing).	71
Using Tabs	71
Types of Tabs	71
Defining and Controlling Tabs	72
Defining Software Tab Positions	73
Defining Hardware Tab Positions	73
Using Attribute Bytes	74
Undoing Edit Interactions	74
UNDO Processing	75
Understanding Differences in SETUNDO Processing	76

Chapter 4. Using Edit Models 77

What Is an Edit Model?.	77
How Models Are Organized	77
How to Use Edit Models	79
Adding, Finding, Changing, and Deleting Models	81
Adding Models	81
Finding Models	85
Changing Models.	86
Deleting Models	86

Part 2. Edit Macros 87

Chapter 5. Using Edit Macros. 89

What Are Edit Macros?.	89
Performing Repeated Tasks	89
Simplifying Complex Tasks	91
Passing Parameters, and Retrieving and Returning Information	92

Chapter 6. Creating Edit Macros 95

CLIST and REXX Edit Macros	95
Edit Macro Commands and Assignment Statements	96
Command Procedure Statements.	96
ISPF and PDF Dialog Service Requests.	97
TSO Commands	97
Program Macros	97
Differences between Program Macros, CLISTs, and REXX EXECs.	98
Passing Parameters in a Program Macro	98
Program Macro Examples	99
Writing Program Macros	99
Running Program Macros	102
Using Commands in Edit Macros	103
Naming Edit Macros.	103
Variables.	103
Edit Assignment Statements	104
Performing Line Command Functions	108

Parameters	109
Passing Parameters to a Macro	109
Using Edit macros in Batch	111
Edit Macro Messages	111
Macro Levels	112
Labels in Edit Macros	112
Referring to Data Lines	114
Referring to Column Positions	115
Defining Macros	115
Using the PROCESS Command and Operand	116
Recovery Macros	118
Return Codes from User-Written Edit Macros	118
Return Codes from PDF Edit Macro Commands	119
Selecting Control for Errors	120

Chapter 7. Testing Edit Macros 121

Handling Errors	121
Edit Command Errors	121
Dialog Service Errors	121
Using CLIST WRITE Statements and REXX SAY Statements	122
Using CLIST CONTROL and REXX TRACE Statements	123
Experimenting with Macro Commands	124

Chapter 8. Sample Edit Macros 127

TEXT Macro	127
PFCAN Macro.	129
BOX Macro	130
IMBED Macro	132
ALLMBRS Macro	135
FINCHGS Macro	138
MASKDATA Macro	141

Part 3. Command Reference 145

Chapter 9. Edit Line Commands 153

Rules for Entering Line Commands.	153
Edit Line Command Notation Conventions	154
Line Command Summary	154
(—Column Shift Left.	156
Syntax	156
Description	156
Example	157
)—Column Shift Right	158
Syntax	158
Description	158
Example	158
<—Data Shift Left	159
Syntax	160
Description	160
Example	160
>—Data Shift Right	162
Syntax	162
Description	162
Example	162
A—Specify an “After” Destination	163
Syntax	164
Description	164
Example	164

B—Specify a “Before” Destination	166
Syntax	166
Description	166
Example	167
BOUNDS—Define Boundary Columns.	168
Syntax	168
Description	168
Example	169
C—Copy Lines	170
Syntax	170
Description	170
Example	171
COLS—Identify Columns	172
Syntax	172
Description	172
Example	173
D—Delete Lines	174
Syntax	174
Description	174
Example	174
F—Show the First Line	175
Syntax	176
Description	176
Example	176
I—Insert Lines.	177
Syntax	177
Description	177
Example	178
L—Show the Last Line(s)	179
Syntax	179
Description	179
Example	179
LC—Convert Characters to Lowercase.	180
Syntax	180
Description	180
Example	181
M—Move Lines	182
Syntax	182
Description	182
Example	183
MASK—Define Masks	184
Syntax	184
Description	184
Example	185
MD—Make Dataline.	186
Syntax	186
Description	186
Example	187
O—Overlay Lines.	188
Syntax	188
Description	188
Example	189
R—Repeat Lines	190
Syntax	191
Description	191
Example	191
S—Show Lines.	193
Syntax	193
Description	193
Example	194
TABS—Control Tabs	195

Syntax	195
Description	195
Examples	195
TE—Text Entry	197
Syntax	197
Description	197
Example	198
TF—Text Flow	200
Syntax	200
Description	200
Example	201
TS—Text Split	202
Syntax	202
Description	202
Examples	202
UC—Convert Characters to Uppercase.	203
Syntax	204
Description	204
Example	204
X—Exclude Lines.	205
Syntax	206
Description	206
Example	206

Chapter 10. Edit Primary Commands 209

Edit Primary Command Notation Conventions	209
Edit Primary Command Summary	209
AUTOLIST—Create a Source Listing	
Automatically	213
Syntax	214
Description	214
Example	214
AUTONUM—Number Lines Automatically	215
Syntax	215
Description	215
Example	216
AUTOSAVE—Save Data Automatically	217
Syntax	217
Description	217
Example	218
BOUNDS—Control the Edit Boundaries	218
Syntax	218
Description	219
Examples	219
BUILTIN—Process a Built-In Command	219
Syntax	219
Description	219
Example	219
BROWSE—Browse from within an Edit Session	220
Syntax	220
Description	220
Example	220
CANCEL—Cancel Edit Changes.	221
Syntax	221
Description	221
Example	221
CAPS—Control Automatic Character Conversion	221
Syntax	221
Description	221
Example	222
CHANGE—Change a Data String	222

Syntax	223	Specific Locate Syntax	256
Description	223	Generic Locate Syntax	256
Examples	224	Examples	257
COMPARE—Edit Compare	224	MODEL—Copy a Model into the Current Data Set	257
Command Syntax.	225	Model Name Syntax	257
Examples	226	Class Name Syntax	258
COPY—Copy Data	227	Example	258
Syntax	227	MOVE—Move Data	260
Description	228	Syntax	260
Example	229	Description	261
CREATE—Create Data	231	Example	262
Syntax	231	NONUMBER—Turn Off Number Mode	264
Description	231	Syntax	264
Example	232	Description	264
CUT—Cut and Save Lines	235	Example	264
Syntax	235	NOTES—Display Model Notes	264
Description	235	Syntax	265
Example	236	Description	265
DEFINE—Define a Name	236	Examples	265
Syntax	236	NULLS—Control Null Spaces.	265
Description	237	Syntax	265
Examples	237	Description	266
DELETE—Delete Lines	238	Examples	266
Syntax	238	NUMBER—Generate Sequence Numbers	266
Description	238	Syntax	266
Examples	238	Description	267
EDIT—Edit from within an Edit Session	239	Examples	268
Syntax	239	PACK—Compress Data.	268
Description	239	Syntax	268
Example	240	Examples	268
END—End the Edit Session	241	PASTE—Move or Copy Lines from Clipboard	268
Syntax	241	Syntax	268
Description	241	Description	269
Example	242	Example	269
EXCLUDE—Exclude Lines from the Display.	242	PRESERVE - Enable Saving of Trailing Blanks	269
Syntax	242	Syntax	269
Description	243	Description	269
Examples	243	Examples	270
FIND—Find a Data String	243	PROFILE—Control and Display Your Profile.	270
Syntax	244	Profile Control Syntax	270
Description	244	Profile Lock Syntax	270
Examples	245	Profile Reset Syntax	271
FLIP—Reverse Exclude Status of Lines	245	Description	271
Syntax	245	Example	272
Description	245	RCHANGE—Repeat a Change	273
Example	246	Syntax	273
HEX—Display Hexadecimal Characters	247	Description	273
Syntax	248	RECOVERY—Control Edit Recovery	273
Description	248	Syntax	273
Examples	248	Description	274
HILITE—Enhanced Edit Coloring	250	RENUM—Renumber Data Set Lines	274
Syntax	250	Syntax	274
Description	253	Description	275
IMACRO—Specify an Initial Macro.	253	Example	276
Syntax	253	REPLACE—Replace Data	277
Examples	253	Syntax	277
LEVEL—Specify the Modification Level Number	254	Description	278
Syntax	254	Example	278
Description	254	RESET—Reset the Data Display	280
Example	254	Syntax	281
LOCATE—Locate a Line	255	Description	281

Examples	281	AUTOSAVE—Set or Query Autosave Mode	309
RFIND—Repeat Find	282	Macro Command Syntax	310
Syntax	282	Assignment Statement Syntax	310
RMACRO—Specify a Recovery Macro	282	Description	310
Syntax	282	Return Codes	310
Description	283	Examples	311
Example	283	BLKSIZE—Query the Block Size	311
SAVE—Save the Current Data	283	Assignment Statement Syntax	311
Syntax	283	Return Codes	311
Description	283	Example	311
Example	284	BOUNDS—Set or Query the Edit Boundaries	311
SETUNDO—Set the UNDO Mode	284	Macro Command Syntax	312
Syntax	284	Assignment Statement Syntax	312
Description	284	Description	312
Example	285	Return Codes	312
SORT—Sort Data	286	Examples	312
Syntax	286	BROWSE—Browse from within an Edit Session	313
Description	286	Macro Command Syntax	313
Examples	287	Description	313
STATS—Generate Library Statistics	288	Return Codes	314
Syntax	288	Examples	314
Examples	288	BUILTIN—Process a Built-In Command	314
SUBMIT—Submit Data for Batch Processing	288	Macro Command Syntax	314
Syntax	288	Description	314
Description	288	Return Codes	314
Examples	288	Examples	314
TABS—Define Tabs	289	CANCEL—Cancel Edit Changes	315
Syntax	289	Macro Command Syntax	315
Example	290	Description	315
UNDO—Reverse Last Edit Interaction	290	Return Codes	315
Syntax	290	Example	315
Description	291	CAPS—Set or Query Caps Mode	315
Example	292	Macro Command Syntax	315
UNNUMBER—Remove Sequence Numbers	293	Assignment Statement Syntax	315
Syntax	294	Description	316
Description	294	Return Codes	316
Example	294	Examples	316
VERSION—Control the Version Number	295	CHANGE—Change a Search String	316
Syntax	295	Macro Command Syntax	317
Description	295	Description	318
Example	295	Return Codes	318
VIEW—View from within an Edit Session	296	Example	319
Syntax	297	CHANGE_COUNTS—Query Change Counts	319
Description	297	Assignment Statement Syntax	319
Example	297	Return Codes	319
		Examples	319
Chapter 11. Edit Macro Commands and		COMPARE—Edit Compare	320
Assignment Statements	299	Macro Command Syntax	320
Edit Macro Command Notation Conventions	299	Return Codes	321
Edit Macro Command Summary	300	Compare Examples	321
AUTOLIST—Set or Query Autolist Mode	307	COPY—Copy Data	322
Macro Command Syntax	307	Macro Command Syntax	322
Assignment Statement Syntax	308	Return Codes	323
Return Codes	308	Examples	323
Examples	308	CREATE—Create a Data Set Member	323
AUTONUM—Set or Query Autonum Mode	308	Macro Command Syntax	323
Macro Command Syntax	308	Description	324
Assignment Statement Syntax	308	Return Codes	324
Description	309	Example	324
Return Codes	309	CTL_LIBRARY—Query Controlled Library Status	324
Examples	309	Assignment Statement Syntax	324

Return Codes	325	END—End the Edit Session	338
Example	326	Macro Command Syntax	338
CURSOR—Set or Query the Cursor Position.	326	Description	339
Assignment Statement Syntax	326	Return Codes	339
Description	326	Example	339
Return Codes	327	EXCLUDE—Exclude Lines from the Display.	339
Examples	327	Macro Command Syntax	339
CUT—Cut and Save Lines	328	Description	340
Syntax	328	Return Codes	341
Description	328	Examples	341
Return Codes	328	EXCLUDE_COUNTS—Query Exclude Counts	341
Examples	329	Assignment Statement Syntax	341
DATA_CHANGED—Query the Data Changed Status.	329	Return Codes	342
Assignment Statement Syntax	329	Example	342
Description	329	FIND—Find a Search String	342
Return Codes	329	Macro Command Syntax	342
Example	329	Description	343
DATA_WIDTH—Query Data Width	330	Return Codes	344
Assignment Statement Syntax	330	Examples	344
Description	330	FIND_COUNTS—Query Find Counts	344
Return Codes	330	Assignment Statement Syntax	344
Example	330	Return Codes	344
DATAID—Query Data ID	331	Example	345
Assignment Statement Syntax	331	FLIP—Reverse Exclude Status of Lines	345
Description	331	Assignment Statement Syntax	345
Return Codes	331	Return Codes	345
Example	331	Examples	345
DATASET—Query the Current Data Set Name	331	FLOW_COUNTS—Query Flow Counts	345
Assignment Statement Syntax	331	Assignment Statement Syntax	345
Return Codes	332	Return Codes	346
Example	332	Example	346
DEFINE—Define a Name	332	HEX—Set or Query Hexadecimal Mode	346
Macro Command Syntax	332	Macro Command Syntax	346
Description	333	Assignment Statement Syntax	346
Return Codes	333	Description	347
Examples	333	Return Codes	347
DELETE—Delete Lines	334	Examples	347
Macro Command Syntax	334	HILITE—Enhanced Edit Coloring	347
Description	334	Macro Command Syntax	348
Return Codes	334	Description	350
Examples	334	Return Codes	350
DISPLAY_COLS—Query Display Columns	335	IMACRO—Set or Query an Initial Macro	351
Assignment Statement Syntax	335	Macro Command Syntax	351
Description	335	Assignment Statement Syntax	351
Return Codes	335	Return Codes	351
Example	335	Examples	351
DISPLAY_LINES—Query Display Lines	336	INSERT—Prepare Display for Data Insertion.	351
Assignment Statement Syntax	336	Macro Command Syntax	352
Return Codes	336	Description	352
Example	336	Return Codes	352
DOWN—Scroll Down	336	Example	352
Macro Command Syntax	336	LABEL—Set or Query a Line Label	352
Description	337	Assignment Statement Syntax	352
Return Codes	337	Description	353
Examples	337	Return Codes	353
EDIT—Edit from within an Edit Session	338	Example	353
Macro Command Syntax	338	LEFT—Scroll Left.	353
Description	338	Macro Command Syntax	354
Return Codes	338	Description	354
Example	338	Return Codes	354
		Example	354

LEVEL—Set or Query the Modification Level	
Number	354
Macro Command Syntax	355
Assignment Statement Syntax	355
Return Codes	355
Examples	355
LINE—Set or Query a Line from the Data Set	355
Assignment Statement Syntax	355
Description	356
Return Codes	356
Examples	356
LINE_AFTER—Add a Line to the Current Data	
Set.	356
Assignment Statement Syntax	356
Description	357
Return Codes	357
Examples	358
LINE_BEFORE—Add a Line to the Current Data	
Set.	358
Assignment Statement Syntax	358
Description	359
Return Codes	359
Examples	359
LINENUM—Query the Line Number of a Labeled	
Line	359
Assignment Statement Syntax	359
Return Codes	359
Description	360
Examples	360
LOCATE—Locate a Line	360
Specific Locate Syntax	360
Generic Locate Syntax	360
Return Codes	361
Examples	361
LRECL—Query the Logical Record Length	362
Assignment Statement Syntax	362
Description	362
Return Codes	362
Example	362
MACRO—Identify an Edit Macro	362
Macro Command Syntax	362
Description	363
Return Codes	363
Examples	363
MACRO_LEVEL—Query the Macro Nesting Level	364
Assignment Statement Syntax	364
Description	364
Return Codes	364
Example	364
MASKLINE—Set or Query the Mask Line	364
Assignment Statement Syntax	364
Description	365
Return Codes	365
Examples	365
MEMBER—Query the Current Member Name	365
Assignment Statement Syntax	365
Return Codes	365
Example	366
MEND—End a Macro in the Batch Environment	366
Macro Command Syntax	366
Description	366
Return Codes	366
Example	366
MODEL—Copy a Model into the Current Data Set	366
Macro Command Model Name Syntax	367
Macro Command Class Name Syntax	367
Return Codes	368
Example	368
MOVE— Move a Data Set or a Data Set Member	368
Macro Command Syntax	368
Description	368
Return Codes	369
Examples	369
NONUMBER—Turn Off Number Mode	369
Syntax	369
Description	369
Return Codes	369
Example	369
NOTES—Set or Query Note Mode	370
Macro Command Syntax	370
Assignment Statement Syntax	370
Return Codes	370
Examples	370
NULLS—Set or Query Nulls Mode	370
Macro Command Syntax	371
Assignment Statement Syntax	371
Description	371
Return Codes	371
Examples	372
NUMBER—Set or Query Number Mode	372
Macro Command Syntax	372
Assignment Statement Syntax	373
Description	374
Return Codes	374
Example	374
PACK—Set or Query Pack Mode	374
Macro Command Syntax	374
Assignment Statement Syntax	375
Return Codes	375
Example	375
PASTE—Move or Copy Lines from Clipboard	375
Syntax	375
Description	376
Return Codes	376
Examples	376
PRESERVE	376
Macro Command Syntax	376
Assignment Statement Syntax	376
Description	377
Return Codes	377
Examples	377
PROCESS—Process Line Commands	377
Macro Command Syntax	377
Description	378
Return Codes	378
Examples	378
PROFILE—Set or Query the Current Profile	379
Macro Command Profile Control Syntax	379
Macro Command Profile Lock Syntax	379
Macro Command Profile Reset Syntax	380
Assignment Statement Syntax	380
Description	380

Return Codes	380	Macro Command Syntax	392
Example	380	Assignment Statement Syntax	393
RANGE_CMD—Query a Command That You Entered	381	Return Codes	393
Assignment Statement Syntax	381	Example	393
Description	381	SEEK—Seek a Data String, Positioning the Cursor	393
Return Codes	381	Macro Command Syntax	393
Example	381	Description	394
RCHANGE—Repeat a Change	381	Return Codes	395
Macro Command Syntax	381	Examples	395
Description	382	SEEK_COUNTS—Query Seek Counts	395
Return Codes	382	Assignment Statement Syntax	395
Example	382	Return Codes	396
RECFM—Query the Record Format.	382	Example	396
Assignment Statement Syntax	382	SESSION	396
Return Codes	383	Assignment Statement Syntax	396
Example	383	Return Codes	396
RECOVERY—Set or Query Recovery Mode	383	SETUNDO—Set UNDO Mode	396
Macro Command Syntax	383	Macro Command Syntax	396
Assignment Statement Syntax	384	Assignment Statement Syntax	397
Return Codes	384	Description	397
Examples	384	Return Codes	397
RENUM—Renummer Data Set Lines	384	Examples	398
Macro Command Syntax	384	SHIFT (—Shift Columns Left	398
Return Codes	385	Macro Command Syntax	398
Examples	385	Description	398
REPLACE—Replace a Data Set Member	386	Return Codes	398
Macro Command Syntax	386	Examples	398
Return Codes	386	SHIFT)—Shift Columns Right	399
Example	386	Macro Command Syntax	399
RESET—Reset the Data Display	386	Description	399
Macro Command Syntax	387	Return Codes	399
Description	387	Examples	399
Return Codes	387	SHIFT <—Shift Data Left	399
Examples	388	Macro Command Syntax	399
RFIND—Repeat Find	388	Description	400
Macro Command Syntax	388	Return Codes	400
Return Codes	388	Examples	400
Example	388	SHIFT >—Shift Data Right.	400
RIGHT—Scroll Right	389	Macro Command Syntax	400
Macro Command Syntax	389	Description	400
Description	389	Return Codes	400
Return Codes	389	Examples	400
Example	390	SORT—Sort Data	401
RMACRO—Set or Query the Recovery Macro	390	Macro Command Syntax	401
Macro Command Syntax	390	Description	401
Assignment Statement Syntax	390	Return Codes	402
Return Codes	390	Examples	402
Example	390	STATS—Set or Query Stats Mode	403
SAVE—Save the Current Data	390	Macro Command Syntax	403
Macro Command Syntax	391	Assignment Statement Syntax	403
Description	391	Return Codes	403
Return Codes	391	Examples	403
Example	391	SUBMIT—Submit Data for Batch Processing	404
SAVE_LENGTH—Set or Query Length for Variable Length Data	391	Macro Command Syntax	404
Assignment Statement Syntax	391	Description	404
Description	391	Return Codes	404
Return Codes	392	Examples	404
Examples	392	TABS—Set or Query Tabs Mode	404
SCAN—Set Command Scan Mode	392	Macro Command Syntax	405
		Assignment Statement Syntax	406
		Return Codes	406

Examples	406
TABSLINE—Set or Query Tabs Line	406
Assignment Statement Syntax	406
Return Codes	407
Examples	407
TENTER—Set Up Panel for Text Entry.	407
Macro Command Syntax	407
Description	408
Return Codes	409
Example	409
TFLOW—Text Flow a Paragraph	409
Macro Command Syntax	409
Return Codes	409
Example	409
TSPLIT—Text Split a Line	409
Macro Command Syntax	409
Description	410
Return Codes	410
Example	410
UNNUMBER—Remove Sequence Numbers	410
Macro Command Syntax	410
Description	410
Return Codes	410
Example	411
UP—Scroll Up.	411
Macro Command Syntax	411
Description	411
Return Codes	412
Examples	412
USER_STATE—Save or Restore User State	412
Assignment Statement Syntax	412
Description	412
Return Codes	413
Examples	413
VERSION—Set or Query Version Number	413
Macro Command Syntax	413
Assignment Statement Syntax	413
Return Codes	413

Examples	414
VIEW—View from within an Edit Session	414
Macro Command Syntax	414
Description	414
Return Codes	414
Examples	414
VOLUME—Query Volume Information	414
Assignment Statement Syntax	415
Return Codes	415
Examples	415
XSTATUS—Set or Query Exclude Status of a Line	415
Assignment Statement Syntax	415
Description	415
Return Codes	416
Examples	416

Part 4. Appendixes 417

Appendix A. Abbreviations for Commands and Other Values. 419

Edit Line Commands	419
Edit Primary Commands	419
Parameters	419
Keywords/Operands	420
Scroll Amounts	420

Appendix B. Edit-Related Sample Macros 421

Sample Macros	421
-------------------------	-----

Index 423

Readers' Comments — We'd Like to Hear from You 433

Figures

1. Panel with an Action Bar Pull-Down Menu	xxxiii	51. TEXT Macro - After Running	129
2. Pop-Up Selected from an Action Bar Pull-Down	xxxiv	52. PFCAN Macro	129
3. Panel with an Action Bar and Point-and-Shoot Fields	xxxiv	53. BOX Macro	130
4. An Unavailable Choice on a Pull-Down	xxxvi	54. BOX Macro - Before Running	132
5. Edit Entry Panel (ISREDM01)	5	55. BOX Macro - After Running	132
6. Creating a New Data Set (ISREDDE2)	11	56. IMBED Macro	133
7. Example Primary Edit Panel (ISREDDE2)	11	57. LIST with Imbed Statements	135
8. Edit Profile Display (ISREDDE2)	22	58. IMBED Macro - After Running	135
9. HILITE Initial Screen (ISREP1)	41	59. ALLMBRS Macro	136
10. Set Overtyping Color panel (ISREP2)	43	60. FINDCHGS Macro	138
11. Set Find String Color panel (ISREP3)	43	61. FINDCHGS Macro - Before Running	140
12. Set Cursor Phrase Color panel (ISREP4)	44	62. FINDCHGS Macro - After Running	141
13. HILITE Specific Language Screens (ISREPC)	45	63. MASKDATA Macro	142
14. HILITE Language Keyword List (ISREPK)	46	64. MASKDATA Macro - Before Running	143
15. Edit Profile Lines with HILITE	46	65. MASKDATA Macro - After Running	144
16. Edit Recovery Panel (ISREDM02)	47	66. Before the ((Column Shift Left) Line Command	157
17. Confirm Replace Panel (ISRERPL2)	50	67. After the ((Column Shift Left) Line Command	157
18. Before FIND Command (ISREDDE2)	61	68. Before the) (Column Shift Right) Line Command	159
19. After FIND Command	62	69. After the) (Column Shift Right) Line Command	159
20. Before CHANGE Command	62	70. Before the < (Data Shift Left) Line Command	161
21. After CHANGE Command	63	71. After the < (Data Shift Left) Line Command	161
22. Before EXCLUDE Command	64	72. Before the > (Data Shift Right) Line Command	163
23. After EXCLUDE Command	64	73. After the > (Data Shift Right) Line Command	163
24. Model Classes Panel (ISREMCLS)	78	74. Before the A (After) Line Command	165
25. CLIST Models Panel (ISREMCMD)	79	75. After the A (After) Line Command	165
26. DISPLAY Service Model	80	76. Before the B (Before) Line Command	167
27. Sample Block Letter Model	82	77. After the B (Before) Line Command	168
28. Panel Models Panel (ISREMPNL)	82	78. Before the BOUNDS Line Command	169
29. Changed Panel Models Panel (ISREMPNL)	83	79. After the BOUNDS Line Command	170
30. Changed)PROC Section of Panel Models Panel (ISREMPNL)	83	80. Before the C (Copy) Line Command	171
31. Source Code for Block Letter Model Selection Panel	85	81. After the C (Copy) Line Command	172
32. DASH Macro	90	82. Before the COLS Line Command	173
33. DASH Macro - Before Running	90	83. After the COLS Line Command	173
34. DASH Macro - After Running	91	84. Before the D (Delete) Line Command	175
35. TESTDATA Macro	91	85. After the D (Delete) Line Command	175
36. TESTDATA Macro - Before Running	92	86. Before the F (Show First Line) Line Command	176
37. TESTDATA Macro - After Running	92	87. After the F (Show First Line) Line Command	177
38. COUNTSTR Macro	93	88. Before the I (Insert) Line Command	178
39. COUNTSTR Macro - Before Running	93	89. After the I (Insert) Line Command	178
40. COUNTSTR Macro - After Running	94	90. Before the L (Show Last Line) Line Command	179
41. SEPLINE REXX Macro	100	91. After the L (Show Last Line) Line Command	180
42. SEPLINE PL/I Macro	101	92. Before the LC (Lowercase) Line Command	181
43. SEPLINE COBOL Macro	102	93. After the LC (Lowercase) Line Command	182
44. TESTDATA Macro with CLIST WRITE Statements	122	94. Before the M (Move) Line Command	183
45. Results of TESTDATA Macro with CLIST WRITE Statements	123	95. After the M (MOVE) Line Command	184
46. TRYIT Macro	124	96. Before the MASK Line Command	185
47. TRYIT Macro - Before Running	125	97. After the MASK Line Command	186
48. TRYIT Macro - After Running	125	98. Before the MD (Make Dataline) Line Command	187
49. TEXT Macro	127		
50. TEXT Macro - Before Running	128		

99. After the MD (Make Dataline) Line Command	188	131. Nested Member Editing Example	241
100. Before the O (Overlay) Line Command	190	132. Example of Data Set	246
101. After the O (Overlay) Line Command	190	133. Example of Data Set with Excluded Lines	247
102. Before the R (repeat) Line Command	192	134. Example of Data Set using FLIP on Excluded Lines	247
103. After the R (Repeat) Line Command	192	135. Member With Hexadecimal Mode Off	249
104. Editing after the R (Repeat) Line Command	193	136. Hexadecimal Display, Vertical Representation	249
105. Before the S (Show) Line Command	194	137. Hexadecimal Display, Data Representation	250
106. After the S (Show) Line Command	195	138. Member With Modification Level of 03	255
107. TAB Line Command Example	196	139. Member With Modification Level Reset to 00	255
108. Before the TE (Text Entry) Line Command	198	140. Profile Display	259
109. After the TE (Text Entry) Line Command	199	141. REXX Models Panel (ISREMRXC).	259
110. Sample Text During Text Entry Mode.	199	142. REXX Model of VGET Service	260
111. Sample Text After Text Entry Mode.	200	143. Member Before Data is Moved.	262
112. Before the TF (Text Flow) Line Command	201	144. Edit Move Panel (ISREMOV1)	263
113. After the TF (Text Flow) Line Command	201	145. Data Set to be Moved.	263
114. Before TS (Text Split) Line Command	203	146. Member After Data Has Been Moved	264
115. After TS (Text Split) Line Command	203	147. Edit Profile Display	272
116. Before the UC (Uppercase) Line Command	205	148. Member Before Lines Are Renumbered	276
117. After the UC (Uppercase) Line Command	205	149. Member After Lines Are Renumbered	277
118. Before the X (Exclude) Line Command	207	150. Member Before Other Member Is Replaced	279
119. After the X (Exclude) Line Command	207	151. Edit - Replace Panel (ISRERPL1)	279
120. Edit Compare Settings Panel	227	152. Member After the Other Member Has Been Replaced	280
121. Member Before Data is Copied	229	153. Other Member Replaced	280
122. Edit Copy Panel (ISRECPY1)	230	154. SETUNDO STORAGE and RECOVERY OFF	285
123. Data Set to be Copied	230	155. Member Before Lines Are Deleted	292
124. Member After Data Has Been Copied	231	156. Member After Lines Are Deleted	293
125. Member Before New Member Is Created	233	157. Member After Lines Have Been Restored	293
126. Edit Create Panel (ISRECRA1)	233	158. Member Before Lines Are Unnumbered	294
127. Member After New Member Has Been Created	234	159. Member After Lines Are Unnumbered	295
128. New Member Created	234	160. Member Before Version Number is Changed	296
129. EDIT Primary Command Example	240	161. Member After Version Number is Changed	296
130. Edit Command Entry Panel (ISREDM03)	240		

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the IBM Corporation, Mail Station P300, 522 South Road, Poughkeepsie, NY 12601-5400, USA. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

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

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

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

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear. For users of BookManager Library Reader for Windows, Version 2.0.2 at 5799-pxy Service Level : S9903ENU provides the best results.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of ISPF.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of ISPF. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

-----Programming Interface information-----

Trademarks

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

ACF/VTAM	MVS/DFP
AD/Cycle	MVS/ESA
AIX	MVS/XA
APL2	MVS/SP
Application Development	Operating System/2
AS/400	OS/2
BookMaster	OS/400
CICS	Personal System/2
CICS OS/2	PS/2
CUA	Presentation Manager
Common User Access	RACF
CSP/AD	Repository Manager
CSP/AE	Repository Manager/MVS
CSP/2AD	Resource Access Control Facility
CSP/370AD	SAA
CSP/370RS	Series/1
C370	System Application Architecture
DATABASE 2	System/370
DB2	System/390
DFSMS/MVS	VisualAge
GDDM	VM/XA
IBM	VTAM
IMS/ESA	

The following terms are trademarks of other companies:

C++
HP
HP-UX
Java
Microsoft
Novell
WordPerfect

American Telephone and Telegraph Company
Hewlett-Packard
Hewlett-Packard
Sun Microsystems, Inc.
Microsoft Corporation
Novell, Inc.
WordPerfect Corporation

Windows is a trademark of Microsoft Corporation.

Preface

This book describes the ISPF editor and provides conceptual, usage, and reference information for the ISPF edit line, primary, and macro commands.

About This Book

This book contains three parts:

- Part 1 introduces and describes how to use the ISPF editor.
- Part 2 describes how to use, write and test edit macros. It also provides and discusses sample CLIST, REXX, and program edit macros.
- Part 3 is a reference for the edit line, primary, and macro commands available for ISPF.

Who Should Use This Book

This book is for application and system programmers who develop programs, and who use the ISPF editor and edit macro instructions. Users who write edit macros should be familiar with coding CLISTs, REXX EXECs, or programs in the MVS environment.

Summary of Changes

OS/390 V2R8.0 ISPF contains the following changes and enhancements:

- ISPF Product and Library Changes
- ISPF Dialog Manager Component Changes
- ISPF PDF Component Changes
- ISPF SCLM Component Changes
- ISPF Client/Server Component Changes

ISPF Product Changes

Changes to the ZENVIR variable. Characters 1 through 8 contain the product name and sequence number in the format *ISPF x.y*, where x.y indicates:

- <= 4.2 means the version.release of ISPF
- = 4.3 means ISPF for OS/390 release 2
- = 4.4 means ISPF 4.2.1 and ISPF for OS/390 release 3
- = 4.5 means ISPF for OS/390 Version 2 Release 5.0
- = 4.8 means ISPF for OS/390 Version 2 Release 8.0

The ZENVIR variable is used by IBM personnel for internal purposes. The x.y numbers DO NOT directly correlate to an ISPF release number in all cases. For example, as shown above, a ZENVIR value of 4.3 DOES NOT mean ISPF Version 4 Release 3. NO stand-alone version of ISPF exists above ISPF Version 4 Release 2 Modification 1.

The ZOS390RL variable contains the OS/390 release on your system.

The ZISPFOS system variable contains the level of ISPF code that is running as part of the OS/390 release on your system. This might or might not match ZOS390RL. For this release, the variable contains **ISPF for OS/390 Version 2 Release 8.0**.

The ZPFKEY system variable contains PFKey values.

Support of CCSIDs 1140 through 1149 was added so the EURO currency symbol can be used in ISPF.

The Samples and Macros libraries each have an index member called @INDEX.

ISPF DM Component Changes

The DM component of ISPF includes the following new functions and enhancements:

- Support added for "VER(&variable, DSNAMEQ)".
- Support added for four-digit year on TBSTATS command and Option 7.4.
- Support added for four-digit year on TBSORT command.
- Message numbers were added to ISPF Line Mode messages.
- Support added for date format in Configuration table.
- Support added for a print utility exit for ISPF termination and Log/List commands.
- Support added for NEXT and PREV commands for Dialog Test Tables.

- The ZSCRNAME modifiable shared variable, containing the current screen name, was added.
- The ZPFKEY variable, which returns the name of the PFKey on exit from a panel, was added.
- The TPUT buffer size variable was added to the configuration table.
- The ISPSPROF system variables were moved to the configuration table.
- Support was added for commands chained after the START command.
- ISPD TLC enhancements:
 - New invocation options: NOPLEB / PLEB, NOMCOMMENT / MCOMMENT.
 - New tags: ATTENTION.
 - Added "German to Swiss German" character transform to create Swiss German panels from German DTL source files.
 - Expanded number of DTL source files for interactive processing from 4 to 12.
 - Allow DTL type comments (<: -- or <!--) in MVS profile data stream.
 - Allow SOURCE tag within ABC, AREA, PANEL, and REGION tags.
 - Added "Options" to action bar of interactive panel as an alternate method of setting conversion options.
 - Added support for macro tags.

ISPF PDF Component Changes

The ISPF PDF component contains the following new functions and enhancements:

- Conversion of the ISPF configuration table to a keyword format.
- Introduction of an interactive tool to create and update the keyword file and to build the needed load module from the keyword file.
- Edit CUT and PASTE commands. Data is saved in data spaces with multiple clipboards available.
- The Edit MOVE, COPY, CREATE, and REPLACE commands now accept a data set name or data set name and member name as a parameter.
- A VSAM editor or browser can be specified in the configuration table and is invoked automatically when a VSAM data set is specified in Options 1, 2, 3, or 11.
- Through the EPDF command, Edit, View, and Browse functions are available from any command line.
- A new edit macro VOLUME command to retrieve the volume of the data set being edited.
- The edit macro RECFM has been enhanced to return the full record format rather than just F or V.
- The new edit macro SESSION command returns EDIT, EDIF, or VIEW, as well as indicating whether the session was initiated from SCLM.
- Edit STATS mode is no longer forced to OFF if a sequential data set is edited. The STATS mode is simply ignored.
- The target data set for the Move/Copy utility or the Edit CREATE and REPLACE commands can be allocated automatically if it does not exist.
- Edit highlighting of FIND strings and the cursor phrase is enabled for data wider than 256 bytes.
- When the View REPLACE command is used to update the member being viewed, the confirmation panel shows whether the member has been updated by someone else during the View session.

- LMMDISP, LMMFIND, and LMMLIST return individual variables for load module statistics when STATS(YES) is specified.
- The COBOL options and PL/I options in Foreground and Batch have been consolidated into 1 COBOL option and 1 PL/I option.
- Data set information processes multivolume data sets with more than 20 volumes.
- Member list REFRESH command added.
- Member list SORT and LOCATE honor the collating sequence table in the PDF translate tables.
- The creation date is displayed on the Confirm Delete panel for VSAM data sets.
- AIX paths are identified in Option 3.4 with *PATH* in the volume column.
- REFLIST function improved.
- A warning message displays on the first data change made while in View.
- RIGHT and LEFT commands are supported in member list to enable the presentation of additional data such as full 4-character year dates.
- The LMMDISP service allows the selection of members that do not exist in the data set being processed.
- SuperC supports VSAM files.
- SuperC supports the FMSTOP performance option which stops on the first mismatch for file compare. FMSTOP is also supported for string searches.

ISPF SCLM Component Changes

The ISPF SCLM component contains the following new functions and enhancements:

- When a language that is not valid is specified on the SPROF panel, a scrollable table display of the valid languages is presented so the user can choose the desired language.
- ISPLNK is a valid CALLMETH for non-BUILD translators.
- A sample DTL parser and translator are available.
- Additional samples added to SAMPLIB.
- An SCLM EDIT service was added to enable editing of SCLM controlled parts from a dialog.
- The data set name and member is added to the SPROF panel (FLMEINFO) to assist users when selecting a language for a part.
- The ability to specify that SCLM temporary load libraries should be allocated as PDSEs has been added.
- Two new exit points added.
- SCLM warning messages issued by the ISPF editor when an SCLM controlled member is edited are only issued if the member's directory entry indicates the member is SCLM controlled.
- SCLM Versioning can be directed to ignore sequence number differences.

ISPF Client/Server Component Changes

The ISPF Client/Server Component enables a panel to be displayed unchanged (except for panels with graphic areas) at a workstation using the native display function of the operating system of the workstation. ISPF manuals call this "running in GUI mode."

The ISPF Client/Server component changes are:

- Support added to provide for an automatic download of the Client/Server component
- Enhanced usability and function of the Client/Server component download panel
- Support added to enable initiation of a workstation connection (without GUI display) while in split screen mode.
- New WSCON and WSDISCON commands to improve entry to the ISPF C/S interface.
- Enable one or more ISPF screens to **Switch** back and forth between GUI and 3270 modes by using the new Switch commands.

ISPF User Interface Considerations

Many changes have been made to the ISPF Version 4 user interface to conform to CUA guidelines. If you prefer to change the interface to look and act more like the Version 3 interface, you can do the following:

- Use the CUAATR command to change the screen colors
- Use the ISPF Settings panel to specify that the TAB or HOME keys position the cursor to the command line rather than to the first action bar item
- Set the command line to the top of the screen by deselecting *Command line at bottom* on the ISPF Settings panel
- Set the primary keys to f13–24 by selecting 2 for Primary range on the Tailor Function Key Definition Display panel
- Use the KEYLIST OFF command to turn keylists off
- Use the PSCOLOR command to change point-and-shoot fields to blue.
- Change the DFLTCOLR field in the PDF configuration table ISRCONFG to disable action bars and or edit highlighting

ISPF Migration Considerations

When migrating to OS/390 V2R8.0 or higher for the first time, you must convert your ISPF customization to the new format. Refer to the section entitled *The ISPF Configuration Table* in the *ISPF Planning and Customizing manual*.

When migrating from one version of ISPF to another, you must be sure to reassemble and re-link the SCLM project definition.

ISPF Profiles

Major changes have been made to the ISPF profiles for ISPF Version 4.2 and OS/390 V2R8.0 ISPF. If you are moving back and forth between a Version 3.3 or Version 3.5 system and a Version 4.2 or an OS/390 V2R8.0 system, you must run with separate profiles.

Year 2000 Support for ISPF

ISPF is fully capable of using dates for the year 2000 and beyond. All of your existing applications should continue to run (some may need minor changes, as explained below) when the year 2000 comes. The base support for the year 2000 was added to OS/390 Version 1 Release 2.0, but the same level of support is

available for ISPF Version 3.5, ISPF Version 4, and OS/390 Version 1 Release 1.0 as well. To get support for the earlier versions, be sure that your system has the correct APARs installed. All ISPF APARs that add or correct function relating to the year 2000 contain the YR2000 identifier in the APAR text. You should search for these APARs to ensure you have all the function available.

What function is included?

- ISPF Dialog variable ZSTDYEAR now correctly shows the year for dates past 1999. Earlier versions always showed the first 2 characters of the year as 19.
- A new ISPF dialog variable (ZJ4DATE) is available for Julian dates with a 4-digit year.
- An ISPF Configuration Table field enables PDF to interpret 2 character year dates as either a 19xx or 20xx date. The default value is 65. Any 2-character year date whose year is less than or equal to this value is considered a 20xx date, anything greater than this value is considered 19xx. To see what value has been set by the ISPF Configuration Table, use the new ZSWIND variable.
- New parameters in the LMMSTATS service (CREATED4 and MODDATE4) for specifying 4-character year dates. All existing parameters still exist and you can continue to use them. If both the 2-character year date parameters (CREATED and MODDATE) and the 4-character year date parameters (CREATED4 and MODDATE4) are specified, the 2-character versions are used.
- Dialog variables ZLC4DATE and ZLM4DATE have been added.
 - You *can* set them before making an LMMREP or LMMADD call. Do this to specify a 4-character *created* or *last modified* date to set in the ISPF statistics.
 - They *are* set by LMMFIND, LMMLIST and LMMDISP to the current value of the created and last modified dates in the ISPF statistics.

What might need to change? Some minor changes to your existing ISPF dialogs might be necessary, especially in ISPF dialogs that use the Library Access Services to manipulate ISPF member statistics.

- For those services that accept both 4-character year dates and 2-character year dates, you can specify one or the other. If you specify both, the 2-character year date is used to avoid affecting existing dialogs. When the 2-character year date is used, the configuration table field mentioned above is used to determine whether the date should be interpreted as 19xx or 20xx.
- ISPF will not necessarily show 4-character dates in all circumstances but it will process them correctly. For example, a member list might only display 2-character year dates but will sort those dates in the proper order.
- SCLM stores dates past the year 1999 in a new internal format. If an accounting file contains dates in this new format, it cannot be processed by a system without year 2000 support. Accounting files without dates past 1999 can be processed with or without the year 2000 support.
- No conversion of the LMF control file is necessary.

Elements and Features in OS/390

You can use the following table to see the relationship of a product you are familiar with and how it is referred to in OS/390 Version 2 Release 8.0. OS/390 V2R8.0 is made up of elements and features that contain function at or beyond the release level of the products listed in the following table. The table gives the name and level of each product on which an OS/390 element or feature is based, identifies the OS/390 name of the element or feature, and indicates whether it is part of the base or optional. For more compatibility information about OS/390 elements see *OS/390 Planning for Installation, GC28-1726*

Product Name and Level	Name in OS/390	Base or Optional
BookManager BUILD/MVS V1R3	BookManager BUILD	optional
BookManager READ/MVS V1R3	BookManager READ	base
MVS/Bulk Data Transfer V2	Bulk Data Transfer (BDT)	base
MVS/Bulk Data Transfer File-to-File V2	Bulk Data Transfer (BDT) File-to-File	optional
MVS/Bulk Data Transfer SNA NJE V2	Bulk Data Transfer (BDT) SNA NJE	optional
IBM OS/390 C/C++ V1R2	C/C++	optional
DFSMSdfp V1R3	DFSMSdfp	base
DFSMSdss	DFSMSdss	optional
DFSMSShsm	DFSMSShsm	optional
DFSMSrmm	DFSMSrmm	optional
DFSMS/MVS Network File System V1R3	DFSMS/MVS Network File System	base
DFSORT R13	DFSORT	optional
EREP MVS V3R5	EREP	base
FFST/MVS V1R2	FFST/MVS	base
GDDM/MVS V3R2 • GDDM-OS/2 LINK • GDDM-PCLK	GDDM	base
GDDM-PGF V2R1.3	GDDM-PGF	optional
GDDM-REXX/MVS V3R2	GDDM-REXX	optional
IBM High Level Assembler for MVS & VM & VSE V1R2	High Level Assembler	base
IBM High Level Assembler Toolkit	High Level Assembler Toolkit	optional
ICKDSF R16	ICKDSF	base
ISPF V4R2M1	ISPF	base
Language Environment for MVS & VM V1R5	Language Environment	base
Language Environment V1R5 Data Decryption	Language Environment Data Decryption	optional

Product Name and Level	Name in OS/390	Base or Optional
MVS/ESA SP V5R2.2		
BCP	BCP or MVS	base
ESCON Director Support	ESCON Director Support	base
Hardware Configuration Definition (HCD)	Hardware Configuration Definition (HCD)	base
JES2 V5R2.0	JES2	optional
JES3 V5R2.1	JES3	base
LANRES/MVS V1R3.1	LANRES	base
IBM LAN Server for MVS V1R1	LAN Server	base
MICR/OCR Support	MICR/OCR Support	base
OS/390 UNIX System Services	OS/390 UNIX System Services	base
OS/390 UNIX Application Services	OS/390 UNIX Application Services	base
OS/390 UNIX DCE Base Services (OSF DCE level 1.1)	OS/390 UNIX DCE Base Services	base
OS/390 UNIX DCE Distributed File Services (DFS) (OSF DCE level 1.1)	OS/390 UNIX DCE Distributed File Services (DFS)	optional
OS/390 UNIX DCE User Data Privacy	OS/390 UNIX DCE User Data Privacy	optional
SOMobjects Application Development Environment (ADE) V1R1	SOMobjects Application Development Environment (ADE)	
SOMobjects Runtime Library (RTL)	SOMobjects Runtime Library (RTL)	base
SOMobjects service classes	SOMobjects service classes	base
Open Systems Adapter Support Facility (OSA/SF) R1	Open Systems Adapter Support Facility (OSA/SF)	base
MVS/ESA RMF V5R2	RMF	optional
OS/390 Security Server	Resource Access Control Facility (RACF) <ul style="list-style-type: none"> • DCE Security Server • OS/390 Firewall Technologies • Lightweight Directory Access Protocol (LDAP) Client and Server • Open Cryptographic Enhanced Plug-ins (OCEP) 	optional
SDSF V1R6	SDSF	optional
SMP/E	SMP/E	base
	Softcopy Print	base
SystemView for MVS Base	SystemView for MVS Base	base
IBM TCP/IP V3R1 <ul style="list-style-type: none"> • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OS/390 Communications Service IP Applications • TCP/IP OS/2 Offload 	TCP/IP <ul style="list-style-type: none"> • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OS/390 Communications Service IP Applications • TCP/IP OS/2 Offload 	base <ul style="list-style-type: none"> • optional • optional • optional • optional • optional • optional
TIOC R1	TIOC	base
Time Sharing Option Extensions (TSO/E) V2R5	TSO/E	base

Product Name and Level	Name in OS/390	Base or Optional
VisualLift for MVS V1R1.1	<ul style="list-style-type: none"> • VisualLift Run-Time Environment (RTE) • VisualLift Application Development Environment (ADE) 	<ul style="list-style-type: none"> • base • optional
VTAM V4R3 with the AnyNet feature	VTAM	base
3270 PC File Transfer Program V1R1.1	3270 PC File Transfer Program	base

The ISPF User Interface

ISPF provides an action bar-driven interface that exploits many of the usability features of Common User Access (CUA) interfaces. Refer to *Object-Oriented Interface Design: IBM Common User Access Guidelines* for additional information.

The panels look different than in Version 3: all screens are in mixed case, and most have action bars at the top. These action bars give you a new way to move around in the product as well as access to command nesting. Command nesting allows you to *suspend* an activity while you perform a new one rather than having to end a function to perform another function.

This chapter primarily explains the action bar-driven interface and the use of ISPF's graphical user interface (GUI).

Some Terms You Should Know

The following terms are used in this book:

action bar. The area at the top of an ISPF panel that contains choices that give you access to actions available on that panel. When you select an action bar choice, ISPF displays a *pull-down menu*.

pull-down menu. A list of numbered choices extending from the selection you made on the action bar. The action bar selection is highlighted; for example, Utilities in Figure 1 on page xxxiii appears highlighted on your screen. You can select an action either by typing in its number and pressing Enter or by selecting the action with your cursor. ISPF displays the requested panel. If your choice contains an *ellipsis (...)*, ISPF displays a *pop-up window*. When you exit this panel or pop-up, ISPF closes the pull-down and returns you to the panel from which you made the initial action bar selection.

ellipsis. Three dots that follow a pull-down choice. When you select a choice that contains an ellipsis, ISPF displays a *pop-up window*.

pop-up window. A bordered temporary window that displays over another panel.

modal pop-up window. A type of window that requires you to interact with the panel in the pop-up before continuing. This includes cancelling the window or supplying information requested.

modeless pop-up window. A type of window that allows you to interact with the dialog that produced the pop-up before interacting with the pop-up itself.

point-and-shoot text. Text on a screen that is cursor-sensitive. See "Point-and-Shoot Text Fields" on page xxxvi for more information.

push button. A rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected (available only when you are running in GUI mode).

function key. In previous releases of ISPF, a programmed function (PF) key. *This is a change in terminology only.*

select. In conjunction with point-and-shoot text fields and action bar choices, this means moving the cursor to a field and simulating Enter.

mnemonics. Action bar choices can be defined with a underscored letter in the action bar choice text. In host mode you can access the action bar choice with the ACTIONS command and parameter 'x', where 'x' is the underscored letter in the action bar choice text. In GUI mode you can use a *hot key* to access a choice on the action bar; that is, you can press the ALT key in combination with the letter that is underscored in the action bar choice text.

How to Navigate in ISPF without Using Action Bars

If you use a non-programmable terminal to access OS/390 V2R8.0 ISPF and you do not want to take advantage of the command nesting function, you can make selections the same way you always have: by typing in a selection number and pressing Enter.

How to Navigate in ISPF Using the Action Bar Interface

Most ISPF panels have action bars at the top; the choices appear on the screen in white by default. Many panels also have point-and-shoot text fields, which appear in turquoise by default. The panel shown in Figure 3 on page xxxiv has both.

Action Bars

Action bars give you another way to move through ISPF. If the cursor is located somewhere on the panel, there are several ways to move it to the action bar:

- Use the cursor movement keys to manually place the cursor on an action bar choice.
- Type **ACTIONS** on the command line and press Enter to move the cursor to the first action bar choice.
- Press F10 (Actions) or the Home key to move the cursor to the first action bar choice.

If mnemonics are defined for action bar choices, you can:

- In 3270 mode, on the command line, type **ACTIONS** and the mnemonic letter that corresponds to an underscored letter in the action bar choice text. This results in the display of the pull-down menu for that action bar choice.
- In 3270 mode, on the command line enter the mnemonic letter that corresponds to an underscored letter in the action bar choice text, and press the function key assigned to the **ACTIONS** command. This results in the display of the pull-down menu for that action bar choice.
- In GUI mode, you can use a *hot key* to access a choice on an action bar or on a pull-down menu; that is, you can press the ALT key in combination with the mnemonic letter that is underscored in the choice text to activate the text.

Use the tab key to move the cursor among the action bar choices. If you are running in GUI mode, use the right and left cursor keys.

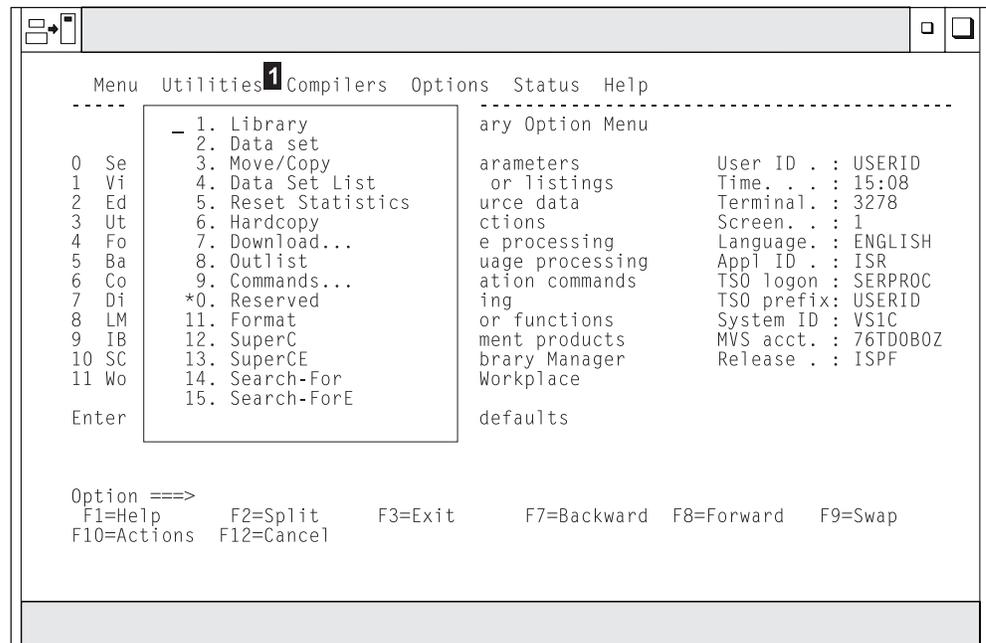
Notes:

1. ISPF does not provide a mouse emulator program. This book uses *select* in conjunction with point-and-shoot text fields and action bar choices to mean moving the cursor to a field and simulating Enter.

Note: Some users program their mouse emulators as follows:

- Mouse button 1 – to position the cursor to the pointer and simulate Enter
 - Mouse button 2 – to simulate F12 (Cancel).
2. If you want the Home key to position the cursor at the first input field on an ISPF panel, type **SETTINGS** on any command line and press Enter to display the ISPF Settings panel. Deselect the **Tab to action bar choices** option.
 3. If you are running in GUI mode, the Home key takes you to the beginning of the current field.

When you select one of the choices on the action bar, ISPF displays a pull-down menu. Figure 1 shows the pull-down menu displayed when you select Utilities on the ISPF Primary Option Menu action bar.



1 The selected action bar choice is highlighted.

Figure 1. Panel with an Action Bar Pull-Down Menu

To select a choice from the Utilities pull-down menu, type its number in the entry field (underlined) and press Enter or select the choice. To cancel a pull-down menu without making a selection, press F12 (Cancel). For example, if you select choice 9, ISPF displays the Command Table Utility pop-up, as shown in Figure 2 on page xxxiv.

Note: If you entered a command on the command line prior to selecting an action bar choice, the command is processed, and the pull-down menu is never displayed. The CANCEL, END, and RETURN commands are exceptions. These three commands are not processed and the cursor is repositioned to the first input field in the panel body. If there is no input field, the cursor is repositioned under the action bar area. If you are running in GUI mode and select an action bar choice, any existing command on the command line is ignored.

The ISPF User Interface

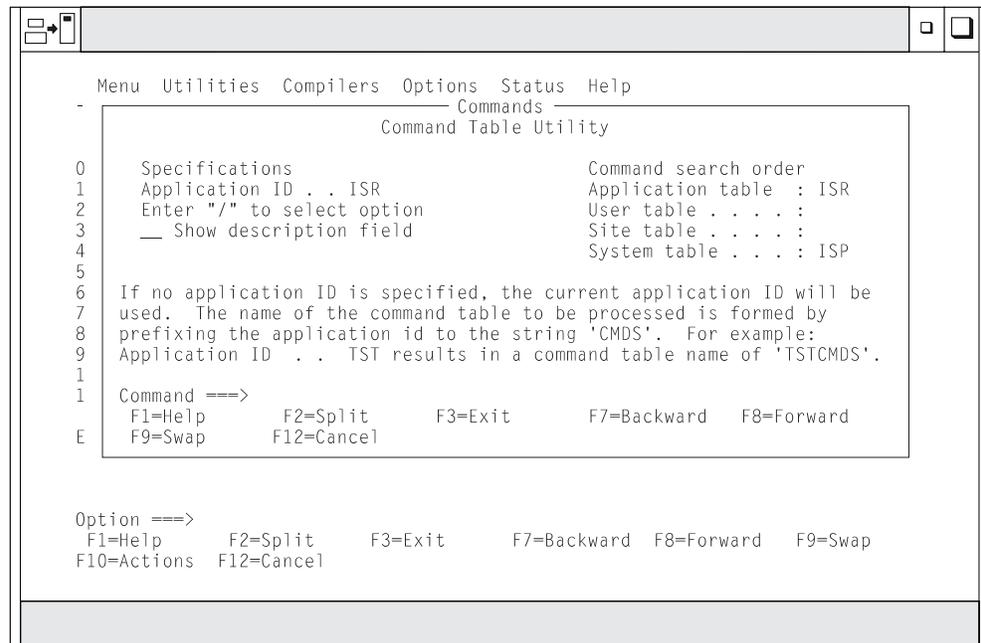
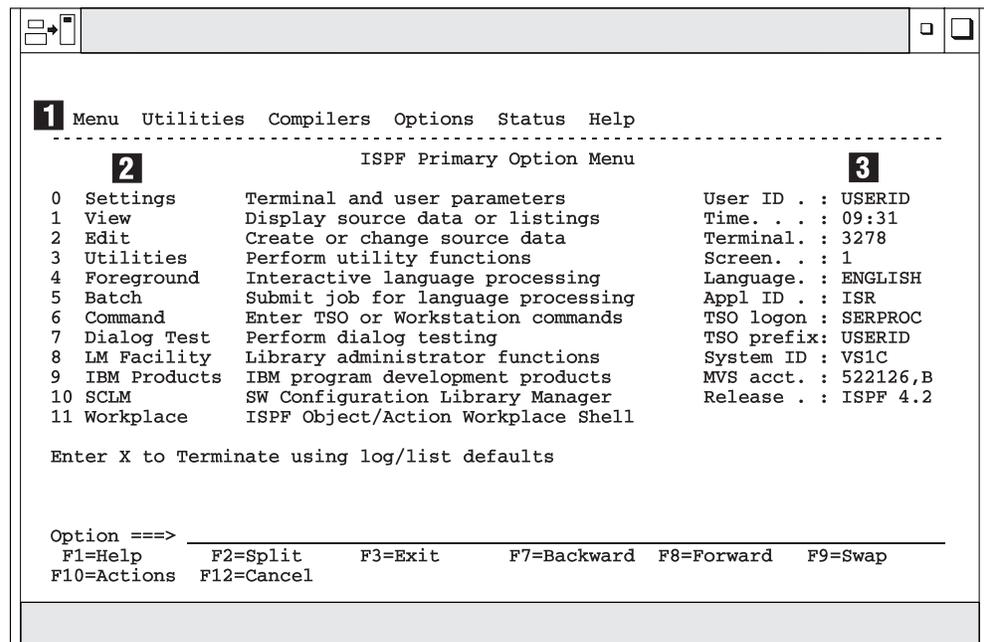


Figure 2. Pop-Up Selected from an Action Bar Pull-Down



- 1** Action bar. You can select any of the action bar choices and display a pull-down.
- 2** Options. The fields in this column are point-and-shoot text fields.
- 3** Dynamic status area. You can specify what you want to be displayed in this area.

Figure 3. Panel with an Action Bar and Point-and-Shoot Fields

Action Bar Choices

The action bar choices available vary from panel to panel, as do the choices available from their pull-downs. However, Menu and Utilities are basic action bar choices, and the choices on their pull-down menus are always the same.

Menu Action Bar Choice

The following choices are available from the Menu pull-down:

Settings	Displays the ISPF Settings panel
View	Displays the View Entry panel
Edit	Displays the Edit Entry panel
ISPF Command Shell	Displays the ISPF Command Shell panel
Dialog Test...	Displays the Dialog Test Primary Option panel
Other IBM Products...	Displays the Additional IBM Program Development Products panel
SCLM	Displays the SCLM Main Menu
ISPF Workplace	Displays the Workplace entry panel
Status Area...	Displays the ISPF Status panel
Exit	Exits ISPF.

Note: If a choice displays in blue (the default) with an asterisk as the first digit of the selection number (if you are running in GUI mode, the choice will be *grayed*), the choice is unavailable for one of the following reasons:

- Recursive entry is not permitted here
- The choice is the current state; for example, RefMode is currently set to Retrieve in Figure 4 on page xxxvi.

The cursor-sensitive portion of a field often extends past the field name. Until you are familiar with this new feature of ISPF, you might want to display these fields in reverse video (use the PSCOLOR command to set Highlight to REVERSE).

Note: You can use the Tab key to position the cursor to point-and-shoot fields by selecting the **Tab to point-and-shoot fields** option on the ISPF Settings panel (Option 0).

Function Keys

ISPF uses CUA-compliant definitions for function keys F1–F12 (except inside the Edit function). F13–F24 are the same as in ISPF Version 3. By default you see the CUA definitions because your **Primary range** field is set to 1 (Lower - 1 to 12).

To use non-CUA-compliant keys, select the **Tailor function key display** choice from the Function keys pull-down on the ISPF Settings (option 0) panel action bar. On the Tailor Function Key Definition Display panel, specify 2 (Upper - 13 to 24) in the **Primary range** field.

The following function keys help you navigate in ISPF:

- F1 Help.** Displays Help information. If you press F1 (and it is set to Help) after ISPF displays a short message, a long message displays in a pop-up window.
- F2 Split.** Divides the screen into two logical screens separated by a horizontal line or changes the location of the horizontal line.

Note: If you are running in GUI mode, each logical screen displays in a separate window.
- F3 Exit** (from a pull-down). Exits the panel underneath a pull-down.
- F3 End.** Ends the current function.
- F7 Backward.** Moves the screen up the scroll amount.
- F8 Forward.** Moves the screen down the scroll amount.
- F9 Swap.** Moves the cursor to where it was previously positioned on the other logical screen of a split-screen pair.
- F10 Actions.** Moves the cursor to the action bar. If you press F10 a second time, the cursor moves to the command line.
- F12 Cancel.** Issues the Cancel command. Use this command to remove a pull-down menu if you do not want to make a selection. F12 also moves the cursor from the action bar to the Option ==> field on the ISPF Primary Option Menu. See *ISPF Dialog Developer's Guide and Reference* for cursor-positioning rules.
- F16 Return.** Returns you to the ISPF Primary Option Menu or to the display from which you entered a nested dialog. RETURN is an ISPF system command.

Selection Fields

OS/390 V2R8.0 ISPF uses the following CUA-compliant conventions for selection fields:

The ISPF User Interface

A single period (.)

Member lists that use a single period in the selection field recognize only a single selection. For example, within the Edit function you see this on your screen:

```
EDIT      USER1.PRIVATE.TEST                ROW 00001 of 00002
Name      VV MM  Created   Changed  Size  Init  Mod  ID
. MEM1    01.00  94/05/12  94/07/22  40    0    0  USER1
. MEM2    01.00  94/05/12  94/07/22  30    0    0  KEENE
```

You can select only one member to edit.

A single underscore (_)

Selection fields marked by a single underscore prompt you to use a slash (/) to select the choice. You may use any non-blank character. For example, the **Panel display CUA mode** field on the ISPF Settings panel has a single underscore for the selection field:

```
Options
Enter "/" to select option
_ Command line at bottom
_ Panel display CUA mode
_ Long message in pop-up
```

Note: If you are running in GUI mode, this type of selection field displays as a check box; that is, a square box with associated text that represents a choice. When you select a choice, a check mark (in OS/2) or an X (in Windows) appears in the check box to indicate that the choice is in effect. You can clear the check box by selecting the choice again.

An underscored field (___)

Member lists or text fields that use underscores in the selection field recognize multiple selections. For example, from the Display Data Set List Option panel, you may select multiple members for print, rename, delete, edit, browse, or view processing.

Command Nesting

Command nesting allows you to *suspend* an activity while you perform a new one rather than having to end a function to perform another function. For example, in previous versions of ISPF, if you are editing a data set and want to allocate another data set, you type =3.2 on the command line and press Enter. ISPF *ends* your edit session before taking you to the Data Set Utility panel. When you have allocated the data set and want to return to your edit session, you type =2 and press Enter; ISPF returns you to the Edit Entry Panel. With OS/390 V2R8.0 ISPF, from your edit session, select the Data set choice from the Utilities pull-down on the Edit panel action bar. ISPF suspends your edit session and displays the Data Set Utility panel. When you have allocated the new data set and end the function, OS/390 V2R8.0 ISPF returns you directly to your edit session rather than to the Edit Entry Panel.

Part 1. The ISPF Editor

Chapter 1. Introducing the ISPF Editor	3	Edit Recovery	47
What is ISPF?	3	Chapter 3. Managing Data	49
What the ISPF Editor Does	4	Creating and Replacing Data	49
How to Use the ISPF Editor	4	Copying and Moving Data	50
Beginning an Edit Session	4	Shifting Data	51
Edit Entry Panel Action Bar	5	Column Shift	51
Edit Entry Panel Fields	7	Column Shifting in Lines that Contain DBCS Strings	52
Creating a New Data Set	10	Data Shift	52
Editing an Existing Data Set	11	Finding, Seeking, Changing, and Excluding Data	53
Using the ISPF Editor Basic Functions	14	Specifying the Search String	54
Ending an Edit Session	15	Simple and Delimited Strings	54
Edit Commands	16	Character Strings	55
Line Commands	16	Picture Strings (String-1)	55
Primary Commands	17	Picture Strings (String-2)	56
Edit Commands and PF Key Processing	17	Effect of CHANGE Command on Column-Dependent Data	57
Edit Macros	18	Using the CHANGE Command With EBCDIC and DBCS Data	57
Editing Data in Controlled Libraries	19	Controlling the Search	57
Packing Data	19	Extent of the Search	57
Chapter 2. Controlling the Edit Environment	21	Starting Point and Direction of the Search	58
What is an Edit Profile?.	21	Qualifying the Search String	59
Using Edit Profile Types	21	Column Limitations	59
Displaying or Defining an Edit Profile	21	Split Screen Limitations	59
Modifying an Edit Profile	23	Excluded Line Limitations	60
Locking an Edit Profile	23	Using the X (Exclude) Line Command with FIND and CHANGE	60
Edit Modes	23	Repeating the FIND, CHANGE, and EXCLUDE Commands	60
Edit Profile Modes	24	Examples	61
Edit Mode Defaults	25	FIND Command Example	61
Site-wide Edit Profile Initialization	25	CHANGE Command Example	62
Creating a ZDEFAULT Edit Profile	26	EXCLUDE Command Example	64
Flagged Lines	27	Excluding Lines	65
Changed Lines	27	Redisplaying Excluded Lines	65
Error Lines	27	Redisplaying a Range of Lines	66
Special Lines	27	Labels and Line Ranges	66
Edit Boundaries	28	Editor-Assigned Labels	66
Initial Macros	29	Specifying a Range	67
Application-Wide Macros	30	Using Labels and Line Ranges	68
Statistics for PDS Members	30	Word Processing	68
Effect of Stats Mode When Beginning an Edit Session	31	Formatting Paragraphs	68
Effect of Stats Mode When Saving Data	31	Using Text Flow on a DBCS Terminal	69
Version and Modification Level Numbers	31	Splitting Lines	70
Sequence Numbers	32	Splitting Lines Within a DBCS String	70
Sequence Number Format and Modification Level	32	Entering Text (Power Typing)	71
Sequence Number Display	33	Entering Text on a DBCS Terminal	71
Initialization of Number Mode	33	Using Tabs	71
Enhanced and Language-sensitive Edit Coloring	34	Types of Tabs	71
Language Support	34	Software and Hardware Tabs	71
Automatic Language Selection	35	Logical Tabs	72
Language Processing Limitations and Idiosyncracies	36	Effect of TABS Commands on Tab Types	72
The HILITE Command/Dialog	38	Defining and Controlling Tabs	72
HILITE Operands	38	Defining Software Tab Positions	73
The HILITE Dialog	40		
Highlighting Status and the Edit Profile	46		

Defining Hardware Tab Positions	73
Limiting the Size of Hardware Tab Columns	73
Using Attribute Bytes	74
Undoing Edit Interactions	74
UNDO Processing	75
Understanding Differences in SETUNDO Processing	76
Chapter 4. Using Edit Models	77
What Is an Edit Model?.	77
How Models Are Organized	77
How to Use Edit Models	79
Adding, Finding, Changing, and Deleting Models	81
Adding Models	81
Finding Models	85
Changing Models.	86
Deleting Models	86

Chapter 1. Introducing the ISPF Editor

This chapter introduces the ISPF Editor. It provides an overview of:

- The ISPF editor functions
- A typical edit session
- Edit commands
- Edit macros.

Note:

Beginning with ISPF Version 4 Release 2, ISPF enables you to more fully utilize your desktop workstation's potential by giving you the ability to edit host data on the workstation, and workstation data on the host. ISPF calls this function *distributed editing*.

The ISPF Workstation Tool Integration dialog, or tool integrator, is a workstation customization tool that enables any workstation application to use data from an MVS host system. After setting up the tool integrator, your workstation-installed applications can interact with the ISPF View and Edit functions and services. Data flow goes both ways with the tool integrator connection. You can work with workstation files on the host or with host files on the workstation.

For more information about distributed editing, refer to the *ISPF User's Guide* and the *ISPF Services Guide*.

What is ISPF?

The Interactive System Productivity Facility (ISPF) is a dialog manager that provides tools to improve program, dialog, and development productivity and control.

The PDF component of ISPF is an integrated work environment used to develop programs, dialogs, and documents. The PDF component provides an MVS-compatible hierarchical library containing numerous productivity-improving functions. Some examples of these functions are:

- ISPF dialog test tools
- Full-screen editor, with a dialog interface called edit macros
- Multiple update access to data sets
- Online tutorials
- Data set management
- Customized library controls.

Note: References in this book to library controls apply to LMF. For information about using SCLM to control libraries, refer to *ISPF Software Configuration and Library Manager (SCLM) Developer's and Project Manager's Guide*.

This book describes the ISPF editor and its dialog interface. A *dialog* is a program running under ISPF. The interface allows a dialog to access the usual ISPF dialog functions and the ISPF editor functions.

What the ISPF Editor Does

You can use the ISPF editor to create, display, and change data stored in ISPF libraries or other partitioned or sequential data sets with the following characteristics:

- Record Format (RECFM):
 - Fixed or variable (non-spanned)
 - Blocked or unblocked
 - With or without printer control characters.
- Logical Record Length (LRECL):
 - From 10 to 32760, inclusive, for fixed-length records
 - From 14 to 32756, inclusive, for variable-length records.

Note: For variable-length records, the amount of editable data in each record is 4 bytes less than the logical record length.

Generally, the editor truncates variable-length lines by removing blanks at the end of each line during a save. However, for variable-length lines containing exactly 8 bytes (normally the line number), the editor adds one blank during a save so that the line length is not zero after removing the line number. If a variable-length line is completely blank and has no line number, a blank is added so that the line length is not zero.

However, with the PRESERVE function, you can save the trailing blanks of variable length files. The **Preserve VB record length** field on the Edit Entry panel and the PRESERVE edit and macro commands enable you to save or truncate the blanks as you prefer.

How to Use the ISPF Editor

This section provides an overview of an edit session and covers:

- Beginning an Edit Session
- Using the ISPF editor Basic Functions
- Ending an Edit Session.

Beginning an Edit Session

To begin using the ISPF editor, select option 2 on the ISPF Primary Option Menu. PDF then displays the Edit Entry panel (Figure 5 on page 5).

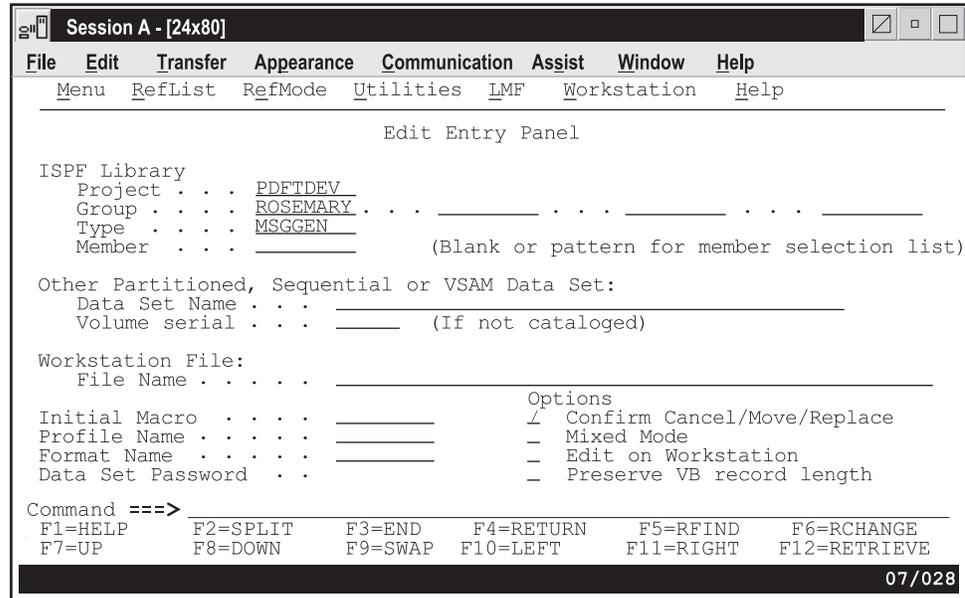


Figure 5. Edit Entry Panel (ISREDM01)

Edit Entry Panel Action Bar

The Edit Entry panel action bar choices function as follows:

Menu See “Menu Action Bar Choice” on page xxxv for information on the Menu pull-down.

Reflist The Reflist pull-down offers the following choices:

- 1 **Reference Data Set List** displays the Reference Data Set List panel, which displays a list of up to 30 data set names you have referenced in PDF panels.
- 2 **Reference Library List** displays the Reference Library List panel.
- 3 **Personal Data Set List** displays the Personal Data Set List panel, of which you can have any number, as long as each has a unique name.
- 4 **Personal Data Set List Open...** displays the **Open** dialog for all Personal Data Sets.
- 5 **Personal Library List** displays the Personal Library List panel, which maintains up to 8 lists, each with a unique name. If more than one list exists, the most recently used list displays.
- 6 **Personal Library List Open...** displays the **Open** dialog for all Personal Library Lists.

Refmode

Refmode sets reference lists to either retrieve or execute mode. The Refmode pull-down offers the following choices:

- 1 **List Execute** sets reference lists, personal data set list and personal library lists into an execute mode. When you select an entry from the list, the information is placed into the ISPF Library or the

How to Use the ISPF Editor

“Other” **Data Set Name** field and an Enter key is simulated. (If this setting is current, the choice is unavailable.)

- 2 **List Retrieve** sets reference lists, personal data set list and personal library lists into a retrieve mode. When you select an entry from the list, the information is placed into the ISPF Library or the “Other” **Data Set Name** field, but the Enter key is *not* simulated. (If this setting is current, the choice is unavailable.)

Utilities

See “Utilities Action Bar Choice” on page xxxvi for information on the Utilities pull-down.

LMF If LMF is installed on your system, you can edit-lock a member of a controlled library that is part of a concatenation sequence, but only if the member does not exist in your private library. Specify one of the following values for **LMF**:

- 1 **Lock - Never.** Tells ISPF not to edit-lock the member and to retain this value for future Edit sessions.
- 2 **Lock - No.** Tells ISPF not to edit-lock the member, but to change this value to YES for the next Edit session.
- 3 **Lock - Yes.** Tells ISPF to edit-lock the member. The member is locked under your user ID.

Edit-locking is important for two reasons:

- Keeping other users from accessing that member while you are editing it.
- Promoting the member back to the controlled library when you have finished editing it. If you do not edit-lock the member, you cannot promote it.

If you save any changes you made while editing the member, it remains locked in your private library. The version of the member stored in the controlled library remains unchanged until you promote the one in your private library. If you leave Edit without saving the changes, they are lost.

Conditions When LMF Locking Is Ignored

ISPF ignores the 3 (Yes) value if:

- The LMF control file (ISRCFIL) is not allocated, which means the **LMF Lock** field has no meaning to ISPF.
- A member in your private library has the same name as the controlled library member that you want to edit-lock. Here, you can either rename, move, or delete the private library member.
- The member is controlled by SCLM only.

Conditions When LMF Locking Causes Errors

The following conditions can cause an error if you specify 3 (Yes):

- The libraries are not concatenated in the proper sequence.
- The library controls are not active.
- The member you want to edit-lock is locked with another user’s user ID.
- The member is controlled by both LMF and SCLM.

Even if an error condition keeps you from locking a member, you can still edit it. Just remember that you cannot promote it back to the controlled library afterwards. ISPF displays a panel that gives you a choice between pressing Enter to edit the member or entering the END command if you decide not to edit.

For information about using the SCLM Edit option to lock data sets or members, refer to the *ISPF Software Configuration and Library Manager (SCLM) Developer's and Project Manager's Guide*

Workstation

Configure ISPF workstation tool integration. For information about the workstation and ISPF, refer to the *OS/390 ISPF User's Guide*.

Help The Help pull-down offers the following choices:

- General
- Types of Data Sets
- Edit entry panel
- Member selection list
- Display screen format
- Scrolling data
- Sequence numbering
- Display modes
- Tabbing
- Automatic recovery
- Edit profiles
- Edit line commands
- Edit primary commands
- Labels and line ranges
- Ending an edit session
- Appendices
- Index.

Edit Entry Panel Fields

You can specify a concatenated sequence of up to four ISPF libraries, but the libraries must have been previously allocated to ISPF with the Data Set utility (3.2).

The fields on this panel are:

Project

The common identifier for all ISPF libraries belonging to the same programming project.

Group The identifier for the particular set of ISPF libraries; that is, the level of the libraries within the library hierarchy.

You can specify a concatenated sequence of up to four existing ISPF libraries.

The editor searches the ISPF libraries in the designated order to find the member and copies it into working storage. If the editor does not find the member in the library, it creates a new member with the specified name.

When you save the edited member, the editor places or replaces it in the first ISPF library in the concatenation sequence, regardless of which library it was copied from.

Type The identifier for the type of information in the ISPF library.

How to Use the ISPF Editor

Member

The name of an ISPF library or other partitioned data set member. Leaving this field blank or entering a pattern causes PDF to display a member list. Refer to *ISPF User's Guide* if you need information about entering a pattern.

Data Set Name

Any fully-qualified data set name, such as 'USERID.SYS1.MACLIB', or a VSAM data set name. If you include your TSO user prefix (defaults to user ID), you must enclose the data set name in apostrophes. However, if you omit the TSO user prefix and apostrophes, your TSO user prefix is automatically added to the beginning of the data set name.

If you specify a VSAM data set, ISPF checks the configuration table to see if VSAM support is enabled. If it is, the specified tool is invoked. If VSAM is not supported by the configuration settings, an error message is displayed.

Volume Serial

A real DASD volume or a virtual volume residing on an IBM 3850 Mass Storage System. To access 3850 virtual volumes, you must also have MOUNT authority, which is acquired through the TSO ACCOUNT command.

Workstation File:

If you have made a connection to the workstation, you can also specify a workstation file name, for example **C: \AUTOEXEC.BAT**, on the Edit Entry Panel. Or you can specify which environment (host or workstation) should be used to edit a data set. With these options, one of four editing situations can occur:

- Edit a host data set on the host
- Edit a host data set on the workstation
- Edit a workstation file on the host
- Edit a workstation file on the workstation.

Edit a Host Data Set on the Host

The editor searches the ISPF libraries in the designated order to find the member and copy it into working storage. If you specified a nonexistent member of an ISPF library, a new member is created with the specified name.

When you save the edited member, the editor places or replaces it in the first ISPF library in the concatenation sequence, regardless of which library it was copied from.

Edit a Host Data Set on the Workstation

The editor searches the ISPF libraries in the designated order to find the member and copy it into working storage. The data set name is converted to a workstation file name, and that name is appended to the workstation's current working directory. The host data set is transferred to the workstation, and the working file is then passed to the user's chosen edit program.

Note: Because of certain JAVA restrictions, if you are using the ISPF Application Server and ISPF Workstation Agent Applet, the editor you use on the workstation is the one provided by ISPF. You cannot choose a different workstation edit

program. For more information about the ISPF Application Server, refer to the *OS/390 ISPF Application Server Guide and Reference*.

When you finish the edit session, the working file is transferred back to the host and stored in the first ISPF library in the concatenation sequence.

Edit a Workstation File on the Host

The editor searches the workstation files to find the desired file and copy it into working storage. The workstation file name is converted to a host data set name, and, if greater than 44 characters, it is truncated to be 44. The workstation file is transferred to the host, where you can edit it.

When you finish the edit session, the working file is transferred back to the workstation and stored.

Edit a Workstation File on the Workstation

This edit proceeds as it normally does on your workstation.

Initial Macro

You can specify a macro to be processed before you begin editing your sequential data set or any member of a partitioned data set. This initial macro allows you to set up a particular editing environment for the Edit session you are beginning. This initial macro overrides any IMACRO value in your profile.

If you leave the **Initial Macro** field blank and your edit profile includes an initial macro specification, the initial macro from your edit profile is processed.

If you want to suppress an initial macro in your edit profile, type NONE in the **Initial Macro** field. See “Initial Macros” on page 29 and “IMACRO—Specify an Initial Macro” on page 253 for more details.

Profile Name

The name of an edit profile, which you can use to override the default edit profile. See the description in “What is an Edit Profile?” on page 21.

Format Name

The name of a format definition or blank if no format is to be used.

Data Set Password

The password for OS password-protected data sets. This is not your RACF* password.

Confirm Cancel/Move/Replace

When you select this field with a “/”, a confirmation panel displays when you request one of these actions, and the execution of that action would result in data changes being lost or existing data being overwritten.

- For MOVE, the confirm panel is displayed if the data to be moved exists. Otherwise, an error message is displayed.
- For REPLACE, the confirm panel is displayed if the data to be replaced exists. Otherwise, the REPLACE command functions like the edit CREATE command, and no confirmation panel is displayed.
- For CANCEL, the confirmation panel is displayed if any data changes have been made, whether through primary commands, line commands, or typing.

How to Use the ISPF Editor

Note: Any commands or data changes pending at the time the CANCEL command is issued are ignored. Data changes are "pending" if changes have been made to the displayed edit data, but no interaction with the host (ENTER, PF key, or command other than CANCEL) has occurred. If no other changes have been made during the edit session up to that point, the confirmation panel is not displayed.

Mixed Mode

When you select this field with a "/" , it specifies that the editor look for shift-out and shift-in delimiters surrounding DBCS data. If you do not select it, the editor does not look for mixed data.

Edit on Workstation

You can select this option to use your workstation as the editing environment for whichever host data set or workstation file you want to edit.

Preserve VB record length

You can select this option to cause the editor to store the original length of each record in variable length data sets and when a record is saved, the original record length is used as the minimum length for the record.

Note: Double-Byte Character Set Support

The ISPF editor supports DBCS alphabets in two ways:

- Formatted data where DBCS characters are in the column positions specified in the format definition created with the Format Utility (option 3.11)
- Mixed characters delimited with the special shift-out and shift-in characters.

If you are using mixed mode and the record length of a data set is greater than 72 bytes, there is a possibility that a DBCS character might encroach on the display boundary. Here, PDF attempts to display the other characters by replacing an unpaired DBCS character byte with an SO or SI character. If there is a possibility that the replaced SO or SI character was erased, the line number of the line is highlighted. If you change the position of the SO and SI characters on the panel, or if you delete the SO and SI characters entirely, the DBCS character on the boundary is removed to keep the rest of the data intact.

Creating a New Data Set

Before you can edit a new sequential data set, you must allocate space for it. When you specify an empty sequential data set or nonexistent member of a partitioned data set, the first edit display contains several empty lines between the Top of Data and Bottom of Data message lines (Figure 6 on page 11). The editor replaces the quote marks on the left of the panel with sequence numbers when you type information on the lines.

See "Creating and Replacing Data" on page 49 and "Word Processing" on page 68 for more information on using the editor to create data.

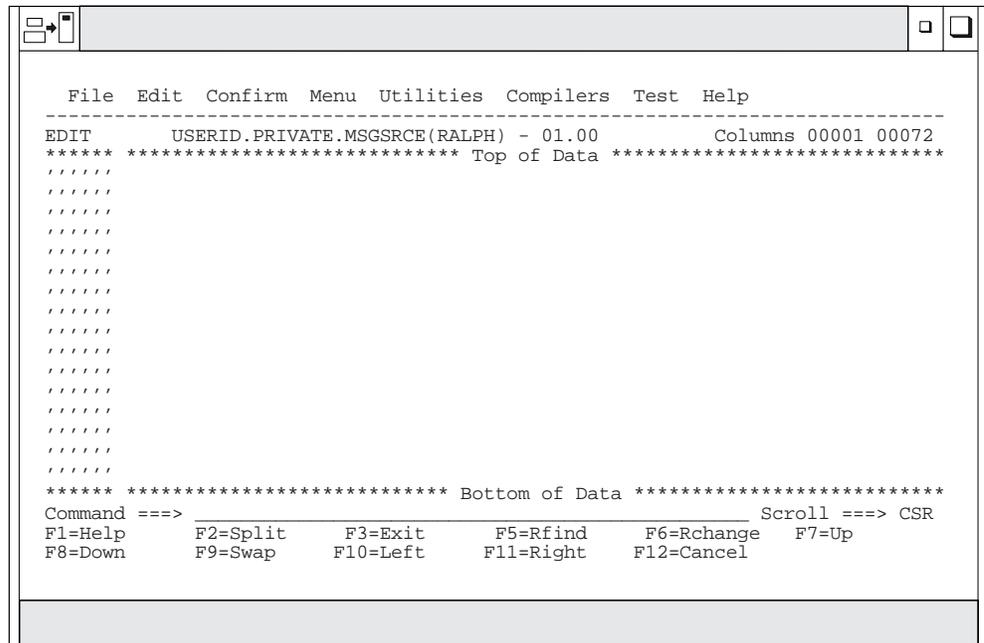


Figure 6. Creating a New Data Set (ISREDDE2)

Editing an Existing Data Set

When you edit an existing data set, ISPF displays the Primary Edit Panel as shown in Figure 7.

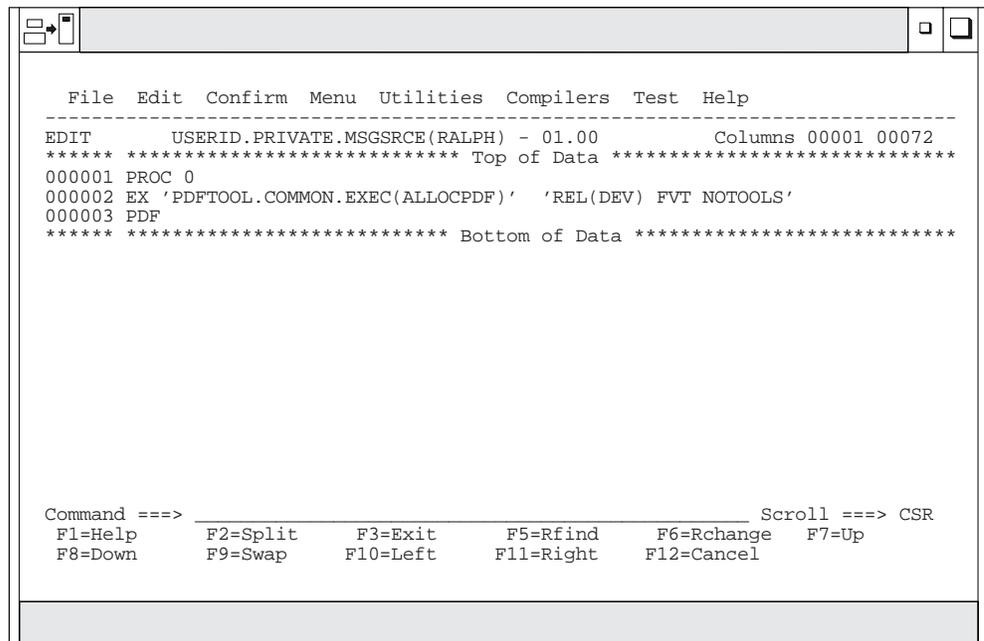


Figure 7. Example Primary Edit Panel (ISREDDE2)

Primary Edit Panel Action Bar Choices: The Primary Edit panel action bar choices function as follows:

File The File pull-down offers you the following choices:

How to Use the ISPF Editor

- 1 **Save** executes the SAVE command.
- 2 **Cancel** executes the CANCEL command (which ignores all changes made to the member) and redisplay the Edit Entry panel.
- 3 **Exit** executes the END command (which saves the data set or member) and redisplay the Edit Entry panel.

Edit The Edit pull-down offers you the following choices:

- 1 **Reset** performs the RESET command.
- 2 **Undo** performs the UNDO command.
- 3 **Hilite** displays the Edit Color Settings pop-up.

Confirm

When selected, causes an additional panel to display to confirm cancellations, moves, or replacements.

- 1 **Set Confirm Cancel/Move/Replace On** causes the additional panel to display.
- 2 **Set Confirm Cancel/Move/Replace Off** prevents the additional panel from displaying.

Menu See "Menu Action Bar Choice" on page xxxv for information on the Menu pull-down.

Utilities

See "Utilities Action Bar Choice" on page xxxvi for information on the Utilities pull-down.

Compilers

Foreground Compilers... offers you the following choices:

- 1 **Assembler** displays the Foreground Assembler panel.
- 2 **COBOL** displays the Foreground COBOL Compiler panel.
- 3 **VS FORTRAN** displays the Foreground VS FORTRAN Compiler panel.
- 5 **PL/I** displays the Foreground PL/I Compiler panel.
- 6 **VS PASCAL** displays the Foreground VS PASCAL Compiler panel.
- 7 ***Binder/Link Editor** displays the Foreground Linkage Edit panel.
- 9 **Script/VS** displays the Script/VS Processor panel.
- 10 ***VS COBOL II debug** displays the Foreground VS COBOL II Interactive DEBUG panel.
- 10A ***OS/VS COBOL debug** displays the COBOL Interactive Debug panel.
- 11 ***FORTRAN Debug** displays the FORTRAN Interactive DEBUG panel.
- 12 **Member Parts List** displays the Foreground Member Parts List panel.
- 13 ***C/370** displays the Foreground C/370 Compiler panel.
- 14 ***REXX 370** displays the Foreground REXX/370 Compiler panel.
- 15 ***ADA/370** displays the Foreground ADA/370 Compiler panel.

- 16 *AD/Cycle C/370 displays the Foreground AD/Cycle C/370 Compiler panel.
- 18 ISPDTLC displays the ISPF Dialog Tag Language conversion utility panel.
- 19 *OS/390 C/C++ displays the C/C++ for MVS/ESA compiler panel, if you have the compiler installed on your system.

Background Compilers... offers you the following choices:

- 1 **Assembler** displays the Batch Assembler panel.
- 2 **COBOL** displays the Batch COBOL Compiler panel.
- 3 **VS FORTRAN** displays the Batch VS FORTRAN Compiler panel.
- 4 **PL/I** displays the Batch PL/I Compiler panel.
- 6 **VS PASCAL** displays the Batch VS PASCAL Compiler panel.
- 7 ***Binder/Link Editor** displays the Batch Linkage Edit panel.
- 10 ***VS COBOL II Debug** displays the Batch VS COBOL II Interactive Debug panel.
- 12 **Member Parts List** displays the Batch Member Parts List panel.
- 13 ***C/370** displays the Batch C/370 Compiler panel.
- 14 ***REXX/370** displays the Batch REXX/370 Compiler panel.
- 15 ***ADA/370** displays the Batch ADA/370 Compiler panel.
- 16 ***AD/Cycle C/370** displays the Batch AD/Cycle C/370 Compiler panel.
- 17 **AD/Cycle COBOL/370** displays the Foreground AD/Cycle COBOL/370 Compiler panel.
- 18 **ISPDTLC** displays the ISPF Dialog Tag Language conversion utility panel.
- 19 ***OS/390 C/C++** displays the ESA compiler panel, if you have the compiler installed on your system.
- 20 ***SOMobjects for MVS** displays the SOMobjects for MVS compiler panel, if you have the compiler installed on your system.

ISPPREP Panel utility displays the PreProcessed Panel Utility.

DTL Compiler displays the ISPF Dialog Tag Language Conversion Utility.

Test The Test pull-down offers you the following choices:

- 1 **Functions...** displays the Dialog Test Function/Selection panel.
- 2 **Panels** displays the Dialog Test Display panel.
- 3 **Variables...** displays the Dialog Test Variables panel.
- 4 **Tables...** displays Dialog Test Tables panel.
- 5 **Log** displays the ISPF Transaction Log panel.
- 6 **Services...** displays the Invoke Dialog Service panel.
- 7 **Traces...** displays the Dialog Test Traces panel.
- 8 **Break Points...** displays the Dialog Test Breakpoints panel.

How to Use the ISPF Editor

- 9 **Dialog Test...** displays the Dialog Test Primary Option panel.
- 10 **Dialog Test appl ID...** displays the Dialog Test Application ID panel.

Help The Help pull-down offers you the following choices:

- General
- Display screen format
- Scrolling Data
- Sequence numbering
- Display modes
- Tabbing
- Automatic recovery
- Edit profiles
- Edit line commands
- Edit Primary commands
- Labels and line ranges
- Ending an edit session
- Appendices
- Index.

Editing the Data Set: When the editor displays existing data, each line consists of a 6-column Line Command field followed by a 72-column data field. The Line Command fields contain the first 6 digits of the sequence numbers in the data. If the data has no sequence numbers, the Line Command fields contain relative numbers that start at 1 and are incremented by 1.

Based on your action, the ISPF editor places the cursor in the most useful position. To help you find the cursor, the editor intensifies the Line Command field that contains the cursor.

If the data contains characters that cannot be displayed, blanks replace those characters on the panel but not in the data. You cannot type over the blanks. You can display and edit undisplayable characters by entering hexadecimal mode or by using the FIND and CHANGE commands with hexadecimal strings. See “HEX—Display Hexadecimal Characters” on page 247 for information on entering hexadecimal mode.

Printer control characters, if present, are displayed and are treated as part of the data. ASA control characters are alphanumeric and you can edit them. Machine control characters, however, cannot be displayed and are replaced on the panel with blanks.

When you are editing existing data, the selected member or sequential data set is read into virtual storage, where it is updated during edit operations. Use of virtual storage for editing work space results in high performance, but might require a large user region. If you use all available storage, an ABEND occurs, and you lose the work space unless recovery mode is on.

Using the ISPF Editor Basic Functions

The ISPF editor is similar to many modern word processors. Its basic functions are simple and can be used immediately:

- To alter data, type over the existing material or use the Ins (Insert) and Del (Delete) keys to add or remove characters.

- To view data that is not displayed, use the scroll commands. The following are PDF default values:

F7/19	Scrolls up.	F10/22	Scrolls left.
F8/20	Scrolls down.	F11/23	Scrolls right.

- To insert a line between existing lines, type I over a number in the Line Command field and press Enter. The Line Command field is the 6-column row displayed on the left side of the panel when you create or edit a data set. The new line is inserted after the one on which you typed the I.

Note: The editor does not distinguish between input mode and edit mode. Use the I or TE line commands to insert new lines, either between existing lines or at the end of the data.

- To delete a line, type D over the number to the left and press Enter.
- To save your work and leave the editor, type END on the command line and press Enter.

Ending an Edit Session

Usually, you complete your editing session with the END command and, based on the values in your edit profile, PDF does the following:

- If autosave mode is on and you have made changes to the data:
 - If both number mode and autonum mode are on, the data is renumbered. If not, the numbers remain unchanged.
 - The data is automatically saved. Special temporary lines, such as =PROF>, =MASK>, ==ERR>, ==CHG>, =BNDS>, =TABS>, ==MSG>, =NOTE=, =COLS>, and ===== lines are not part of the data and are not saved. However, you can convert =COLS>, ==MSG>, =NOTE=, and ===== lines to data lines and save them as part of the data set by using the MD (make dataline) line command before entering END.
 - If stats mode is on and the data is a member of an ISPF library or other partitioned data set, the statistics are either generated or updated, depending on whether statistics were previously maintained for the member. If the member is an alias, the alias indicator is turned off.
 - If autolist mode is on, a source listing of the data is recorded in the ISPF list data set for eventual printing.
- If autosave mode is off with the PROMPT operand, a prompting message is displayed. You can issue SAVE to save the data or CANCEL to end the edit session without saving the data.
- If autosave mode is off with the NOPROMPT operand, the data is not saved. The result is the same as that which occurs if you enter a CANCEL command. (You can opt to confirm cancelations by selecting that option from the Primary Edit panel action bar Confirm choice.)
- PDF returns to the previous panel, which is either a member list or the Edit Entry panel. If a member list is displayed, the member you just edited appears at the top of the list.

You can end editing without saving by using CANCEL.

By default, the editor truncates variable-length lines by removing blanks at the end of each line during a save. However, for variable-length lines containing exactly 8 bytes (normally the line number), the editor adds one blank during a save so that

How to Use the ISPF Editor

the line length is not zero after removing the line number. If a variable-length line is completely blank and has no line number, a blank is added so that the line length is not zero.

If you select **Preserve VB record length** on the edit entry panel, or specify PRESERVE on the edit service, the editor stores the original length of each record in variable length data sets and when a record is saved, the original record length is used as the minimum length for the record. The minimum line length can be changed by using the SAVE_LENGTH edit macro command. The editor always includes a blank at the end of a line if the length of the record is zero or eight.

Because VIEW is a special type of edit session, it is important to note that the use of the REPLACE or CREATE commands from within VIEW always honors the setting of the **Preserve VB record length** option on the edit entry panel. This setting can be overridden by using the PRESERVE primary command.

Attention:

CANCEL cancels all changes made since the beginning of the edit session or the last SAVE command, whichever is most recent.

The RETURN command is logically equivalent to the repeated use of the END command. PDF performs the same actions at the end of the edit session.

When a space ABEND such as D37 occurs, ISPF unallocates the data set so that you can swap to another screen or user ID and reallocate the data set. This does not occur for data sets that were edited using the DDNAME parameter of the EDIT service.

Edit Commands

You can use two kinds of commands to control editing operations: line commands and primary commands.

Line Commands

Line commands affect only a single line or block of lines. You enter line commands by typing them in the Line Command field on one or more lines and pressing Enter. The Line Command field is usually represented by a column of 6-digit numbers on the far left side of your display. When you are editing an empty data set or member, however, the Line Command field contains quotes. This field can also be used to define labels and to display flags that indicate special lines, such as the =NOTE= flag, which indicates a note line.

You can use line commands to:

- Insert or delete lines
- Repeat lines
- Rearrange lines or overlay portions of lines
- Simplify text entry and formatting
- Define an input mask
- Shift data
- Include or exclude lines from the display
- Control tabs and boundaries for editing
- Convert some types of special temporary lines to data lines.

You can enter edit line commands as primary commands on the command line by prefixing them with a colon (:) and placing the cursor on the target line. For example, if you enter `:D3` on the command line and move your cursor to line 12 of the file, the three lines 12, 13, and 14 are deleted from the file. This technique is normally used for PF key assignments.

See Chapter 3. Managing Data for ways you can use line commands to manipulate data and Chapter 9. Edit Line Commands for the line command syntax.

Primary Commands

Primary commands affect the entire data set being edited. You enter primary commands by typing them on the Command line (Command ==>), usually located on line 2, and pressing Enter. Any command entered on the edit command line is first intercepted by ISPF. If the command entered is an Edit Primary Command or an Edit Macro, PDF processes the command

You can use primary commands to:

- Control your editing environment
- Find a specific line
- Find and change a character string
- Combine several members into one
- Split a member into two or more members
- Submit data to the job stream
- Save the edited data or cancel without saving
- Sort data
- Delete lines
- Access dialog element models
- Run an edit macro.

You can prefix any primary command with an ampersand to keep the command displayed on the Command line after the command has processed. This technique allows you to repeat similar commands without retyping the command. For example, if you type:

```
Command ==> &CHANGE ALL ABCD 1234
```

the command is displayed after the change has been made, which allows you then to change the operands and issue another CHANGE command. You can recall previous commands with the ISPF RETRIEVE command.

See Chapter 3. Managing Data for some of the ways you can use primary commands to manipulate data and Chapter 10. Edit Primary Commands for the primary command syntax.

Edit Commands and PF Key Processing

In the Edit function there are some differences between the way ISPF processes commands when they are entered from the command line as compared to when they are entered by a combination of the command line and a function (PF) key. In most applications, when you press a PF key, ISPF concatenates the contents of the command line to the definition of the function key. The result is handled as a single command by ISPF or by the application.

When you use a PF key defined as a scroll command (UP, DOWN, LEFT, or RIGHT) the system processes the command as follows:

Edit Commands

- If the concatenation of the scroll command PF key definition and the contents of the command line does not create a valid scroll command:
 - If the word after the scroll command PF key definition begins with a numeric character (0-9), you get a message telling you the scroll amount was not valid.
 - Otherwise, edit processes the contents of the command line as an edit command, then processes the scroll command using the default scroll amount. In this case, the processing of the command line contents as an edit command bypasses the command table, because the command table is used to resolve the scroll key.
- If the concatenation of the scroll command PF key definition and the contents of the command line does create a valid scroll command edit scrolls the screen the specified amount.

If you manually type a scroll command on the command line (you do not use any PF keys) and it has an operand, the operand is checked for validity. However, in the case of a scroll operand that is not valid, the operand is not processed as a separate edit command as it is when used with a PF key.

Edit Macros

Edit macros are primary commands that you write. You can save time and keystrokes by using macros to perform often-repeated tasks. To run a macro, type its name and any operands on the Command line, and press Enter. Your installation may have written and documented common macros for your use. Of course, you can also write your own edit macros.

The rules for running a specific macro, and the expected results, depend on the particular macro. Your installation is responsible for documenting these rules and results. If you want to write your own macros, read Part 2. Edit Macros and Chapter 11. Edit Macro Commands and Assignment Statements.

ISPF enables the installer of the program to specify an edit macro that runs for all users. If a macro name is specified in the ISPF configuration table, then that macro runs before any macros specified in the users' profiles, in programs that invoke edit, or on the edit entry panels.

The site-wide macro can be used to alter existing profiles, enforce site-wide standards, track edit usage, deny edit and view of a data set member, or for any other purposes for which edit macros are designed. Site-wide macros normally end with a return code of 1 (one) in order to place the cursor on the command line. Site-wide macros must be available to each user in the appropriate data set concatenation (SYSPROC, STEPLIB, and so forth) or in Linklist or LPA (program macros only).

A user can also set an application-wide macro if he chooses. See "Application-Wide Macros" on page 30 for more information.

The effect of running a macro depends on the implementation of the macro. Results such as cursor positioning, output messages, and so on, may or may not conform to the results that you expect from built-in edit commands.

Editing Data in Controlled Libraries

When you edit controlled libraries you may use, as previously assigned, either the Library Management Facility (LMF) or the Software Configuration and Library Manager (SCLM).

If LMF is not active on your system and you attempt to start it, the system displays an error panel. Contact your library administrator, database administrator, or system programmer to correct the problem. The editor allows you to access the data, but at your own risk. You are not able to promote changes made to the controlled libraries when LMF is inactive.

For information about editing libraries that are controlled under LMF, refer to *ISPF Library Management Facility*. For information about editing libraries that are controlled under SCLM, refer to *ISPF Software Configuration and Library Manager (SCLM) Developer's and Project Manager's Guide*.

Packing Data

Data can be saved in either packed or standard format. You can control the format by using the PACK primary command to change the edit profile. The editor reads the data in and you can edit it the way you normally would. When you end the editing session, the data is packed and stored. See "PACK—Compress Data" on page 268 and "PACK—Set or Query Pack Mode" on page 374 for more information.

The packed data format has the advantage of saving space. It allows for a more efficient use of DASD by replacing repeating characters with a sequence that shows the repetition.

The disadvantage is that space is saved at the expense of additional processing when the data is read or written. Also, the data cannot be directly accessed by programs. You must access the data through PDF dialogs and library access services. For example, a packed CLIST or REXX EXEC does not run properly because pack mode analysis is not done before passing the CLIST or REXX EXEC to the system.

Note: The library access services referred to in this section apply to LMF. Services for SCLM are described in *ISPF Software Configuration and Library Manager (SCLM) Developer's and Project Manager's Guide*.

Data that is packed by PDF Version 3 Release 3 or later might not be able to be read by releases prior to PDF Version 2 Release 2.

Edit Macros

Chapter 2. Controlling the Edit Environment

This chapter describes the editing environment and how you can customize that environment to best suit your needs.

The PDF component defaults control much of the editing environment. However, you can use line and primary commands to change number and statistical fields on a data display panel and to determine how the data appears.

What is an Edit Profile?

An edit profile controls your edit session through modes and temporary lines. These modes and lines convert data to uppercase (caps mode), automatically renumber lines of data (autonum mode), or specify the left and right boundaries used by other commands (=BNDS> line).

The library type (the last of the data set name qualifiers), record format (fixed or variable), or the record length can implicitly specify an edit profile. You can choose an edit profile in three ways:

- Issue the PROFILE command with a profile name as parameter
- Fill in the **Profile** field on the Edit Entry panel
- Supply a PROFILE keyword and name when calling the EDIT service, such as:
ISPEXEC EDIT PROFILE(name) ...

Using Edit Profile Types

Different kinds of data can have several different edit profiles. With this capability, you could set up an edit profile for COBOL programs, a different edit profile for memos, and a third edit profile for test data. Your installation determines how many different edit profiles are available to you. Typically, 25 edit profiles are available.

If you attempt to create more edit profiles than defined by your installation, the least-used edit profile is deleted first. Locked edit profiles are not deleted unless all your edit profiles are locked. In that case, the least-used locked edit profile is deleted first. Again, if you continue to add edit profiles, all of the unlocked edit profiles are deleted before locked edit profiles.

You can control the use of profiles from the Edit Entry panel. If you leave the **Profile Name** field blank, the profile name defaults to the data set type, which is the last qualifier in the data set name. If you type a profile name, it overrides the data set type qualifier. In either case, if a profile of that name currently exists, it is used. If it does not exist, a new profile is defined. The initial contents of the new profile include the default mode settings, all-blank mask and tabs, and default bounds. To eliminate the profile lines from your panel, use the RESET command.

Displaying or Defining an Edit Profile

You can display none, all, or part of an edit profile by entering the following command:

```
PROFILE [name] [number]
```

Displaying or Defining an Edit Profile

where *name* is the name of the edit profile that you want to display and *number* is a number from 0 to 9. If you omit both operands the editor displays the first five lines of the profile at the top of the data area.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 -----COLUMNS 00001 00072
***** ***** Top of Data *****
=PROF> ...EXEC (FIXED - 80)...RECOVERY ON...NUMBER ON STD.....
=PROF> ...CAPS ON...HEX OFF...NULLS OFF...TABS OFF.....
=PROF> ...AUTOSAVE ON...AUTONUM OFF...AUTOLIST OFF...STATS ON.....
=PROF> ...PROFILE UNLOCK...IMACRO NONE...PACK OFF...NOTE ON.....
=PROF> ...HILITE OFF.....
=TABS> - - - * - - -
=MASK>
=BNDS>
=COLS> -----1-----2-----3-----4-----5-----6-----7-----
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST' THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
Command ===> Scroll ===>
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 8. Edit Profile Display (ISREDDE2)

Note: See “Primary Edit Panel Action Bar Choices” on page 11 for information on the action bar choices on this panel.

The first five lines of the edit profile (Figure 8) are the current mode settings. The remaining lines are the current contents of the =TABS>, =MASK>, and =BNDS> lines, with the =COLS> positioning line. When no operands are entered, the first five lines, which contain the =PROF> flags, are always displayed. However, the =MASK> and =TABS> lines do not appear if they contain all blanks; if the =MASK> and/or =TABS> lines do contain data, they appear, followed by the =COLS> line.

The =BNDS> line does not appear if it contains the default boundary positions. It does appear when the bounds are set to something other than the default, and no 'number' parameter is entered into the PROFILE command.

Note: If enhanced edit coloring is not enabled for the edit session, the profile line displaying HILITE status is not shown. If highlighting is available, and if you explicitly set the language, then the language appears in RED on color terminals.

If you include the name of an existing profile, the editor immediately switches to the specified profile and displays it.

If you include a new profile name, the editor defines a profile using the current modes, options and temporary lines.

The number operand controls the number of lines shown in the profile display. If you type the number 0, the profile is not displayed. If you type a number from 1 through 8, that number of lines of the profile is displayed. If you type the number 9, the complete profile is displayed, even if the =MASK> and =TABS> lines are blank

and the =BNDS> line contains the defaults. Since masks are ignored when using a format name, the "=MASK>" line is not displayed by the profile command in formatted edit sessions.

Modifying an Edit Profile

You modify an edit profile by entering commands to set various modes, options, and temporary lines. Whenever you change an edit profile value, PDF saves the value (unless the edit profile is locked). The next time you edit data using the edit profile, the data is retrieved and the environment is set up again. This is easier than it sounds. First, there are defaults for all the modes, and, in most cases, you do not need to change them. Second, if you decide that you want to change a mode, you just enter the appropriate command. The edit profile is automatically changed and saved for you. See "Edit Modes" for more information about the edit modes.

Locking an Edit Profile

Once you have an edit profile exactly the way you want it, you can lock it. To do this, type PROFILE LOCK and press Enter. The edit profile is saved with all the current modes, options, and temporary lines, and it is marked so that the saved copy of the edit profile is not changed. Usually, each time you begin an editing session the edit profile you start with is exactly the way you locked it. The exceptions are caps, number, stats, and pack, which are made to match the data and are noted with messages. You can change a mode during an editing session, but if the edit profile is locked, the change affects only the current session; it does not affect any later sessions.

If you have locked your current edit profile, you cannot change the initial macro name with IMACRO. For information on IMACRO, see "IMACRO—Specify an Initial Macro" on page 253. For information on the LOCK operand, see "PROFILE—Control and Display Your Profile" on page 270.

Edit Modes

The edit modes control how your edit session operates. To set these modes, use the associated primary commands. For example, if you are editing a COBOL program that is in uppercase and you want all your input to be converted to uppercase, set caps mode on by entering CAPS ON.

The following list summarizes the primary commands you use to display and change your edit profile. See Chapter 10, Edit Primary Commands for a complete description and for the operands you can type with the commands.

PROFILE

Displays the current setting of each mode in this list and controls whether changes to these settings are saved.

AUTOLIST

Controls whether a copy of the saved data is automatically stored in the ISPF list data set.

AUTONUM

Controls whether lines of data are automatically renumbered when the data is saved.

Edit Modes

AUTOSAVE

Controls whether data is saved when you enter END.

CAPS Controls whether alphabetic characters are stored in uppercase when the data is saved.

HEX Controls whether data is displayed in hexadecimal format.

HILITE

Controls the use of enhanced edit color.

IMACRO

Names an edit macro used at the start of the edit session.

NOTES

Controls whether tutorial notes are included in an Edit model.

NULLS

Controls whether blank spaces at the end of a line are written to the panel as blanks or nulls. The difference is that nulls allow you to insert data; blanks do not.

NUMBER

Controls the generation of sequence numbers in a data set.

PACK Controls whether ISPF packs (compresses) the data when it is saved.

RECOVERY

Controls the recovery of an edit session following a system failure.

SETUNDO

Controls the method of saving changes for the UNDO command.

STATS

Controls whether statistics for a data set are generated.

TABS Controls tab settings for aligning data.

Edit Profile Modes

The data you edit controls four special edit profile modes. These modes are set when data is first edited or new data is copied in.

Caps mode

The editor sets caps mode on if it detects that a member to be edited contains no lowercase characters and sets caps mode off if the member does contain lowercase characters.

Number mode

The editor sets number mode on and changes number options if it detects that the data contains valid sequence numbers. It sets number mode off if the data does not contain valid sequence numbers.

Pack mode

The editor sets pack mode on if the data being edited was previously saved in packed format and sets pack mode off if the data was not previously saved in packed format.

Stats mode

The editor sets stats mode on if the member being edited currently has ISPF statistics and sets stats mode off if the member did not previously have ISPF statistics.

The ISPF editor changes the special data modes even if the original edit profile of the member edit profile is locked. However, for locked profiles, it does not save the changes to the profile.

For your convenience, the editor changes the special data modes automatically to correspond to the data. This allows you to have a single data set and to use the default edit profile, even though some members may contain programs (CAPS ON) while other members contain text (CAPS OFF). Some of the members may have statistics to be maintained, while other members are stored without statistics. Some members may be in packed data format, while others are in standard data format. And finally, and perhaps most important, some members may be sequence-numbered, while others are not.

When the editor changes your edit profile to correspond to the data, special message lines appear. If you want to override the change, enter the appropriate command. For example, if the editor changes caps mode from on to off because it finds lowercase characters in the data, you just type CAPS ON and press Enter to reset it.

If you have special requirements, you might not want the editor to change the special modes. You may want to have caps mode on, even if the data contains lowercase data, or you may want to generate statistics on output, regardless of whether the member originally had statistics. If so, you can write an initial macro to specify how the editor is to run these special modes. You would then use IMACRO to associate the initial macro with the edit profile. See “Initial Macros” on page 29 for more information on initial macros.

Edit Mode Defaults

PDF saves several different edit modes in an edit profile. The user can specify the desired edit profile on the Edit Entry Panel. If the **Profile** field is left blank, the data set type is used as the profile name.

To preinitialize a set of edit profiles for first-time users, do the following:

1. Enter PDF.
2. Select the Edit option.
3. Set the edit profile with the defaults you chose.

For example, to set your “COBOL FIXED 80” profile, edit a member of a partitioned data set that has a RECFM of F or FB, a LRECL of 80, and a type qualifier of COBOL (or enter COBOL as the profile name on the Edit Entry Panel).

ISPF provides two methods for setting defaults for new edit profiles. You can set up a profile called ZDEFAULT in the ISPTLIB concatenation, or you can modify the edit profile defaults in the ISPF configuration table. IBM **strongly recommends** using the ISPF configuration table method because it is easier to maintain than the ZDEFAULT method. The ZDEFAULT method can still be used by individual users.

Site-wide Edit Profile Initialization

When no ZDEFAULT profile exists in the ISPTLIB concatenation and the user has no edit profile member in the ISPPROF concatenation, new edit profiles are created based on the settings in the ISPF configuration table. Using the configuration table, you can change any of the defaults for new edit profiles and you can override (force) settings for PACK, RECOVERY, RECOVERY WARN,

Edit Modes

SETUNDO, AUTOSAVE, and IMACRO in existing profiles. When a setting is forced the editor **WILL CHANGE** the users' profiles, so be very careful if you override the IMACRO setting. IBM recommends that you use the site-wide initial macro instead of forcing the initial macro in each user's profile.

It is helpful to understand when the ZDEFAULT profile is used and where it exists in a user's concatenations. The ZDEFAULT profile exists as a row of the edit profile table named xxxEDIT, where xxx is the application profile.

If ZDEFAULT exists in the edit profile table in the ISPTLIB concatenation, and the user has NO edit profile table in the ISPPROF allocation, the ZDEFAULT profile is copied from ISPTLIB into the user's edit profile when the user's edit profile is created. Therefore, many of your existing users might already have a ZDEFAULT profile in their edit profile. Individual users can delete their ZDEFAULT profiles using the PROFILE RESET command from within an edit session. Doing so allows them to use the site-wide configuration for new profiles. You can also use a site-wide edit initial macro to issue a PROFILE RESET for all users. ISPF does not ship any edit profiles.

Note: If you use the force settings such as PACK OFF, edit macro commands that attempt to change forced settings will not receive a failing return code, but the settings will not change.

Creating a ZDEFAULT Edit Profile

Set up a special edit profile named ZDEFAULT (enter ZDEFAULT as the profile name on the Edit Entry Panel). The ZDEFAULT profile is the one used for the initial settings whenever a new edit profile is generated, regardless of the RECFM and LRECL values. For example, if you do not have an ASM profile and you edit an ASM data set, an ASM profile is generated using ZDEFAULT for the initial settings. If no ZDEFAULT profile exists, it is automatically generated with the following settings:

Modes set on:

CAPS STATS NUMBER

Modes set off:

RECOVERY HEX NULLS TABS AUTONUM AUTOLIST PACK

Profile set to:

UNLOCK

IMACRO set to:

None

SETUNDO set to:

STG

HILITE set to:

ON AUTO (CURSOR, FIND, PAREN and LOGIC matching are inactive)

The number of profiles you can establish is described in the configuration table. See "Displaying or Defining an Edit Profile" on page 21 for more details. When you finish, exit PDF. Your entire set of edit profiles is saved in your profile library (referenced by ddname ISPPROF) as the ISREDIT member.

Flagged Lines

Flagged lines are lines that contain highlighted flags in the line command area. These lines can be divided into the following categories:

- Changed lines
- Error lines
- Special lines.

The flags in the line command area are not saved when you end an edit session.

Changed Lines

==CHG> Shows lines that were changed by a CHANGE or RCHANGE command.

Error Lines

==ERR> Shows lines in which PDF finds an error when you enter a line, primary, or macro command. For example, when you enter a CHANGE command, there is not enough room on the line to make the change.

Special Lines

Special lines can be divided into two categories:

- Edit profile lines (the values associated with these lines are stored in your edit profile):
 - =PROF>** Contains the settings of the individual edit modes. This line is not saved as part of your data set or member. See “Edit Modes” on page 23 for more information.
 - =TABS>** Defines tab positions. This line is not saved as part of your data set or member.
 - =MASK>** Can contain data to be inserted into your data set or member when you use the I (insert) line command. This line is not saved as part of your data set or member.
 - =BNDS>** Specifies left and right boundaries that are used by other commands. This line is not saved as part of your data set or member.
 - =COLS>** Identifies the columns in a line.

The column identification line can be saved as part of the data set or member if you use the MD (make dataline) line command to convert it to a data line.
- Message, note, and information lines:
 - ==MSG>** Message lines inform you of changes to the edit profile. These changes are caused by inconsistencies between the data to be edited and the edit profile settings. Message lines also warn you that the UNDO command is not available when edit recovery is off.

You can insert message lines manually by using an edit macro that contains the LINE_AFTER and LINE_BEFORE assignment statements.

Message lines are not saved as part of the data set or member unless you use the MD (make dataline) line command to convert them to data lines.

Flagged Lines

=NOTE= Note lines display information when you insert edit models. However, these lines do not appear if the edit profile is set to NOTE OFF.

You can insert note lines manually by using an edit macro that contains the LINE_AFTER and LINE_BEFORE assignment statements.

Note lines are not saved as part of the data set or member unless you use the MD (make dataline) line command to convert them to data lines.

===== Temporary information lines are lines you can add to provide temporary information that is not saved with the data. They can be inserted into an edit session by using an edit macro containing the LINE_AFTER and LINE_BEFORE assignment statements.

Information lines are not saved as part of the data set or member unless you use the MD (make dataline) line command to convert them to data lines.

Edit Boundaries

Boundary settings control which data in a member or data set is affected by other line, primary, and macro commands. You can change the boundary settings by using either the BOUNDS line command, primary command, or macro command. Table 1 shows commands that work within the column range specified by the current boundary setting:

Table 1. Commands for Use with Boundary Setting Column Range

Line Commands	Primary Commands	Macro Commands	
<	CHANGE	CHANGE	SHIFT <
>	EXCLUDE	EXCLUDE	SHIFT >
(FIND	FIND	SHIFT (
)	LEFT	LEFT	SHIFT)
O	RCHANGE	RCHANGE	SORT
TE	RFIND	RFIND	TENTER
TF	RIGHT	RIGHT	TFLOW
TS	SORT	SEEK	TSPLIT
			USER_STATE

This column range is in effect unless you specify overriding boundaries when entering a command. Refer to the individual command descriptions for the effect the current bounds settings have.

If you do not explicitly set bounds, the editor uses the default bounds. These bounds change as the number mode changes. If you have changed the bounds settings for a data set and would like to revert to the default settings, you can use any BOUNDS command to do so. Table 2 shows the default bounds settings for various types of data sets:

Table 2. Default Bounds Settings for Data Sets

RECFM	Data Set Type	Number Mode	BND S When LRECL=80	BND S Using Other LRECL

FIXED	ASM	ON STD	1, 71	1, LRECL-8
		OFF	1, 71	1, LRECL
	COBOL	OFF	1, 80	1, LRECL
		ON STD	1, 72	1, LRECL-8
		ON COBOL STD	7, 72	7, LRECL-8
		ON COBOL	7, 80	7, LRECL
	OTHER	ON STD	1, 72	1, LRECL-8
		OFF	1, 80	1, LRECL
VARIABLE	ALL	ON STD	9, record length	N/A
		OFF	1, record length	N/A

If the default boundaries are in effect, they are automatically adjusted whenever number mode is turned on or off. If you have changed the bounds from the default settings, they are not affected by the setting of number mode.

If a left or right scroll request would cause the display to be scrolled 'past' a left or right bound, the scrolling stops at the bound. A subsequent request then causes scrolling beyond the bound.

This scrolling feature is especially useful when you are working with data that has sequence numbers in the left hand columns. It allows left and right scrolling up to (but not past) the bounds so that the sequence numbers are normally excluded from the display.

If you specify an invalid value for either the left or right boundary when changing the current boundary settings, the editor resets the value for that boundary to the default. The following constitute invalid boundary values:

- A right boundary value that is greater than the logical record length of a fixed-block file if the file is unnumbered.
- A right boundary value that is greater than the logical record length-8 of a fixed-block file if the file with standard numbers.
- A right boundary value that is greater than the logical record length-4 of a variable-block file.
- A left boundary value that is less than or equal to 8 for a variable-block file with standard numbers
- A left boundary value that is less than or equal to 6 for a file that is numbered with COBOL numbers.

Initial Macros

The editor runs an initial macro after it reads but before it displays data. The macro might initialize empty data sets, define program macros, or initialize function keys.

For example, if you want caps mode on, even if the data contains lowercase data, create an initial macro with a CAPS ON command. The editor first reads the edit profile and the data, then it sets caps mode to correspond to the data. Next, it runs your initial macro, which overrides the edit profile setting of caps mode.

You can specify an initial macro in one of the following ways:

Initial Macros

- Store the macro name in the edit profile with the IMACRO command:

```
Command====> IMACRO INITMAC
```

See “IMACRO—Specify an Initial Macro” on page 253 for more information on the IMACRO command.

- Specify the initial macro name on the Edit Entry panel:

```
INITIAL MACRO ====> initmac
```

- Specify the initial macro name on the EDIT service call:

```
ISPEXEC EDIT DATASET(dsname) MACRO(initmac) ...
```

Once specified, the initial macro runs at the beginning of each edit session that uses the profile. It may be overridden by an initial macro typed in the **INITIAL MACRO** field on the Edit Entry panel or specified on the EDIT service call. You can type **NONE** in the **INITIAL MACRO** field to suppress the initial macro defined in the profile.

If the current profile is locked, the IMACRO command cannot be run.

Remember that commands referencing display values (DISPLAY_COLS, DISPLAY_LINES, DOWN, LEFT, RIGHT, UP, LOCATE) are invalid in an initial macro because no data has been displayed.

If the initial macro issues either an END or CANCEL command, the member is not displayed.

Application-Wide Macros

You can specify a macro to run at the beginning of your edit sessions by placing a variable called **ZUSERMAC** in either the shared or profile pool. **ZUSERMAC** must contain the name of the macro and cannot include any operands. **ZUSERMAC** must not be longer than 8 characters long.

If **ZUSERMAC** exists in the profile or shared pool, the macro it specifies is run after the site-wide initial macro, and before the initial macro specified on the edit panel, on EDIT service command, or in the edit profile.

If you want to remove the user application-wide macro, you can issue the VERASE service to remove **ZUSERMAC** from the shared or profile pool.

Statistics for PDS Members

If stats mode is on, PDF creates and maintains statistics for partitioned data set members. The following sections explain the effect stats mode has on your statistics, first when you are beginning an edit session and then when you are saving data.

Note: Stats mode is ignored for sequential data sets.

Included in the statistics are version and modification levels. These numbers can be useful in controlling library members. See “Sequence Number Format and Modification Level” on page 32 for a discussion of how the generation of statistics affects the format of sequence numbers.

Effect of Stats Mode When Beginning an Edit Session

Whenever a member is retrieved for editing, the ISPF editor checks the setting of stats mode. PDF does not display any warning messages if the stats mode and the member are consistent. For example:

- If the stats mode is on and the member has statistics
- If the stats mode is off and the member does not have statistics.

If the stats mode and the member are not consistent, however, PDF displays a warning message. For example:

- If stats mode is on and the member has no statistics, PDF displays a warning message, but does not change the stats mode.
- If stats mode is off and the member has statistics, PDF automatically turns on stats mode and displays a message indicating the mode change.

Effect of Stats Mode When Saving Data

If stats mode is on when you save the member, PDF updates the statistics, or creates statistics if the member did not previously have them.

If stats mode is off when you save the member, PDF does not store any statistics; any previous statistics are destroyed.

Stats mode is saved in the edit profile.

Version and Modification Level Numbers

Two of the statistics that the editor creates and maintains for members of ISPF libraries and partitioned data sets (when stats mode is on) are the version and modification level numbers. These numbers are displayed in the form VV.MM at the top of the edit panel following the data set name.

When the editor creates statistics for a new member, the default version and modification level numbers are 01 and 00, respectively. Otherwise, the values are taken from the previous statistics stored with the member.

You can change the version number with the VERSION command.

The modification level number appears in the last 2 digits of the line numbers for new or changed lines to provide a record of activity. The number is automatically incremented by one when the first change is made to the data. It can also be changed explicitly with the LEVEL command. The numbers for both can range from 00 to 99, inclusive. After the modification level number reaches 99, it does *not* increment by one to return to level 00.

The editor normally increments the modification level the first time that data is changed. This incrementing is suppressed if:

- You have set the modification level with a LEVEL command before making the first change.
- Statistics did not previously exist, and the editor has set the modification level to 0 for a new member.

If both stats mode and standard sequence number mode are on, the current modification level replaces the last two positions of the sequence number for any

Version and Modification Level Numbers

lines that are changed. At the time the data is saved, it is also stored for any lines that already are marked with a modification level higher than the current modification level. If you type LEVEL 0, press Enter, and then save the data, all lines are reset to level 0. See “LEVEL—Specify the Modification Level Number” on page 254 for more information.

Sequence Numbers

Each line on the panel represents one data record. You can generate and control the numbering of lines in your data with the following commands:

AUTONUM

Automatically renumbers data whenever it is saved, preserving the modification level record.

NUMBER

Turns number mode on or off, and selects the format.

RENUM

Rennumbers all lines, preserving the modification level number.

UNNUMBER

Turns off numbering and blanks the sequence number fields on all lines. This deletes all modification level records.

Sequence Number Format and Modification Level

Sequence numbers can be generated in the standard sequence field, the COBOL sequence field, or both:

- The *standard sequence field* is the last 8 characters for fixed-length records, or the first 8 characters for variable-length records, regardless of the programming language. Use NUMBER ON STD to generate sequence numbers in the standard sequence field.

For members of partitioned data sets, the format of standard sequence numbers depends on whether statistics are being generated. If statistics are being generated, standard sequence numbers are 6 digits followed by a 2-digit modification level number. The level number flag reflects the modification level of the member when the line was created or last changed. If, for example, a sequence number field contains 00040002, the line was added or last changed at modification level 02. The sequence number is 000400.

If stats mode is off, or if you are editing a sequential data set, standard sequence numbers are 8 digits, right-justified within the field.

Sequence Numbers

- The *COBOL sequence field* is always the first 6 characters of the data and is valid only for fixed-length records. Use the NUMBER ON COBOL or NUMBER ON STD COBOL to generate COBOL sequence numbers. **Attention:**

If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. Or, you can use CANCEL at any time to end the edit session without saving the data. COBOL sequence numbers are always 6 digits and are unaffected by the setting of stats mode.

Sequence numbers usually start at 100 and are incremented by 100. When lines are inserted, the tens or units positions are used. If necessary, one or more succeeding lines are automatically renumbered to keep the sequence numbers in order.

Sequence Number Display

For numbered data, the Line Command field displayed to the left of each line duplicates the sequence number in the data. Normally, the editor automatically scrolls left or right to avoid showing the data columns that contain the sequence numbers. However, you can explicitly scroll left or right to display the sequence numbers. The DISPLAY operand of the NUMBER and RENUMBER commands also causes the editor to display the sequence numbers.

For example, assume that the data has COBOL numbers in columns 1 through 6 and the number mode is NUMBER ON COBOL. When the data is displayed, column 7 is the first column displayed. If you change number mode to NUMBER OFF, the data is scrolled so that column 1 is the first column displayed. If you then change number mode to NUMBER ON, the data is scrolled back to column 7. But if you change number mode to NUMBER ON DISPLAY, the sequence numbers in columns 1 through 6 remain displayed. The sequence numbers in columns 1 through 6 become part of the data window, but cannot be modified.

Initialization of Number Mode

When you retrieve data for editing, the editor determines whether it contains sequence numbers. The editor always examines the standard sequence field. It examines the COBOL sequence field if the data set type (the lowest level qualifier in the data set name) is COBOL.

If all lines contain numeric characters in either the standard or COBOL sequence field positions, or both, and if the numbers are in ascending order, the editor assumes the data is numbered and turns on number mode. Otherwise, the editor turns off number mode.

If the first setting of the number mode differs from the setting in the edit profile, a message indicating that the editor has changed the mode is displayed. For new members or empty sequential data sets, the first setting of number mode is determined by the current edit profile. For a new edit profile, the default is NUMBER ON for standard sequence fields, and NUMBER ON COBOL if the data set type is COBOL.

Enhanced and Language-sensitive Edit Coloring

The editor provides language-sensitive coloring as a productivity aid for users who are editing program source. It is used in a variety of programming languages. Some coloring enhancements are also useful for editing data other than program source.

Note: Language-sensitive and enhanced coloring of the edit session is only available when enabled by the installer or the person who maintains the ISPF product. For information on enabling the enhanced color functions, see *ISPF Planning and Customizing*

These enhancements allow programmers to immediately see simple programming errors, such as mismatched quotes or parentheses, unclosed comments, and mismatched logical constructs. The language-sensitive component allows you to take advantage of the editor's coloring capabilities for a number of programming languages simultaneously. Enhanced coloring is also a general productivity aid, because it improves your ability to locate text quickly.

The editor provides enhanced highlighting in the following areas:

1. Programming language constructs, including the following:
 - Keywords for each individual language
 - Comments
 - Quoted strings (using both single and double quotes)
 - Compiler directives (C, COBOL, PL/I, and PASCAL only)
 - Special characters that the user chooses.
2. Language-sensitive program logic features, such as logical blocks and IF/ELSE logic.
3. Any strings that match the previous FIND operation or that would be found by an RFIND or RCHANGE request.
4. Default color for the data area in non-program files.
5. The phrase containing the cursor in the data area.
6. Characters that have been input since the previous Enter or function key entry was pressed.

Note: Highlighting is *not* available for edit sessions that involve the following:

- Only CURSOR and FIND highlighting is valid for data sets with record lengths greater than 255
- Mixed mode edit sessions (normally used when editing DBCS data)
- Formatted data.

Language Support

The following languages are supported for language-sensitive coloring:

- Assembler
- BookMaster
- C
- COBOL
- ISPF Dialog Tag Language (DTL)
- ISPF Panels (non-DTL)
- ISPF Skeletons
- JCL (Job Control Language)
- Pascal

- REXX
- PL/I
- OTHER, which includes languages that use constructs similar to PL/I, such as DO, BEGIN, END, SELECT, and so forth. Limited support for CLIST is provided with the OTHER language. OTHER does not support any compiler directives.

Automatic Language Selection

If you choose not to set the language explicitly, the editor can *automatically* determine the language of the part being edited. The language is determined by looking at the first non-blank string in the file. In cases where ambiguity exists between languages, as in the case C and JCL (both may start with //) or PL/I and REXX (both may start with a /* comment), the last qualifier of the data set name may be used to determine the language. Rules for automatic language recognition are as follows:

Assembler

Asterisk in column 1 or a recognized opcode of CSECT, DSECT, MACRO, TITLE, START or COPY.

Note: *PROCESS in column 1 is recognized as PL/I.

BookMaster

First character is . or : in column 1.

C Any of the following:

- First string is #
- First string is // and data set type is not .CNTL, .JCL, or ISPCTLx
- First string is /* and data set type is .C.

COBOL

First non-blank is a * or / in column 7.

ISPF DTL

First non-blank character is <.

ISPF Panel

First string is) in column 1, followed by a panel section name, or the first string is % in column 1.

ISPF Skeleton

) in column 1 in a file that does not seem to be a panel.

JCL Any of the following:

- //anything followed by the word COMMAND, DD, ELSE, ELSEIF, EXEC, IF, INCLUDE, JCLLIB, JOB, OUTPUT, PROC, SET, XMIT, or any word beginning with the characters 'MSG'
- /* in column 1
- // in column 1, and the data set type is .CNTL, .JCL, or ISPCTLx
- Any of the following in column 1:

```
*$
/*JOBPARM
/*MESSAGE
/*NETACCT
/*NOTIFY
/*OUTPUT
/*PRIORITY
```

Enhanced Edit Coloring

```
/*ROUTE  
/*SETUP  
/*SIGNOFF  
/*SIGNON  
/*XEQ  
/*XMIT
```

PASCAL

First string is (*, or the first string is /* and the data set name ends in .PASCAL.

PL/I First string is % or /* or the first string is *PROCESS in column 1. The use of carriage control characters in column one may cause PL/I detection to fail. For data sets names with a final qualifier starting with "PL", automatic language detection is retried ignoring column one if the first non-blank characters occur in column one, and no language can be detected. See REXX, C, and Panel for more information.

REXX First string is a /* comment containing REXX, or the first string is a /* comment, and the data set type is .EXEC or .REXX.

Other First word is PROC, CONTROL, ISPEXEC, or ISREDIT.

HILITE AUTO selects a language based on the first non-blank line, and in some cases, the last qualifier of the data set name.

The PDF component only scans a maximum of 72 bytes of data per line to determine the language. If the data which would determine the language is past the 72nd column, the PDF component may incorrectly determine the language.

Language Processing Limitations and Idiosyncracies

Because the PDF component does not provide true parsing, the built-in language scanner does not operate as a syntax checker. Keywords or built-in function names that are used as variables, and therefore not used in a language context, *will* be highlighted as keywords. For example, in context sensitive languages, such as PL/I, the word 'ELSE' may be used as a variable name. PDF treats 'ELSE' as a keyword in all cases, both for highlighting and logic determination.

In addition, the varying implementations and release schedules of the supported languages may result in keyword highlighting that does not reflect the latest version of the language.

Note: Nested comments are not recognized in any language. When sequence numbers are in use, the editor only highlights the editable data. The sequence numbers are shown in the overtyping color.

Also, because the language scanners of edit highlighting do not provide true parsing, when an unmatched end tag is encountered and the LOGIC option is enabled, subsequent end tags might be highlighted as unmatched, even if they appear to be properly matched.

Recognized Special Symbols: Special characters can be highlighted for each specific language. The characters are only highlighted if they are not part of another class of constructs such as a comment, a string, or a compiler directive. The default set of characters for each language follows:

Assembler

```
-+*/=<>&^|;
```

```

BookMaster      &.,!?$
C               -+*/=<>&~|:;!%?#[] \
COBOL          .
DTL            <>()=
Panel         &
Skel          &
JCL           (),|<>~&=
Pascal       -+*/=<>&~|:[]
PL/I         -+*/=<>&~|:
REXX        -+*/=<>&~|:%\
Other       -+*/=<>&~|:

```

These character sets may be changed by each user using the HILITE dialog.

Assembler: Highlighting is performed only in columns 1 through 72.

Specific keywords are not highlighted. Any word where an opcode would be expected is highlighted as a keyword.

BookMaster: Only BookMaster tags that begin with a colon (:) are highlighted. All tags should be terminated by a period, because ISPF highlights up to the next period. Dot control words (.xx) are never highlighted.

The keyword list supplied by the ISPF comprises the tags used to do logic matching (:xxx/:exxx). Tags that have an optional end tag must have a matching end tag in the edited data for logical highlighting to work. The LOGIC option highlights unmatched end tags (:exxx tags which do not have a corresponding :xxx tag) in reverse video pink.

BookMaster tags are not checked for validity. If you specify a colon (:) as a special character to highlight, the editor does not recognize BookMaster tags.

C: C++ comments (//) are recognized.

Logical highlighting highlights curly braces ({ and }).

Keywords are case-sensitive in C. Only the lower case versions of keywords are highlighted.

COBOL: Highlighting is performed only in columns 7 through 72.

Both single quotes (') and double quotes (") are treated as unique open and close quote characters, although some COBOL languages only specifies double quotes as string delimiters. Compiler directives (also called compiler-directing statements) are supported for IBM SAA AD/Cycle COBOL/370 Version 1.1.

DTL: Only items in tags are highlighted. Any less than sign (<) is assumed to start a tag. This may cause highlighting errors if the '<' symbol appears outside of a DTL tag.

Panels and Skeletons:

Enhanced Edit Coloring

Quoted strings are terminated at the end of a line. For the most part, the PDF component does not parse panels or skeletons. Usually any data on a line that starts with a ')' in column 1 is highlighted as a keyword.

JCL: Because automatic language determination recognizes C++ comments (//), JCL is recognized only if any of the following conditions is met:

- The last qualifier of the data set name is JCL, CNTL, or PROCLIB or ISPCTLx (where x is any character)
- The 2nd non-blank 'word' of the 1st non-blank line is DD, JOB, EXEC, or PROC
- The 2nd non-blank 'word' of the 1st non-blank line starts with 'MSG'. This is for JCL with no JOB card, but with MSGLEVEL or MSGCLASS.
- The first three characters in the first non-blank line are //*.

Conditional JCL logic (IF/ELSE) is highlighted, but is not supported by the LOGIC option.

PL/I: For fixed length record format data sets, column 1 is not scanned after the first non-blank line, except to search for *PROCESS statements.

REXX: Logic highlighting does not support a terminating semicolon in the IF expression, or a semicolon before the THEN or ELSE instructions.

In addition, IF statements which have the THEN keyword on the following line but do not have a continuation character at the end of the IF expression will cause highlighting errors.

For example, although the following statements are valid in REXX, the ELSEs will be highlighted as a mismatched ELSEs.

```
IF a=b; THEN say 'ok'; ELSE; say 'Not OK';
IF a=b
  THEN say 'ok';
  ELSE say 'Not OK';
```

Other: When OTHER is in effect, ISPF tries to determine if the program is a CLIST by checking for a first word of PROC, CONTROL, ISPEXEC or ISREDIT. If ISPF determines that the data being edited is a CLIST, then CLIST comment closure and continuation rules apply.

The HILITE Command/Dialog

PDF supports enhanced and language-sensitive coloring in edit through a new edit primary and macro command called HILITE. However, the basic functions of HILITE cannot be accessed through a dialog that utilizes the GUI interface.

HILITE Operands

ON Sets program coloring ON and turns LOGIC off.

OFF Sets coloring OFF, with the exception of cursor highlighting.

LOGIC

Turns on both IF and DO logic matching. When logic matching is active, only comments are specially colored. All other code, other than logic keywords, is shown in the default color.

IFLOGIC

Turns on IF/ELSE logic matching.

DOLOGIC

Turns on DO/END logic matching.

NOLOGIC

Same as ON.

AUTO

Allows PDF to determine the language.

DEFAULT

Highlights the data in a single color.

OTHER

Highlight the data as a pseudo-PL/I language. Limited CLIST support is also provided by OTHER.

ASM Highlights the data as Assembler.

BOOK

Highlights the data as BookMaster.

C Highlights the data as C.

COBOL

Highlights the data as COBOL

DTL Highlights the data as Dialog Tag Language.

JCL Highlights the data as MVS Job Control Language.

PANEL

Highlights the data as ISPF Panel Language.

PASCAL

Highlights the data as Pascal.

PLI Highlights the data as PL/I.

REXX Highlights the data as REXX.

SKEL Highlights the data as ISPF Skeleton Language.

RESET

Resets defaults (AUTO, ON, Find and Cursor on).

CURSOR

Toggles highlighting of the phrase that contains the cursor.

FIND Toggles highlighting FIND strings.

PAREN

Turns on parenthesis matching. When parenthesis matching is active, only comments are specially colored. All other code is displayed in the default color. Note that extra parenthesis highlighting is always active when highlighting is active.

SEARCH

Finds the first unmatched END, ELSE, or). For C language programs this command also finds the first unmatched }. The search for mismatches only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file before issuing the HI SEARCH command.

Note: The logic setting affects the search results. For example, if DOLOGIC is on, only mismatched ENDS are found. If IFLOGIC is on, only mismatched ELSEs are found.

Enhanced Edit Coloring

DISABLED

Turns off all HILITE features and removes all action bars. This benefits performance at the expense of function. Since DISABLED status is not stored in the edit profile, you need to reenter this operand each time you enter the editor.

The HILITE Dialog

The HILITE command with no operands displays a dialog that enables you to do the following:

- Specify a specific language to be used for coloring or enable automatic language detection.
- Assign colors for different language elements on a language-by-language basis or for all languages at once.
- Enable or disable logic or parenthesis matching.
- Turn FIND coloring on or off and assign the color for FIND highlighting.
- Turn cursor coloring on or off and assign the color for cursor phrase highlighting.
- Specify special symbols to be highlighted on a language-by-language basis.
- View keyword lists for each language.

Note: Keyword lists and default highlighted symbols for each language are supplied by IBM. A facility that involves assembly and link editing of an installation-modified keyword or symbol list does exist to add or remove keywords. However, IBM does not supply facilities for adding additional languages. The keyword and symbol lists, and directions for changing them are in member ISRPXASM in the IBM-supplied ISPF sample library.

The functions of the HILITE dialog are provided by the your selection of pull-down choices from action bars. Selection of pull-down choices results in pop-up windows that enable you to supply the desired coloring information and gain access to additional pull-down choices.

The HILITE panels are accompanied by descriptions of the available pull-down choices:

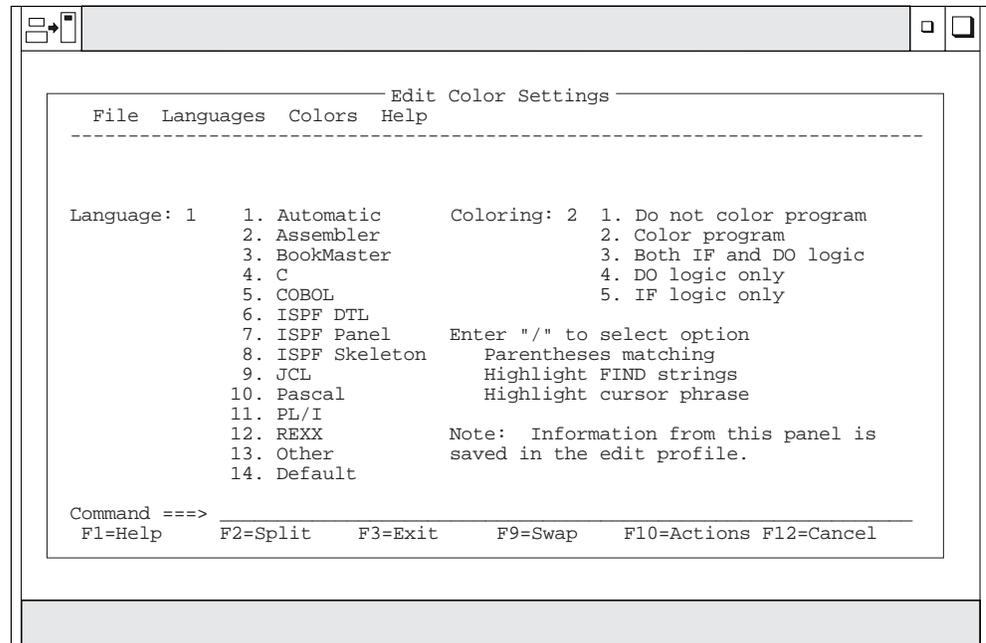


Figure 9. HILITE Initial Screen (ISREP1)

You can reach this panel by issuing HILITE from an edit panel, or by selecting **Hilite...** from the Edit pull-down.

HILITE Initial Panel Action Bar: The action bar choices on the HILITE Initial panel are:

File

Restart application

Resets all settings on all panels back to the point that HILITE was invoked.

Default All Settings

Resets all settings on this panel back to the point that HILITE was invoked.

Save and Exit

Saves changes and exits application.

Cancel

Ends application and discards changes.

The LANGUAGES pulldown allows you to change the way that specific supported languages are highlighted, including the symbols which are highlighted and the colors that are used for the various language elements.

Note: ALL changes the colors for all of the languages at once.

Languages

All (changes all languages)...

Assembler...

BookMaster...

C...

COBOL...

IDL...

ISPF DTL...

ISPF Panel...

ISPF Skeleton...

Enhanced Edit Coloring

JCL...

Pascal...

PL/1...

REXX...

Other...

See "Language Support" on page 34 for a description of the Other... choice.

Default...

Used when AUTO is specified, but no language can be determined.

Colors

Overtyping Color...

Changes the color used for typed data. See Figure 10.

Find String Color...

Changes the color used to find strings. See Figure 11.

Cursor Phrase Color...

Changes the color of the phrase which contains the color. See Figure 12.

Note: On a PC, the terminal emulator can affect the color. Some terminals do not support features such as "blink"; if this is selected with a color, another color might display.

Help Immediately enters help panels, which offers these choices:

- Overview
- HILITE command
- Supported Languages
- Automatic Language Determination
- Additional Functions
- Supported Comment Types
- FIND and CURSOR highlighting
- Logic Highlighting
- C and IDL Language Notes
- Assembler Notes
- PL/I Notes
- BookMaster Notes
- Panel Notes
- Skeleton Notes
- Miscellaneous Notes.

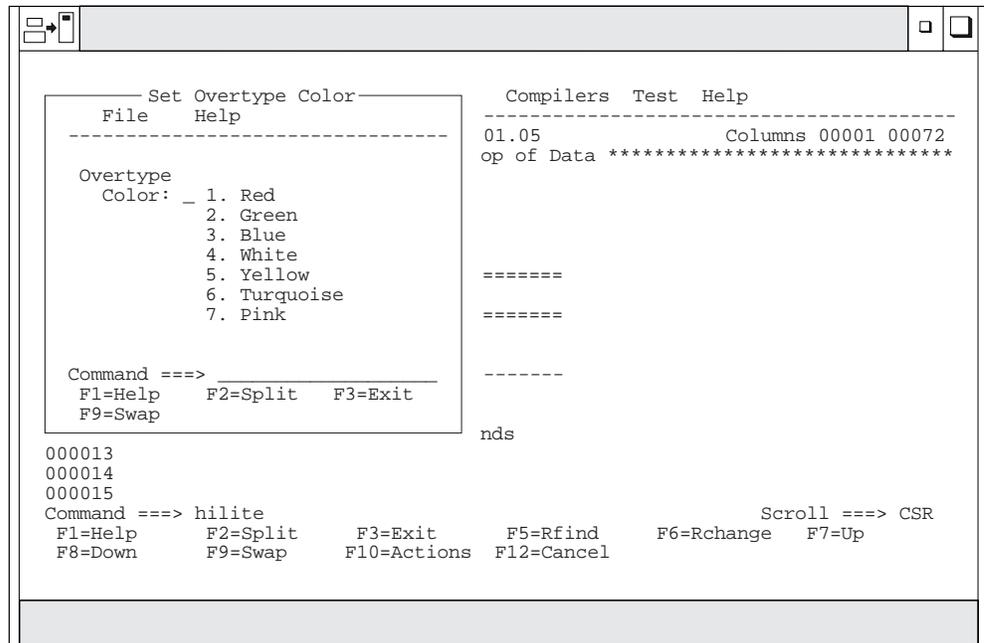


Figure 10. Set Overtyping Color panel (ISREP2)

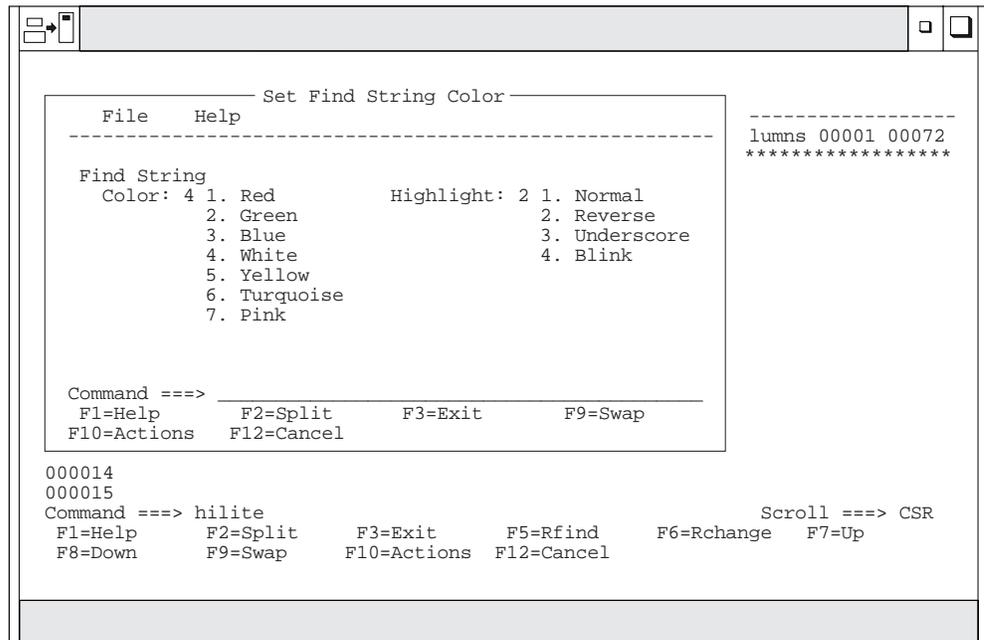


Figure 11. Set Find String Color panel (ISREP3)

Enhanced Edit Coloring

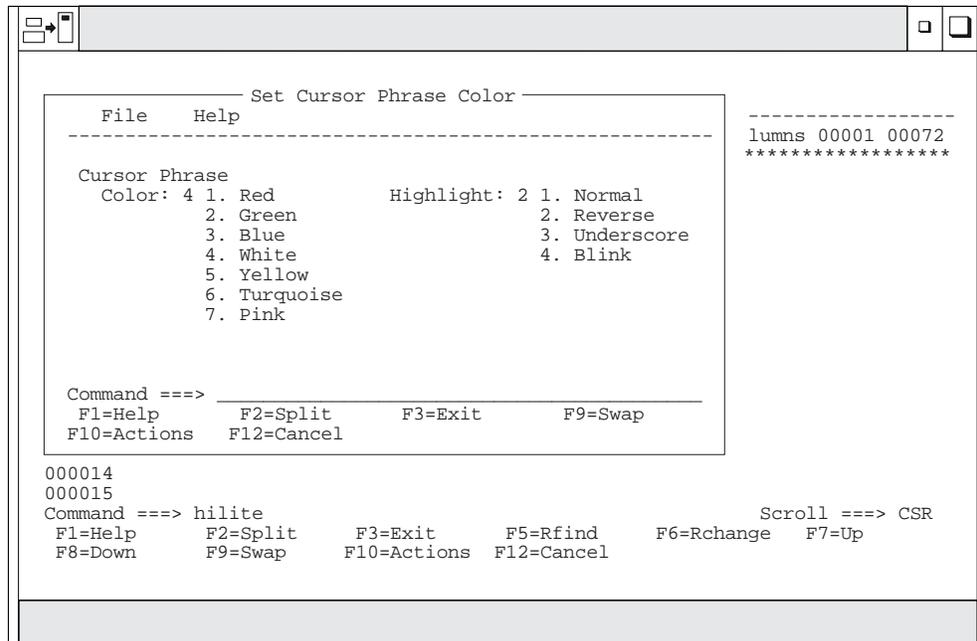


Figure 12. Set Cursor Phrase Color panel (ISREP4)

Set Overtyp, Find String, Cursor Phrase Color Action Bars: These action bar choices function as follows:

- File** The File pull-down offers these choices:
 - Reset** Resets the settings on this panel to the values they had when the panel first appeared.
 - Default** Sets the values to the IBM-supplied defaults.
 - Save and Exit** Exits this panel. Changes will be saved when the HILITE dialog completes, unless Cancel is specified.
 - Cancel** Exits this panel and discards changes.
- Help** Immediately enters help panels for the HILITE command and dialog.

After selecting a specific language from the Languages pull-down on the HILITE Initial panel (Figure 9 on page 41), Figure 13 appears:

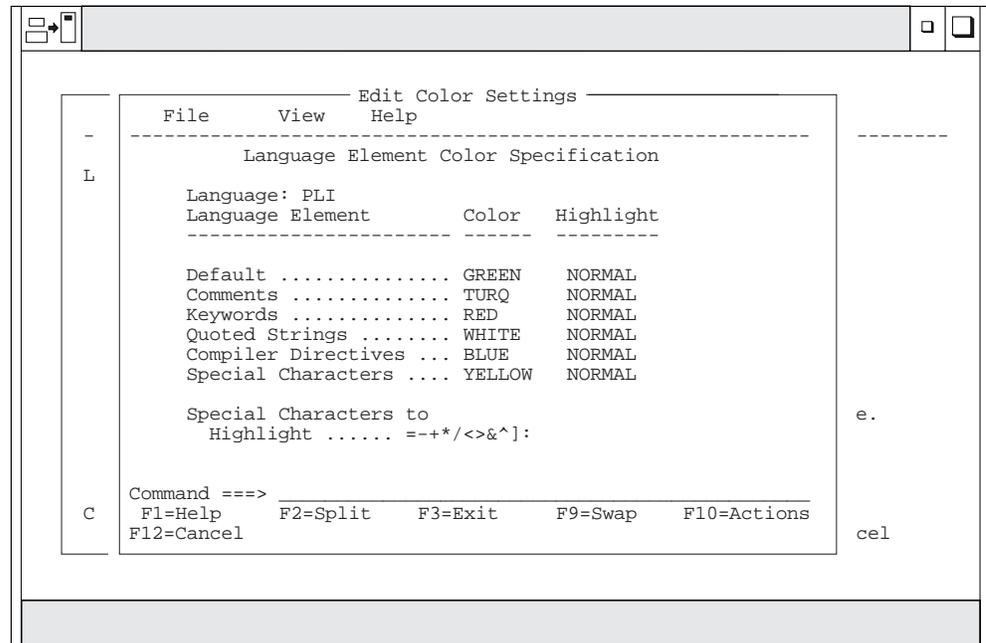


Figure 13. HILITE Specific Language Screens (ISREPC)

If the JCL language is selected, the Compiler Directives field is replaced by a DD * and Data Lines field in the pop-up window.

When a new color is typed in, the input field is shown in that color when you press Enter.

Note: If a field is not applicable to a language, the field is supplied with a *n/a*.

Edit Color Settings Action Bar: The Edit Color Settings action bar choices function as follows:

File The File pull-down offers these choices:

Restart 'language'

Resets colors and symbols to the settings they had upon entry to this panel.

Defaults

Resets colors and symbols to default values.

Save and Exit

Exits this panel. Changes will be saved when the HILITE dialog completes, unless Cancel is specified.

Cancel

Exits this panel and discards changes.

View The View pull-down choice is:

View Keywords

Displays a list of keywords for a particular language. See Figure 14 for an example of a Language Keyword list.

Help Immediately enters help panels.

If no keywords exist for a given language choice, a message is displayed instead of a Language Keyword list.

Enhanced Edit Coloring

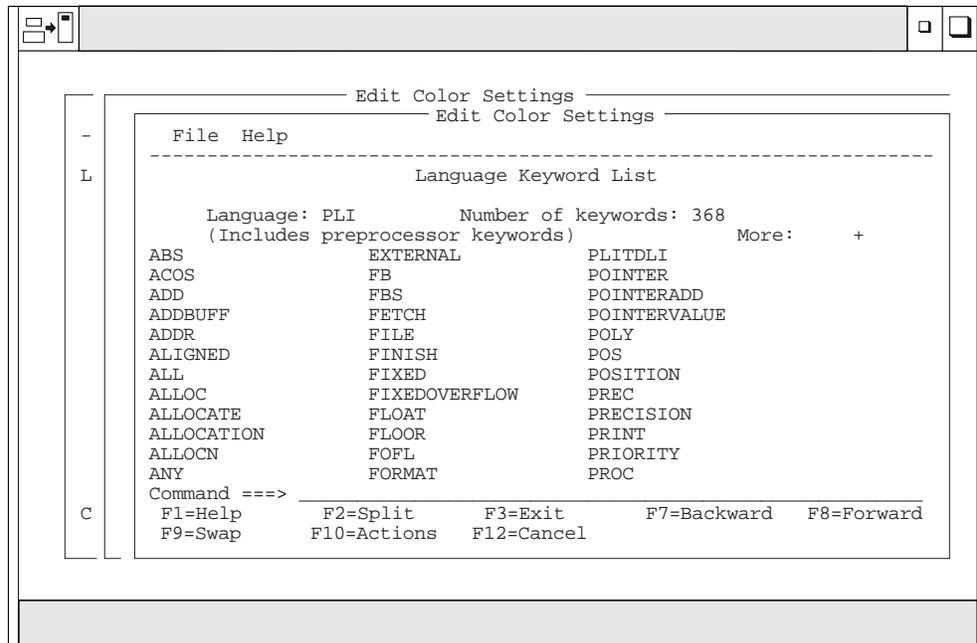


Figure 14. HILITE Language Keyword List (ISREPK)

Language Keyword List Action Bar: The Language Keyword List action bar choices function as follows:

File The File pull-down choice is:

Cancel

Exit this panel. (No changes are possible on this panel.)

Help Immediately enters help panels.

Highlighting Status and the Edit Profile

Colors are assigned to each character in the data area when the data appears. As you type in characters, they appear in the 'overtyping' color. When the Enter key or a F key is pressed, the file is scanned again and the new characters are displayed in the appropriate colors for the type of data being edited. The actual color definitions and symbol sets for each language affect the entire ISPF session. However, only the language, coloring type (ON/OFF status), and logic type are saved in the edit profile.

A new edit profile line, as shown in Figure 15, has been added which shows the status of edit highlighting. If edit highlighting is not available, the profile line is not shown. If highlighting is available, and you explicitly set the language, then the language appears in RED on color terminals.

```
....HILITE PLI LOGIC PAREN CURSOR FIND.....
or
....HILITE PLI PAREN FIND.....
or
....HILITE OFF.....
```

Figure 15. Edit Profile Lines with HILITE

The information shown on the PROFILE command is saved as part of the edit profile.

Edit Recovery

Edit recovery is the PDF component's method of helping you recover data that could otherwise be lost. For example, you would use edit recovery to re-establish the edit session at the point of failure after a power outage or system failure.

You can turn on edit recovery mode by doing either of the following:

- Entering the RECOVERY primary command:
Command ==> RECOVERY ON
- Running an edit macro that contains the RECOVERY macro command:
ISREDIT RECOVERY ON

If recovery mode is on when a system crash occurs, automatic recovery takes place the next time you attempt to use edit. Recovery mode is remembered in your edit profile.

Note: Turning recovery mode on causes the data to be written to a temporary backup file. This is independent of whether changes have been made to the data.

When you begin an edit session, if there is data to recover, the the Edit Recovery panel appears, shown in Figure 16.

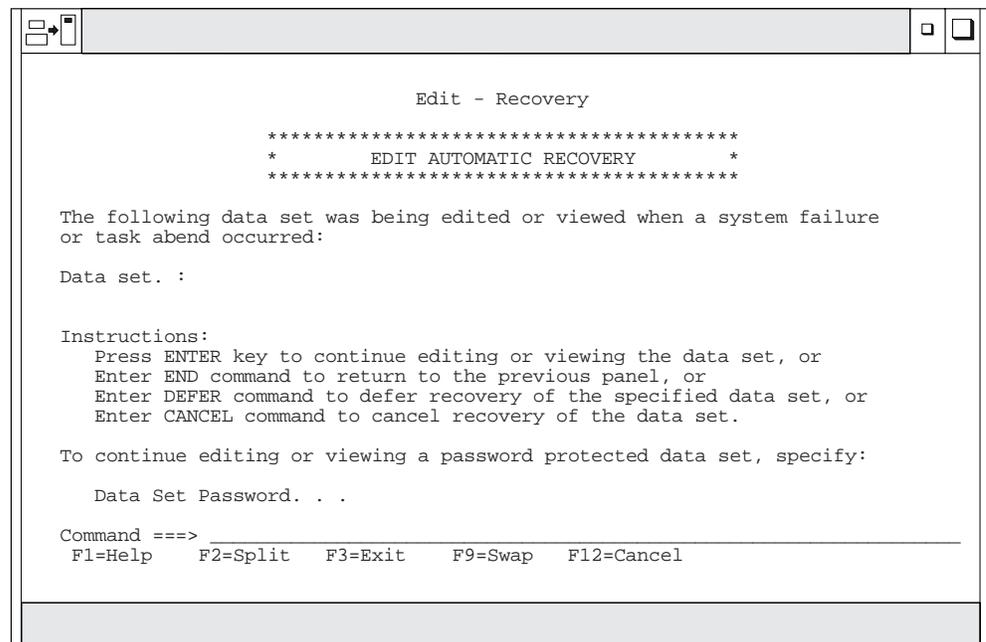


Figure 16. Edit Recovery Panel (ISREDM02)

Note: Refer to *ISPF User's Guide* for information about the **Data Set Password** field.

Edit Recovery

If you continue with, defer, or cancel recovery and you have other data to be recovered, the Edit Recovery panel is displayed again for the next data set. You can control the number of data sets to be recovered with the edit recovery table, a system data set that contains entries for each level of nested editing sessions that can be recovered. For information on changing edit recovery operands, refer to *ISPF Planning and Customizing*

Note: You cannot recursively edit data while you are in an edit session which is the result of an edit recovery.

Attention:

If the data set to be recovered was edited by another user before you continue with edit recovery, the changes made by the other user are lost if you save the data.

If you press Enter to continue editing the data set, the editor runs a recovery macro if you had previously specified one by using the RMACRO primary or macro command. See "Recovery Macros" on page 118 and the descriptions of the RMACRO primary and macro commands for more information.

In spite of edit recovery's benefit in recovering data, there are times when you might not want to use it. You might want to turn edit recovery off in the following situations:

- Operating with recovery mode off eliminates the I/O operations that maintain the recovery data and can therefore result in improved response time.
- Besides recording actual data changes, recovery mode records temporary changes, such as excluding lines and defining labels. These temporary changes are recorded to allow UNDO to undo other edit interactions besides those that change data. Therefore, when edit recovery is on, the recording of both data and temporary changes affects the amount of DASD space that is used.

You can turn off edit recovery mode by doing either of the following:

- Entering the RECOVERY primary command:
Command ==> RECOVERY OFF
- Running an edit macro that contains the RECOVERY macro command:
ISREDIT RECOVERY OFF

See Chapter 10. Edit Primary Commands for details on using RECOVERY.

Chapter 3. Managing Data

This chapter gets you started using some of the basic line and primary commands to manipulate data.

The basic functions of the ISPF editor are similar to those of a word processor. You can create, copy, move, search, and replace data, as well as perform several other word processing functions by using the line and primary commands described in this chapter.

Creating and Replacing Data

Use the CREATE and REPLACE primary commands to specify a member to be written from the data being edited. CREATE adds a new member to a partitioned data set or a new sequential data set. REPLACE rewrites a member or sequential data set. The process of creating and replacing data is very similar. However, remember that when you replace data, the original data is deleted and replaced with the new data.

There are two ways you can use CREATE or REPLACE:

1. You can type either CREATE or REPLACE on the Command line, followed by the name of a member or the name of a data set and member, to be created or replaced. You can add line labels that show the lines to be copied. If you omit the labels, you can use the C (copy) or M (move) line commands to specify which lines are to be copied or moved. Then press Enter. See “CREATE—Create Data” on page 231 and “REPLACE—Replace Data” on page 277 for the complete syntax of the commands.
2. If you omit the member name or data set name and member, and just type CREATE or REPLACE and specify the lines to be used to create or replace the member, the editor displays a panel requesting the name of the member or data set you want created or replaced.

If you try to create or replace data that has inconsistent attributes, the Edit - Confirm Create Edit - Confirm Replace panel that warns you of the inconsistency and gives you an opportunity to cancel the create and replace commands. Figure 17 shows an Edit - Confirm Replace panel that was displayed for a user who tried to replace a sequential data set with a member of a partitioned data set.

Copying and Moving Data

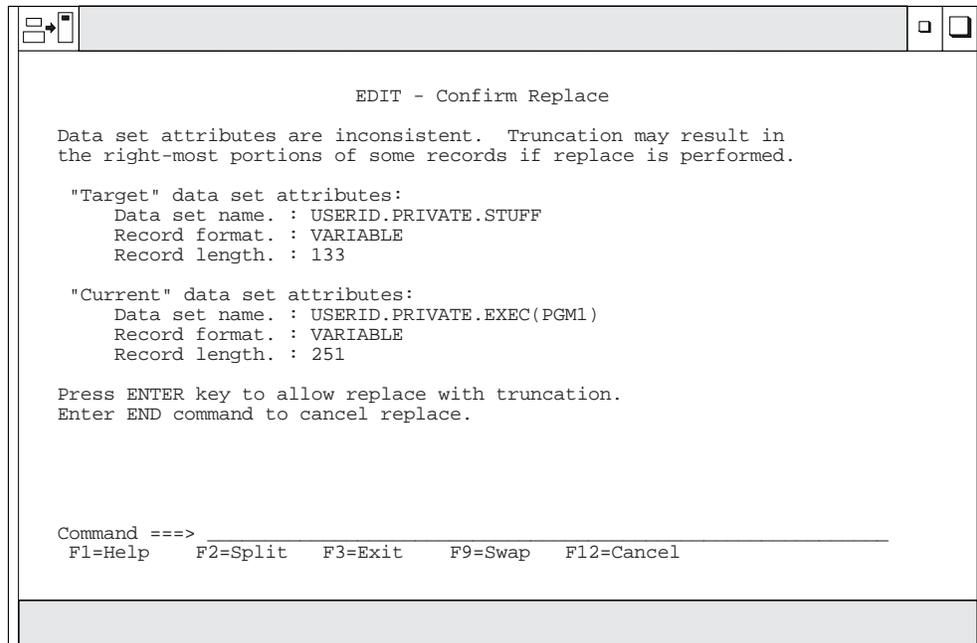


Figure 17. Confirm Replace Panel (ISRERPL2)

Copying and Moving Data

While you are editing, you can copy or move another data set or member into the current data by using the COPY or MOVE primary commands. The process of moving and copying data is very similar. However, remember that when you move data, the original information no longer exists in the member or data set that it is being moved from.

When moving or copying large data sets, you can reduce the processing time significantly by specifying NUMBER OFF before the operation and NUMBER ON afterwards.

This topic explains how to use the COPY and MOVE primary commands. See “C—Copy Lines” on page 170 and “M—Move Lines” on page 182 for information about the line commands.

The two ways to perform a move or copy operation are:

- You can type either COPY or MOVE, followed by *name* and either AFTER *label* or BEFORE *label*, where *name* is the name of the member or data set to be copied or moved and *label* is a label that is defined in the line command area. The label can be defined by PDF, such as .ZFIRST for the first line of data, or it can be one that you have defined. If you omit the label, you can use the A (after) or B (before) line command to specify where the information is to go. When you press Enter, the member is copied or moved. See “COPY—Copy Data” on page 227 and “MOVE—Move Data” on page 260 for the complete syntax of the commands.
- If you omit the member name or data set name, and just type the command and the destination of the operation (using either the AFTER label or BEFORE label operand or the A or B line command), the editor displays a panel on which you

can specify the name of the member to be copied or moved. The only difference between the Edit Move and Edit Copy panels is that with Copy, you can specify the number of lines you want copied.

Shifting Data

When you edit data, the editor automatically shifts characters on a line to the left or right to accommodate insertions or deletions. This shifting can be either *implicit* or *explicit*. Implicit shifts occur when the CHANGE command *string-2* length is different from the *string-1* length. Explicit shifts occur when you use the following commands:

- Line commands
 - (Column Shift Left
 -) Column Shift Right
 - < Data Shift Left
 - > Data Shift Right

- Macro commands
 - Shift** (Column Shift Left
 - Shift**) Column Shift Right
 - Shift** <
 - Data Shift Left
 - Shift** >
 - Data Shift Right

See the descriptions of these commands for the syntax and examples of usage.

Two columns is the default for shift operations. When shifting a block of lines more or less than the default, enter the amount on the first or last line of the block. If you enter it in both places, the line shifts only if:

- Both amounts are the same, or
- The amounts differ, but one is the default (2). Here, the lines shift according to the non-default amount.

If the shift amounts are different and neither amount is the default, an error message appears and the shift is not performed.

Shifting occurs within column boundaries. The default boundaries are typically the first and last columns in which you can type source code for the particular programming language. See “Edit Boundaries” on page 28 for a discussion of default boundaries and the procedures for changing them.

Column Shift

The simplest kind of shift is a column shift. Column shifting moves all characters within the bounds without altering their relative spacing. Characters shifted past the bounds are deleted. That is, blanks are inserted at the bound from which the characters are being shifted, and the characters are deleted at the opposite bound. So, this shift is called a *destructive* shift because information shifts within column boundaries without regard to its contents, and can result in the loss of data with no error being noted.

If the UNDO mode was on before you entered the shift command, you can recover by using the UNDO command. Otherwise, you can use CANCEL.

Column Shifting in Lines that Contain DBCS Strings

The following rules apply:

- If half of a DBCS character is in the shift, it is excluded from the operation; the shift count is changed automatically.
- If a column shift causes a DBCS string and an EBCDIC string to be connected, a shift-out or shift-in character, as appropriate, is inserted between the strings. The shift count is changed automatically.
- If left, right, or both boundaries are set, a DBCS character can cross the boundary. The DBCS character that crosses the boundary is excluded from the operation, and the shift count is changed automatically.
- If a request to shift an odd number of columns causes an odd-length DBCS string, the requested shift number is discarded. The shift is processed up to the next field boundary within the boundary, if any. If no field boundary is found, the line number is replaced with the following intensified warning message: ==ERR>. Also, the short message for an incomplete data shifting error is displayed.

If you are using the column shifting or data shifting line command while editing a formatted data set, you should note the following points:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
 - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
 - If the right boundary does not fall on the same field as the left boundary, it is set to point to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.
- If you use the data shift or column shift line command to shift a DBCS field and you specify an odd-length shift amount, the shift amount is decreased by one to preserve DBCS data integrity.
- If a shift cannot be completed, it is partially done and the line number is replaced by the following intensified warning message: ==ERR>. Remove the message by issuing the RESET primary command, or type over the message or data on that line.
- If a request to shift an odd number of bytes causes an odd-length DBCS string, the shift volume is decreased by one and the operation is performed. The line number is replaced with the following intensified warning message: ==ERR>.

Data Shift

Data shifting attempts to shift the body of a program statement without shifting the label or comments, and prevents loss of data. This shift is *non-destructive* because it stops before it shifts a non-blank character past the bound. This shift is explicitly done with the < and > line commands, and the SHIFT < and SHIFT > macro commands. The CHANGE command can cause an implicit shift of the same nature.

For data shift left attempts that exceed the current BOUNDS setting, text stops at the left bound and PDF marks the shifted lines with ==ERR> flags. If an error occurs in an excluded line, you can find the error with LOCATE, and remove the error flag by using RESET.

Data shifts are designed to work with typical program sources. In doing so, it makes certain general assumptions about the format of the source code. For instance, the editor assumes:

- Anything beginning at the left bound is a label and should not be shifted.
- If there are two or more consecutive blanks, one can be added or deleted.
- Blanks within quotes (' or ") are to be treated as non-blanks.
- Source statements appear on the left followed by comments on the right.
- Single blanks are used between source code and comment words. Therefore, the only strings of multiple blanks appear between the source code and the comment, and between the comment and its ending delimiter (if there is one). In the following example, LABEL and */ are at the left and right bounds, respectively:

```
LABEL: DO I=1 TO 5;          /* The comment... */
      A=A+B(I);            /* The comment... */
      END;
```

Keeping the previous assumptions in mind, the editor attempts to move only the source code statement when shifting data. The label and comments are left unchanged. However, if necessary, it shifts the comment also.

Although the editor always uses these assumptions, data shifting is not language-sensitive. It only makes generalities about syntax and individual code entry style.

Finding, Seeking, Changing, and Excluding Data

FIND, SEEK, CHANGE, and EXCLUDE allow you to find a specified search string, change one search string to another, or exclude a line containing a specified search string. These commands provide powerful editing functions because they operate on a complete data set rather than on a single line.

The characteristics of each command follow:

FIND Causes all lines that it finds to be displayed, and moves the cursor (scrolling if necessary) to the first occurrence of the search string.

SEEK A special form of FIND that can only be used in an edit macro. It is different from FIND in that it does not change the exclude status of the lines found.

CHANGE

Causes the same effect as FIND, but it also has a second string operand (*string-2*). During a search, whenever *string-1* is found, the editor replaces that string with *string-2*. Data to the right is shifted, if necessary.

EXCLUDE

Causes lines that match the search not to be displayed. These lines remain in the data, however. Unlike FIND and CHANGE, it does not require a search string if you use the ALL operand. EXCLUDE ALL is often used with FIND and CHANGE because they cause excluded lines to be redisplayed. Use RESET to cause all lines to be redisplayed.

The syntax of each command is a variation of that listed below. See the command descriptions in Chapter 10. Edit Primary Commands and Chapter 11. Edit Macro Commands and Assignment Statements for the exact syntax.

Finding, Seeking, Changing, and Excluding Data

```
string [range] [NEXT ] [CHARS ] [X ] [col-1] [col-2]]
                [ALL ] [PREFIX] [NX]
                [FIRST] [SUFFIX]
                [LAST ] [WORD ]
                [PREV ]
```

Specifying the Search String

The primary control for any search is the search string because it represents the value for which you are looking. Two operands, *string-1* and *string-2*, are required for the CHANGE command to specify the new value of the string once it is found. The rules for specifying *string-1* and *string-2* are the same, except that if you type a single asterisk for *string-2*, the previous value is used again.

You can define *string-1* and *string-2* to be EBCDIC, DBCS, and mixed strings in any combination. If you delimit a DBCS search string (*string-1*) with SO and SI characters, the SO and SI characters are not used as part of the string. If you specify a mixed string that contains no EBCDIC characters, the string is treated as a DBCS string; that is, the SO and SI characters are not used as part of the string.

The editor allows you to specify the following kinds of strings:

Simple string

Any series of characters not starting or ending with a quote (' or ") and not containing any embedded blanks, commas, or asterisks.

Delimited string

Any string enclosed (delimited) by either single quotes (') or double quotes ("). The beginning and ending delimiters must be the same character.

Hexadecimal string

Any delimited string of valid hexadecimal characters, preceded or followed by the character X, such as X'C27B'.

Character string

Any delimited string of characters, preceded or followed by the character C, such as C'conditions for'. See "Character Strings" on page 55 for more information.

Picture string

Any delimited string of picture characters, preceded or followed by the character P, such as P'. '. See "Picture Strings (String-1)" on page 55 and "Picture Strings (String-2)" on page 56 for more information.

Note: The Edit FIND, CHANGE, and EXCLUDE commands do not work with a search argument that contains the command delimiter, even if string delimiters are used. You can specify a hexadecimal search string or use ISPF Option 0.1 to change the command delimiter to a different character.

Simple and Delimited Strings

If the string is a simple or delimited string, the characters are treated as being both upper and lowercase even if caps mode is off. For example, this command:

```
find ALL 'CONDITION NO. 1'
```

successfully finds the following:

Finding, Seeking, Changing, and Excluding Data

CONDITION NO. 1
Condition No. 1
condition no. 1
coNDitION n0. 1

Also, all of the following commands have the same effect:

```
FIND 'Edit Commands'  
FIND 'EDIT COMMANDS'  
FIND 'edit commands'
```

You must use delimiters if a string contains imbedded blanks or commas, or if a string is the same as a command or keyword. You delimit strings with quotes, either ' or ". For example, to change the next occurrence of every one to all, type:

```
Command ==> CHANGE 'every one' 'all'
```

Note: When using a DBCS terminal, if you specify a text string that contains any SO and SI characters, the string is considered a character string.

Character Strings

Use a character string in a FIND, CHANGE, or EXCLUDE command if you want the search to be satisfied by an exact character-by-character match. Lowercase alphabetic characters match only with lowercase alphabetic characters, and uppercase alphabetic characters match only with uppercase.

For example, FIND C'XYZ' finds the characters XYZ only, not xyz.

Picture Strings (String-1)

A picture string in a FIND, CHANGE, or EXCLUDE command allows you to search for a particular kind of character without regard for the specific character involved. You can use special characters within the picture string to represent the kind of character to be found, as follows:

String Meaning

P'='	Any character
P'~'	Any character that is not a blank
P'.'	Any character that cannot be displayed
P'##'	Any numeric character, 0-9
P'-'	Any non-numeric character
P'@'	Any alphabetic character, uppercase or lowercase
P'<'	Any lowercase alphabetic character
P'>'	Any uppercase alphabetic character
P'\$'	Any special character, neither alphabetic nor numeric.

If you are using an APL or TEXT keyboard, you can use the following additional characters in a picture string:

P'␣'	Any APL-specific or TEXT-specific character
P'_'	Any underscored non-blank character.

A picture string can include alphanumeric characters, which represent themselves, mixed with other characters. If the character does not have a special meaning (such as @ standing for any alphabetic), the character is treated as itself.

Finding, Seeking, Changing, and Excluding Data

When using a DBCS terminal, you cannot specify a DBCS field as the subject of a picture string for the FIND operation.

Picture String Examples:

- To find a string of 3 numeric characters:

```
FIND P'###'
```

- To find any 2 characters that are not blanks but are separated by a blank:

```
FIND P' - '
```

- To find any character that cannot be displayed:

```
FIND P'.'
```

- To find a blank followed by a numeric character:

```
FIND P' #'
```

- To find a numeric character followed by AB:

```
FIND P'#AB'
```

- To find the next character in column 72 that is not a blank:

```
FIND P' -' 72
```

- To change any characters in columns 73 through 80 to blanks:

```
CHANGE ALL P'=' ' ' 73 80
```

- To find the next line with a blank in column 1 and a character in column 2 that is not a blank:

```
FIND P' -' 1
```

When you use the special characters = or . and a character that cannot be displayed is found, that character's hexadecimal representation is used in the confirmation message that appears in the upper-right corner of the panel. For example:

```
FIND P'..''
```

could result in the message CHARS X'0275' FOUND.

Picture Strings (String-2)

In a CHANGE command, *string-2* can be a picture string with the following rules and restrictions:

- The length of *string-2* must be the same as the length of *string-1*.
- The only valid special characters are =, >, and <.

String Meaning

P'=' Equal to the corresponding character in *string-1*

P'>' Converts the corresponding character in *string-1* to uppercase

P'<' Converts the corresponding character in *string-1* to lowercase.

Picture String Examples:

- To change an alphabetic, alphabetic, numeric, numeric string so that the alphabetic characters become uppercase characters and the numeric characters are unchanged:

```
CHG P'@##' P'>>=='
```

- To change all characters to uppercase:

```
CHG ALL P'<' P'>'
```

Effect of CHANGE Command on Column-Dependent Data

Column-dependent data is groups of non-blank source data separated by two or more blanks, such as a table. When you use CHANGE to change column-dependent data, ISPF attempts to maintain positional relationships. For instance, if you change a long word to a short word, the editor pads the short word with blanks. This padding maintains the column position of any data to the right of the change by preventing it from shifting left.

When only one blank separates words, as in most text data, padding does not occur. Changing a long word to a short word causes data to the right of the change to shift left.

Using the CHANGE Command With EBCDIC and DBCS Data

If you are editing a data set that contains both EBCDIC and DBCS data, you should note the following rules about CHANGE strings:

- The SO and SI characters that delimit the CHANGE string are used as part of the string only if necessary. If you specify replacement of an EBCDIC string with a DBCS string, they are used. If you specify replacement of a DBCS string with another DBCS string, they are not used.
- If you specify in a CHANGE string that an SO or SI character be changed to another character, the result is unpredictable.
- If you specify a CHANGE string that causes a field length of zero and the boundary falls between the SO and SI characters, the SO/SI or SI/SO character strings that are next to each other are replaced with a DBCS blank. If the boundary does not fall between the SO and SI characters, the SO/SI or SI/SO characters that are next to each other are removed.
- If the lengths of the two strings specified in CHANGE are different, the following occurs:
 - If *string-1* is shorter than *string-2*, the data to the right of *string-1* is shifted to the left up to some breakpoint. Breakpoints include the border between an EBCDIC field and a DBCS field, a double or single blank, or the right boundary set by a BOUNDS command.
 - If *string-1* is longer than *string-2*, blanks in the record to the right of *string-1* are used to make room. When blanks in a DBCS field are used, they are used in units of 2 bytes.
- If a DBCS field crosses the right boundary, CHANGE can cause an odd-length DBCS field. If this happens, the right boundary is ignored and the operation takes place.

Controlling the Search

After you specify the search string, you can then specify how much of the data you want to search, as well as the starting point and direction of the operation.

Extent of the Search

You can limit the lines to be searched by first assigning a label to the first and last lines to be searched, and then specifying the labels on the command (range operand).

Finding, Seeking, Changing, and Excluding Data

If you want to limit the search to a single line, assign a label to it, and then specify the label twice to show the first and last line of the range. For more information about labels, see “Labels and Line Ranges” on page 66.

Starting Point and Direction of the Search

To control the starting point and direction of the search, use one of the following operands:

- NEXT** Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string-1*. NEXT is the default.
- ALL** Starts at the top of the data and searches ahead to find all occurrences of *string-1*. The long verification message, which PDF displays when you enter the HELP command in response to the short verification message, shows the number of occurrences found. If you use this operand with CHANGE, the lines changed are marked with ==CHG> flags, and lines that cannot be changed are marked with ==ERR> flags. The status of these lines can be used by LOCATE and changed by RESET.
- FIRST** Starts at the top of the data and searches ahead to find the first occurrence of *string-1*.
- LAST** Starts at the bottom of the data and searches backward to find the last occurrence of *string-1*.
- PREV** Starts at the current cursor location and searches backward to find the previous occurrence of *string-1*.

If you specify NEXT, ALL, or FIRST, the direction of the search is forward. When you press the assigned function keys, the RFIND or RCHANGE commands find or change the next occurrence of the designated string. If you specify LAST or PREV, the direction of the search is backward. When you specify those operands, the editor finds or changes the previous occurrence of the string.

The search proceeds until the editor finds one or all occurrences of *string-1*, or the end of data.

If you omit the ALL operand on the CHANGE command, the editor searches only for the first occurrence of *string-1* after the current cursor location. If the cursor is not in the data area of the panel, the search starts at the beginning of the first line currently displayed. Scrolling is performed, if necessary, to bring the string into view.

After you make the change, the cursor is positioned at the end of the changed string; a verification message is displayed in the upper right-hand corner of the panel.

Depending on the direction of the search, if the string is not found between the current cursor location and the end or beginning of data, a message is displayed and an audible alarm, if installed, is sounded.

If *string-1* is not found, one of the following actions takes place:

- A NO *string-1* FOUND message is displayed in the upper right-hand corner of the panel.
- If CHANGE or EXCLUDE was repeated using RFIND or RCHANGE, either a BOTTOM OF DATA REACHED or a TOP OF DATA REACHED message appears, depending on the direction of the search. When these messages appear, you can enter

Finding, Seeking, Changing, and Excluding Data

RFIND or RCHANGE again to continue the search by wrapping to the top or bottom of the data. If *string-1* is still not found, a NO *string-1* FOUND message is displayed.

Qualifying the Search String

You can specify additional characteristics of *string-1* by using the operands PREFIX, SUFFIX, CHARS, and WORD. You can abbreviate PREFIX, SUFFIX, and CHARS to PRE, SUF, and CHAR, respectively.

CHARS

Locates *string-1* anywhere the characters match. This is the default.

PREFIX

Locates *string-1* at the beginning of a word.

SUFFIX

Locates *string-1* at the end of a word.

WORD

String-1 is delimited on both sides by blanks or other non-alphanumeric characters.

In the following example, the editor would find the underscored strings only:

```
CHARS 'DO' - DO DONT ADO ADOPT 'DO' (DONT)
```

```
PREFIX 'DO' - DO DONT ADO ADOPT 'DO' (DONT)
```

```
SUFFIX 'DO' - DO DONT ADO ADOPT 'DO' (DONT)
```

```
WORD 'DO' - DO DONT ADO ADOPT 'DO' (DONT)
```

If you do not specify an operand, the default is CHARS.

Column Limitations

The *col-1* and *col-2* operands allow you to search only a portion of each line, rather than the entire line. These operands, which are numbers separated by a comma or by at least one blank, show the starting and ending columns for the search. The following rules apply:

- If you specify neither *col-1* nor *col-2*, the search continues across all columns within the current boundary columns.
- If you specify *col-1*, the editor finds the string only if the string starts in the specified column.
- If you specify both *col-1* and *col-2*, the editor finds the string only if it is entirely within the specified columns.

Split Screen Limitations

When *string-1* is not found within the data that is displayed on the screen, the search operation scrolls the data so that *string-1* appears on the second displayed

Finding, Seeking, Changing, and Excluding Data

line of the data area. If only one line of data is showing in split screen mode, the data on the second line (thus, *string-1*) cannot be seen and the cursor is placed on the command line.

Excluded Line Limitations

You can limit the lines to be searched by first using the X or NX operands:

X Scan only lines that are excluded from the display.
NX Scan only lines that are not excluded from the display.

If you omit these operands, both excluded and nonexcluded lines are searched. When you issue a FIND or CHANGE command that includes searching excluded lines, all lines found are displayed. EXCLUDE can also find labels assigned to excluded lines.

Using the X (Exclude) Line Command with FIND and CHANGE

You can use the X (exclude) line command with FIND and CHANGE to display only those lines containing the search string or those lines that have been changed. For example, if your data set contains 99,999 lines or less, type X99999 in the line command area of the first line to exclude all of the lines from the display. Then enter a CHANGE command, such as:

```
COMMAND ==>> CHANGE ALL XYZ ABC
```

All lines containing search string XYZ are redisplayed with XYZ changed to ABC and with the cursor at the end of the first string changed.

Similarly, you can enter a FIND command:

```
Command ==>> FIND ALL XYZ
```

Here, all lines containing the search string XYZ are redisplayed with the cursor at the beginning of the first string found.

Repeating the FIND, CHANGE, and EXCLUDE Commands

The easiest way to repeat FIND, CHANGE, and EXCLUDE without retyping them is to assign those commands to function keys. The defaults are:

```
F5/17 RFIND  
F6/18 RCHANGE
```

The search begins at the cursor. If the cursor has not moved since the last FIND, CHANGE, or EXCLUDE command, the search continues from the string that was just found. Instead of retyping *string-1*, you can type an asterisk to specify that you want to use the last search string. If you decide to type RCHANGE or RFIND on the Command line instead of using a function key, position the cursor at the desired starting location before pressing Enter.

All three commands share the same *string-1*. Therefore:

```
Command ==>> FIND ABC
```

followed by:

```
Command ==>> CHANGE * XYZ
```

Finding, Seeking, Changing, and Excluding Data

first shows you where ABC is, and then replaces it with XYZ. However, you can do this more easily by typing:

```
Command ==> CHANGE ABC XYZ
```

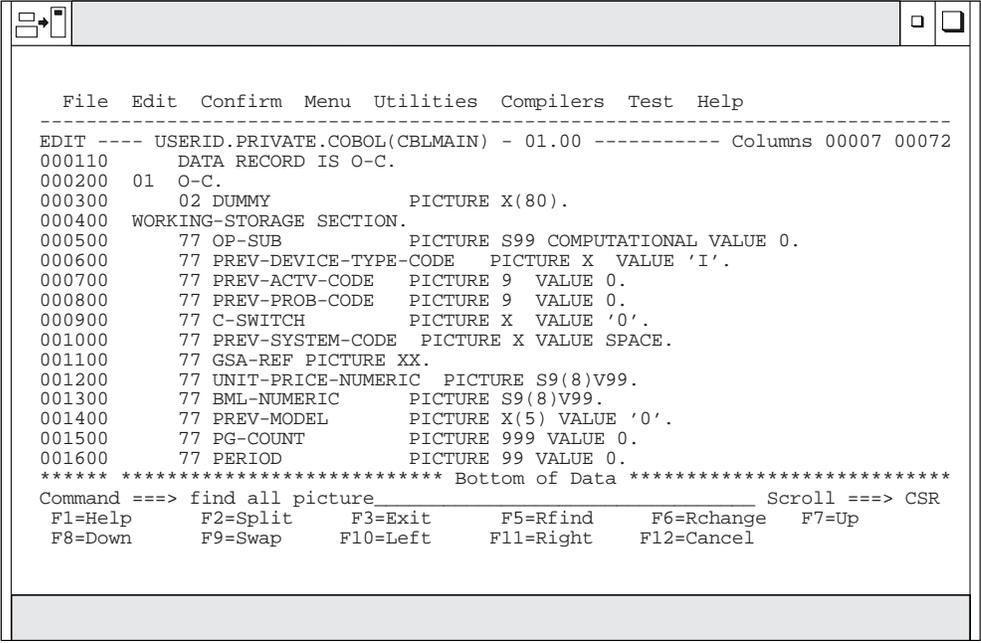
Then press F5/17 to repeat FIND. The editor finds the next occurrence of ABC. You can either press F5/17 to find the next ABC, or F6/18 to change it. Continue to press F5/17 to find remaining occurrences of the string.

The previous value of a search string, specified by an asterisk or by use of RFINDD or RCHANGE, is retained until you end your editing session.

Examples

FIND Command Example

To find all occurrences of “picture” in a member such as the one shown in Figure 18, type find all picture on the Command line.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- Columns 00007 00072
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
001200 77 UNIT-PRICE-NUMERIC PICTURE S9(8)V99.
001300 77 BML-NUMERIC PICTURE S9(8)V99.
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> find all picture_____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 18. Before FIND Command (ISREDD2)

After you press Enter, the editor searches for the string starting at the top of the data, places the cursor at the beginning of the first occurrence, and displays the number of occurrences as shown in Figure 19.

Finding, Seeking, Changing, and Excluding Data

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- 19 CHARS 'PICTURE'
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
001200 77 UNIT-PRICE-NUMERIC PICTURE S9(8)V99.
001300 77 BML-NUMERIC PICTURE S9(8)V99.
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F11=Left F12=Right F12=Cancel
```

Figure 19. After FIND Command

CHANGE Command Example

To change “numeric” to “numeric-int” type `chg numeric numeric-int` all on the Command line as shown in Figure 20.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- Columns 00007 00072
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
001200 77 UNIT-PRICE-NUMERIC PICTURE S9(8)V99.
001300 77 BML-NUMERIC PICTURE S9(8)V99.
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> chg all numeric numeric-int Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 20. Before CHANGE Command

Finding, Seeking, Changing, and Excluding Data

The editor changes all occurrences of the string starting at the top of the data and inserts a ==CHG> flag next to each changed line, as shown in Figure 21.

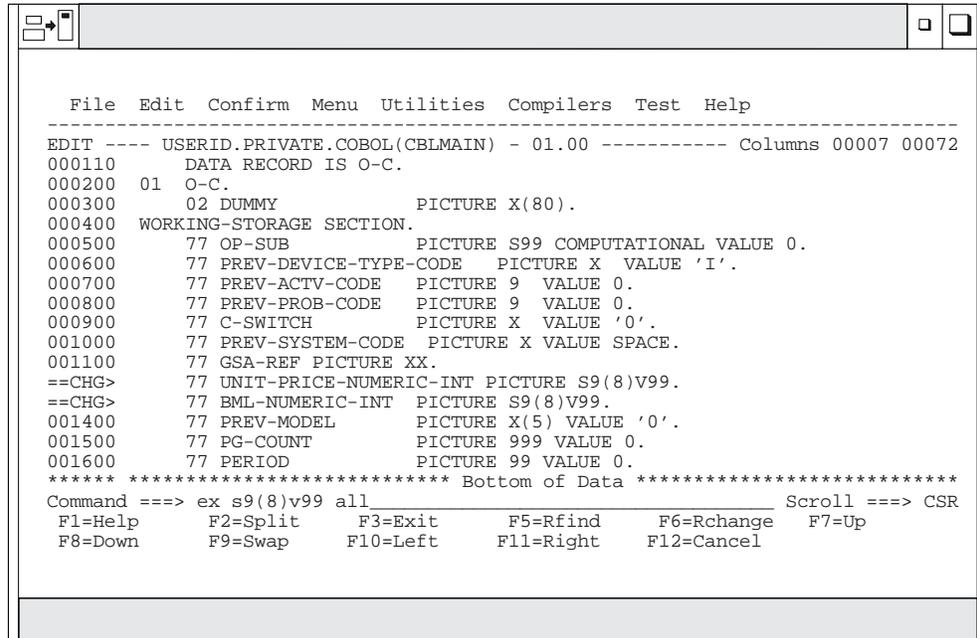
```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- CHARS 'NUMERIC' changed
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
==CHG> 77 UNIT-PRICE-NUMERIC-INT.PICTURE S9(8)V99.
==CHG> 77 BML-NUMERIC-INT PICTURE S9(8)V99.
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 21. After CHANGE Command

Finding, Seeking, Changing, and Excluding Data

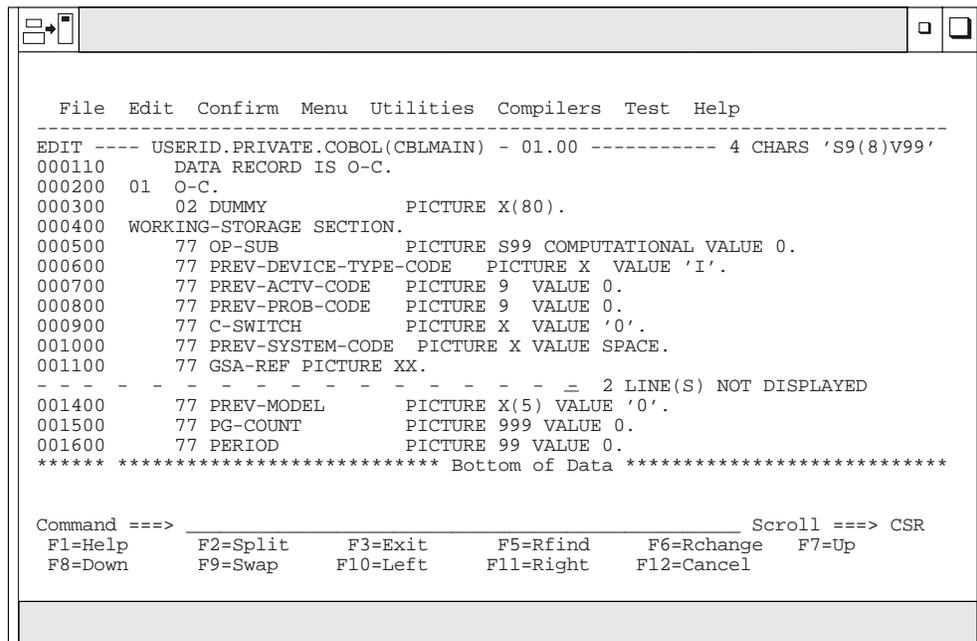
EXCLUDE Command Example

When you enter an EXCLUDE command like `ex s9(8)v99 all` on the Command line (Figure 22), the editor excludes all lines with that string starting at the top of the data (Figure 23).



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- Columns 00007 00072
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
==CHG> 77 UNIT-PRICE-NUMERIC-INT PICTURE S9(8)V99.
==CHG> 77 BML-NUMERIC-INT PICTURE S9(8)V99.
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> ex s9(8)v99 all Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 22. Before EXCLUDE Command



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.COBOL(CBLMAIN) - 01.00 ----- 4 CHARS 'S9(8)V99'
000110 DATA RECORD IS O-C.
000200 01 O-C.
000300 02 DUMMY PICTURE X(80).
000400 WORKING-STORAGE SECTION.
000500 77 OP-SUB PICTURE S99 COMPUTATIONAL VALUE 0.
000600 77 PREV-DEVICE-TYPE-CODE PICTURE X VALUE 'I'.
000700 77 PREV-ACTV-CODE PICTURE 9 VALUE 0.
000800 77 PREV-PROB-CODE PICTURE 9 VALUE 0.
000900 77 C-SWITCH PICTURE X VALUE '0'.
001000 77 PREV-SYSTEM-CODE PICTURE X VALUE SPACE.
001100 77 GSA-REF PICTURE XX.
- - - - - 2 LINE(S) NOT DISPLAYED
001400 77 PREV-MODEL PICTURE X(5) VALUE '0'.
001500 77 PG-COUNT PICTURE 999 VALUE 0.
001600 77 PERIOD PICTURE 99 VALUE 0.
***** Bottom of Data *****
Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 23. After EXCLUDE Command

Excluding Lines

You can exclude lines from a data set using the X (exclude) line command as well as the EXCLUDE primary command.

When you are editing a program that exceeds the screen size, it is often difficult to determine whether the control structure and indentation levels are correct. Excluding lines allows you to remove one line or a block of lines from the display so that you can see the general control structure. The lines are excluded from the display, but are not deleted from the data. Excluded lines are treated as valid data lines.

The X line command can have the syntax:

X[n]

or

XX

The first form allows you to exclude one line (X) or any number of lines (Xn).

The second form allows you to exclude a block by typing XX on the first and last lines of the block of lines that you want to exclude. The first and last lines do not need to be on the same page; after typing the first XX you can scroll to the second XX.

You can enter any line command that usually operates on a single line in the line command area of the excluded lines message. For example, if you enter the D (delete) line command, the complete block of excluded lines is deleted.

Redisplaying Excluded Lines

To display all excluded lines, enter the RESET EXCLUDED primary command. Alternatively, you can display one or more excluded lines again by entering the S (show), F (first), or L (last) line commands, typing over the dashes in the line command area. If these commands are typed outside the dashes of the command line area, no action is taken.

You can add a number following any of these line commands to cause more than one line to appear again:

S[n]

F[n]

L[n]

FIND and CHANGE also cause any excluded lines that meet the search criteria to appear again.

The S line command causes the editor to scan block of excluded lines, and one or more lines is selected to be appear again. The selected lines are those with the leftmost indentation levels; that is, the lines that contain the fewest leading blanks. If you type S3, for example, the three lines with the leftmost indentation level are displayed again. If more than three lines exist at this indentation level, only the first three are displayed.

Excluding Lines

Note: If you enter an S line command to display all but one line of an excluded block, then that line is also displayed. This could result in more lines being displayed than the number you requested. For example, if five lines are excluded in a block, an S4 command causes all five lines to be displayed.

Redisplaying a Range of Lines

The FLIP command lets you reverse the exclude status of a specified group of lines in a file or of all the lines in the file. This is useful when you have used the 'X ALL;FIND ALL xyz' command to find lines containing a string (xyz) and want to see the lines which do not contain the string. You can also use FLIP to show excluded note, message, and information lines.

You can enter one or two labels to specify the range of lines whose include status you want to reverse. If no labels are specified, the exclude status of all of the lines is reversed.

To reverse the exclude status of all the lines in a file, use the following syntax:

```
Command ==> flip
```

To reverse the exclude status of specified lines, use the following syntax:

```
Command ==> flip .a .b
```

The lines between labels .a and .b are redisplayed.

Labels and Line Ranges

A label is an alphabetic character string used to name lines or strings of data for easy reference. Because labels remain with the lines to which they are assigned, they are especially useful in keeping track of lines whose numbers might change. Most labels are assigned in macros, but certain labels are automatically assigned by the ISPF editor.

You can assign a label to a line by typing the label over the line number on the left side of the panel. The label is displayed in place of the number whenever the line is being displayed. If you then move the line, the label moves with it. You cannot type a label on a non-data line or on the line that is displayed to show one or more lines is excluded.

A label must begin with a period, and be followed by no more than 5 alphabetic characters (8 for edit macros), the first of which cannot be a Z. Labels beginning with Z are reserved for use by the editor. No special or numeric characters are allowed.

To eliminate a single label, blank it out. To eliminate all labels, use the RESET LABEL command.

An edit macro can assign labels to lines that the macro references frequently. See "Labels in Edit Macros" on page 112 for details.

Editor-Assigned Labels

The editor automatically assigns special labels that begin with the letter Z. Only the editor can assign a special label.

These built-in labels are:

.ZCSR The data line on which the cursor is currently positioned.

.ZFIRST

The first data line (same as relative line number 1). Can be abbreviated **.ZF**.

.ZLAST

The last data line. Can be abbreviated **.ZL**.

Unlike other labels, **.ZCSR**, **.ZFIRST**, and **.ZLAST** do not stay with the same line. Label **.ZCSR** stays with the cursor, and labels **.ZFIRST** and **.ZLAST** remain with the current first and last lines.

Specifying a Range

Labels allow you to specify a line or a range of lines on a primary command. You can specify two labels to define a range of lines to be processed on the following commands:

CHANGE	FIND	RESET
DELETE	LOCATE	SORT
EXCLUDE	REPLACE	SUBMIT

The range operand is always optional. If you do not specify a range, it defaults to **.ZFIRST** and **.ZLAST**. For example, the command:

```
CHANGE ALL 'TEST' 'FINAL'
```

starts at the first line of the data being edited and scans all lines up to and including the last line, changing all occurrences of **TEST** to **FINAL**.

However, the command:

```
CHANGE .ZCSR .ZLAST ALL 'TEST' 'FINAL'
```

specifies a range, and is thus interpreted differently. The command changes only the last part of the data.

When you use labels to specify a range, you must always use two labels to define the first and last lines, inclusively. To process a single line, repeat the label:

```
CHANGE ALL " " "_" .A .A
```

The command in the previous example is interpreted as, “Change all blanks to underscores on the **.A** line”.

The order in which you specify the labels is not important. The editor assumes that the line closer to the beginning of the data set is the first line of the range, and the line closer to the end of the data set is the last.

A common error when using a range is to assume that the search begins at the first character of the line with the first label. Remember, however, that the default is **NEXT** and that the search starts at the cursor location. Lines outside the range are logically the same as the **TOP OF DATA** and **BOTTOM OF DATA** lines. Use the **FIRST**, **LAST**, or **PREV** operands to ensure that the search begins within the range.

Labels and Line Ranges

Using Labels and Line Ranges

The following examples show the results of using labels to identify ranges of lines. They show that the order of both labels and other operands is not important, and that you can type both labels and operands in either uppercase or lowercase.

- The following command locates the first line flagged ==CHG> between the line labeled .start and the line with the cursor on it:

```
locate first chg .start .zcsr
```
- The following command changes the last occurrence of pre to post between the first line and the line marked with the .here label:

```
change last pre post .here .zfirst
```
- The following command changes all occurrences of pre to post from the .mylab line to the last line of the data set:

```
change pre post all .mylab .zl
```
- The following command finds the word higher between the .start line and the .end line:

```
find higher word .start .end
```

Word Processing

This section is a general overview of three line commands for word or text processing: TF (text flow), TS (text split), and TE (text entry). The editor also provides three corresponding edit macro commands: TFLOW, TSPLIT, and TENTER. For the sake of simplicity, only the line commands are referred to. However, the descriptions apply to the macro commands, as well.

TF, TS, and TE assume that the data is grouped in paragraphs. A paragraph is a group of lines that begin in the same column. The first line of a paragraph is excluded from the grouping. The editor interprets any indentation or blank line as representing a new paragraph. It also recognizes word processor control words that are used by the Document Composition Facility as the beginning of a paragraph. These control words begin with a period, a colon, or an ampersand.

If you use text line commands frequently, you can assign both the TS and TF commands to function keys. Use KEYS to reassign the keys. For example:

```
F10 ==> :TS  
F11 ==> :TF
```

Now you can split text by moving the cursor to the desired split point within a line and pressing F10. Having typed the new material, press F11 to restructure the text from the line containing the cursor to the end of the paragraph.

Formatting Paragraphs

The TF (text flow) line command formats paragraphs. It assumes that the sentences are roughly in paragraph form with a ragged right margin when it attempts to recognize groupings. TF can be followed by a number (TF72 for example) that specifies the desired right side column for the paragraph. If you do not specify a number, the right side of the panel is used unless you have set bounds different from the default. In that case, the right boundary is used. The editor assumes that because the first line of a paragraph may be at a different indentation level than the remainder of the paragraph, the starting column of the second line is the left side of the paragraph.

When formatting paragraphs, the editor:

- Moves text so that each line contains the maximum number of words. TF limits its activity to within the bounds. Thus, it can be used to flow text within a border.
- Keeps any blanks between words.
- Assumes one blank between the word at the end of a line and the word on the next line except when the line ends with a period. In that case, the editor inserts two blanks.

The end of the paragraph is denoted by a blank line, a change in indentation, or the special characters period (.), colon (:), ampersand (&), or left caret (<) in the left boundary column. These special characters are used as Document Composition Facility (SCRIPT/VS) control word delimiters.

The restructure operation removes trailing blanks on a line by using words from the following line. It does not, however, remove embedded blanks within a line. Accordingly, if one or more words in a line are to be removed, delete the words rather than type over them.

The text to be restructured is taken from within the currently-defined column boundaries. Any text outside the bounds is not included in the restructuring. The restructured text is also positioned within the current boundaries. If the original text was indented from the left boundary, that indentation is preserved.

Using Text Flow on a DBCS Terminal

You can restructure paragraphs containing lines that include DBCS strings based on the following rules:

- If a character in a DBCS string encroaches on the rightmost column position for the restructured text, the string is divided before that character. An SI character is added at the end of the line, and an SO character is added at the beginning of the new line.
- If the boundaries are defined and a DBCS character is on the boundary, the DBCS character is in the text flow operation. An SO or SI character is added to both lines to ensure that DBCS character strings remain enclosed with SO and SI characters.
- If the mask contains DBCS fields and some of the DBCS fields cross the left, right, or both boundaries, the result may be unpredictable.
- If a DBCS string crosses the left, right, or both boundaries, the result may be unpredictable.
- When a text flow operation causes a field length of zero, the SO/SI or SI/SO character strings that are next to each other are removed.

If you use the TF line command on a line while editing a formatted data set, you should note that:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
 - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
 - If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on

the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.

- If you specify the column number with the TF command, and if the column falls on the first byte of a DBCS character in a DBCS field, the column number increases by one.

Splitting Lines

The TS (text split) line command splits a line into two lines. The cursor shows where the line is to be split. The editor moves the characters to the right of the cursor or to a new line following the original line and aligns the new line with the left side of the paragraph. As mentioned earlier, the left side of a paragraph is determined by looking for a pattern in the lines preceding or succeeding a paragraph.

If the line being split is the first line in a paragraph, the new line is aligned with the rest of the lines in the paragraph. If there are no other lines in the paragraph, the portion of the line to the right of the cursor aligns itself with the first portion of the line.

One or more blank lines are inserted after the line being split, depending on what you specify when you enter the TS command. Note that the TSPLIT macro command inserts only one blank line.

To rejoin lines, use the TF (text flow) line command. See “Formatting Paragraphs” on page 68 for more information.

Splitting Lines Within a DBCS String

You can split a line within a DBCS string based on the following rules:

- When splitting at a DBCS character, an SI character is added to the end of the line and an SO character is added at the beginning of the new line.
- If the cursor is placed at the SO character, the SO character becomes the first character to be moved.
- If the cursor is placed at the SI character, the character following the SI character becomes the first character to be moved.
- If the mask contains DBCS fields and some of the DBCS fields cross the left, right, or both column boundaries, the result is unpredictable.

If you use the TS line command while editing a formatted data set, you make special considerations for the current boundaries. These boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:

- If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
- If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.

Entering Text (Power Typing)

The TE (text entry) line command allows you to *power type*. When using this command, the display is filled with blank lines. The line number field normally on the left of the display disappears, so that you can type all of your data as if it were one continuous line. Because the editor is doing the formatting, you can continue typing and ignore the wrap around on the display. Any explicit cursor movement is interpreted as your personal formatting and results in embedded blanks.

The editor assumes that you are typing text as paragraphs. If you explicitly move the cursor down and leave a blank line, the editor assumes that the blank line should be there. The text that follows the blank line is consequently a new paragraph. Similarly, if you leave a specified number of blanks between words, the editor leaves them there. Also, if you tab to the beginning of the next line before completing the current line, the editor does not flow these sentences together. Remember that skipping a line specifies the start of a new paragraph.

Note: You cannot use logical or hardware tabs during text entry.

When you press Enter, the text is flowed in the same manner as the TF (text flow) line command, except that it uses the bounds as the right and left sides of the paragraphs.

Entering Text on a DBCS Terminal

If you are using the TE line command in a formatted data set, you should note that:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
 - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
 - If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.
- The attribute of the field where the left boundary falls is used for the text input area attribute. The new input data is reformatted to fit within the current boundaries.

Using Tabs

This section discusses hardware, software, and logical tabs, defining and controlling tabs, defining tab positions, and using attribute bytes.

Types of Tabs

Software and Hardware Tabs

The editor uses software and hardware tabs to reposition the cursor within the current display window. You can define tabs with the TABS line command. Use underscores (`_`) or hyphens (`-`) to define software tabs and asterisks (`*`) to define hardware tabs.

Logical Tabs

The editor uses logical tabs to reposition strings of data. You can use TABS primary and macro commands, and the TABS assignment statement to define a special character. The tab character locates the beginning of each string. Edit repositions the strings one space to the right of hardware tab positions.

Notes:

1. You cannot use the command delimiter that you defined on the Terminal Characteristics panel (option 0.1) as a special tab character.
2. Tabs are not functional when you are using the TE (text entry) line command.

Effect of TABS Commands on Tab Types

If you are using hardware or logical tabs, the TABS line command must be used with one of the other TABS commands or the TABS assignment statement. For example, hardware tab positions defined by the TABS line command do not take effect until tabs mode is turned on, which the line command cannot do. Conversely, a logical tab character defined with the TABS primary or macro command, or the TABS assignment statement, cannot be used to position data strings horizontally unless hardware tab positions are defined with the TABS line command. However, if you are using software tabs, you do not need to turn tabs mode on. The TABS primary and macros commands, and the TABS assignment statement, have no effect on software tabs.

Defining and Controlling Tabs

Three TABS commands help you quickly position the cursor where you want to start typing. These commands are the TABS line command, primary command, and macro command. There is also a TABS assignment statement.

You type the TABS line command in the line command area over the line numbers. This command:

- Displays the =TABS> (tab-definition) line
- Defines tab positions for software, hardware, and logical tabs.

You type the TABS primary command on the Command line. The TABS macro command is processed from within an edit macro. The TABS primary and macro commands can:

- Turn tabs mode on and off
- Define the logical tab character
- Control the insertion of attribute bytes at hardware tab positions that have been defined with the TABS line command.

The TABS assignment statement is processed from within an edit macro. It can do everything that the TABS macro command can do. In addition, the TABS assignment statement can retrieve the setting of tabs mode and place it in a variable.

You can use PROFILE to check the setting of tabs mode and the logical tab character.

Defining Software Tab Positions

If you display the =TABS> line and type software tab definitions, they take effect immediately. Each line contains a software tab or a tab field at the designated column positions. The TABS primary command has no effect on software tab definitions.

To define software tab positions:

1. Type TABS in the line command area and press Enter.
2. Type an underscore (_) or a hyphen (-) at each desired column position on the =TABS> line.
3. Press Enter again to start the tabs.

You can move the cursor from one column position to the next by continuing to press Enter. See “Using Software and Hardware Tabs” on page 196 for an example of using software tabs.

Defining Hardware Tab Positions

Hardware tab definitions do not take effect until you turn on tabs mode by using the TABS primary command. The asterisks define the column positions, but the insertion of attribute bytes (hardware tabs) or the repositioning of data strings (logical tabs) does not occur unless tabs mode is on.

To define hardware tab positions:

1. Type TABS in the line command area and press Enter.
2. Type an asterisk (*) at each desired column position on the =TABS> line.
3. Press Enter again.

When tabs mode is turned on using either the ON or ALL operand, the Tab Forward and Tab Backward keys can be used to move the cursor to the space following the next attribute byte.

Note: If the ALL operand is not used, attribute bytes are inserted only in spaces that contain a blank or null character, causing the Tab Forward and Tab Backward keys to recognize only these tab definitions.

When tabs mode is turned on using the *tab-character* operand, the Tab Forward and Tab Backward keys do not recognize hardware tab definitions because no attribute bytes are inserted.

Limiting the Size of Hardware Tab Columns

To limit the size of hardware tab columns, type consecutive asterisks between columns to define *hardware tab fields*. The consecutive asterisks:

- Allow you to determine the length of the data string to be typed in a column
- Cause the cursor to automatically move to the next column when the current column is full.

This procedure works only with asterisks (hardware tabs). When you type hyphens or underscores (software tabs), PDF does not insert attribute bytes. Because attribute bytes cannot be typed over, they limit the tab column size.

Using Tabs

Insert the asterisks from the point where you want the column to end to the point where the next column begins. For instance, suppose you want to limit each tab column to five spaces. You could do so by following these steps:

1. Type COLS in the line command area and press Enter. A partial =COLS> line with positions 9 through 45 is shown in the following example:

```
=COLS> -1-----2-----3-----4-----+
```

2. Type TABS ALL on the Command line and press Enter again. This command causes PDF to insert an attribute byte at each hardware tab position defined by an asterisk (*).

3. Using the TABS line command, change the =TABS> line as follows:

```
=COLS> -1-----2-----3-----4-----+
=TABS>          *      *****      *****
```

With the =TABS> line altered as shown, the cursor automatically skips to the next tab column when 5 characters, blank spaces, or a combination of both are typed in each column.

Using Attribute Bytes

Attribute bytes overlay characters only on the display; the attribute bytes are never recorded in the data. If your data set contains DBCS fields, however, attribute bytes can invalidate them. If you start hardware tabs and insert an attribute byte in the middle of a DBCS field, you invalidate the DBCS field, and it is displayed as an EBCDIC field. When you turn tabs mode off, the attribute bytes are removed and the overlaid character at each tab position is displayed again.

When you are in formatted data edit mode, TABS is ignored.

In tabs mode, you temporarily remove the attribute bytes from a single line. There are two ways to do this:

- Blank out the entire Line Command field using the Erase EOF key.
- Place the cursor directly under one of the attribute bytes and press Enter. When you press Enter again, the attribute bytes are reinserted.

Undoing Edit Interactions

If you enter an edit primary, line, or macro command, or type over existing data by mistake, you can restore your data with the UNDO primary command. UNDO has no operands.

Each time you enter UNDO it undoes one interaction. A single interaction might be a data change and Enter key, a data change and function key, or the invocation of an edit macro. All changes caused by an edit macro are considered to be one interaction. You can continue to undo interactions, one at a time, until you have reversed all changes made back to the beginning of your edit session unless you have done a save or undo recycled. If you have done a save or if undo recycled, you can only undo interactions back to that point. At that point, if you enter UNDO again, a message informs you that there are no more interactions to undo.

UNDO has certain limitations. Edit interactions that the command does not undo are:

- Changes that are made by an initial edit macro or recovery edit macro.
- Edit interactions before any data changes are made.

Undoing Edit Interactions

- Edit interactions in previous edit sessions.
- Reset of changed flags (==CHG>) by use of RESET or by typing over the command line area.
- Changes you make to other data sets or members by using the CREATE, REPLACE, or MOVE commands. Because UNDO affects only the member or data set that you are editing, it removes lines from your display if they were inserted there by MOVE. However, it does not put those lines back into the data set or member from which they came.

See “UNDO—Reverse Last Edit Interaction” on page 290 for a discussion of UNDO limitations.

UNDO is reset by SAVE. This means that you can UNDO interactions for the current edit session until you save your data. After the save, you can undo only interactions made following the time you saved your data.

UNDO can be run from data kept in storage or from the recovery file (as in previous releases) depending on what you specify in the Edit Profile for the data you are entering. The SETUNDO primary or macro command is used to control the profile setting. To use UNDO, you must have either RECOVERY on or SETUNDO on. You can undo only those changes made after RECOVERY or SETUNDO was turned on.

SETUNDO allows you to specify how changes you make during your edit session are to be recorded and used by UNDO. You can specify SETUNDO STORAGE or SETUNDO RECOVER. SETUNDO STORAGE specifies UNDO from storage. SETUNDO RECOVERY specifies UNDO from recovery and turns recovery on if it is off. See “SETUNDO—Set the UNDO Mode” on page 284 for more details. “Understanding Differences in SETUNDO Processing” on page 76 explains how the SETUNDO operands differ.

If not enough storage is available to run UNDO from storage but RECOVERY is on, UNDO processing continues to be available by using the recovery file. This makes UNDO available for very large files. It also provides users of machines with less storage with the benefit of UNDO for their larger files.

Note: If you have specified RECOVERY OFF and your installation allows UNDO from storage, the message that UNDO is unavailable does not display when you enter an edit session. If UNDOSIZE = 0, the message appears as before.

The UNDOSIZE specifies the number of kilobytes allowed for saving edit transactions for UNDO and the value is in the configuration table. For more details, refer to *ISPF Planning and Customizing*

If UNDOSIZE is set to zero, all undo documented functions work as in ISPF/PDF Version 3.3 and previous releases. This means that the Profile lines do **not** show the status of SETUNDO, and that warning messages will be shown informing you that UNDO is unavailable until RECOVERY is turned on.

UNDO Processing

When the storage allocated for changes is exhausted, UNDO *recycles* itself and puts up the message UNDO RECYCLED. Recycling is the process of saving the current image of the file as a new base from which to work. UNDO is then available after

Undoing Edit Interactions

the next transaction. No transactions made before the recycling can be undone. This is because UNDO saves an image of the original file and keeps an incremental list of changes to that image.

If there is not enough storage to save the initial image, then UNDO attempts to use the recovery file for undo processing. If recovery is off or suspended, the message UNDO SUSPENDED is shown with an alarm, and the profile status line is changed to SETUNDO SUSP. If recovery is available, the message UNDO FROM RECOVERY is shown with an alarm, and the profile status line is changed to SETUNDO REC. This affects the display but does not affect the edit profile values.

To resume SETUNDO STG, enter the SETUNDO primary command. If there is still not enough storage to hold the original copy of the file, the recycling procedure is repeated.

Note: Edit recovery can no longer process edit recovery files created under previous releases of ISPF/PDF. A panel is displayed, but no other action is taken if an old recovery file is used.

Understanding Differences in SETUNDO Processing

SETUNDO STORAGE and SETUNDO RECOVERY work essentially the same way; however, there are some important differences. SETUNDO REC is available only after the edit recovery file is initialized, that is, until the first data change is made. Because SETUNDO STG keeps its record of changes in storage, it does not incur the same performance penalty as using the SETUNDO REC.

SETUNDO STG can start to save editing changes earlier than SETUNDO REC, because even non-data changes, such as setting line labels, adding note lines, and inserting blank lines, cause SETUNDO STG to initialize its record of changes. You can undo these changes using UNDO even if no data changes have been made. When SETUNDO REC is in effect, only changes made after and including the first change to edit data can be undone.

UNDO reverses changes made during a single edit transaction. It is important to note, however, that changes to the profile, such as HEX ON, LEVEL, and CAPS, are not undone separately. A data change followed by one or more profile changes is usually considered a single transaction. For example, if you change the data and then the profile, and then enter UNDO, the data and profile return to their statuses before the data change. Profile changes usually cannot be undone if they are not preceded by a data change. SETUNDO STG and SETUNDO REC may work slightly differently in this regard. Since SETUNDO STG keeps the record of changes in storage, it is not a substitute for recovery. To recover the edit session after a system failure, you must have recovery on during the edit session. SETUNDO STG and RECOVERY ON can be in effect simultaneously, however, after a system crash and a recovery, no transactions can be undone using SETUNDO STG because the in-storage record will be empty.

If you are running both SETUNDO STG and RECOVERY ON, the UNDO command causes the last change to be backed out using the in-storage record of edit changes, and the recovery data set to be reinitialized. If you issue a SETUNDO REC command, after you use UNDO (from storage), there will be no more transactions to UNDO since the recovery file has been reinitialized.

Chapter 4. Using Edit Models

This chapter describes the PDF component edit models and tells you how to use them.

What Is an Edit Model?

A *model* is a predefined set of statements for a dialog element that you can include in the data you are editing and then modify to suit your needs. When you enter the MODEL command, you can select the correct segment for the data type being edited.

The PDF component is shipped with an initial set of models for panels, messages, skeletons, and command and program processing of ISPF and PDF component services. You can add more. There are no models of edit macro commands and assignment statements.

A model has two parts:

Data lines

These are the actual lines that are placed in the data you are editing. For example, the data might be a dialog service call or a panel format. You can update fields in the source statements by inserting names, parameters, and so forth.

The models also include source statement comments for models of dialog service calls to document the meanings of the possible return codes from the service. The comments are in a valid format for the particular kind of model. These comments give you the information you need to develop error-handling logic for your function. Sometimes they provide parameter descriptions for other kinds of models.

Notes Notes provide tutorial information about how to complete source code statements. You can specify whether you want the notes displayed during the edit session by using the NOTES command or the NOTES or NONOTES operand on the MODEL command. To remove notes from the panel, issue RESET. To convert the notes to data so that they can be saved with your data set, use the MD (make dataline) line command.

How Models Are Organized

Models are organized and named according to a hierarchy based on the type and version of the dialog element they represent. Each part of the model's name corresponds to a level in the hierarchy.

The first part of the logical name is the model *class*. There is a model class for each data set type qualifier that can store a dialog element. The Model Classes panel, Figure 24 on page 78, lists the classes defined for the models distributed by the PDF component. This panel prompts you when you need to set the desired model class, if you do not name the class explicitly.

Model Hierarchy

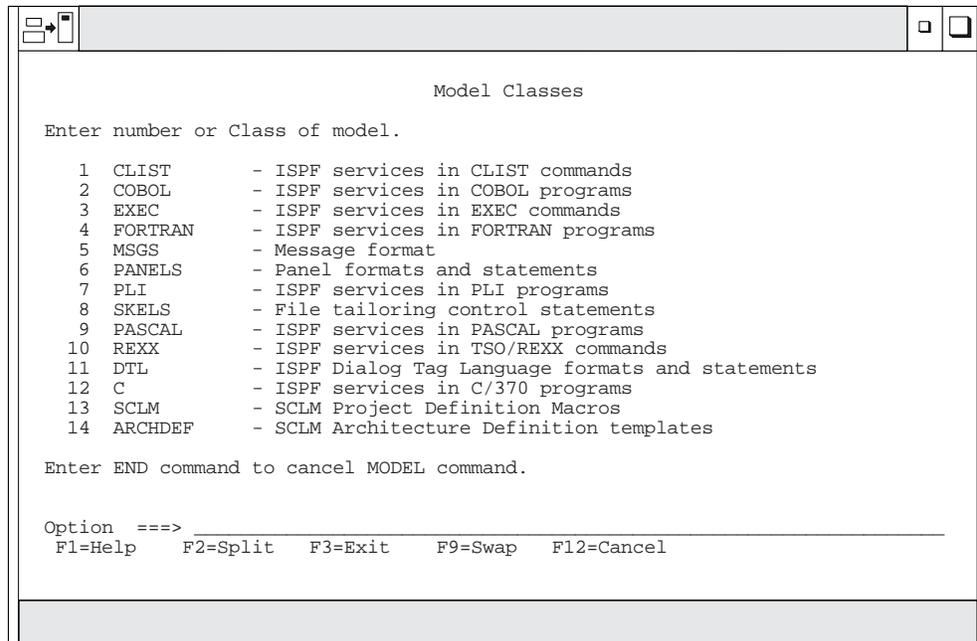


Figure 24. Model Classes Panel (ISREMCLS)

You can use the default for this part of the logical name whenever the edit profile name matches the class of the model desired.

The second part of the logical name is the model *name*, which identifies the specific model within the model class. Frequently, it uniquely identifies a model and completes the logical name. To uniquely identify a model, you can define optional *qualifiers*. Qualifiers are used, for example, to differentiate among the various kinds of panel verification (VER) statements.

A hierarchy of selection panels defines the hierarchy of models. The different parts of the logical name of a model are selections on the panels that you can choose either by keyword name or option identifier. This allows you to be prompted by selection panels if you do not know the logical name of the model you want or to bypass the display of these panels if you do know the name.

Usually, you do not need to worry about the model class. You must specify it only if you want to use a class that is different from the edit profile name. The model function of the editor recognizes PANELS as a valid type qualifier for panel models, so you do not need to specify the class when requesting a panel model from a data set with a type qualifier of PANELS (assuming you allow the edit profile name to default to panels).

Assume, however, that you call your panels screens and maintain them in a data set with a type of SCREENS. When you want to use a model to develop a new panel, you enter the MODEL command. The model function does not recognize SCREENS as a model class, so you are prompted to identify the class you want, which is the PANELS class in this situation.

Once you have specified a class, whether by panel selection or by use of the MODEL CLASS command, that class remains in effect until you change it. The two ways to change the class specification are by typing a data set name with a different type qualifier, or by leaving the Edit Entry panel.

How to Use Edit Models

You use models to assist you in defining a dialog element. To use a model, first edit your data. Then determine where you want to place the model. If you are editing existing data, define a label or use the A (after) or B (before) line command to show where the model goes. You do not need to use the A or B command when you have a new data set. Then type MODEL on the Command line and press Enter.

If you know the logical name of the model you want, you can use it to directly access the model. Type MODEL mmm, where mmm is the name of the model. For example, if you want the model for LMCLOSE, you would specify MODEL LMCLOSE. If you enter MODEL with no parameters, PDF displays a series of selection panels, from which you select the model name and any qualifiers.

The original data is then displayed with the model in place. You can type over or use line commands to change the data lines in the model to meet your needs.

As an example, assume that you are writing a dialog function using CLIST commands and you want to have the CLIST display a panel. You are editing your CLIST member, called USERID.PRIVATE.CLIST(DEMO1). Since your data set type, CLIST, matches the class of models you want, you can allow the model class to default. If you enter MODEL without a model name, the CLIST Models panel, Figure 25, appears.

Note: The following models for library access services shown in Figure 25 apply to LMF only: LMPROM, LMHIER, LMACT, LMDEACT, LMREVIEW.

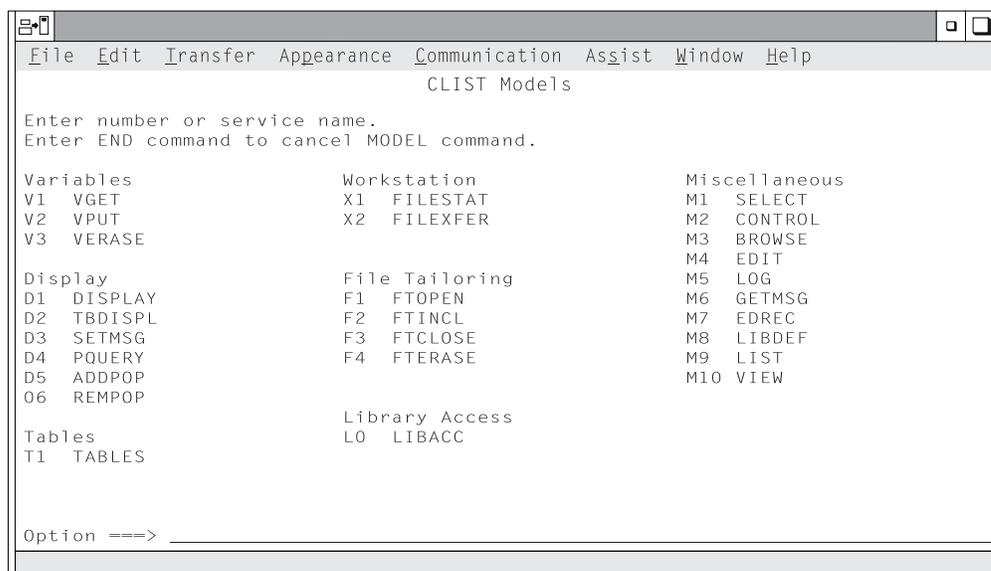


Figure 25. CLIST Models Panel (ISREMCMD)

If you select option D1 (DISPLAY), the editor inserts the model for the DISPLAY service in your CLIST at the location you specify with a label or an A or B line command. Notes are identified by the characters =NOTE= in the line command area (Figure 26 on page 80).

How to Use Edit Models

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT USERID.PRIVATE.CLIST(SCREEN) - 01.01 Columns 00001 00072
***** ***** Top of Data *****
000001
000002 ISPEXEC DISPLAY PANEL(PANELNAM) MSG(MSG-ID) +
000003 CURSOR(FIELDNAM) CSRPOS(POS#) +
000004 COMMAND(COMMANDS) RETBUFFR(BUF-NAME) +
000005 RETLGTH(LNG-NAME) MSGLOC(MSG-FIELD)
=NOTE=
=NOTE= PANELNAM - OPTIONAL, NAME OF THE PANEL TO BE DISPLAYED.
=NOTE= MSG-ID - OPTIONAL, IDENTIFIER OF A MESSAGE TO BE DISPLAYED ON
=NOTE= THE PANEL.
=NOTE= FIELDNAM - OPTIONAL, NAME OF THE FIELD WHERE THE CURSOR IS TO BE
=NOTE= POSITIONED.
=NOTE= POS# - OPTIONAL, POSITION OF CURSOR IN FIELD. DEFAULT IS 1.
=NOTE= COMMANDS - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE CHAIN
=NOTE= OF COMMANDS.
=NOTE= BUF-NAME - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE
=NOTE= REMAINING PORTION OF THE COMMAND CHAIN TO BE STORED
=NOTE= IF AN ERROR OCCURS.
Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PANELS(DEMO1) - 01.01 ----- Columns 00009 00080
***** ***** Top of Data *****
=NOTE=
=NOTE= EXAMPLE: ISPEXEC DISPLAY PANEL(PANEL1) MSG(MSG101) CURSOR(FLD1)
=NOTE=
000050 IF &LASTCC ^= 0 THEN /* RETURN CODES */ +
000060 DO /* 4 - COMMAND NOT FOUND */ +
000070 END /* 8 - END OR RETURN COMMAND USED */ +
000080 ELSE /* - PANEL WAS GENERATED FROM TAGS */ +
000090 /* AND EXIT COMMAND USED */ +
000100 /* 12 - PANEL, MESSAGE, OR CURSOR FIELD */ +
000110 /* NOT FOUND */ +
000120 /* 16 - DATA TRUNCATION OR TRANSLATION */ +
000130 /* ERROR */ +
000140 /* 20 - SEVERE ERROR */ +
***** ***** Bottom of Data *****
Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 26. DISPLAY Service Model

With the notes as a guide, you can edit the CLIST to change the DISPLAY service call parameters for your function. The error-handling source code shown serves as a skeleton which you can update. Finally, use RESET to eliminate the notes from the panel, leaving the service call, the error-handling logic, and the comments. Some models also include examples in NOTE lines. Use the MD line command to turn NOTE lines into data lines.

Adding, Finding, Changing, and Deleting Models

Models are implemented in a general fashion, so your installation can apply and use the concept for other tasks besides dialog development. You can create a set of PL/I call models for your IMS applications, or a set of report format models for your sales forecasting application. You can also create models for the JCL statements that you use most frequently.

Similarly, you may find that the models provided for panel formats do not correspond to the standards for your local installation or for your particular application. You can change the distributed panel models to match your own requirements.

This section describes how you can add a new model to your skeleton library, change an existing model, or delete an existing model.

Adding Models

To create a new model, you must:

1. Determine the data set name and member name for the model. For actual use, the model must be in a skeleton library.
2. Create the source code for the model. Consider whether you should create all new source code or whether you should change an existing model under a new name.

When you create a COBOL model, make sure number mode is on. Then, when you save the model, turn number mode off.

3. Make the model accessible from a model selection panel by having its selection call the program ISRECMBR with the actual model member name as its parameter. This involves:
 - Changing an existing model selection panel to add the new panel.
 - Creating a new model selection panel. If you do this, you must add the new panel to the hierarchy of selection panels by changing one of the higher-level panels.
 - No change, if you are replacing an existing model with an updated model with the same name.

As an example of adding a model, assume that you want to create a model for multiple-line block letters. Since you intend to use these block letters on panels, the model becomes part of the panel model class.

To build a model block letter, use the editor to create a new member in your skeleton library. For this example, the member name is BLKI. By manipulating input, you can develop the letter I (Figure 27 on page 82).

Adding, Finding, Changing, and Deleting Models

```
      IIIIIIIII
        II
        II
        II
        II
        II
      IIIIIIIII
)N
)N  the letter I for logo
```

Figure 27. Sample Block Letter Model

Once the model for each letter is built, you must update the selection panel in the prompting sequence that deals with panel model selection. Figure 28 shows the displayed form of this panel, panel ISREMPNL in the system panel library.

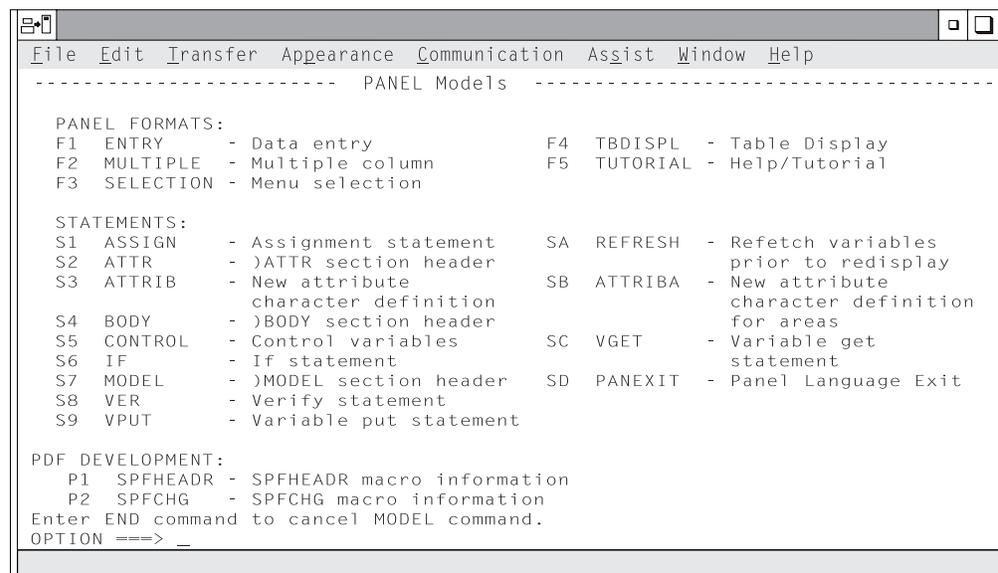


Figure 28. Panel Models Panel (ISREMPNL)

Copy the panel shown in Figure 28 into your panel data set and change it by adding a format F1, BLOCKLTR. See Figure 29 on page 83 for an example.

Adding, Finding, Changing, and Deleting Models

```

----- PANEL MODELS -----
STATEMENTS:
S1 ASSIGN - Assignment statement      S12 VGET - Variable get statement
S2 ATTR - )ATTR section header        S13 PANEXIT - Panel Language Exit
S3 ATTRIB - New attribute
      character definition            S14 ABC - Action bars
S4 BODY - )BODY section header        S15 KEYLIST - Keylist specification
S5 CONTROL - Control variables        S16 PDC - Action bar pull-down
S6 IF - If statement                 S17 VEDIT - Validate a variable
S7 MODEL - )MODEL section header     S18 CUAATTR - CUA attributes
S8 VER - Verify statement
S9 VPUT - Variable put statement
S10 REFRESH - Refetch variables
      prior to redisplay
S11 ATTRIBA - New attribute
      character definition
      for areas

Enter END command to cancel MODEL command.

Option ==> _____
F1=Help   F2=Split   F3=Exit   F9=Swap   F12=Cancel

```

Figure 29. Changed Panel Models Panel (ISREMPNL)

If there are several new models, this panel should be updated so that when you select F2, a new Block Letter selection panel is displayed. Therefore, you should change the)PROC section of panel ISREMPNL to include item F2. See Figure 30 for an example.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- XXXXXX.XXXXXXX.PANELS(ISREMPNL) - 21.12 ----- Columns 00001 00072
000030 REFRESH(ZCMD)
000031 )PROC
000032 IF (&ZCMD = 'SELECTION')
000033     &TMP = TRUNC (&ZCMD, '.')
000034     &ZCMD = TRUNC (&ZCMD, 8)
000035     &ZSEL = TRANS(TRUNC (&ZCMD, '.'))
000036     F1, 'PGM(ISRECMBR) PARM(ISREMMF1)'
000037     ENTRY, 'PGM(ISRECMBR) PARM(ISREMMF1)'
000038     F2, 'PANEL(BLKLTRS)' /* NEED TO QUALIFY THIS */
000039     MULTIPLE, 'PANEL(BLKLTRS)' /* PANEL FOR COLUMNS ID. */
000040     F3, 'PGM(ISRECMBR) PARM(ISREMMF3)'
000041     SELECT, 'PGM(ISRECMBR) PARM(ISREMMF3)' /* AUTOMATIC SINGLE COLUMN*/
000042     SELECTIO, 'PGM(ISRECMBR) PARM(ISREMMF3)' /* FOR 8 OR LESS SELECTION*/
000043     F4, 'PGM(ISRECMBR) PARM(ISREMMF4)'
000044     TDISPL, 'PGM(ISRECMBR) PARM(ISREMMF4)'
000045     F5, 'PGM(ISRECMBR) PARM(ISREMMF5)'
000046     TUTORIAL, 'PGM(ISRECMBR) PARM(ISREMMF5)'
Command ==> _____ Scroll ==> CSR
F1=Help   F2=Split   F3=Exit   F5=Rfind   F6=Rchange   F7=Up
F8=Down   F9=Swap    F10=Left  F11=Right  F12=Cancel

```

Figure 30. Changed)PROC Section of Panel Models Panel (ISREMPNL)

This concept allows you and other users to have sets of individual models, and allows the installation to have its own set of general models, without having multiple copies of the PDF model selection panels. For each model class, the installation could provide two additional entries on the selection panel: one for

Adding, Finding, Changing, and Deleting Models

installation-wide models and one for your models. Each entry could point to a selection panel, with each user having a copy of the selection panel to customize for individual use.

Note that the entry for F2, BLOCKLTR, points to a new panel, BLKLTRS, which you must now build.

You can change an existing panel model to create the new panel. Figure 31 on page 85 shows how the new panel might be typed. Note particularly the)INIT and)PROC sections of the coding. In the)PROC section of panel BLKLTRS, the target for all valid selections is the program ISRECMBR. The parameter passed to this program is different for each separate, but valid, selection and is the name of the model for that selection. Thus, for our example, the model name for selection 1 or I is BLKI.

You should follow the)INIT source code and the end source code in the)PROC section shown in Figure 31 on page 85 for all new panels.

```

)ATTR
)BODY
%----- BLOCK LETTER -----
%OPTION ==> _ZCMD +
%
% 1 +I          - Block letter I
% 2 +J          - Block letter J
% 3 +K          - Block letter K
%
%
+Enter%END+command to cancel MODEL command.+
%
)INIT
.CURSOR = ZCMD
.HELP = ISRxxxxx
IF (&ISRMSPL = 'RETURN ')
.RESP = END
)PROC
&ZSEL = TRANS(TRUNC (&ZCMD, '.'))
1, 'PGM(ISRECMBR) PARM(BLKI)'
1, 'PGM(ISRECMBR) PARM(BLKI)'
2, 'PGM(ISRECMBR) PARM(BLKJ)'
J, 'PGM(ISRECMBR) PARM(BLKJ)'
3, 'PGM(ISRECMBR) PARM(BLKK)'
K, 'PGM(ISRECMBR) PARM(BLKK)'
*, '?' )
IF (&ZSEL = '?')
.MSG = ISRYM012
&ISRMEND = 'N' /* SET THE END INDICATOR TO NO */
IF (.RESP = END) /* IF ENDING, WHY ... WHO CAUSED */
IF (&ISRMONCL = 'Y') /* MAKE SURE ITS NOT A CLASS OP. */
IF (&ISRMSPL = 'RETURN ') /* MAKE SURE ITS NOT END ON MBR. */
&ISRMEND = 'Y' /* NO - ITS BECAUSE USER HIT END */
)END

```

Figure 31. Source Code for Block Letter Model Selection Panel

Finding Models

Before you change or delete a model, you must determine the physical name of the model in the skeleton library. Refer to *ISPF Planning and Customizing* for a list of the names of the models of dialog elements distributed with PDF. In addition, you can use the following method to find the member name for any model.

You can find the member name for any model in the)PROC section of the final selection panel used to get it. The member name is the parameter passed to ISRECMBR, the program called when you choose that selection.

Adding, Finding, Changing, and Deleting Models

To determine the name of the model selection panel so that you can look at it to find the model member name, use the PANELID command when that panel is displayed. Then use the Browse or Edit options to look at the member of the panel library with that name.

Changing Models

To change a model that currently exists, copy the existing model from the skeleton data set into your own data set. Then use the editor to change the model in the same way you would change any text data set.

Note: Any lines that are to contain notes must have)N in positions 1 and 2, followed by one or more blanks, as shown in the following example.

```
VARIABLE = VALUE
)N      VARIABLE - A DIALOG VARIABLE OR A CONTROL VARIABLE.
)N      VALUE   - A LITERAL VALUE CONTAINING: SUBSTITUTABLE
)N                VARIABLES, A DIALOG VARIABLE, A CONTROL
)N                VARIABLE, OR AN EXPRESSION CONTAINING A
)N                BUILT-IN FUNCTION.
)N      EXAMPLES: &DEPT = 'Z59'   &A = &B   &C = ' '
```

When the model is later accessed using MODEL, the lines with)N indicators are flagged with =NOTE= in the line command area (Figure 26 on page 80).

Deleting Models

You can delete models by deleting the references to them. To delete the references, remove the entry referencing the model in both the)BODY and)PROC sections of the model selection panel.

Generally, you can leave the model itself in the skeleton library. However, if you are deleting a substantial number of models, you can delete those members from the library and then compress it.

Part 2. Edit Macros

Chapter 5. Using Edit Macros	89	Specifying NOPROCESS in the Macro Statement	116
What Are Edit Macros?	89	Specifying a Destination	117
Performing Repeated Tasks	89	Specifying a Range	117
Simplifying Complex Tasks	91	Example	118
Passing Parameters, and Retrieving and Returning Information	92	Recovery Macros	118
Chapter 6. Creating Edit Macros	95	Return Codes from User-Written Edit Macros	118
CLIST and REXX Edit Macros	95	Return Codes from PDF Edit Macro Commands	119
Edit Macro Commands and Assignment Statements	96	Selecting Control for Errors	120
Using the REXX ADDRESS Instruction	96	Chapter 7. Testing Edit Macros	121
Command Procedure Statements.	96	Handling Errors	121
ISPF and PDF Dialog Service Requests.	97	Edit Command Errors	121
TSO Commands	97	Dialog Service Errors	121
Program Macros	97	Using CLIST WRITE Statements and REXX SAY Statements	122
Differences between Program Macros, CLISTs, and REXX EXECs.	98	Using CLIST CONTROL and REXX TRACE Statements	123
Passing Parameters in a Program Macro	98	Experimenting with Macro Commands	124
Program Macro Examples	99	Chapter 8. Sample Edit Macros	127
Writing Program Macros	99	TEXT Macro	127
Running Program Macros	102	PFCAN Macro.	129
Using Commands in Edit Macros	103	BOX Macro	130
Naming Edit Macros.	103	IMBED Macro	132
Variables.	103	ALLMBRS Macro	135
Variable Substitution.	104	FINDCHGS Macro	138
Character Conversion	104	MASKDATA Macro	141
Edit Assignment Statements	104		
Value	104		
Keyphrase	105		
Overlays and Templates	106		
Using Edit Assignment Statements	106		
Passing Values.	107		
Manipulating Data With Edit Assignment Statements	107		
Differences Between Edit, CLIST, and REXX Assignment Statements	108		
Performing Line Command Functions	108		
Parameters	109		
Passing Parameters to a Macro	109		
Using Edit macros in Batch	111		
Edit Macro Messages	111		
Macro Levels	112		
Labels in Edit Macros	112		
Using Labels	112		
Referring to Labels	113		
Passing Labels.	114		
Referring to Data Lines.	114		
Referring to Column Positions	115		
Defining Macros	115		
Defining an Alias.	115		
Resetting Definitions.	115		
Replacing Built-In Commands	116		
Implicit Definitions	116		
Using the PROCESS Command and Operand	116		

Chapter 5. Using Edit Macros

This chapter documents general-use programming interfaces and associated guidance information.

This chapter describes edit macros and describes several examples of their use.

What Are Edit Macros?

You can use edit macros, which look like ordinary editor commands, to extend and customize the editor. You create an edit macro by placing a series of commands into a data set or member of a partitioned data set. Then you can run those commands as a single macro by typing the defined name in the command line.

Edit macros can be either CLISTs or REXX EXECs written in the CLIST or REXX command language, or program macros written in a programming language (such as FORTRAN, PL/I, or COBOL). This manual uses the CLIST command language for most of its examples, with a few examples in REXX. Examples of program macros are in “Program Macros” on page 97.

Edit macros can also contain edit assignment statements that communicate between a macro and the editor. These statements are made up of two parts, keyphrases and values, that are separated by an equal sign. Edit assignment statements are described in “Edit Assignment Statements” on page 104.

Edit macros have access to the dialog manager and system services. Because edit macros are CLISTs, or REXX EXECs, programs, they have unlimited possibilities.

Note: All edit macros must have an ISREDIT MACRO statement as the first edit command. For more information see “Macro Command Syntax” on page 362.

You can use edit macros to:

- Perform repeated tasks
- Simplify complex tasks
- Pass parameters
- Retrieve and return information.

The remainder of this chapter presents examples of these tasks.

Note: To run an edit macro against all members of a PDS you can use a program containing a loop that uses a LMMLIST service to obtain the names of PDS members. For each member issue a ISPEXEC edit command with the initial macro keyword. For an example, see Figure 59 on page 136.

Performing Repeated Tasks

You can use an edit macro to save keystrokes when you frequently perform a task. A simple example would be using a macro to delete every line that begins with a dash (-) in column 1. You could scan the data and manually delete each line, or you could write a macro that does the same thing much faster. The edit macro in

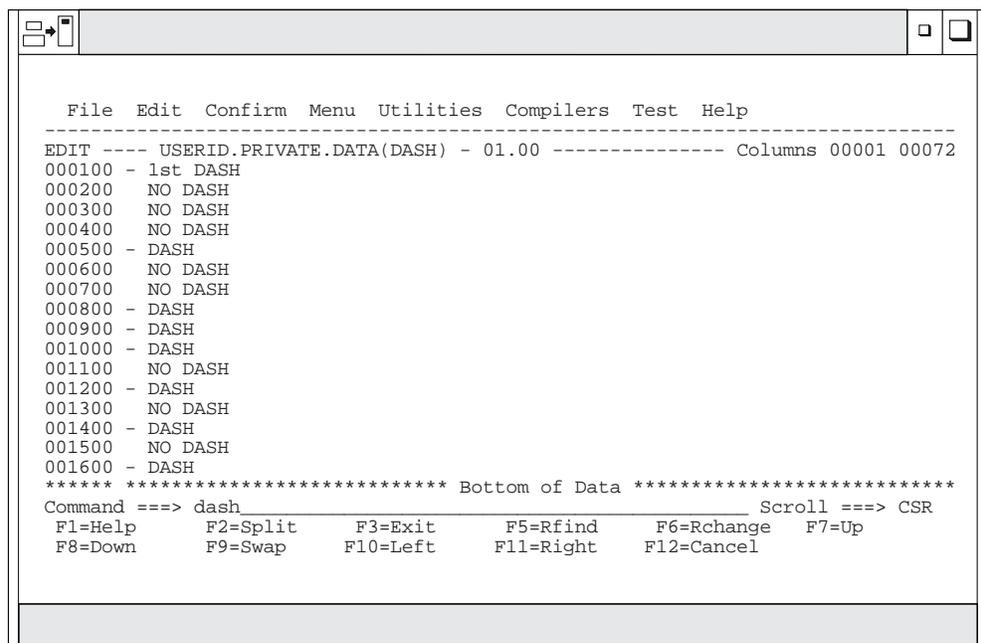
What Are Edit Macros?

Figure 32 processes the commands necessary to delete the lines and requires only that you enter the DASH macro.

```
/*
/* DASH MACRO - DELETE LINES WITH A '-' IN COLUMN 1
/*
/* EXCEPT FIRST '-'
/*
ISREDIT MACRO
ISREDIT RESET EXCLUDED /* Ensure no lines are excluded */
ISREDIT EXCLUDE ALL '-' 1 /* Exclude lines with '-' in col1 */
ISREDIT FIND FIRST '-' 1 /* Show the first such line */
ISREDIT DELETE ALL EXCLUDED /* Delete all lines left excluded */
EXIT CODE(0)
```

Figure 32. DASH Macro

When you run this macro, it deletes all lines beginning with a dash, except the first one. To run the macro, type dash on the Command line (Figure 33). The dash macro deletes all lines that began with a dash except the first one (Figure 34 on page 91).



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(DASH) - 01.00 ----- Columns 00001 00072
000100 - 1st DASH
000200 NO DASH
000300 NO DASH
000400 NO DASH
000500 - DASH
000600 NO DASH
000700 NO DASH
000800 - DASH
000900 - DASH
001000 - DASH
001100 NO DASH
001200 - DASH
001300 NO DASH
001400 - DASH
001500 NO DASH
001600 - DASH
***** ***** Bottom of Data *****
Command ==> dash
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 33. DASH Macro - Before Running

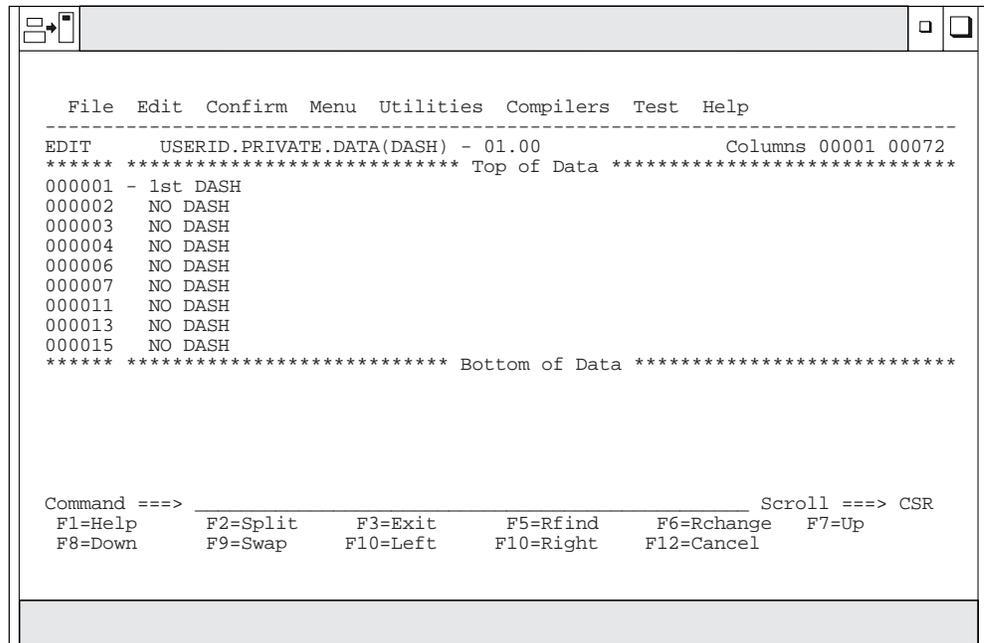


Figure 34. DASH Macro - After Running

Simplifying Complex Tasks

If you need to perform an involved task, you can include logic in your edit macro. For instance, the TESTDATA macro shown in Figure 35 creates variations of the same line by first finding the succeeding test string number, and then changing each occurrence, using ascending numbers one through nine.

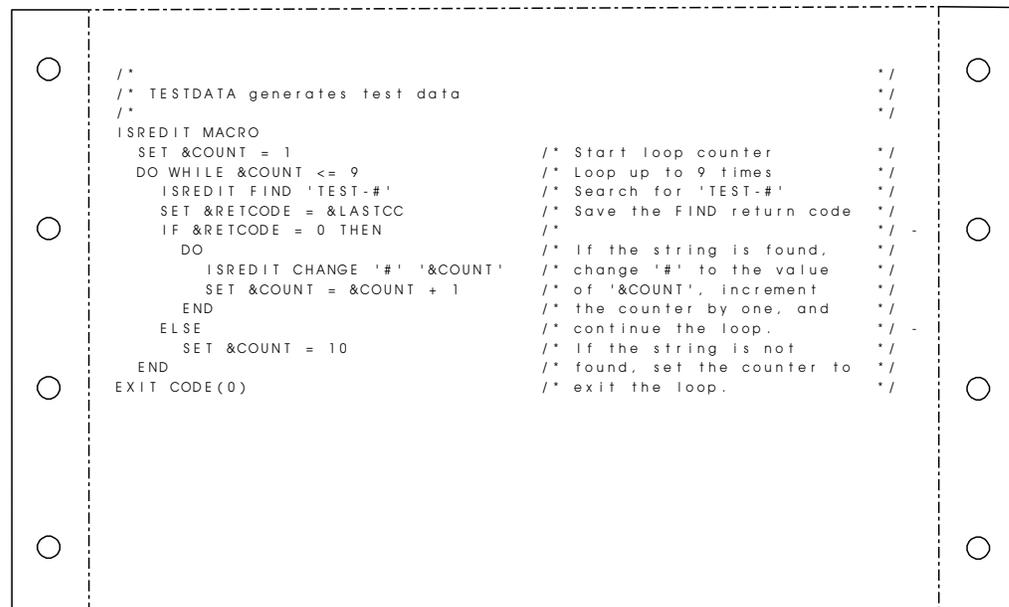
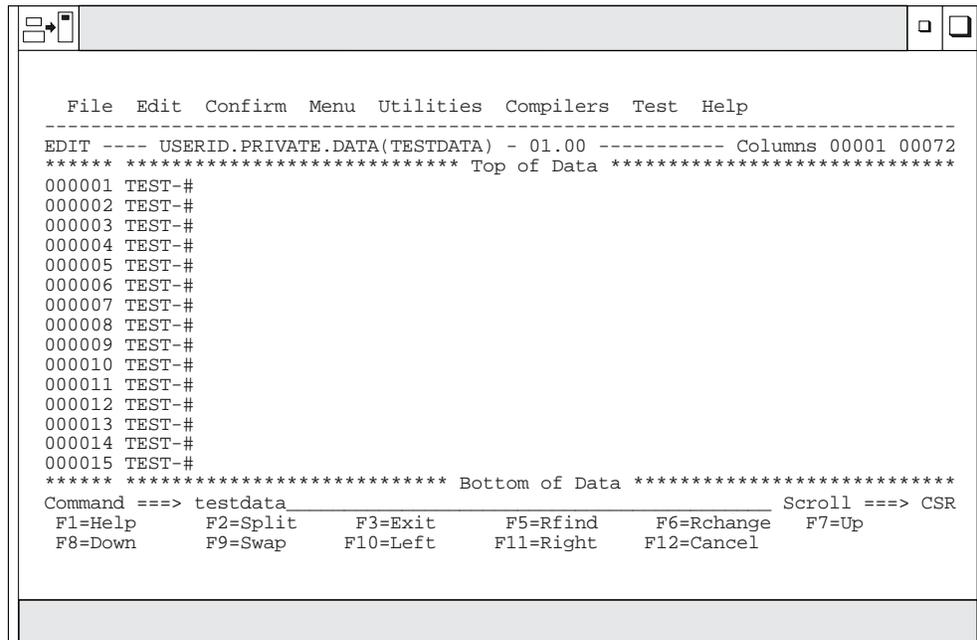


Figure 35. TESTDATA Macro

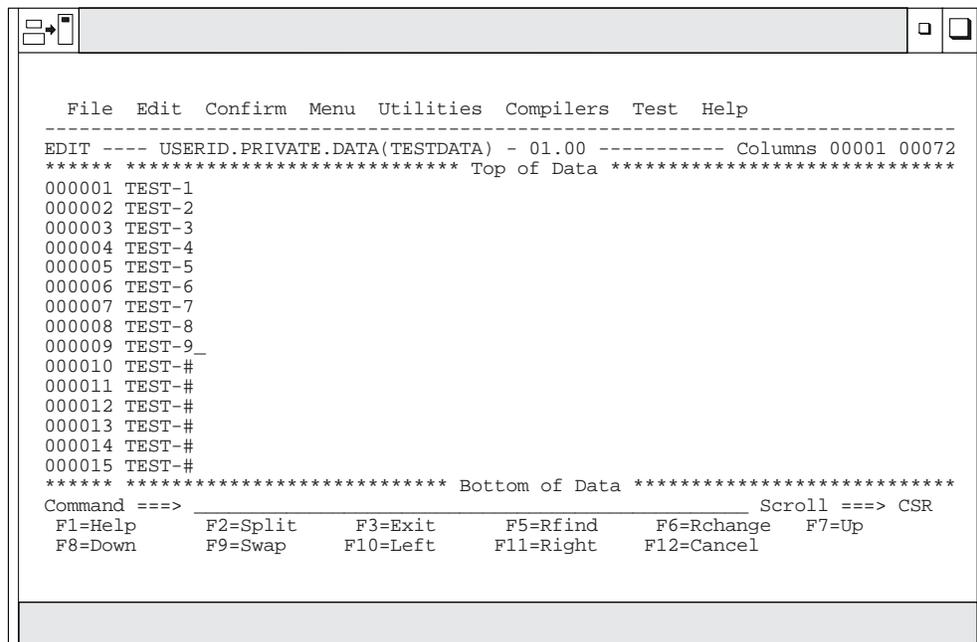
What Are Edit Macros?

To run the test macro, type testdata on the Command line (Figure 36). The macro numbers the first nine lines of data (Figure 37).



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-#
000002 TEST-#
000003 TEST-#
000004 TEST-#
000005 TEST-#
000006 TEST-#
000007 TEST-#
000008 TEST-#
000009 TEST-#
000010 TEST-#
000011 TEST-#
000012 TEST-#
000013 TEST-#
000014 TEST-#
000015 TEST-#
***** ***** Bottom of Data *****
Command ==> testdata                               Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right    F12=Cancel
```

Figure 36. TESTDATA Macro - Before Running



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-1
000002 TEST-2
000003 TEST-3
000004 TEST-4
000005 TEST-5
000006 TEST-6
000007 TEST-7
000008 TEST-8
000009 TEST-9_
000010 TEST-#
000011 TEST-#
000012 TEST-#
000013 TEST-#
000014 TEST-#
000015 TEST-#
***** ***** Bottom of Data *****
Command ==>                               Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right    F12=Cancel
```

Figure 37. TESTDATA Macro - After Running

Passing Parameters, and Retrieving and Returning Information

You can also write macros to get information from other users and from the editor, and to display messages to other users. The COUNTSTR macro, as shown in

Figure 38, finds occurrences of the string *TEST* from the previous example, counts them, and prepares a return message.

```

/*
/* COUNTSTR counts the number of occurrences
/* of a string, and returns a message
/*
/*
ISREDIT MACRO (PARMSTR)
  ISREDIT SEEK ALL &PARMSTR
  IF &LASTCC > 12 THEN DO
    SET &ZEDSMMSG = &STR(SEEK ERROR)
    SET &ZEDLMSG = &STR(STRING NOT FOUND)
  END
  ELSE DO
    ISREDIT (COUNT) = SEEK_COUNTS
    SET &COUNT = &COUNT
    SET &ZEDSMMSG = &STR("&PARMSTR" FOUND &COUNT TIMES)
    SET &ZEDLMSG = &STR("THE STRING "&PARMSTR" WAS FOUND +
                        &COUNT TIMES.)
  END
  ISPEXEC SETMSG MSG(ISRZ000)
  EXIT CODE(0)

```

Figure 38. COUNTSTR Macro

To run the COUNTSTR macro, type `countstr TEST` on the Command line (Figure 39). The macro does not change the data but displays return messages to show the number of times it found the string. The editor always displays the short message in the upper right-hand corner of the screen. Enter `HELP` (the default is `F1`) to produce the long message (Figure 40 on page 94).

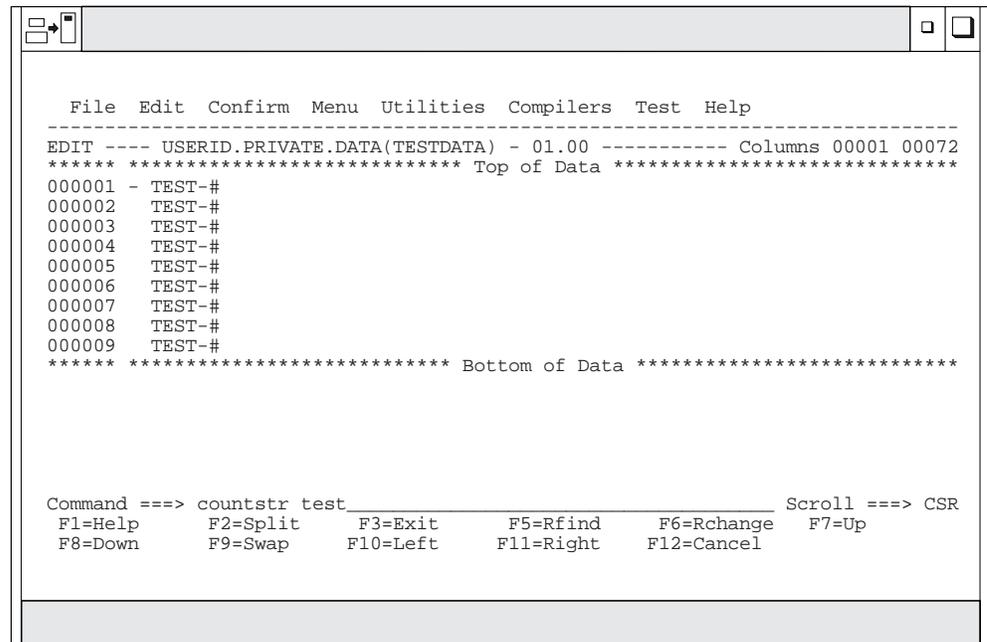
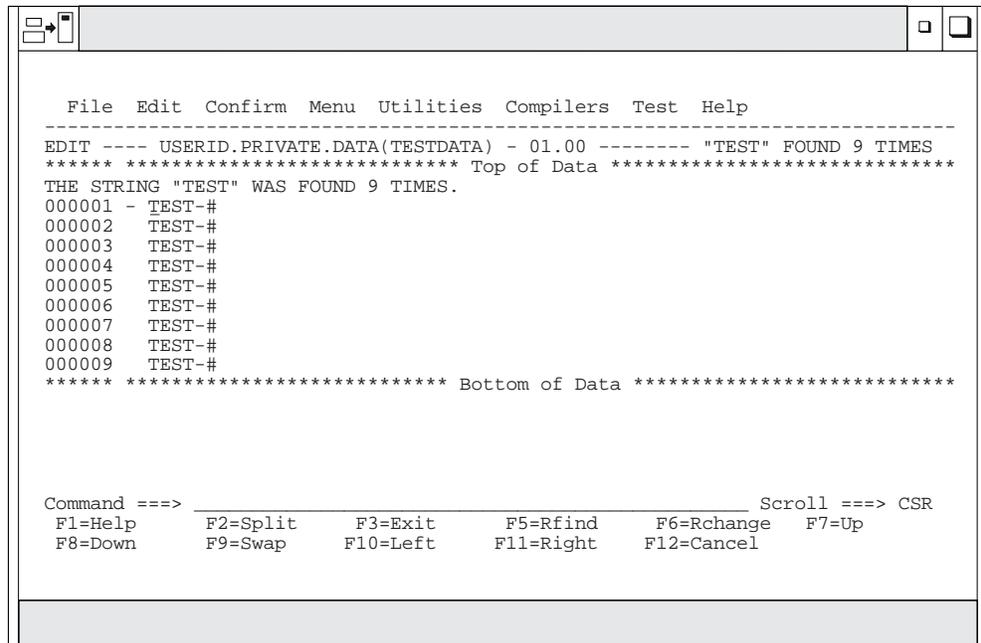


Figure 39. COUNTSTR Macro - Before Running

What Are Edit Macros?



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.00 ----- "TEST" FOUND 9 TIMES
***** ***** Top of Data *****
THE STRING "TEST" WAS FOUND 9 TIMES.
000001 - TEST-#
000002 TEST-#
000003 TEST-#
000004 TEST-#
000005 TEST-#
000006 TEST-#
000007 TEST-#
000008 TEST-#
000009 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 40. COUNTSTR Macro - After Running

Chapter 6. Creating Edit Macros

This chapter documents general-use programming interfaces and associated guidance information.

Edit macros are ISPF dialogs that run in the ISPF editor environment.

CLIST edit macros must be in partitioned data sets in at least one of the following concatenations: SYSUPROC, ALTLIB (for data sets activated as CLISTs), or SYSPROC. Data sets in these concatenations can contain either CLIST edit macros, REXX edit macros, or a combination of the two. However, REXX edit macros in these concatenations must include a REXX comment line (*/* REXX */*) as the first line of each edit macro to distinguish them from CLIST edit macros. This comment line can contain other words or characters if necessary, but it must include the string REXX.

Note: For more information about the ALTLIB concatenation, refer to *TSO Extensions Version 2 Command Reference*

REXX edit macros must also be in partitioned data sets. Besides the concatenations in the previous list for CLIST edit macros, REXX edit macros can exist in the following concatenations: SYSUEXEC, ALTLIB (for data sets activated as EXECs), and SYSEXEC. Data sets in these concatenations can contain only REXX EXECs.

For example, if an application activates an application-level library with the following commands:

```
ALTLIB ACTIVATE APPLICATION(EXEC) DA(DS1 DS2 DS3)
ALTLIB ACTIVATE APPLICATION(CLIST) DA(DSA DSB DSC)
```

then data sets DS1, DS2, and DS3 must contain only REXX EXECs. However, data sets DSA, DSB, and DSC can contain either REXX EXECs or CLISTs; if these data sets contain REXX EXECs, the first line of each EXEC must be a REXX comment line.

As in an ISPF dialog, program macros must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library.

CLIST and REXX Edit Macros

A CLIST edit macro is made up of CLIST statements and a REXX edit macro is made up of REXX statements. Each statement falls into one of the following categories:

- Edit macro commands
- CLIST or REXX command procedure statements and comments
- ISPF and PDF dialog service requests
- TSO commands.

All statements are initially processed by the TSO command processor, which scans them and does symbolic variable substitution. It is important to recognize the different kinds of CLIST and REXX statements listed because:

- They are processed by different components of the system.
- They have different syntax rules and error handling.

CLIST and REXX Edit Macros

- Their descriptions are in different manuals.

Edit Macro Commands and Assignment Statements

Any statement in an edit macro that begins with ISREDIT is assumed to be an edit macro command or assignment statement. When such a statement is found, the CLIST or REXX command processor does symbolic substitution and then passes it to the editor. The editor processes it, performing any requested functions. Examples of two edit macro commands are:

CLIST Statements

```
ISREDIT FIND "TEST475"  
ISREDIT PROCESS
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT FIND TEST475'  
'ISREDIT PROCESS'
```

Examples of two edit macro assignment statements are:

CLIST Statements

```
ISREDIT BOUNDS = 1,60  
ISREDIT (WIDTH) = LRECL
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT BOUNDS = 1,60'  
'ISREDIT (WIDTH) = LRECL'
```

A description of each edit macro command and assignment statement is in Chapter 11. Edit Macro Commands and Assignment Statements.

Using the REXX ADDRESS Instruction

If you have several edit macro commands within a REXX EXEC, you can change the command environment to the PDF editor with the instruction ADDRESS ISREDIT. All subsequent commands in the EXEC are passed directly to the editor. The following examples show how you can pass the same edit macro commands using different environments:

ISPEXEC Environment

```
ADDRESS ISPEXEC  
'ISREDIT BOUNDS = 1,60'  
'ISREDIT (WIDTH) = LRECL'
```

ISREDIT Environment

```
ADDRESS ISREDIT  
'BOUNDS = 1,60'  
'(WIDTH) = LRECL'
```

For information on using the REXX ADDRESS instruction, refer to *TSO/E Version 2 REXX Reference*

Command Procedure Statements

Command procedure statements handle CLIST and REXX variables and control flow within a CLIST or REXX EXEC. When a command procedure statement is found, it is processed by the TSO command processor. Some of the command procedure statements commonly seen in PDF edit macros are:

- Assignment statements
- IF-THEN-ELSE statements
- DO-WHILE-END statements
- EXIT statements.

For a complete list and description of command procedure statements for CLIST and REXX, refer to *TSO Extensions CLISTs TSO/E Version 2 REXX Reference* and *TSO/E Version 2 REXX User's Guide*

ISPF and PDF Dialog Service Requests

Any statement in an edit macro beginning with ISPEXEC is assumed to be an ISPF or PDF component dialog service request. When such a statement is found, the TSO command processor does symbolic substitution. It then passes the command to the appropriate ISPF or PDF component service to be processed. Some examples of service requests that might be in a PDF component edit macro are:

CLIST Statements

```
ISPEXEC SETMSG ...
ISPEXEC VPUT ...
ISPEXEC DISPLAY ...
ISPEXEC EDIT ...
ISPEXEC LMINIT ...
```

REXX Statements

```
ADDRESS ISPEXEC
'SETMSG ...'
'VPUT ...'
'DISPLAY ...'
'EDIT ...'
'LMINIT ...'
```

For more information on ISPF services, refer to *ISPF Services Guide*. For more information on PDF services, refer to *ISPF Examples*.

TSO Commands

Any statement that is not recognized as a command procedure statement and does not begin with ISPEXEC or ISREDIT is assumed to be a TSO command. TSO commands can be either CLISTs, REXX EXECs, or programs. When the command processor finds a TSO command, it processes the command. Examples of TSO commands are:

CLIST Statements

```
ALLOCATE ...
FREE ...
DELETE ...
RENAME ...
```

REXX Statements

```
ADDRESS TSO
'ALLOCATE ...'
'FREE ...'
'DELETE ...'
'RENAME ...'
```

For more information on TSO commands, refer to *TSO Extensions Command Language Reference*.

Program Macros

Besides writing edit macros as CLISTs and REXX EXECs, you can also write edit macros in programming languages, just as you write program dialogs. These are called *program macros*.

PDF accepts all languages supported by ISPF. Refer to *ISPF Dialog Developer's Guide and Reference* or *ISPF Examples* for more information.

There are four basic reasons to write and debug a program macro:

- A macro runs faster in a language that can be precompiled than in the CLIST or REXX interpretive languages. This can be valuable for macros that you run many times.
- A macro that has to deal with data containing symbols can confuse an interpretive language processor. Particularly, ampersands in the data can cause problems.
- A macro that has complex logic can be handled better in a programming language.

Program Macros

- To pass mixed data or strings (those that contain both EBCDIC and DBCS characters) as parameters, you must use a program macro. Although CLIST does not allow mixed data strings, there are edit macro commands and assignment statements that allow you to supply data or string operands. The edit macro commands and assignment statements that allow you to supply data or string operands are:

CHANGE	LINE	MASKLINE
EXCLUDE	LINE_AFTER	SEEK
FIND	LINE_BEFORE	TABSLINE

Differences between Program Macros, CLISTs, and REXX EXECs

Program macros have special characteristics that you should consider before coding:

- Variables are not self-defining in program macros, as they are in CLISTs and REXX EXECs. The VDEFINE, VCOPY, and VREPLACE dialog services must be called to identify variables looked at or set by the program.
- If you write a REXX EXEC or a program macro that accepts parameter input, the macro must be aware that the input may be in lowercase. Variable values are automatically converted to uppercase by the CLIST processor.
- Program macros are not implicitly defined, while CLIST and REXX macros are. When you use a command name that is not a built-in or previously-defined primary command, the editor searches the SYSUEXEC, SYSUPROC, ALTLIB, SYSEXEC, and SYSPROC concatenations (for CLISTs and REXX EXECs) for a member with the same name. If it exists, it is assumed to be a macro.

No automatic search is done for program macros. Therefore, there are two ways to tell the editor to run a macro as a program macro. You can precede the name with an exclamation point (!) if it is less than 8 characters, or you can use the DEFINE command to define the name as a program macro. Program macros are treated as ISPF dialogs, and must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library.

- Program macros can run without being verified as macros; the MACRO statement can follow calls to dialog services.
- The editor scans edit statements within program macros to do variable substitution similar to the CLIST processor. Only one level of substitution is done. This is the default; use the SCAN assignment statement to prevent it.

Passing Parameters in a Program Macro

Program macros process edit commands by using the ISPLINK or ISPEXEC interface. ISPLNK and ISPEX are the interface names used in FORTRAN and Pascal programs. Parameters are passed to the ISREDIT service as follows:

```
CALL ISPLINK ('ISREDIT',length,buffer)
```

```
CALL ISPEXEC (length,'ISREDIT command')
```

where the following definitions apply:

'ISREDIT'

The service name.

length A fullword number indicating the length of the command buffer. When a zero length is passed, the maximum buffer length is 255 bytes.

buffer Can contain any edit command that is valid from a macro, typed with the same syntax used in a CLIST or REXX EXEC.

command

Any PDF edit command that is valid from a macro, typed with the same syntax used in a CLIST or REXX EXEC.

Program Macro Examples

The following examples show three different methods of coding a FIND command for a program macro. They are typed using PL/I syntax:

```
CALL ISPLINK ('ISREDIT',LEN0,'¢FIND XYZ¢')
CALL ISPLINK ('ISREDIT',LEN8,'FIND XYZ')
CALL ISPEXEC (LEN16,'ISREDIT FIND XYZ')
```

where:

LEN0 A fullword program variable with a value of 0.

LEN8 A fullword program variable with a value of 8.

LEN16
A fullword program variable with a value of 16.

In each of the previous examples, the remainder of the command is typed as a literal value.

The first two examples use the ISPLINK syntax. In the ISPLINK call, ISREDIT is passed as the first parameter and is omitted from the command buffer.

The first example uses a special interface. A zero length can be passed, but only when the command is delimited by a special character. A special character cannot be an alphanumeric character. If the length is zero and if a valid delimiter is the first character in the command buffer, a scan of the command is done to find the next occurrence of that character. The command length is the number of characters between the two delimiters. Here, the cent sign (¢) is used as a delimiter. When a zero length is passed, the maximum buffer length is 255 bytes.

In the second example, an explicit length of 8 is used and the command buffer contains the command without delimiters.

The third example uses the ISPEXEC syntax. This syntax always requires the length of the command buffer to be passed. The command buffer includes the ISREDIT prefix, and is typed the same way as a CLIST or REXX command.

Writing Program Macros

When you write a program macro, it can help to first type it as a CLIST or REXX macro to debug the logic and the command statements. The example that follows is called SEPLINE, a simple macro that separates each line in a set of data with a line of dashes. The REXX syntax is shown in Figure 41 on page 100, the PL/I program is shown in Figure 42 on page 101, and the COBOL program is shown in

Program Macros

Figure 43 on page 102. Notice that a VDEFINE is not required for the variable SAVE, which is referenced only by the ISPF editor.

```
/* REXX                                     */
/*                                           */
/* SEPLINE separates lines with a line of dashes. */
/*                                           */
TRACE
ADDRESS ISPEXEC
'ISREDIT MACRO'

'ISREDIT (SAVE) = USER_STATE'
'ISREDIT RESET'
'ISREDIT EXCLUDE ----- 1 ALL'
'ISREDIT DELETE ALL X'
LASTL = 1
LINE = 0
LINX = COPIES('-',70)

LL = LASTL + 1
DO WHILE LINE < LL
'ISREDIT LINE_AFTER 'LINE' = (LINX)'
'ISREDIT (LASTL) = LINENUM .ZLAST'
LL = LASTL + 1
LINE = LINE + 2
END
'ISREDIT USER_STATE = (SAVE)'
EXIT
```

Figure 41. SEPLINE REXX Macro

```

/*
/* SEPLINE - EDIT MACRO PROGRAM TO INSERT SEPARATOR LINES
/*
SEPLINE: PROC OPTIONS(MAIN);
/*
DECLARE
  LINEX CHAR (70) INIT ((70)'-'),      /* SEPARATOR LINE --- */
  LASTL FIXED BIN(31,0) INIT (0),      /* LAST LINE OF TEXT */
  LINE  FIXED BIN(31,0) INIT (0),      /* CURRENT LINE NUMBER */
  LENO  FIXED BIN(31,0) INIT (0),      /* LENGTHS - 0      */
  LEN1  FIXED BIN(31,0) INIT (1),      /* LENGTHS - 1      */
  LEN4  FIXED BIN(31,0) INIT (4),      /* LENGTHS - 4      */
  LEN70 FIXED BIN(31,0) INIT (70);     /* LENGTHS - 70     */
/*
DECLARE
  ISPLINK ENTRY OPTIONS(ASM,INTER,RETCODE); /* LINK TO ISPF
/*
CALL ISPLINK ('VDEFINE','(LASTL)',LASTL,'FIXED',LEN4);
CALL ISPLINK ('VDEFINE','(LINE)',LINE,'FIXED',LEN4);
CALL ISPLINK ('VDEFINE','(LINEX)',LINEX,'CHAR',LEN70);

CALL ISPLINK('ISREDIT',LENO,'% MACRO %');
CALL ISPLINK('ISREDIT',LENO,'% (SAVE) = USER_STATE %');
CALL ISPLINK('ISREDIT',LENO,'% RESET %');
CALL ISPLINK('ISREDIT',LENO,'% EXCLUDE ----- 1 ALL %');
CALL ISPLINK('ISREDIT',LENO,'% DELETE ALL X %');

LASTL = 1;
LINE = 0;

DO WHILE (LINE < (LASTL + 1));
  CALL ISPLINK('ISREDIT',LENO,'% LINE_AFTER &LINE = (LINEX) % ');
  CALL ISPLINK('ISREDIT',LENO,'% (LASTL) = LINENUM .ZLAST %');
  LINE = LINE + 2;
END;

  CALL ISPLINK('ISREDIT',LENO,'% USER_STATE = (SAVE) %');
END SEPLINE;

```

Figure 42. SEPLINE PL/I Macro

Program Macros

```
ID DIVISION.
PROGRAM - ID. SEPLINE.
*
*           EDIT MACRO PROGRAM TO INSERT SEPARATOR LINES
*
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING - STORAGE SECTION.

01 LINEX   PIC X(70) VALUE ALL "-".
* SEPARATOR LINE -----
01 LASTL   PIC 9(6) VALUE 0 COMP.
* LAST LINE OF TEXT
01 LYNE    PIC 9(6) VALUE 0 COMP.
* CURRENT LINE NUMBER

01 ISREDIT PIC X(8) VALUE "ISREDIT ".
01 VDEFINE PIC X(8) VALUE "VDEFINE ".
01 ZLASTL  PIC X(8) VALUE "(LASTL)".
01 ZLINE   PIC X(8) VALUE "(LINE)".
01 ZLINEX  PIC X(8) VALUE "(LINEX)".
01 FIXED   PIC X(8) VALUE "FIXED ".
01 CHAR    PIC X(8) VALUE "CHAR ".
01 LENO    PIC 9(6) VALUE 0 COMP.
01 LEN4    PIC 9(6) VALUE 4 COMP.
01 LEN70   PIC 9(6) VALUE 70 COMP.

01 EM1     PIC X(10) VALUE "¢ MACRO ¢".
01 EM2     PIC X(24) VALUE "¢ (SAVE) = USER_STATE ¢".
01 EM3     PIC X(10) VALUE "¢ RESET ¢".
01 EM4     PIC X(25) VALUE "¢ EXCLUDE -----1 ALL 0".
01 EM5     PIC X(18) VALUE "¢ DELETE ALL X ¢".
01 EM6     PIC X(30) VALUE "¢ LINE_AFTER &LINE = (LINEX) ¢".
01 EM7     PIC X(28) VALUE "¢ (LASTL) = LINENUM .LAST ¢".
01 EM8     PIC X(23) VALUE "¢ USER_STATE = (SAVE) ¢".

PROCEDURE DIVISION.
CALL "ISPLINK" USING VDEFINE ZLASTL LASTL FIXED LEN4.
CALL "ISPLINK" USING VDEFINE ZLINE LYNE FIXED LEN4.
CALL "ISPLINK" USING VDEFINE ZLINEX LINEX CHAR LEN70.
CALL "ISPLINK" USING ISREDIT LENO EM1.
CALL "ISPLINK" USING ISREDIT LENO EM2.
CALL "ISPLINK" USING ISREDIT LENO EM3.
CALL "ISPLINK" USING ISREDIT LENO EM4.
CALL "ISPLINK" USING ISREDIT LENO EM5.

MOVE 1 TO LASTL.
MOVE 0 TO LYNE.
PERFORM LOOP UNTIL LYNE IS NOT LESS THAN (LASTL + 1).
CALL "ISPLINK" USING ISREDIT LENO EM8.
GOBACK.

LOOP.
CALL "ISPLINK" USING ISREDIT LENO EM6.
CALL "ISPLINK" USING ISREDIT LENO EM7.
ADD 2 TO LYNE.
```

Figure 43. SEPLINE COBOL Macro

Running Program Macros

The ISPF editor assumes that any unknown primary command is a macro, and it also assumes that the macro has been implemented as a CLIST or REXX EXEC. You can define a macro as a program macro either by entering a DEFINE command or by prefixing the macro name with an exclamation point (!) when you type the macro name on the Command line.

If a macro named FINDIT is a CLIST or REXX EXEC macro, for example, you can run it by typing FINDIT on the Command line and pressing Enter. If it is a program macro, you can type !FINDIT, or FINDIT if it had previously been defined as a

program macro by the DEFINE command. The first time you enter a macro with an exclamation point (!) prefix implicitly defines that macro as a program macro. Thereafter, you can omit the prefix.

To use the DEFINE command to define a program as a macro, type:

```
Command ==> DEFINE name PGM MACRO
```

and press Enter. The operands can be typed in either order. The following, for example, is also valid:

```
Command ==> DEFINE name MACRO PGM
```

Using Commands in Edit Macros

You can use most primary commands in an edit macro if you precede it with ISREDIT. Table 6 on page 300 shows the macro commands available to use. There are differences, though, between entering a command on the Command line and processing the same command in a macro as one of a series:

- When you enter a command on the Command line, the result of the command is displayed in either an informational or an error message. If you process the same command in a macro, messages are not displayed, and the lines actually displayed may be different from a command entered on the Command line.
- When you issue a series of commands as a macro, the display does not change with each command. The lines displayed are the end result of the macro running, not the individual commands.
- Some commands have additional operands permitted in a macro that cannot be used interactively.

Besides these differences, there are certain guidelines to remember when creating edit macros. The following topics apply to CLIST, REXX, and program macros.

Naming Edit Macros

Edit macro names can be any valid CLIST, REXX, or program name. Using the DEFINE ALIAS command, you can assign command names for running the edit macros that are different from the actual name.

When choosing names and aliases, avoid defining names that might conflict with the DEFINE command operands and their abbreviations. You can do this by implicitly defining the macros: precede program macros with an exclamation point (!); do not use explicit definitions for CLIST or REXX macros.

Variables

Variables function in edit macros the same way they do in CLISTs and REXX EXECs. The only exceptions are dialog variables—variables that communicate with ISPF and the PDF component—which can only have names from 1 to 8 characters in length. The following presents a brief introduction on using variables; for more detailed information on variables in CLISTs, refer to *TSO Extensions CLISTs*. For information on variables in REXX EXECs, refer to *TSO/E Version 2 REXX Reference* and *TSO/E Version 2 REXX User's Guide*.

When coding macros in CLIST or REXX, remember that all ISREDIT statements are processed for variable substitution before the editor sees the statements. Enclose the variables in parentheses when variable substitution should not occur, such as

Using Commands in Edit Macros

in cases when ISREDIT statements expect a variable name and not its value. For CLIST variables, omit the ampersand; for REXX variables, use quotes.

Variable Substitution

Scan mode controls the automatic replacement of variables in command lines passed to the editor. Use the SCAN assignment statement either to set the current value of scan mode (for variable substitution), or to retrieve the current value of scan mode and place it in a variable.

When scan mode is on, command lines are scanned for ampersands (&). If an ampersand followed by a non-blank character is found, the name following the ampersand (ended by a blank or period) is assumed to be a dialog variable name, such as '&NAME'. or '&NAME'; the value from the variable pool is substituted in the command for the variable name before the command is processed. The period after the variable allows concatenation of the variable value without an intervening blank delimiter. Remember this when using program macros that do not have the CLIST processor to substitute variable values.

Character Conversion

A CLIST automatically converts all character strings to uppercase before passing them to the editor. Therefore, if you want an edit macro command or assignment statement that you process from a CLIST to find a character string in lowercase, you must precede the command or statement with the TSO CONTROL ASIS statement. This statement passes lowercase characters to the editor.

Edit Assignment Statements

You use edit assignment statements to communicate between macros and the editor. An assignment statement consists of two parts, *values* and *keyphrases*, which are separated by an equal sign. The value segment represents data that is in the macro, and the keyphrase segment represents data in the editor. You can use assignment statements to pass data from the edit macro to the editor, or to transfer data from the editor to the edit macro.

Data is always transferred from the right-hand side of the equal sign in an assignment statement to the left side. Therefore, if the keyphrase is on the right, data known to the editor is put into CLIST or REXX variables on the left. In this situation, the yyy would be a keyphrase, and the xxx would be the value.

CLIST Statement

```
ISREDIT xxx = yyy
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT xxx = yyy'
```

Value

The value part of an edit macro assignment statement can be one of the following:

- A *literal* character string can be one of the following:

Simple string

Any series of characters not enclosed within quotes (either ' or "), parentheses, or less-than (<) and greater-than signs (>), and not containing any embedded blanks or commas.

Delimited string

Any string starting and ending with a quote (either ' or "), but not containing embedded quotes. The delimiting quotes are not considered to be part of the data.

- A *dialog variable name* enclosed in parentheses (varname). If the dialog variable name is on the right, the entire contents of the variable are considered part of the data, including any quotes, apostrophes, blanks, commas, or other special characters. If the dialog variable name is on the left, its content is totally replaced.

Notes:

1. In the CLIST environment, the CLIST variable pool and the dialog function variable pool are merged. Therefore, variables in parentheses are the same as ampersand variables, except that the editor does the symbolic substitution rather than the CLIST processor.
2. In the REXX environment, the REXX variable pool and the dialog function variable pool are also merged. Therefore, quoted variable names in parentheses are the same as unquoted variable names, except that the editor does the symbolic substitution rather than the REXX processor.
3. In a program macro, you must use the VDEFINE service for any variables that are passed to the editor.

Keyphrase

A keyphrase is either a single keyword, or a keyword followed by a line number or label. The keyphrase can be either a single-valued keyphrase or a double-valued keyphrase.

Keyphrase Syntax: Single-valued keyphrases can have the following syntax:

```
ISREDIT keyphrase = keyphrase
ISREDIT keyphrase = value
ISREDIT keyphrase = keyphrase + value
ISREDIT keyphrase = value + value
```

Double-valued keyphrases can have the following syntax:

```
ISREDIT (varname,varname) = keyphrase
ISREDIT keyphrase = value-pair
```

where value-pair is one of the following:

- Two literals, which can be separated by a comma or blank. For examples:

CLIST Statements

```
ISREDIT CURSOR = 1,40
ISREDIT CURSOR = 1 40
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT CURSOR = 1,40'
'ISREDIT CURSOR = 1 40'
```

Apostrophes or quotes cannot be used when specifying two numeric values. All of the following, for example, are incorrect:

CLIST Statements

```
ISREDIT CURSOR = '1','40'
ISREDIT CURSOR = '1,40'
```

REXX Statements

```
ADDRESS ISPEXEC
"ISREDIT CURSOR = '1','40'"
"ISREDIT CURSOR = '1,40'"
```

- Two variable names enclosed in parentheses and separated by a comma or blank, where each variable contains a single value:

(varname,varname) or *(varname varname)*

Using Commands in Edit Macros

In any edit assignment statement containing a two-valued keyphrase, either of the variables or values in a pair can be omitted. The general syntax then becomes:

```
ISREDIT (varname) = keyphrase
ISREDIT keyphrase = single-value
ISREDIT (,varname) = keyphrase
ISREDIT keyphrase = ,single-value
```

Note: Even though you can use blanks instead of commas to separate paired variables or values, you must use a leading comma whenever the first variable or value has been omitted.

Overlays and Templates

The transfer of information from one side of the equal sign to the other can involve combining several variables or values. This transfer is called an *overlay*. When you perform overlays, there are certain guidelines to remember.

When two values (or a keyphrase and a value) are on one side of an equal sign and separated by a plus sign (+), only non-blank characters in the value on the right overlay corresponding positions in the value on the left. For example:

CLIST Statements

```
ISREDIT LINE .ZCSR = LINE + '//'
ISREDIT MASKLINE = MASKLINE + <40 '&STR(/*)' 70 '&STR(*)'>
```

REXX Statements

```
ADDRESS ISPEXEC
"ISREDIT LINE .ZCSR = LINE + '//'
"ISREDIT MASKLINE = MASKLINE + <40 '/*' 70 '*/'>"
```

The first example causes two slashes to replace the first two column positions of the current line (the line containing the cursor). The remainder of the line is unchanged. The second example uses a *template* to cause columns 40-41 of the current mask line to be replaced with /* and columns 70-71 to be replaced with */. Again, remember that the template replaces the corresponding positions on the left only if those left positions are blank. The template shown in the preceding example has the form:

```
<col-1 literal-1 col-2 literal-2 ... >
```

It can be designed with *col-1* and *col-2* indicating a starting column position, and *literal-1* and *literal-2* indicating the data to start in that column. The entire template is delimited with less-than (<) and greater-than (>) signs. A template can be designed by using variable names (enclosed in parentheses) for either *col-1*, *col-2*, *literal-1*, *literal-2*, or for all four. All of the following forms are valid:

```
<(colvar-1) (datavar-1) (colvar-2) (datavar-2) ... >
<(colvar-1,datavar-1) (colvar-2,datavar-2) ... >
<(colvar-1) literal-1 col-2 (datavar-2) ... >
```

Using Edit Assignment Statements

You can use an assignment statement to pass edit parameters to a macro or to allow a macro to set an edit parameter. If the edit parameter keyphrase is on the right of the assignment statement, the edit parameter is passed to the macro. If the edit parameter keyphrase is on the left of the assignment statement, the edit parameter is changed to the value on the right. In the following assignment statement, the edit parameter keyphrase is CAPS. The editor assigns the current CAPS edit mode status (ON or OFF) to the variable CAPMODE.

Using Commands in Edit Macros

CLIST Statement

```
ISREDIT (CAPMODE) = CAPS
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT (CAPMODE) = CAPS'
```

In the preceding example statements, the parentheses around CAPMODE indicate to the ISPF editor that the enclosed name is the name of a symbolic variable. If the name happened to be preceded with an ampersand (&), rather than enclosed in parentheses, the CLIST processor would replace the name of the variable with its actual value, and the editor would not see the name. In a REXX statement, the variable name must be within quotes so that the name, not the value, is passed. Only names with 8 or fewer characters are allowed by the ISPF editor.

When the editor finds a variable name in parentheses in a position where a value is required, it substitutes the value assigned to that variable. In the following examples the edit macro sets the edit CAPS mode:

CLIST Statements

```
ISREDIT CAPS = ON  
ISREDIT CAPS = (CAPMODE)  
ISREDIT CAPS = &CAPMODE
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT CAPS = ON'  
'ISREDIT CAPS = (CAPMODE)'  
'ISREDIT CAPS = 'capmode'
```

The CLIST and REXX command processors replace the variable CAPMODE with its assigned value before the ISPF editor processes the statement. This makes the last statement equivalent to the first statement; in this case, the variable has a value of ON.

The second statement differs in that the editor receives the variable name and retrieves its value from the dialog variable pool.

Passing Values

Some information can best be passed back and forth between the editor and the macro in pairs. The following examples show assignment statements that pass two values:

CLIST Statements

```
ISREDIT (LB, RB) = BOUNDS  
ISREDIT BOUNDS = (LB, RB)
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT (LB, RB) = BOUNDS'  
'ISREDIT BOUNDS = (LB, RB)'
```

In the first statement, the current left and right boundaries are stored into the variables LB (LEFTBND) and RB (RIGHTBND). In the second statement, the values from the variables LB and RB are used to change the current boundaries.

For more information on which edit macro commands take one variable and which take two, see Chapter 11. Edit Macro Commands and Assignment Statements.

Manipulating Data With Edit Assignment Statements

You can use assignment statements to obtain, replace, or add data being edited.

To copy a line, use:

CLIST Statement

```
ISREDIT LINE_AFTER 5 = LINE 2
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT LINE_AFTER 5 = LINE 2'
```

Using Commands in Edit Macros

To copy line 1 from the data set into the variable LINEDATA, use:

CLIST Statement	REXX Statements
ISREDIT (LINEDATA) = LINE 1	ADDRESS ISPEXEC 'ISREDIT (LINEDATA) = LINE 1'

To replace the first line in the data set, using the data from the variable LINEDATA, use:

CLIST Statement	REXX Statements
ISREDIT LINE 1 = (LINEDATA)	ADDRESS ISPEXEC 'ISREDIT LINE 1 = (LINEDATA)'

To add a new line after line 1 in the data set using the variable NEWDATA, use:

CLIST Statement	REXX Statements
ISREDIT LINE_AFTER 1 = (NEWDATA)	ADDRESS ISPEXEC 'ISREDIT LINE_AFTER 1 = (NEWDATA)'

Differences Between Edit, CLIST, and REXX Assignment Statements

Note the following differences between edit, CLIST, and REXX assignment statements:

- Edit assignment statements are preceded by ISREDIT. CLIST assignment statements are preceded by SET. If the **address isredit** command is in effect, edit assignment statements within a REXX exec do not need to be preceded by ISREDIT.
- In edit assignment statements, a keyphrase must appear on either the left or right side of the equal sign. A keyphrase is either a single keyword, or a keyword followed by a line number or label. See “Keyphrase” on page 105 if you need more information.
- When coding edit assignment statements, variable names to be passed to the editor are enclosed in parentheses so that the PDF component is passed the name of the variable, not its value. Sometimes two variable names may appear within the parentheses.
- Arithmetic expressions are not allowed in an edit assignment statement, but in certain cases a plus sign (+) can be used to show partial overlay of a line. See “Overlays and Templates” on page 106 if you need more information.

Performing Line Command Functions

You cannot issue line commands directly from an edit macro. For example, you cannot use the M (move) line command within an edit macro.

However, you can perform most of the functions provided by line commands by writing an edit macro. By using edit assignment statements or by issuing primary commands, you can perform functions such as move, copy, or repeat. For example, if you want to move a line, you can assign the line to a CLIST or REXX variable, delete the original line using the DELETE command, and assign the variable to a new line in the data.

Some commands can be processed only from within a macro. These commands provide functions done with line commands from the keyboard. Table 3 on page 109 identifies the commands, the corresponding line commands, and the

functions performed.

Table 3. Edit Macro Commands Corresponding to Line Commands

Edit Macro Statement	Corresponding Line Command	Function
INSERT	I	Inserts temporary lines
SHIFT ((Shifts columns left
SHIFT))	Shifts columns right
SHIFT <	<	Shifts data left
SHIFT >	>	Shifts data right
TENTER	TE	Starts text entry mode
TFLOW	TF	Performs text flow
TSPLIT	TS	Performs text split

For example:

CLIST Statement	REXX Statements
ISREDIT TFLOW 1	ADDRESS ISPEXEC 'ISREDIT TFLOW 1'

causes the paragraph starting on line 1 to be flowed in the same way as a TF (text flow) line command would if entered on the first line.

For more information on line command functions in edit macros, see Chapter 11. Edit Macro Commands and Assignment Statements.

Parameters

If you want to supply information to a macro as parameters, you must identify these parameters on the ISREDIT MACRO statement by enclosing them in parentheses. For example, if you have the following macro command in an edit macro named FIXIT:

CLIST Statement	REXX Statements
ISREDIT MACRO (MEMNAM)	ADDRESS ISPEXEC 'ISREDIT MACRO (MEMNAM)'

when you enter:

Command =====> FIXIT ABCD

the value ABCD is assigned to the variable MEMNAM.

Passing Parameters to a Macro

A parameter can be either a simple string or a quoted string. It can be passed by using the standard method of putting variables into shared and profile pools (use VPUT in dialogs and VGET in initial macros). This method is best suited to parameters passed from one dialog to another, as in an edit macro.

You can enter parameters along with an edit macro name as a primary command by using the MACRO command. This command allows you to identify the names of one or more variables to contain any passed parameters.

Using Commands in Edit Macros

It is an error to enter parameter values for a macro without parameter variables. If you make this mistake, the editor displays a message. It is not an error if you supply more or fewer parameters than the number of variables that are included on the MACRO command. When you are writing a macro, check for omissions and the order of parameters.

Multiple parameters are placed into one or more variables based on the number of variables specified in the MACRO command. If you include more than one variable name, the editor stores the parameters in order (the first parameter in the first variable, the second in the second, and so on). Note that assignment to variables is by position only.

If there are more parameters entered than there are variables available, the editor stores the remaining parameters as 1 character string in the last variable. If you include only one variable name on the MACRO command, that variable contains all the parameters entered with the macro name. If there are more variable names than parameters, the unused variables are set to nulls.

Multiple parameters are separated by a blank or comma, or a quoted string that is separated by a blank or comma. Quotes can be single (') or double ("). If you want your FIXIT macro to accept two parameters, for example, you can include the following command:

CLIST Statement

```
ISREDIT MACRO (PARM1,PARM2,REST)
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT MACRO (PARM1,PARM2,REST)'
```

This means that if you enter:

```
Command =====> FIXIT GOOD BAD AND UGLY
```

variable PARM1 is assigned the value GOOD, PARM2 is assigned the value BAD, and REST is assigned the value AND UGLY.

If the parameters passed were GOOD BAD, variable REST would be null. Also, if the parameters are enclosed in quotation marks, such as:

```
Command =====> FIXIT 'GOOD BAD' 'AND UGLY'
```

PARM1 would be set to GOOD BAD, PARM2 would be set to AND UGLY, and REST would be null.

For another example, see the TRYIT macro (Figure 46 on page 124). If the MACRO statement contains two variables (ISREDIT MACRO (COMMAND,PARM)), entering:

```
Command =====> TRYIT RESET
```

sets the variables Command to RESET and PARM to null. Conversely, the following command:

```
Command =====> TRYIT FIND A
```

sets Command to FIND and PARM to A. To find out what was actually typed on the command line, a macro may examine the variable ZEDITCMD, which is in the shared variable pool. ZEDITCMD is a character variable, the length of which depends on the length of the command entered. Therefore, you should either VDEFINE ZEDITCMD to be sufficiently large to hold the expected command, or use the VCOPY service to get the length.

Using Edit macros in Batch

You can run PDF edit macros in batch by submitting JCL which allocates all of the necessary ISPF libraries (refer to *ISPF Dialog Developer's Guide and Reference*), and runs a command which calls the EDIT service with an initial macro. This initial macro can do anything that can be done by an initial macro in an interactive session. However, in batch, the macro should end with an ISREDIT END or ISREDIT CANCEL statement. These statements insure that no attempt is made to display the edit screen in batch.

A simple initial macro to change strings in batch might look like the following:

```
ISREDIT MACRO
ISREDIT CHANGE JANUARY FEBRUARY ALL
ISREDIT END
```

Edit Macro Messages

You can display messages from an edit macro the same way you do from an ISPF dialog.

- Use SETMSG, which causes the message to appear on whatever panel is displayed next.
- Use DISPLAY with the MSG keyword. This is useful if the macro displays panels of its own.

PDF provides three generic messages for use in dialogs where you want to generate the message text or when you do not want a separate message library.

```
ISRZ000 '&ZEDSMMSG' .ALARM = NO .HELP = ISR2MACR
'&ZEDLMSG'
```

```
ISRZ001 '&ZEDSMMSG' .ALARM = YES .HELP = ISR2MACR
'&ZEDLMSG'
ISRZ002 '&ZERRSM' .ALARM = &ZERRALRM .HELP = &ZERRHM
'&ZERRLM'
```

For example, if you want your macro to sound an alarm, and to issue the short message INVALID PARAMETER and the long message PARAMETER MUST BE 4 DIGITS, use the following statements:

CLIST Statements

```
SET &ZEDSMMSG = &STR(INVALID PARAMETER)
SET &ZEDLMSG = &STR(PARAMETER MUST BE 4 DIGITS)
ISPEXEC SETMSG MSG(ISRZ001)
```

REXX Statements

```
ADDRESS ISPEXEC
zedsmg = 'Invalid Parameter'
zedlmsg = 'Parameter must be 4 digits'
'SETMSG MSG(ISRZ001)'
```

Note: ZEDLMSG only displays when you enter the HELP command.

Using Commands in Edit Macros

Macro Levels

Each macro operates on a separate and unique level. A person at the keyboard always operates at level 0. If that person starts a macro, it operates at level 1; the macro started by a level-1 macro operates at level 2, and so on. The level is the degree of macro nesting. Edit macros are primary commands; thus, nested macros are started by prefixing them with ISREDIT.

A macro can determine its own level with the following assignment statement:

```
ISREDIT (varname) = MACRO_LEVEL
```

The current level number is stored in the specified variable. ISPF supports up to 255 levels of macro nesting.

Labels in Edit Macros

A label is an alphabetic character string used to name lines. It is especially useful for keeping track of a line whose relative line number may change because labels remain set on a line even when relative line numbers change. The following special labels are automatically assigned by the editor. A label must begin with a period (.) and be followed by no more than 8 alphabetic characters, the first of which cannot be Z. No special characters or numeric characters are allowed.

The special labels that are automatically assigned by the editor all begin with the letter Z. Labels beginning with Z are reserved for editor use only.

The editor-assigned labels are:

.ZCSR	The data line on which the cursor is currently positioned.
.ZFIRST	The first data line (same as relative line number 1). Can be abbreviated .ZF .
.ZLAST	The last data line. Can be abbreviated .ZL .
.ZFRANGE	The first line in a range specified by you.
.ZLRANGE	The last line in a range specified by you.
.ZDEST	The destination line specified by you.

Note: Unlike other labels, **.ZCSR**, **.ZFIRST**, and **.ZLAST** do not stay with the same line. Label **.ZCSR** stays with the cursor, and labels **.ZFIRST** and **.ZLAST** point to the current first and last lines, respectively.

Using Labels

In a macro, you can assign a label to a line by using the LABEL assignment statement. For example:

CLIST Statements

```
SET &LNUM = 10  
ISREDIT LABEL &LNUM = .HERE
```

REXX Statements

```
ADDRESS ISPEXEC  
lnum = 10  
'ISREDIT LABEL' lnum '= .HERE'
```

This assigns the label **.HERE** to the line whose relative line number is contained in variable **LNUM** (line 10 here). The **.HERE** label allows the macro to keep track of a line whose relative line number may change. When the macro finishes running, the **.HERE** label is removed.

Using Commands in Edit Macros

Labels can be used as part of a keyphrase instead of a line number. For example:

CLIST Statements

```
ISREDIT LINE .NEXT = (DATAVAR)
ISREDIT LINE_AFTER .XYZ = (DATAVAR)
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT LINE .NEXT = (DATAVAR) '
'ISREDIT LINE_AFTER .XYZ = (DATAVAR) '
```

The first example stores new data into the line that currently has the label `.NEXT`. The second example creates a new line after the line whose label is `.XYZ`, and stores data into the new line.

A macro can determine if a label exists. Using the `LINENUM` assignment statement, you can obtain the current relative line number of a labeled line. If the label does not exist, the return code (`&LASTCC` for CLIST or `RC` for REXX) is 8. For example:

CLIST Statements

```
ISREDIT (LNUM2) = LINENUM .ABC
IF &LASTCC = 8 THEN WRITE NO .ABC LABEL
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT (LNUM2) = LINENUM .ABC '
IF RC = 8 THEN SAY 'No .ABC label'
```

This example stores the relative line number of the line with label `.ABC` into variable `LNUM2` and tests to see if that label did exist.

Labels have a variety of uses. For example, because both the `FIND` and `SEEK` commands position the cursor at the search string after the macro has been started, you may want to assign the data from the line on which the cursor is positioned to the variable `CSRDATA`. To do so, use the following statement:

CLIST Statements

```
ISREDIT FIND 'IT'
ISREDIT (CSRDATA) = LINE .ZCSR
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT FIND IT'
'ISREDIT (CSRDATA) = LINE .ZCSR'
```

The label `.ZCSR` names the line in which the cursor is positioned. The `.ZCSR` label is moved to a new line when one of the following commands moves the cursor: `FIND`, `CHANGE`, `SEEK`, `EXCLUDE`, `TSPLIT` or `CURSOR`. The labels `.ZFIRST` and `.ZLAST` can also move when data is added or deleted.

If you assign a labeled line a new label that is blank, the previous label becomes unassigned (if both labels are at the same level). For example:

CLIST Statement

```
ISREDIT LABEL .HERE = ' '
```

REXX Statements

```
ADDRESS ISPEXEC
"ISREDIT LABEL .HERE = ' '"
```

removes the label from the line.

If a label in use is assigned to another line, the label is moved from the original line to the new line (if the new assignment is at the same level as the original).

Referring to Labels

A nested macro can refer to all labels assigned by higher-level macros and to labels that you assign. When a macro assigns labels, they are associated by default with the assigning macro level. The labels are automatically removed when the macro finishes running. The labels belong to the level at which they are assigned and can have the same name as the labels at other levels without any conflict.

Using Commands in Edit Macros

When a macro ends, the labels at the current nesting level are deleted. To set a label for the next higher level, the macro can issue the `MACRO_LEVEL` assignment statement to obtain the current level and decrease the level by 1.

A macro can determine the level of a label with the `LABEL` assignment statement, as shown in the following syntax:

```
ISREDIT (varname1,varname2) = LABEL lptr
```

The label assigned to the referenced line is stored in the first variable and its level is stored in the second variable. If a label is not assigned to the line, a blank is stored in both variables.

Passing Labels

You can create a label at any level above its current level by explicitly stating the level:

```
ISREDIT LABEL lptr = label [level]
```

Here, if the label previously existed at the explicitly specified level, its old definition is lost. A label assigned at a higher level remains after the macro ends and is available until the level at which it was assigned ends or the label is explicitly removed.

If a macro sets a label without indicating a level, or if its value is equal to or greater than the level at which the macro is running, the label is set at the macro level that is currently in control and does not affect any labels set in a higher level.

If a macro queries a label without specifying a level, or uses the label as a line pointer, the search for the label starts at the current macro level and goes up, level by level, until the label defined closest to the current level is found.

If you specify a level parameter that is outside the currently active levels, it is adjusted as follows: a value less than zero is set to zero; a value greater than the current nesting level is set to the current nesting level. This means that a higher-level macro cannot set a label at the level of the macro that it is going to start.

Referring to Data Lines

You can refer to data lines either by a relative line number or by a symbolic label. Note that special lines (`MASK` lines, `TABS` lines, `COLS` lines, `BOUNDS` lines, `MSG` lines, and others) are not considered data lines. You cannot assign labels to them, and they do not have relative line numbers. Also, you cannot directly reference these lines in a macro, even though they are displayed. Excluded lines are regarded as data lines.

Relative line numbers are not affected by sequence numbers in the data, nor are they affected by the current setting of number mode. The first line of data is always treated as line number 1, the next line is line number 2, and so on. The `TOP OF DATA` line is considered line number 0.

When you insert or delete lines, the lines that follow change relative line numbers. If you insert a new line after line 3, for example, it becomes relative line 4 and what was relative line 4 becomes relative line 5, and so on. Similarly, if line 7 is deleted, the line that was relative line 8 becomes relative line 7, and so on.

Referring to Column Positions

Column positions in edit macros are not the same as they appear on the panel; they refer only to the editable portions of the data. When number mode is on, sequence numbers are not part of the data, and thus are not editable. For example, if NUMBER COBOL ON mode is in effect, the first six positions of each line contain the sequence number. The first data character is in position 7, which is considered relative column 1. When number mode is off, the line number portion is editable, so here position 1 becomes column 1 and position 7 becomes column 7. These are not the column values displayed on the edit panel. This discrepancy can influence the use of column numbers as parameters from the keyboard. Column numbers must be converted according to number mode. See “Edit Boundaries” on page 28 for the conversions.

If your macro must access the sequence numbers as data, include statements that save the current number mode, set number mode off, and then restore the original number mode.

When a macro retrieves the current cursor position, a relative column number of zero is returned if the cursor is outside the data portion of the line. When a macro sets the cursor column to zero, the cursor is placed in the Line Command field on the left side of the designated line.

Defining Macros

You can use DEFINE to give macros names that are different from their data set names, make aliases for built-in edit commands, identify macros as program macros, or set a command as disabled. DEFINE commands are usually issued in an initial macro.

For more information, refer to the description of the DEFINE command in Chapter 11. Edit Macro Commands and Assignment Statements.

Defining an Alias

To establish an alias or alternate name for a primary command, enter a DEFINE followed by the new name, the ALIAS operand, and then the original command name. For example, the following command:

```
Command ==> DEFINE FILE ALIAS SAVE
```

establishes FILE as an alias for SAVE, allowing you to enter FILE to save the data currently being edited instead of SAVE.

Resetting Definitions

To reset the last definition for a command and return the command to its previous status, use the DEFINE command with the RESET operand. For example, having established FILE as an alias for SAVE, you can enter:

```
Command ==> DEFINE FILE RESET
```

to cause FILE to be flagged as an invalid command. When defining a command as DISABLED, you cannot reset the disabled function.

Using Commands in Edit Macros

Replacing Built-In Commands

To replace an existing edit command, with a macro, you also use DEFINE. For example:

CLIST Statement	REXX Statements
ISREDIT DEFINE FIND ALIAS MYFIND	ADDRESS ISPEXEC 'ISREDIT DEFINE FIND ALIAS MYFIND'

This links the command name to an edit macro.

To use the built-in edit command, precede the command with BUILTIN. For example, to process the built-in FIND command, include the following statement:

CLIST Statement	REXX Statements
ISREDIT BUILTIN FIND...	ADDRESS ISPEXEC 'ISREDIT BUILTIN FIND ...'

where the ellipses represent other FIND command operands, such as the search string.

Implicit Definitions

When you or your macro issue a command unknown to the editor, PDF searches for a CLIST or REXX EXEC with that name. If the editor finds the command, it is implicitly defines it as an edit macro.

Program macros can be implicitly defined by preceding the name of the macro with an exclamation point (!). Remember that the name must be 7 characters or less, excluding the exclamation point. Program macros are similar to ISPF dialogs in that they must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library. See “Program Macros” on page 97 for more information.

Using the PROCESS Command and Operand

The PROCESS command provides a way to alter the usual sequence of events in an edit macro. It is related to the PROCESS operand on the MACRO command. PROCESS is the default for the MACRO command. PROCESS specifies that display data and line commands be processed before another statement is processed. If you specify NOPROCESS, the editor defers processing the panel data and line commands until it finds an ISREDIT PROCESS command later in the macro, or until the macro ends. You can use PROCESS to create a “before-and-after” effect. If you specify NOPROCESS at the beginning of a macro, edited data appears without the changes made from the keyboard—creating a “before” effect. Once you specify PROCESS, changes that were made from the keyboard appear—creating an “after” effect.

The syntax of the ISREDIT MACRO statement is:

```
ISREDIT MACRO [(var1[,var2...])] [PROCESS|NOPROCESS]
```

Specifying NOPROCESS in the Macro Statement

NOPROCESS is useful if you want to process statements before the display data or line commands are processed. It enables you to perform initial verification of parameters or capture lines before they are changed from the panel.

Using Commands in Edit Macros

It is also useful if you want to include an ISREDIT PROCESS command to specify whether the macro expects, and handles, line commands that identify either a range of lines, a destination line, or both. This linking is the method by which the editor allows a macro command to interact with line commands in the same way that the built-in MOVE and REPLACE commands do. With the ISREDIT PROCESS command, the editor can process line commands that you have entered, performing significant error and consistency checking.

Specifying a Destination

If you include the following process statement in an edit macro:

CLIST Statement

```
ISREDIT PROCESS DEST
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT PROCESS DEST'
```

the macro expects you to specify a destination line. A destination line is always specified using either A (after) or B (before). The editor sets the dialog variable .ZDEST to the line preceding the destination. However, if neither A nor B is specified, .ZDEST is set to the last data line. In this situation, a return code shows that no destination was specified.

Specifying a Range

If you use the following syntax for a PROCESS macro command in an edit macro:

```
ISREDIT PROCESS RANGE operand
```

the macro expects to receive a specified range of lines to process. The operand following the RANGE operand identifies either one or two commands that are to be accepted. For example, the command PROCESS RANGE Q Z allows the line commands Q or Z (but not both) to be processed with this macro. The line commands could take any of the following forms:

- Q or Z, to specify a single line.
- QQ or ZZ, to specify a block of lines. This form is obtained by doubling the last letter of the single-line command.
- Qn or Zn where *n* is a number that specifies a series of lines.

After the PROCESS command is completed, the dialog variable .ZFRANGE is automatically set to the first line of the specified range. The dialog variable .ZLRANGE is set to the last line of the specified range. These labels can refer to the same line. If no range is entered, the range defaults to the entire data set. In this situation, a return code shows that no range was specified.

Two line command names can be specified for PROCESS. In this situation, use the RANGE_CMD assignment statement to return the value of the command entered. For example, if you issue the following PROCESS command:

CLIST Statement

```
ISREDIT PROCESS RANGE Z $
```

REXX Statements

```
ADDRESS ISPEXEC  
'ISREDIT PROCESS RANGE Z $'
```

The RANGE_CMD assignment statement returns either a Z or a \$.

The names of line commands that define the range can be 1 to 6 characters, but if the name is 6 characters long, it cannot be used as a block format command by

Using Commands in Edit Macros

doubling the last character. The name can contain any alphabetic or special character except blank, hyphen (-), apostrophe ('), or period (.). It cannot contain any numeric characters.

Example

In the example that follows, the NOPROCESS operand on the MACRO command defers processing of the panel data until the line with the cursor is assigned to a variable. After the PROCESS command, the line contains any changes that you made.

CLIST Statements

```
ISREDIT MACRO NOPROCESS
ISREDIT (BEFORE) = LINE .ZCSR
ISREDIT PROCESS
ISREDIT (AFTER) = LINE .ZCSR
IF &STR(&BEFORE) = &STR(&AFTER) THEN -
...
ELSE -
...
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT MACRO NOPROCESS'
'ISREDIT (BEFORE) = LINE .ZCSR'
'ISREDIT PROCESS'
'ISREDIT (AFTER) = LINE .ZCSR'
IF BEFORE = AFTER THEN
...
ELSE
...
```

See “PROCESS—Process Line Commands” on page 377.

Recovery Macros

After a system failure, you might want to restore the command definitions and aliases that you were using when the system failed, but you do not want to destroy the profile changes you made during the edit session before the failure.

To help to recover after a system failure, you can provide a recovery macro which can restore command definitions and aliases while not destroying profile changes made before the failure. The recovery macro, like an initial macro, runs after the data has been read but before it is displayed. However, the macro is run whenever the recovery data set is being edited.

You can specify a recovery macro:

- By entering the RMACRO primary command:
Command ==> RMACRO name
- In your initial macro by using the RMACRO assignment statement:
ISREDIT RMACRO = name

where *name* sets the name of the macro for the edit session. The name operand is used to specify the name of the macro to be run after a data set has been recovered.

Note: Recovery macros are only in effect for the duration of a particular Edit session. They must be specified again each time a new member or data set is edited.

Return Codes from User-Written Edit Macros

A macro can issue the following return codes. These return codes affect the Command line and cursor position on the next display of edit data:

Return Codes from User-Written Edit Macros

- 0** Shows normal completion of the macro. The cursor position is left as set by the macro. The Command line is blanked.
- 1** Shows normal completion of the macro. The cursor is placed on the Command line and the line is blanked. Use this return code to make it easy to enter another macro or edit command on the Command line.

4 and 8

Treated by the ISPF editor as return code 0. No special processing is done.

12 and higher

Error return codes. The cursor is placed on the Command line and the macro command remains. When used with these return codes, the dialog manager SETMSG service prompts you for an incorrect or omitted parameter.

Any invocation of a disabled macro command issues a return code of 12. See the DEFINE command for more information on disabled commands.

20 and higher

Indicate a severe error. The meanings of the severe return codes are:

- 20** Command syntax error or Dialog service routine error.
- 24** Macro nesting limit of 255 exceeded (possible endless loop; see the BUILTIN macro command).
- 28** Command found either preceding the ISREDIT MACRO command, or following the ISREDIT END or ISREDIT CANCEL command.

Each command description in Chapter 11. Edit Macro Commands and Assignment Statements includes a list of return codes that are possible for the command. Because &LASTCC (CLIST) or RC (REXX) is set for every statement, you must either test it in the statement immediately following the command that sets it, or you must save its value in another variable. Use a command such as:

```
SET &RETCODE = &LASTCC
```

The variable (&RETCODE or RETCODE) can then be tested anywhere in the macro until it is changed.

Return Codes from PDF Edit Macro Commands

Every CLIST edit macro command sets variable &LASTCC with a return code. REXX edit macros set variable RC. The return codes range from 0 to 20.

- 0** Shows normal completion of the command.

2, 4, and 8

Information return codes. They show a special condition that is not necessarily an error. These return codes can be tested or ignored, depending on the requirements of the macro.

For some cases of RC=8, the ISPF system variables ZERRSM (short error message text) and ZERRLM (long error message text) are set. For more information on ZERRSM and ZERRLM, see *ISPF Dialog Developer's Guide and Reference*

12 and higher

Error return codes. Normally an error return code causes the macro to end abnormally and an error panel to appear. The error panel shows the kind of error and lists the statement that caused the error condition.

Return Codes from PDF Edit Macro Commands

The ISPF system variables ZERRSM (short error message text) and ZERRLM (long error message text) are set for error return codes. For more information on ZERRSM and ZERRLM, see *ISPF Dialog Developer's Guide and Reference*

Often, the only two possible return codes are 0 and 20. The CAPS command is an example of such a command. Any valid form of CAPS issues a return code of 0.

Selecting Control for Errors

As explained in "Return Codes from PDF Edit Macro Commands" on page 119, every edit macro statement causes variable &LASTCC (CLIST) or RC (REXX) to be set to a return code. Return codes of 12 or higher are considered errors (except for the PROCESS edit macro command return code of 12), and the default is to end macros that issue those return codes.

Sometimes you need to handle errors at the time that they occur. The error is expected and the edit macro logic can handle the problem. If you want to handle all errors that might occur in your macro, you can include the following statement:

```
ISPEXEC CONTROL ERRORS RETURN
```

If errors occur, control returns to the macro. On the other hand, to return error handling to the default mode, include the following:

```
ISPEXEC CONTROL ERRORS CANCEL
```

If an error occurs, the macro ends.

If you want to do both, you can include any number of ISPEXEC CONTROL statements in your macro to turn error handling on and off.

Chapter 7. Testing Edit Macros

This chapter documents general-use programming interfaces and associated guidance information.

This chapter tells you how to include statements in your edit macros to capture and handle error conditions.

Using the information in the preceding chapters, you should be able to write and run an edit macro that uses CLIST or REXX logic and processes simple edit commands. However, even an experienced edit macro writer occasionally includes a bug that causes a macro to end abnormally (ABEND), or writes a macro that does not work as expected. When this occurs, you must debug your macro, just as you would debug any other kind of program you write.

Handling Errors

There are two kinds of errors that you may encounter when you debug macros—edit command errors and dialog service errors. Both kinds of errors are controlled by the ISPEXEC CONTROL ERRORS RETURN command. For more specific information, refer to *ISPF User's Guide*

Edit Command Errors

The editor detects edit command errors and displays either an edit macro error panel with an error message, or a return code. If an edit command error occurs, the macro ends abnormally with the following results:

- When you are using the ISPF editor with ISPF test mode off, you return to the edit session.
- If ISPF test mode is on, the PDF component is also in test mode. You can override the abnormal end and attempt to continue by typing YES on the PDF edit macro error panel and pressing Enter. If ISPEXEC CONTROL ERRORS RETURN has been processed, the error panel does not appear, and the macro automatically continues.

Dialog Service Errors

ISPF detects dialog service errors and displays a message identifying the error with the statement which caused the error. If a dialog service error occurs, the edit session ends abnormally with the following results:

- When you are using the PDF component with ISPF test mode off, the ISPF Primary Option Menu is displayed.
- If you are using the PDF component with ISPF test mode on, you can override the abnormal end and attempt to continue by typing YES on the ISPF dialog error panel and pressing Enter. In either case, if ISPEXEC CONTROL ERRORS RETURN has been processed, no panel appears and the editor sends a return code instead of ending the dialog.

Note: If you enter ISPF with TEST as an operand, or use Dialog Test (option 7), ISPF remains in test mode until you end the ISPF session.

Using CLIST WRITE Statements and REXX SAY Statements

The CLIST WRITE statement and the REXX SAY statement can be valuable tools in tracking down edit macro problems. A WRITE statement or a SAY statement is simply a line of text inserted into your macro that creates a message on your screen while the macro is running. With these statements, you can identify the position of the statement within the macro, and display the value of variables.

For example, if you are having trouble debugging the CLIST TESTDATA macro from Figure 35 on page 91, adding some WRITE statements may help locate the problem (Figure 44).

```

/*                                     */
/*                                     */
/* TESTDATA - generates test data     */
/*                                     */
ISREDIT MACRO
  SET &COUNT = 1                      /* Initialize loop counter */
  DO WHILE &COUNT <= 9                /* Loop up to 9 times     */
    ISREDIT FIND 'TEST-#'              /* Search for 'TEST-#'   */
    SET &RETCODE = &LASTCC             /* Save the FIND return code */
    WRITE RESULT OF FIND, RC = &RETCODE
    IF &RETCODE = 0 THEN                /* If string was found,   */
      DO                                 /*                         */
        ISREDIT CHANGE '#' '&COUNT' /* Change # to a digit and */
        SET &COUNT = &COUNT + 1 /* increment loop counter */
        WRITE COUNT IS NOW UP TO &COUNT
      END
    ELSE                                 /* If string is not found, */
      SET &COUNT = 10              /* Set counter to exit loop */
  END
EXIT CODE(0)

```

Figure 44. TESTDATA Macro with CLIST WRITE Statements

Remember that the macro TESTDATA creates test data with variations of the same line by putting ascending numbers 1 through 9 in the data. When WRITE statements are included in the data, a step-by-step breakdown of the procedure appears on your screen.

If there are no errors in the TESTDATA macro, the return codes and count appear on your screen in TSO line mode. Asterisks at the bottom of the screen prompt you to press Enter and return to ISPF full-screen mode (Figure 45 on page 123).

```

RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 2
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 3
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 4
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 5
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 6
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 7
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 8
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 9
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 10
*** _

```

Figure 45. Results of TESTDATA Macro with CLIST WRITE Statements

Using CLIST CONTROL and REXX TRACE Statements

You can display a statement from a macro as it is being interpreted and run. Use either of the following:

- A CLIST CONTROL statement with the LIST, SYMLIST, or CONLIST operand
- A REXX TRACE statement with the A, I, L, O, R, or S operand.

These statements produce messages on your display screen similar to the WRITE and SAY statements discussed in the previous section. However, several differences should be noted:

- For the CLIST CONTROL statement:
 - LIST displays commands and subcommands (including ISREDIT statements) after substitution but before processing. This allows you to see an ISREDIT statement in the form that the editor sees the statement.
 - CONLIST displays a CLIST statement (for example, IF, DO, SET) after substitution but before processing. You might be able to tell why an IF statement did not work properly by using CONLIST.
 - SYMLIST displays both CLIST and command lines before symbolic substitution, allowing you to see the lines as written.

Use the NOLIST, NOSYMLIST, and NOCONLIST operands to prevent the display of statements. Refer to *TSO Extensions CLISTs* for more details.

- For the REXX TRACE statement:
 - The A operand traces all clauses displaying the results of each clause.
 - The I operand traces the intermediate results, displaying both the statement and the results.
 - The L operand traces labels in your edit macro.
 - The O operand stops, or turns off, the trace.
 - The R operand, which is used most often, traces all clauses and expressions.

Using CLIST CONTROL and REXX TRACE Statements

- The S operand scans each statement, displaying it without processing it.

Refer to *TSO/E Version 2 REXX Reference* and *TSO/E Version 2 REXX User's Guide* for more details.

Experimenting with Macro Commands

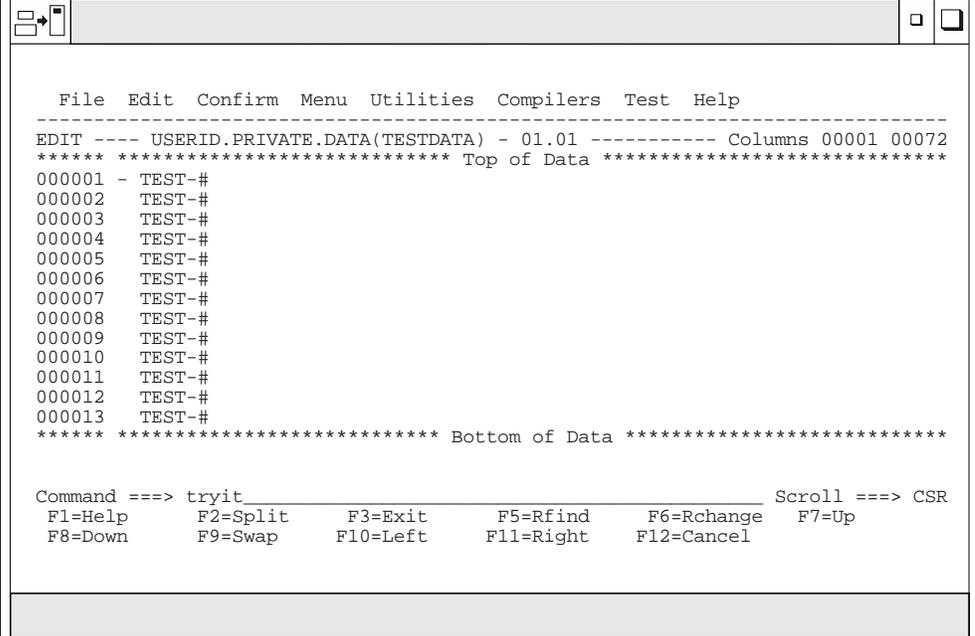
Use the TRYIT macro (Figure 46) to experiment with edit macros. TRYIT is handy when you want to see how a command or assignment statement works but do not actually want to write an entire macro. TRYIT processes the command and issues return codes that show whether it succeeded. To start the macro, type TRYIT on the Command line, followed by a command, and press Enter. If you enter TRYIT with the RESET operand, the variable &COMMAND is set to RESET; if you enter it as TRYIT FIND A, the variable &COMMAND is set to FIND A.

```
/*
/* TRYIT is a simple macro for trying out edit macro
/* statements
/*
ISREDIT MACRO (COMMAND)
SET &RETCODE=0 /* Initialize return code */
IF &STR() = &STR(&COMMAND) THEN /* If no command specified*/ -
WRITE MISSING COMMAND PARAMETER /* indicate problem */
ELSE /* Else parameter exists; */ -
DO /* invoke edit command */
ISREDIT &COMMAND /* Save the return code */
SET &RETCODE = &LASTCC /* from command invocation*/
WRITE &COMMAND RETURN CODE IS &RETCODE/* and indicate */
END /* its value to the user */
EXIT CODE(&RETCODE)
```

Figure 46. TRYIT Macro

The TRYIT macro tests both the SEEK and AUTONUM commands (Figure 47 on page 125). When you run the macro, it displays the return codes from the commands on your screen (Figure 48 on page 125).

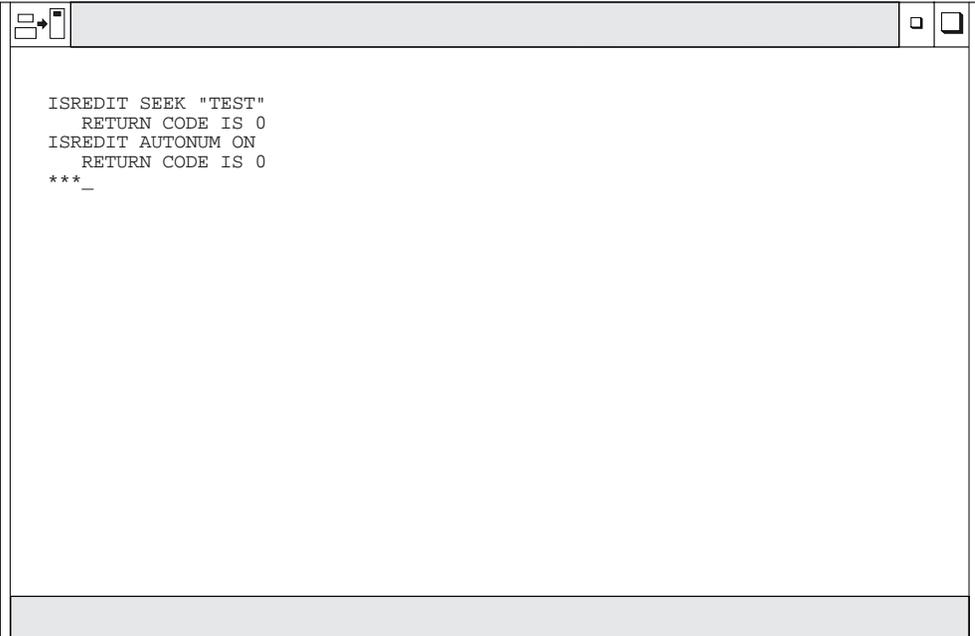
Experimenting with Macro Commands



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.01 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 - TEST-#
000002 TEST-#
000003 TEST-#
000004 TEST-#
000005 TEST-#
000006 TEST-#
000007 TEST-#
000008 TEST-#
000009 TEST-#
000010 TEST-#
000011 TEST-#
000012 TEST-#
000013 TEST-#
***** ***** Bottom of Data *****

Command ==> tryit_ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 47. TRYIT Macro - Before Running



```
ISREDIT SEEK "TEST"
RETURN CODE IS 0
ISREDIT AUTONUM ON
RETURN CODE IS 0
***
_
```

Figure 48. TRYIT Macro - After Running

Experimenting with Macro Commands

Chapter 8. Sample Edit Macros

This chapter documents general-use programming interfaces and associated guidance information.

TEXT Macro

The TEXT macro (Figure 49) initializes the edit profile values and function keys for text entry. You can enter it from the Command line or use it in an initial macro. This macro sets F12 to BOX. The BOX macro is described later in this chapter. It does not otherwise affect the running of the TEXT macro.

```
/*
/*TEXT initializes the profile and PF keys for text work
/*
/*
ISREDIT MACRO
ISREDIT NUMBER OFF /* Set number mode off
ISREDIT TABS OFF /* Set tabs off
ISREDIT NULLS OFF /* Set nulls off
ISREDIT BOUNDS /* Default bounds
ISREDIT CAPS OFF /* Set caps off
ISREDIT RECOVERY ON /* Set recovery mode on
/*
ISPEXEC VGET (ZPF24) PROFILE /* Ensure this is the
/* profile value
SET SAVEPF24 = &ZPF24 /* Save it for later
/* restoration
ISPEXEC VPUT (SAVEPF24) /* by PFEND and PFCAN
/*
SET &ZPF24 = BOX /*Set PF 12 to BOX
ISPEXEC VPUT (ZPF24) PROFILE /* and save in profile
/*
ISREDIT DEFINE END ALIAS PFEND /* Do DEFINES to reset
ISREDIT DEFINE CANCEL ALIAS PFCAN /* the PF key at exit
ISREDIT DEFINE QUIT ALIAS CANCEL /* Note that QUIT=PFCAN
EXIT CODE(0) /*
```

Figure 49. TEXT Macro

The following list explains the logical sections of the TEXT macro:

1. MACRO identifies this CLIST as a macro:
ISREDIT MACRO
2. The commands that follow MACRO set edit profile values; the boundaries are set to the first and last columns of data:
ISREDIT NUMBER OFF
ISREDIT TABS OFF
ISREDIT NULLS OFF
ISREDIT BOUNDS
ISREDIT CAPS OFF
ISREDIT RECOVERY ON
3. The SET statements save the current value and set ISPF variable &ZPF24 to BOX:
SET SAVEPF24 = &ZPF24
SET &ZPF24 = BOX

The &ZPF24 variable controls the function of the F12 key (for terminals with 12 function keys) or the F24 key (for terminals with 24 function keys). The BOX

TEXT Macro

command is processed when F12 or F24 is pressed. Since no native edit command exists with the name BOX, PDF searches for a CLIST or REXX EXEC named BOX.

4. The VPUT service sets the &ZPF24 variable in the profile pool, causing it to take effect.

```
ISPEXEC VPUT (ZPF24) PROFILE
```

5. DEFINE is used to define macros that are to be run when certain edit commands are entered. For example, because of the first DEFINE command, the PFEND macro is run when you enter END.

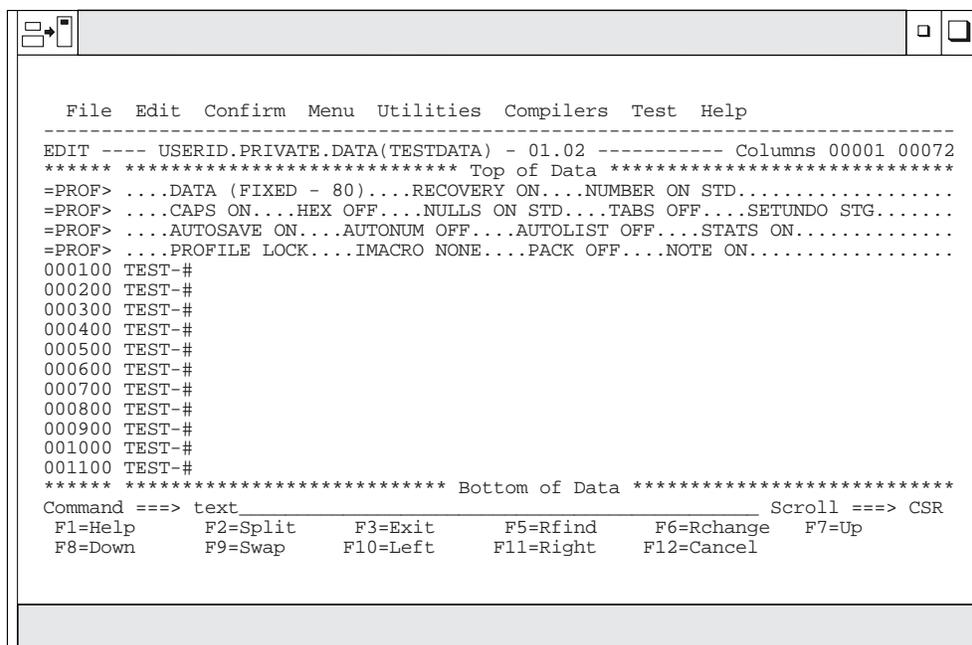
```
ISREDIT DEFINE END ALIAS PFEND
ISREDIT DEFINE CANCEL ALIAS PFCAN
ISREDIT DEFINE QUIT ALIAS CANCEL
```

Notice that since QUIT is defined after CANCEL, both QUIT and CANCEL have become aliases of PFCAN. See “PFCAN Macro” on page 129 to learn about the PFCAN macro.

6. The EXIT statement sets a return code of 0.

```
EXIT CODE(0)
```

To run the TEXT macro, type text on the Command line as shown in Figure 50:



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.02 ----- Columns 00001 00072
***** ***** Top of Data *****
=PROF> ...DATA (FIXED - 80)...RECOVERY ON...NUMBER ON STD.....
=PROF> ...CAPS ON...HEX OFF...NULLS ON STD...TABS OFF...SETUNDO STG.....
=PROF> ...AUTOSAVE ON...AUTONUM OFF...AUTOLIST OFF...STATS ON.....
=PROF> ...PROFILE LOCK...IMACRO NONE...PACK OFF...NOTE ON.....
000100 TEST-#
000200 TEST-#
000300 TEST-#
000400 TEST-#
000500 TEST-#
000600 TEST-#
000700 TEST-#
000800 TEST-#
000900 TEST-#
001000 TEST-#
001100 TEST-#
***** ***** Bottom of Data *****
Command ==> text
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 50. TEXT Macro - Before Running

Figure 51 shows how the macro switches the NUMBER and CAPS mode OFF to prepare for text entry.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.02 ----- Columns 00001 00072
***** ***** Top of Data *****
=PROF> ...DATA (FIXED - 80)...RECOVERY ON...NUMBER OFF.....
=PROF> ...CAPS OFF...HEX OFF...NULLS ON STD...TABS OFF...SETUNDO STG.....
=PROF> ...AUTOSAVE ON...AUTONUM OFF...AUTOLIST OFF...STATS ON.....
=PROF> ...PROFILE LOCK...IMACRO NONE...PACK OFF...NOTE ON.....
000001 TEST-#
000002 TEST-#
000003 TEST-#
000004 TEST-#
000005 TEST-#
000006 TEST-#
000007 TEST-#
000008 TEST-#
000009 TEST-#
000010 TEST-#
000011 TEST-#
***** ***** Bottom of Data *****
Command ==>> _____ Scroll ==>> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 51. TEXT Macro - After Running

PFCAN Macro

The PFCAN macro listed in Figure 52 cancels an edit session, but first it resets F12, which was previously defined by the TEXT macro. TEXT defines F12 to start the BOX macro in Figure 53 on page 130. TEXT and PFCAN can be used in conjunction to save keystrokes.

```

/* PFCAN Reset PF 12, which was defined by */
/* the TEXT macro. */
ISREDIT MACRO /*
  SET ZPF24 = &SAVEPF24 /* Reset PF 12 to its */
  ISPEXEC VPUT (ZPF24) PROFILE /* default value */
  ISREDIT BUILTIN CANCEL /* Cancel the Edit */
                          /* session */
EXIT /*

```

Figure 52. PFCAN Macro

The following list explains the logical sections of the PFCAN macro:

1. F12 is reassigned to its previous setting:

PFCAN Macro

```
ISREDIT VPUT (ZPF24) PROFILE
```

2. The native Edit CANCEL command is processed. If BUILTIN did not precede CANCEL on this statement, PFCAN would issue a CANCEL command that would cause PFCAN to be called recursively.

```
ISREDIT BUILTIN CANCEL
```

BOX Macro

The BOX macro draws a box with its upper left corner at the cursor position. This macro comes in handy when you want to make a note to yourself or others reading the data. You can start the BOX macro in one of three ways:

- Type BOX on the Command line as an edit primary command and press Enter.
- Type KEYS on the Command line, press Enter, set a function key to the BOX macro, and enter the END command.
- Use the TEXT macro, defined earlier, which sets up the function key for BOX and defines the profile values for text entry.

If you have defined a function key for BOX, position the cursor on a data line where you want the box drawn. Press the function key that you have defined to start the BOX macro. After the box is drawn, the cursor is positioned inside, ready for you to type enough text to fill the box.

If any of the macro commands fail, a warning message appears.

```
/*BOX - Draw a box with its upper left corner at the          */
/* cursor position                                           */
/*                                                           */
ISREDIT MACRO
ISREDIT (ROW, COL) = CURSOR                                /* Get cursor position*/
/*                                                           */
ISPEXEC CONTROL ERRORS RETURN                             /*No macro error panel*/
/* Draw box over                                           */
/* existing lines                                          */
/*                                                           */
ISREDIT LINE &ROW = LINE + < &COL '+-----'>
ISREDIT LINE &EVAL(&ROW+1) = LINE + < &COL '|>
ISREDIT LINE &EVAL(&ROW+2) = LINE + < &COL '|>
ISREDIT LINE &EVAL(&ROW+3) = LINE + < &COL '|>
ISREDIT LINE &EVAL(&ROW+4) = LINE + < &COL '|>
ISREDIT LINE &EVAL(&ROW+5) = LINE + < &COL '+-----'>
/*                                                           */
IF &MAXCC > 0 THEN                                       /* If error occurred */ -
DO                                                       /* while overlaying */
SET ZEDSMMSG = &STR(INCOMPLETE BOX) /* lines
SET ZEDLMSG = &STR(NOT ENOUGH LINES/COLUMNS +
TO DRAW COMPLETE BOX)
ISPEXEC SETMSG MSG(ISRZ001) /* Issue error message*/
END
SET &COL = &COL + 2 /* Position cursor
SET &ROW = &ROW + 1 /* within the box
ISREDIT CURSOR = (ROW, COL) /*
EXIT CODE (0)
```

Figure 53. BOX Macro

The following list explains the logical sections of the BOX macro:

1. The variables &ROW and &COL are set to the cursor position.
ISREDIT (ROW, COL) = CURSOR
2. The dialog service allows the macro to handle severe errors, allowing a message to be displayed when the cursor is placed too close to the end of the data. The LINE assignment statement fails if the row it is setting does not exist.
ISREDIT CONTROL ERRORS RETURN

3. The LINE assignment statements overlay existing data on a line with the characters which form a box. LINE uses a merge format to include the existing line data and then a template to put the overlaying data at the cursor column position. The CLIST &EVAL function increments the relative line numbers before the statement is passed to the editor.

```

ISREDIT LINE &ROW          = LINE + < COL '+-----+'>
ISREDIT LINE &EVAL(&ROW+1) = LINE + < COL '|         |'>
ISREDIT LINE &EVAL(&ROW+2) = LINE + < COL '|         |'>
ISREDIT LINE &EVAL(&ROW+3) = LINE + < COL '|         |'>
ISREDIT LINE &EVAL(&ROW+4) = LINE + < COL '|         |'>
ISREDIT LINE &EVAL(&ROW+5) = LINE + < COL '+-----+'>

```

4. The CLIST IF statement checks the &MAXCC variable, and if it is nonzero, calls the dialog service SETMSG to display a message. &MAXCC is a variable updated by the CLIST processor to contain the highest condition code.

```
IF &MAXCC > 0 THEN
```

5. The message used in SETMSG is one of two messages (ISRZ000 and ISRZ001) reserved for macro use. Each message uses two variables:
- &ZEDSMMSG to set the text for the short message (up to 24 characters) that is displayed when the macro ends.
 - &ZEDLMSG to set the text for the long message that appears when the HELP command is entered.

Message ISRZ001 sounds the alarm to indicate an error; message ISRZ000 does not sound the alarm.

```

DO
  SET ZEDSMMSG = &STR(INCOMPLETE BOX)
  SET ZEDLMSG = &STR(NOT ENOUGH LINES/COLUMNS +
    TO DRAW COMPLETE BOX)
  ISPEXEC SETMSG MSG(ISRZ001)
END

```

6. These statements position the cursor within the box to simplify entering text when the panel is redisplayed.

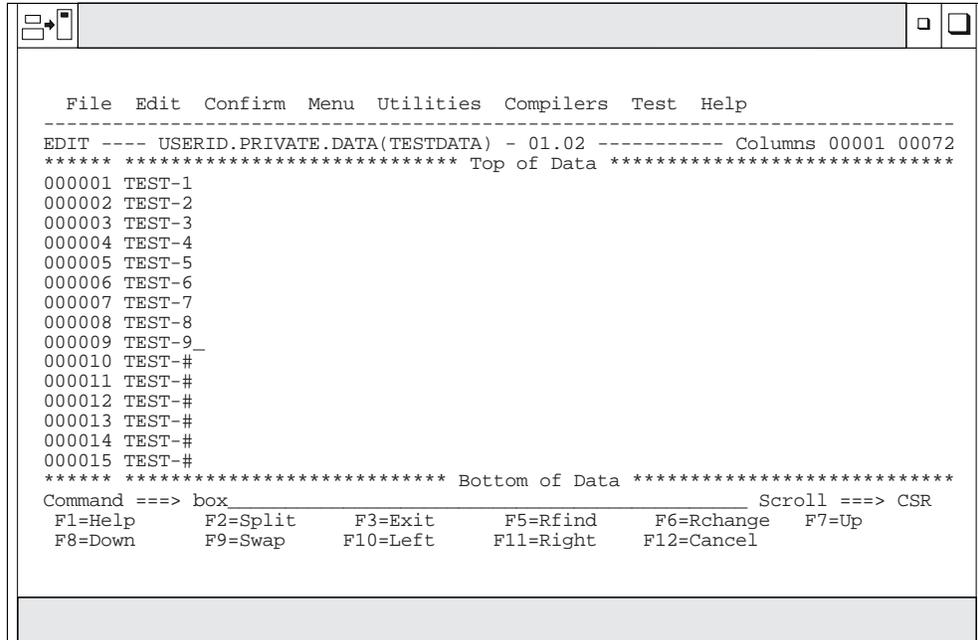
```

SET &COL = &COL + 2
SET &ROW = &ROW + 1
ISREDIT CURSOR = (ROW,COL)

```

The example in Figure 54 shows the cursor placed on line 000009 next to the number 9 before starting the macro.

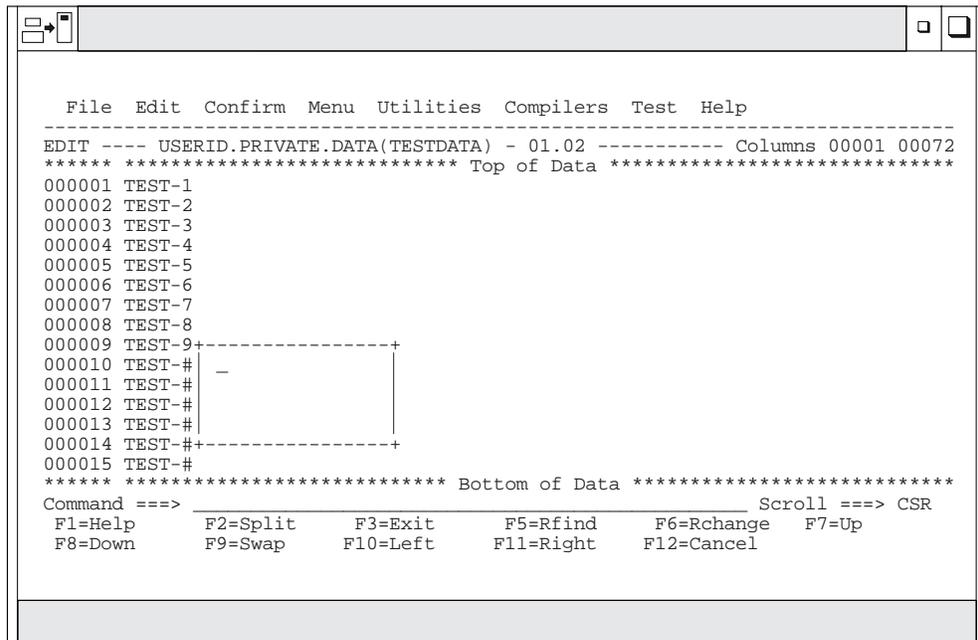
BOX Macro



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.02 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-1
000002 TEST-2
000003 TEST-3
000004 TEST-4
000005 TEST-5
000006 TEST-6
000007 TEST-7
000008 TEST-8
000009 TEST-9
000010 TEST-#
000011 TEST-#
000012 TEST-#
000013 TEST-#
000014 TEST-#
000015 TEST-#
***** ***** Bottom of Data *****
Command ==> box
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel  Scroll ==> CSR
```

Figure 54. BOX Macro - Before Running

When you press Enter, a box appears beside the cursor, as shown in Figure 55.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.02 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-1
000002 TEST-2
000003 TEST-3
000004 TEST-4
000005 TEST-5
000006 TEST-6
000007 TEST-7
000008 TEST-8
000009 TEST-9+-----+
000010 TEST-#  |
000011 TEST-#  |
000012 TEST-#  |
000013 TEST-#  |
000014 TEST-#+-----+
000015 TEST-#
***** ***** Bottom of Data *****
Command ==>
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel  Scroll ==> CSR
```

Figure 55. BOX Macro - After Running

IMBED Macro

The IMBED macro (Figure 56) builds a list of imbed (.im) statements found in the member that is entered as an operand. The list is created at the end of the member currently being edited. The imbed statements are indented under a MEMBER identifier line.

You can start this macro by editing a member, typing IMBED and the name of the member that contains the imbed statements as the operand, and pressing Enter.

```

/* IMBED - Creates a list of imbed statements */
/*
ISREDIT MACRO (MEMBER) /* Member name passed */
/* as input */
ISREDIT LINE_AFTER .ZL='MEMBER &MEMBER' /*Add member ID line */
ISREDIT (LINENBR) = LINENUM .ZL /* Get line number */
/*
ISREDIT COPY AFTER .ZL &MEMBER /* Copy member at end */
ISREDIT (NEWLL) = LINENUM .ZL /* Get new last line# */
/*
IF &LINENBR = &NEWLL THEN /* If no data was -
EXIT CODE(8) /* copied, then exit */
ELSE /*
DO /*
ISREDIT LABEL &EVAL(&LINENBR + 1) /* Label first line */ -
= .FIRST /* copied */
ISREDIT RESET EXCLUDED /* Make sure there are*
/* no previously */
/* excluded lines */
/*
ISREDIT EXCLUDE ALL .FIRST .ZL /* Exclude newly */
/* copied lines */
ISREDIT FIND ALL .IM 1 .FIRST .ZL /* Show lines */
SET FINDRC = &LASTCC /* containing ".im" */
/* in column 1 */
ISREDIT DELETE ALL X .FIRST .ZL /* Delete any lines */
/* still excluded */
ISREDIT (NEWLL) = LINENUM .ZL /* Update last line */
/* number after delete*/
IF &FINDRC = 0 THEN /* If ".im" was found */ -
DO WHILE (&LINENBR < &NEWLL) /* for all remaining */
/* copied lines */
SET LINENBR = &LINENBR + 1 /* Shift all .im lines*/
ISREDIT SHIFT (&LINENBR ) 8 /* right 8 */
END
EXIT CODE(1) /* Place cursor on */
/* command line */

```

Figure 56. IMBED Macro

The following list explains the logical sections of the IMBED macro:

1. Add a line that identifies the member to be searched at the end of IMBEDLIST. The .ZL (or .ZLAST) is always associated with the last line in the data.
ISREDIT LINE_AFTER .ZL = 'MEMBER &MEMBER'
2. Retrieve the line number of the identifier line just added into &LINENBR.
ISREDIT (LINENBR) = LINENUM .ZL
3. Now copy, at the end of IMBEDLIST, the member name that was passed as an input parameter.
ISREDIT COPY AFTER .ZL &MEMBER

IMBED Macro

4. &NEWLL is set to the new last line number of IMBEDLIST.
ISREDIT (NEWLL) = LINENUM .ZL
5. Check to see if any lines were added by the copy. Exit from the macro if no lines were added.
IF &LINENBR = &NEWLL THEN
EXIT CODE(8)
6. Set the .FIRST label on the first line copied. This label is available only to this macro; you do not see it.
ISREDIT LABEL &EVAL(&LINENBR + 1) = .FIRST
7. Excluded lines are deleted later. Therefore, make sure that no lines in the data set are excluded.
ISREDIT RESET EXCLUDED
8. Exclude all lines that were just copied: all the lines in the range .FIRST to .ZL.
ISREDIT EXCLUDE ALL .FIRST .ZL
9. The FIND command is used to find all occurrences of .im starting in column 1 of the copied lines. This shows (unexcludes) the lines to keep. If .im was not found on any line, &FINDRC will be 4.
ISREDIT FIND ALL .IM 1 .FIRST .ZL
SET FINDRC = &LASTCC
10. All the lines still excluded are now deleted.
ISREDIT DELETE ALL X .FIRST .ZL
11. Obtain the last line number again, because it will have changed if lines were deleted.
ISREDIT (NEWLL) = LINENUM .ZL
12. If .im lines were found, loop using a column shift to indent them under the member identifier line. Note that &LINENBR is still associated with the identifier line.
IF &FINDRC = 0 THEN
DO WHILE (&LINENBR < &NEWLL)
SET LINENBR = &LINENBR + 1
ISREDIT SHIFT &LINENBR) 8
END

LIST is a member with several imbed statements; see Figure 57.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(LIST) - 01.00 ----- Columns 00001 00068
***** ***** Top of Data *****
000001 .im imbedname1
000002 *****
000003 .im imbedname2
000004 *****
000005 .im imbedname3
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
    
```

Figure 57. LIST with Imbed Statements

When you run the IMBED macro by typing IMBED LIST on the Command line of TESTDATA, a list of the imbeds in LIST appears at the end of the data. See Figure 58.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      USERID.PRIVATE.SCRIPT(TESTDATA) - 01.03      Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-#
000002 TEST-#
000003 TEST-#
000004 TEST-#
000005 TEST-#
000006 TEST-#
000007 TEST-#
000008 TEST-#
000009 TEST-#
000010 TEST-#
000011 MEMBER LIST
000012          .im imbedname1
000013          .im imbedname2
000014          .im imbedname3
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F10=Right   F12=Cancel
    
```

Figure 58. IMBED Macro - After Running

ALLMBRS Macro

The ALLMBRS macro (Figure 59 on page 136) uses PDF library access services to determine each member name in the library being edited.

ALLMBRS Macro

This macro invokes the edit service for each member in the library, except the member currently being edited, passing a user-specified edit macro on the edit service invocation. The ALLMBRS *macname* command, where *macname* is the name of the macro to be invoked against each member, starts the service.

This macro can aid in making repetitive changes to all members of a data set, or in searching all members for a specific string of data.

```
/*REXX*****  
/* ISPF edit macro to process all members of partitioned data set, */  
/* running a second, user-specified, ISPF edit macro against each */  
/* member. */  
/* */  
/* To run: */  
/* Enter "ALLMBRS macname" on the command line, where macname is */  
/* the macro you want run against each member. */  
/******  
  
'ISREDIT MACRO (NESTMAC)'  
  
/******  
/* Get dataid for data set and issue LMOPEN */  
/******  
'ISREDIT (DATA1) = DATAID'  
'ISREDIT (CURMEM) = MEMBER'  
Address ispexec 'LMOPEN DATAID('data1') OPTION(INPUT)'  
member = ' '  
lmrc = 0  
  
/******  
/* Loop through all members in the PDS, issuing the EDIT service for */  
/* each. The macro specified on the ALLMEMS invocation is passed as */  
/* an initial macro on the EDIT service call. */  
/******  
Do While lmrc = 0  
  Address ispexec 'LMMLIST DATAID('data1') OPTION(LIST),  
                MEMBER(MEMBER) STATS(NO)'  
  
  lmrc = rc  
  If lmrc = 0 & member ^= curmem Then  
    do  
      Say 'Processing member' member  
      Address ispexec 'EDIT DATAID('data1') MEMBER('member')  
                    MACRO('nestmac)'  
    end  
  End  
  
/******  
/* Free the member list and close the dataid for the PDS. */  
/******  
Address ispexec 'LMMLIST DATAID('data1') OPTION(FREE)'  
Address ispexec 'LMCLOSE DATAID('data1)'  
  
Exit 0
```

Figure 59. ALLMBRS Macro

To start the ALLMBRS macro, edit a member (either new or existing), type ALLMBRS *macname*, where *macname* is the name of the macro you wish to invoke against each member of the data set, and press enter. For example, if the name of the macro to be invoked is IMBED, type:

```
Command ==> ALLMBRS IMBED
```

The following list explains the logical sections of the ALLMBRS macro:

1. The MACRO command identifies NESTMAC as the variable to contain the name of the macro that is passed on the edit service invocation for each member. If no parameter is passed to ALLMBRS, NESTMAC is blank.

```
ISREDIT MACRO (NESTMAC)
```
2. The DATAID assignment statement returns a data ID in the variable DATA1. The data ID identifies the concatenation of data sets currently being edited.

```
ISREDIT (DATA1) = DATAID
```
3. The name of the member currently being edited is returned in CURMEM.

```
ISREDIT (MEMBER) = CURMEM
```
4. The data set (or sets) identified by the data ID obtained earlier is opened for input to allow the LMMLIST service to be called later. No return code checking is done because it is presumed that if the data set is being edited, it can be successfully processed by LMOPEN.

```
Address ispexec 'LMOPEN DATAID('data1') OPTION(INPUT)'
```
5. The variable to hold the name of the next member to be processed, and the return code from the LMMLIST service are initialized.

```
member = ' '  
lmrc = 0
```
6. The exec loops to process all members returned by LMMLIST. Variable LMRC is set to 4 when the end of the member list is reached, stopping the loop.

```
Do While lmrc = 0
```
7. Obtain the next member in the list. If this is the first invocation of LMMLIST, the first member in the list is returned. The member name is returned in variable MEMBER, and variable LMRC is set to the return code from LMMLIST.

```
Address ispexec 'LMMLIST DATAID('data1') OPTION(LIST),  
MEMBER(MEMBER) STATS(NO)'  
lmrc = rc
```
8. If LMMLIST returns a 0, indicating a member name was returned, and if the member returned is not the member currently being edited, the member is processed.

```
If lmrc = 0 Then  
do
```
9. The Rexx SAY statement is used to write line-I/O messages. As the macro processes each member, the member name appears on the terminal to keep you informed about what is happening. An alternative to the SAY statement would be to display a panel showing the member name after issuing the ISPEXEC CONTROL DISPLAY LOCK service.

```
Say 'Processing member' member
```
10. The EDIT service is invoked on the member returned by LMMLIST. The macro specified on invocation of ALLMBRS is passed as an initial macro on the edit service.

```
Address ispexec 'EDIT DATAID('data1') MEMBER('member')  
MACRO('nestmac)'
```
11. When the LMMLIST service returns a non-zero value, the loop is exited and the cleanup begins. LMMLIST is called to free the member list, and the LMCLOSE service is called to close the data set or sets associated with the data ID.

```
Address ispexec 'LMMLIST DATAID('data1') OPTION(FREE)'  
Address ispexec 'LMCLOSE DATAID('data1)'
```

FINDCHGS Macro

The FINDCHGS macro (Figure 60) identifies the lines most recently changed by showing only those lines and excluding all others. When no level is passed, the latest level is assumed. A label range can also be passed to FINDCHGS to limit the search. This macro relies on the modification level maintained by the editor for members with numbers and ISPF statistics.

Operands can also be specified. For example, to show lines with level 8 or greater on a line range:

Command ==> FINDCHGS 8 .FIRST .LAST

```

/*
/* FINDCHGS shows the most recent changes to a data set.
/*
/*
ISREDIT MACRO (SEARCH,PARMS) /* Macro accepts args: */
/* level & label range */
ISREDIT (SAVE) = USER_STATE /* Save user info/csr pos*/
ISREDIT (NUMBER, NUMTYPE) = NUMBER /* Get the number mode */
SET SYSVAL = &NUMTYPE /* Parse the number type*/
READDVAL STD COBOL DISPLAY
ISREDIT (STATS) = STATS /* Get the stats mode */
ISREDIT (LEVEL) = LEVEL /* Get the current level*/
IF &SEARCH = &STR() | &SUBSTR(1:1,&SEARCH) = &STR(.) THEN -
DO /* If first arg is null */
SET PARMS = &STR(&SEARCH &PARMS) /* or a label, no level */
/* was specified */
/* Move the first arg
/* back into the parms */
/* Default to the
/* current level */
SET SEARCH = &LEVEL
END

IF &STATS = OFF | &NUMBER = OFF | &STD = NOSTD THEN -
DO /* If level not possible*/
SET ZEDSMMSG = &STR (INVALID DATA)
SET ZEDLMSG = &STR (BOTH NUMBER AND STATS MODE MUST BE ON)
ISPEXEC SETMSG MSG(ISRZ001) /* Set an error message */
EXIT CODE(8)
END
IF &DATATYPE(&SEARCH) = CHAR THEN -
DO /* First arg not number */
SET ZEDSMMSG = &STR (INVALID ARG)
SET ZEDLMSG = &STR(SEARCH ARGUMENT MUST BE FIRST)
ISPEXEC SETMSG MSG(ISRZ001) /* Set an error message */
EXIT CODE(8)
END

ISREDIT NUMBER = OFF /* The nums become data */
ISREDIT (RECFM) = RECFM /* Get record format */
IF &RECFM = F THEN -
DO /* If record format is */
ISREDIT (LRECL) = LRECL /* fixed, get maximum */
SET COL1 = &LRECL - 1 /* column in data. Use */
SET COL2 = &LRECL /* the last 2 columns */
END /* to find the lvl */
ELSE DO
SET COL1 = 7 /* Assume RECFM = V */
SET COL2 = 8
END

ISREDIT EXCLUDE ALL /* Exclude all lines */
DO WHILE &SEARCH <= &LEVEL
ISREDIT FIND ALL 'SEARCH' &COL1 &COL2 &PARMS/*find the level */

SEARCH = &SEARCH + 1 /* Increment level num */
END
EXIT

ISREDIT USER_STATE = (SAVE) /* Restore saved user */
/* values */
EXIT CODE (1) /* Place CSR on cmd line*/

```

Figure 60. FINDCHGS Macro

The following list explains the logical sections of the FINDCHGS macro:

1. FINDCHGS allows three optional parameters to be passed: a search level and two labels (a label range). If all three are passed, PARMs contains two labels.
ISREDIT MACRO (SEARCH,PARMS)

2. The following statements save user information, number mode and type, last find string, cursor location, and other profile and status information. Also, stats mode and the current modification level for parameter checking are retrieved, and the three-part number type is divided into three variables.

```
ISREDIT (SAVE) = USER_STATE
ISREDIT (NUMBER, NUMTYPE) = NUMBER
SET SYSDVAL = &NUMTYPE
READDVAL STD COBOL DISPLAY
ISREDIT (STATS) = STATS
ISREDIT (LEVEL) = LEVEL
```

3. FINDCHGS requires that the modification level be entered first if it is specified. This check allows the level to default to the current (highest) modification level. A label range can be specified without a level number; PARMs is reset to capture both labels.

```
IF &SEARCH = &STR() | &SUBSTR(1:1,&SEARCH) = &STR(;) THEN -
DO
    SET PARMs = &STR(&SEARCH &PARMs)
    SET SEARCH = &LEVEL
END
```

4. Check to see if the member modification level is maintained. If not, issue an error message and exit the macro.

```
IF &STATS = OFF | &NUMBER = OFF | &STD = NOSTD THEN -
DO
    SET ZEDSMsG = &STR(INVALID DATA)
    SET ZEDLMSG = &STR(BOTH NUMBER AND STATS MODE MUST BE ON)
    ISPEXEC SETMSG MSG(ISRZ001)
    EXIT CODE(8)
END
```

5. A CLIST DATATYPE function is used to check if the first parameter is valid (a number). If it is not valid, issue an error message and exit from the macro.

```
IF &DATATYPE(&SEARCH) = CHAR THEN -
DO
    SET ZEDSMsG = &STR(INVALID ARG)
    SET ZEDLMSG = &STR(SEARCH STRING MUST BE FIRST)
    ISPEXEC SETMSG MSG(ISRZ001)
    EXIT CODE(8)
END
```

6. Now that validity checks have been passed you can set number mode off. This allows you to treat the number field, which contains the level number, as data.

```
ISREDIT NUMBER = OFF
```

7. Set &COL1 and &COL2 to the columns containing the level numbers.

```
ISREDIT (RECFM) = RECFM
IF &RECFM = F THEN -
DO
    ISREDIT (LRECL) = LRECL
    SET COL1 = &LRECL - 1
    SET COL2 = &LRECL
END
ELSE DO
    SET COL1 = 7
    SET COL2 = 8
END
```

8. Exclude all lines.

```
ISREDIT EXCLUDE ALL
```

FINDCHGS Macro

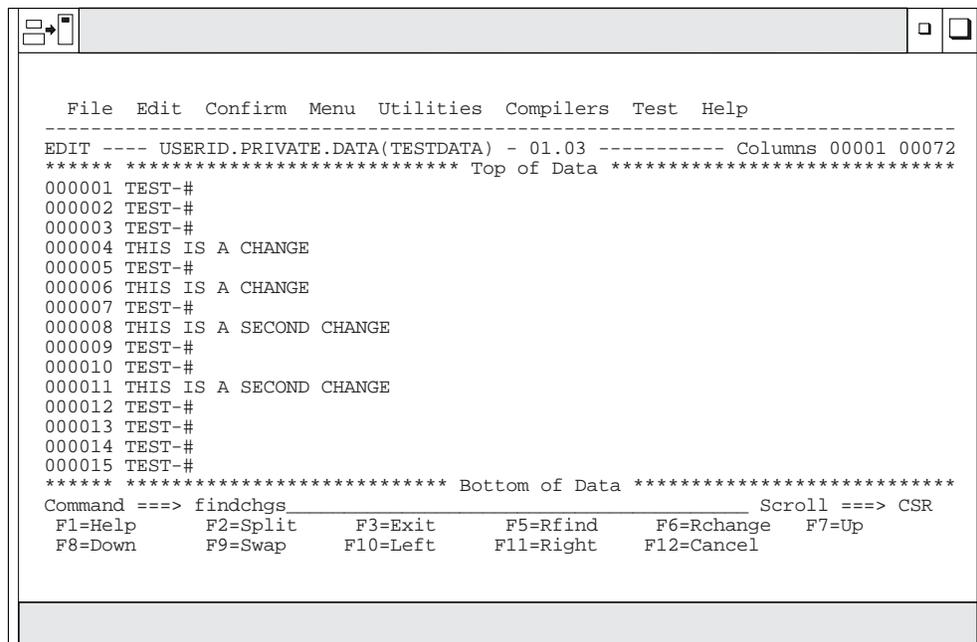
9. For each level, find all occurrences of the current modification level. If a label range was specified, it is in the PARMs variable. All lines with matching levels are excluded.

```
DO WHILE &SEARCH <= &LEVEL
  ISREDIT FIND ALL '&SEARCH' &COL1 &COL2 &PARMS
  SEARCH = &SEARCH + 1
END
```

10. Restore user values, especially number mode.

```
ISREDIT USER_STATE = (SAVE)
```

In the example in Figure 61 the data contains lines that you have changed. When you press Enter, the FINDGHGS macro displays the changed lines and



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 TEST-#
000002 TEST-#
000003 TEST-#
000004 THIS IS A CHANGE
000005 TEST-#
000006 THIS IS A CHANGE
000007 TEST-#
000008 THIS IS A SECOND CHANGE
000009 TEST-#
000010 TEST-#
000011 THIS IS A SECOND CHANGE
000012 TEST-#
000013 TEST-#
000014 TEST-#
000015 TEST-#
***** ***** Bottom of Data *****
Command ==> findchgs          Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap       F10=Left    F11=Right    F12=Cancel
```

Figure 61. FINDCHGS Macro - Before Running

excludes the others, as shown in Figure 62 on page 141.

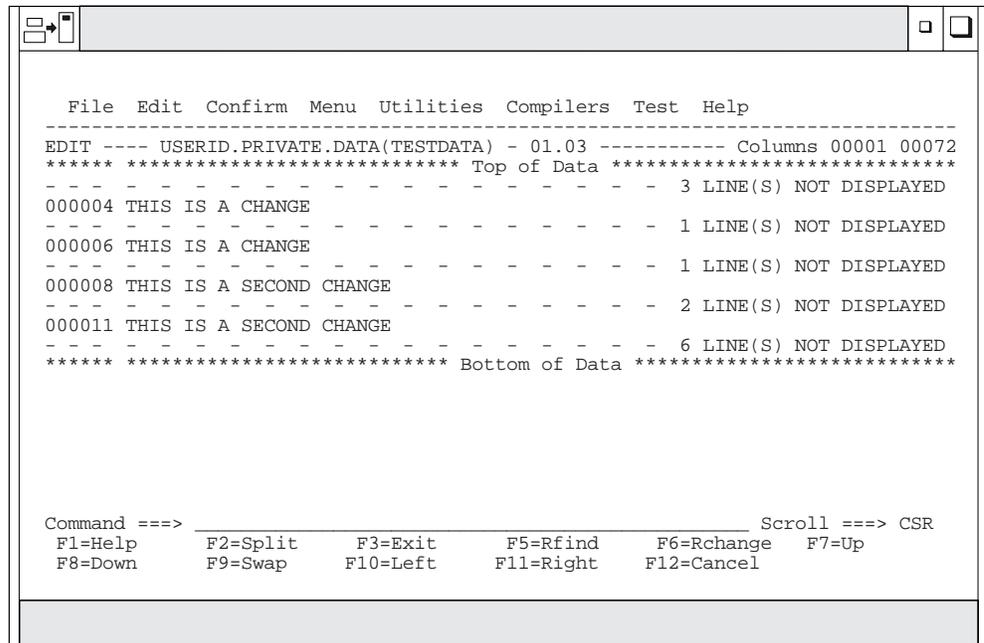


Figure 62. FINDCHGS Macro - After Running

MASKDATA Macro

The MASKDATA macro (Figure 63 on page 142) allows data in the mask line to overlay lines. It can be used to place a comment area over existing lines in a member.

Before starting this macro, you must first specify two things: a mask line and the range of lines it overlays. See “MASKLINE—Set or Query the Mask Line” on page 364 for information on creating mask lines.

Specify the range of lines by using either an OO or \$\$ line command. You can use O, OO, On, or \$, \$\$, \$n, where *n* is the number of lines.

An O line command specifies that mask line data overlays only blanks in the line data. A \$ line command specifies that non-blank mask line data overlays the line data. Once the mask line and range of lines have been specified, type MASKDATA on the Command line and press Enter.

MASKDATA Macro

```

/*
/* MASKDATA - Overlay a line with data from the mask line.
/* Use either line command 0 or $ to indicate
/* which line to overlay. 0 causes
/* nondestructive overlay, and $ causes a
/* destructive overlay.
/*
/*
/* Wait to process
/* "O" and "$" reserved
/* for macro
/* If specified, get
/* command entered
/* and line number
/* range
/* Loop to merge data
/* based on which
/* line command was
/* entered. If $
/* overlay data- else
/* do not overlay
/* Increment line num
/*
/* Set prompt messages
/*
/* Set return code to
/* 12 to keep command
/* in command area
*/
ISREDIT MACRO NOPROCESS
ISREDIT PROCESS RANGE 0 $
IF &LASTCC = 0 THEN
DO
ISREDIT (CMD) = RANGE_CMD
ISREDIT (FIRST) = LINENUM .ZFRANGE
ISREDIT (LAST) = LINENUM .ZLRANGE
DO WHILE &FIRST LE &LAST
IF &CMD = $ THEN
ISREDIT LINE &FIRST = (LINE) + MASKLINE
ELSE
ISREDIT LINE &FIRST = MASKLINE + (LINE)
SET FIRST = &FIRST + 1
END
SET RC = 0
ELSE
DO
SET ZEDSMMSG = &STR(ENTER "O"/"$" LINE CMD)
SET ZEDLMSG = &STR("MASKDATA" REQUIRES AN "O" OR +
"$" CMD TO INDICATE LINE(S) MERGED WITH MASKLINE)
ISPEXEC SETMSG MSG(ISRZ001)
SET RC = 12
END
EXIT CODE(&RC)

```

Figure 63. MASKDATA Macro

The following list explains the logical sections of the MASKDATA macro:

1. The NOPROCESS keyword on the MACRO command allows the macro to control when user input (changes to data and line commands) is processed.
ISREDIT MACRO NOPROCESS
2. Now process user input and check if certain line commands are entered. The O and \$ following the RANGE keyword specify the line commands to be processed by this macro.
ISREDIT PROCESS RANGE 0 \$
3. A zero return code shows that you entered an O or \$ in any of its valid forms: 00-00, 0n, and so forth.
IF &LASTCC = 0 THEN
4. &CMD is set to O or \$, whichever command was entered.
ISREDIT (CMD) = RANGE_CMD

- &LINE1 and &LINE2 contain the first and last line numbers of the lines specified by the user line commands.

```
ISREDIT (FIRST) = LINENUM .ZFRANGE
ISREDIT (LAST) = LINENUM .ZLRANGE
DO WHILE &FIRST LE &LAST
```

- Each line that you specify is merged with data from the mask line. Note the use of the LINE keyphrase on both sides of the assignment. The line command entered controls how the data is merged. An O specifies that the mask line data only overlays where the line contains blanks. A \$ specifies that non-blank mask line data overlays line data.

```
IF &CMD = $ THEN
  ISREDIT LINE &FIRST = (LINE) + MASKLINE
ELSE
  ISREDIT LINE &FIRST = MASKLINE + (LINE)
```

- When no line command is entered, issue a prompt message. Set a return code of 12 to keep MASKDATA displayed on the Command line.

```
SET ZEDSMMSG = &STR(ENTER "O"/"$" LINE CMD)
SET ZEDLMSG = &STR("MASKDATA" REQUIRES AN "O" OR +
  "$" CMD TO INDICATE LINE(S) MERGED WITH MASKLINE)
ISPEXEC SETMSG MSG(ISRZ001)
SET RC = 12
```

In the example shown in Figure 64, the mask line is specified and the range of lines is set with the destructive \$\$ line command.

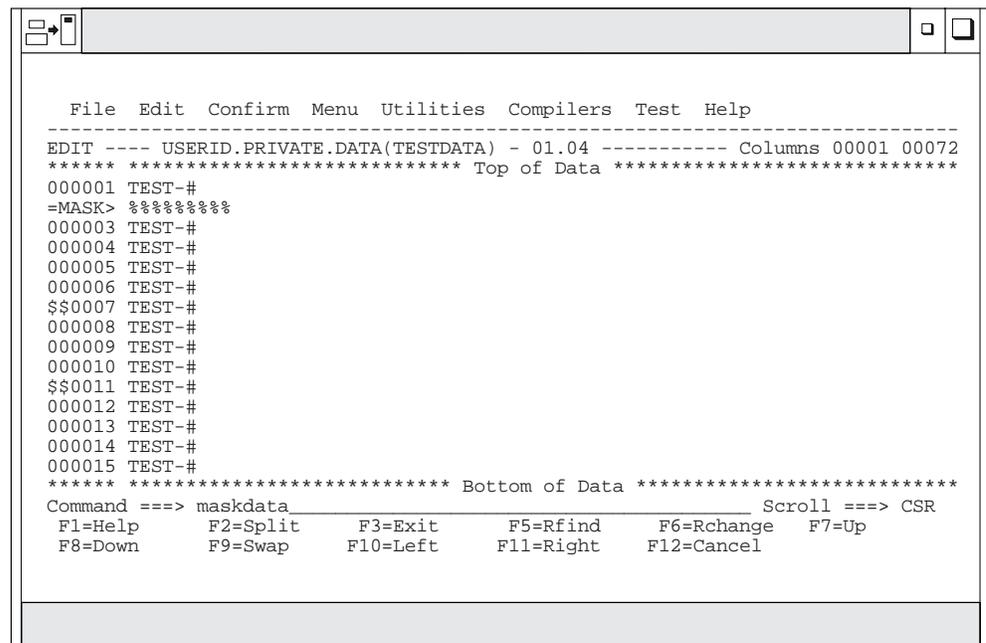


Figure 64. MASKDATA Macro - Before Running

When you press Enter, the macro overlays the mask line onto the specified range of lines, as shown in Figure 65 on page 144.

MASKDATA Macro

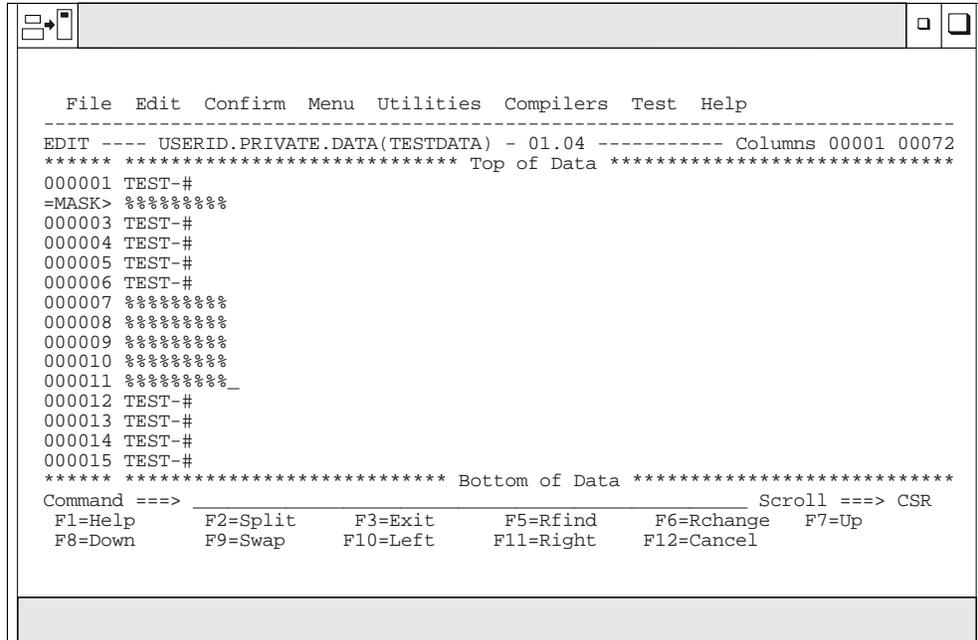


Figure 65. MASKDATA Macro - After Running

Part 3. Command Reference

Chapter 9. Edit Line Commands	153	LC—Convert Characters to Lowercase	180
Rules for Entering Line Commands	153	Syntax	180
Edit Line Command Notation Conventions	154	Description	180
Line Command Summary	154	Example	181
(—Column Shift Left	156	M—Move Lines	182
Syntax	156	Syntax	182
Description	156	Description	182
Example	157	Example	183
)—Column Shift Right	158	MASK—Define Masks	184
Syntax	158	Syntax	184
Description	158	Description	184
Example	158	Example	185
<—Data Shift Left	159	MD—Make Dataline	186
Syntax	160	Syntax	186
Description	160	Description	186
Example	160	Example	187
>—Data Shift Right	162	O—Overlay Lines	188
Syntax	162	Syntax	188
Description	162	Description	188
Example	162	Example	189
A—Specify an “After” Destination	163	R—Repeat Lines	190
Syntax	164	Syntax	191
Description	164	Description	191
Example	164	Example	191
B—Specify a “Before” Destination	166	S—Show Lines	193
Syntax	166	Syntax	193
Description	166	Description	193
Example	167	Example	194
BOUNDS—Define Boundary Columns	168	TABS—Control Tabs	195
Syntax	168	Syntax	195
Description	168	Description	195
Example	169	Examples	195
C—Copy Lines	170	Using Software and Hardware Tabs	196
Syntax	170	Using Software Tab Fields	196
Description	170	TE—Text Entry	197
Example	171	Syntax	197
COLS—Identify Columns	172	Description	197
Syntax	172	Example	198
Description	172	TF—Text Flow	200
Example	173	Syntax	200
D—Delete Lines	174	Description	200
Syntax	174	Example	201
Description	174	TS—Text Split	202
Example	174	Syntax	202
F—Show the First Line	175	Description	202
Syntax	176	Examples	202
Description	176	UC—Convert Characters to Uppercase	203
Example	176	Syntax	204
I—Insert Lines	177	Description	204
Syntax	177	Example	204
Description	177	X—Exclude Lines	205
Example	178	Syntax	206
L—Show the Last Line(s)	179	Description	206
Syntax	179	Example	206
Description	179		
Example	179	Chapter 10. Edit Primary Commands	209

Edit Primary Command Notation Conventions	209	Description	238
Edit Primary Command Summary	209	Examples	238
AUTOLIST—Create a Source Listing		EDIT—Edit from within an Edit Session	239
Automatically	213	Syntax	239
Syntax	214	Description	239
Description	214	Example	240
Example	214	END—End the Edit Session	241
AUTONUM—Number Lines Automatically	215	Syntax	241
Syntax	215	Description	241
Description	215	Example	242
Example	216	EXCLUDE—Exclude Lines from the Display	242
AUTOSAVE—Save Data Automatically	217	Syntax	242
Syntax	217	Description	243
Description	217	Examples	243
Example	218	FIND—Find a Data String	243
BOUNDS—Control the Edit Boundaries	218	Syntax	244
Syntax	218	Description	244
Description	219	Examples	245
Examples	219	FLIP—Reverse Exclude Status of Lines	245
BUILTIN—Process a Built-In Command	219	Syntax	245
Syntax	219	Description	245
Description	219	Example	246
Example	219	HEX—Display Hexadecimal Characters	247
BROWSE—Browse from within an Edit Session	220	Syntax	248
Syntax	220	Description	248
Description	220	Examples	248
Example	220	HILITE—Enhanced Edit Coloring	250
CANCEL—Cancel Edit Changes	221	Syntax	250
Syntax	221	Description	253
Description	221	IMACRO—Specify an Initial Macro	253
Example	221	Syntax	253
CAPS—Control Automatic Character Conversion	221	Examples	253
Syntax	221	LEVEL—Specify the Modification Level Number	254
Description	221	Syntax	254
Example	222	Description	254
CHANGE—Change a Data String	222	Example	254
Syntax	223	LOCATE—Locate a Line	255
Description	223	Specific Locate Syntax	256
Examples	224	Generic Locate Syntax	256
COMPARE—Edit Compare	224	Examples	257
Command Syntax	225	MODEL—Copy a Model into the Current Data Set	257
Examples	226	Model Name Syntax	257
COPY—Copy Data	227	Class Name Syntax	258
Syntax	227	Example	258
Description	228	MOVE—Move Data	260
Example	229	Syntax	260
CREATE—Create Data	231	Description	261
Syntax	231	Example	262
Description	231	NONUMBER—Turn Off Number Mode	264
Example	232	Syntax	264
CUT—Cut and Save Lines	235	Description	264
Syntax	235	Example	264
Description	235	NOTES—Display Model Notes	264
Example	236	Syntax	265
DEFINE—Define a Name	236	Description	265
Syntax	236	Examples	265
Description	237	NULLS—Control Null Spaces	265
Stacking DEFINE Commands	237	Syntax	265
Examples	237	Description	266
DELETE—Delete Lines	238	Examples	266
Syntax	238	NUMBER—Generate Sequence Numbers	266

Syntax	266
Description	267
Examples	268
PACK—Compress Data	268
Syntax	268
Examples	268
PASTE—Move or Copy Lines from Clipboard	268
Syntax	268
Description	269
Example	269
PRESERVE - Enable Saving of Trailing Blanks	269
Syntax	269
Description	269
Examples	270
PROFILE—Control and Display Your Profile	270
Profile Control Syntax	270
Profile Lock Syntax	270
Profile Reset Syntax	271
Description	271
Example	272
RCHANGE—Repeat a Change	273
Syntax	273
Description	273
RECOVERY—Control Edit Recovery	273
Syntax	273
Description	274
RENUM—Renumber Data Set Lines	274
Syntax	274
Description	275
Example	276
REPLACE—Replace Data	277
Syntax	277
Description	278
Example	278
RESET—Reset the Data Display	280
Syntax	281
Description	281
Examples	281
RFIND—Repeat Find	282
Syntax	282
RMACRO—Specify a Recovery Macro	282
Syntax	282
Description	283
Example	283
SAVE—Save the Current Data	283
Syntax	283
Description	283
Example	284
SETUNDO—Set the UNDO Mode	284
Syntax	284
Description	284
Example	285
SORT—Sort Data	286
Syntax	286
Description	286
Sorting Data Without Operands	286
Limiting the SORT Command	287
Sorting DBCS Data	287
Examples	287
STATS—Generate Library Statistics	288
Syntax	288

Examples	288
SUBMIT—Submit Data for Batch Processing	288
Syntax	288
Description	288
Examples	288
TABS—Define Tabs	289
Syntax	289
Example	290
UNDO—Reverse Last Edit Interaction	290
Syntax	290
Description	291
Example	292
UNNUMBER—Remove Sequence Numbers	293
Syntax	294
Description	294
Example	294
VERSION—Control the Version Number	295
Syntax	295
Description	295
Example	295
VIEW—View from within an Edit Session	296
Syntax	297
Description	297
Example	297

Chapter 11. Edit Macro Commands and Assignment Statements

Assignment Statements	299
Edit Macro Command Notation Conventions	299
Edit Macro Command Summary	300
AUTOLIST—Set or Query Autolist Mode	307
Macro Command Syntax	307
Assignment Statement Syntax	308
Return Codes	308
Examples	308
AUTONOM—Set or Query Autonom Mode	308
Macro Command Syntax	308
Assignment Statement Syntax	308
Description	309
Return Codes	309
Examples	309
AUTOSAVE—Set or Query Autosave Mode	309
Macro Command Syntax	310
Assignment Statement Syntax	310
Description	310
Return Codes	310
Examples	311
BLKSIZE—Query the Block Size	311
Assignment Statement Syntax	311
Return Codes	311
Example	311
BOUNDS—Set or Query the Edit Boundaries	311
Macro Command Syntax	312
Assignment Statement Syntax	312
Description	312
Return Codes	312
Examples	312
BROWSE—Browse from within an Edit Session	313
Macro Command Syntax	313
Description	313
Return Codes	314
Examples	314

BUILTIN—Process a Built-In Command	314	Return Codes	330
Macro Command Syntax	314	Example	330
Description	314	DATAID—Query Data ID	331
Return Codes	314	Assignment Statement Syntax	331
Examples	314	Description	331
CANCEL—Cancel Edit Changes	315	Return Codes	331
Macro Command Syntax	315	Example	331
Description	315	DATASET—Query the Current Data Set Name	331
Return Codes	315	Assignment Statement Syntax	331
Example	315	Return Codes	332
CAPS—Set or Query Caps Mode	315	Example	332
Macro Command Syntax	315	DEFINE—Define a Name	332
Assignment Statement Syntax	315	Macro Command Syntax	332
Description	316	Description	333
Return Codes	316	Return Codes	333
Examples	316	Examples	333
CHANGE—Change a Search String	316	DELETE—Delete Lines	334
Macro Command Syntax	317	Macro Command Syntax	334
Description	318	Description	334
Return Codes	318	Return Codes	334
Example	319	Examples	334
CHANGE_COUNTS—Query Change Counts	319	DISPLAY_COLS—Query Display Columns	335
Assignment Statement Syntax	319	Assignment Statement Syntax	335
Return Codes	319	Description	335
Examples	319	Return Codes	335
COMPARE—Edit Compare	320	Example	335
Macro Command Syntax	320	DISPLAY_LINES—Query Display Lines	336
Return Codes	321	Assignment Statement Syntax	336
Compare Examples	321	Return Codes	336
COPY—Copy Data	322	Example	336
Macro Command Syntax	322	DOWN—Scroll Down	336
Return Codes	323	Macro Command Syntax	336
Examples	323	Description	337
CREATE—Create a Data Set Member	323	Return Codes	337
Macro Command Syntax	323	Examples	337
Description	324	EDIT—Edit from within an Edit Session	338
Return Codes	324	Macro Command Syntax	338
Example	324	Description	338
CTL_LIBRARY—Query Controlled Library Status	324	Return Codes	338
Assignment Statement Syntax	324	Example	338
Return Codes	325	END—End the Edit Session	338
Example	326	Macro Command Syntax	338
CURSOR—Set or Query the Cursor Position	326	Description	339
Assignment Statement Syntax	326	Return Codes	339
Description	326	Example	339
Return Codes	327	EXCLUDE—Exclude Lines from the Display	339
Examples	327	Macro Command Syntax	339
CUT—Cut and Save Lines	328	Description	340
Syntax	328	Return Codes	341
Description	328	Examples	341
Return Codes	328	EXCLUDE_COUNTS—Query Exclude Counts	341
Examples	329	Assignment Statement Syntax	341
DATA_CHANGED—Query the Data Changed Status	329	Return Codes	342
Assignment Statement Syntax	329	Example	342
Description	329	FIND—Find a Search String	342
Return Codes	329	Macro Command Syntax	342
Example	329	Description	343
DATA_WIDTH—Query Data Width	330	Return Codes	344
Assignment Statement Syntax	330	Examples	344
Description	330	FIND_COUNTS—Query Find Counts	344
		Assignment Statement Syntax	344

Return Codes	344	Return Codes	359
Example	345	Examples	359
FLIP—Reverse Exclude Status of Lines	345	LINENUM—Query the Line Number of a Labeled Line	359
Assignment Statement Syntax	345	Assignment Statement Syntax	359
Return Codes	345	Return Codes	359
Examples	345	Description	360
FLOW_COUNTS—Query Flow Counts	345	Examples	360
Assignment Statement Syntax	345	LOCATE—Locate a Line	360
Return Codes	346	Specific Locate Syntax	360
Example	346	Generic Locate Syntax	360
HEX—Set or Query Hexadecimal Mode	346	Return Codes	361
Macro Command Syntax	346	Examples	361
Assignment Statement Syntax	346	LRECL—Query the Logical Record Length	362
Description	347	Assignment Statement Syntax	362
Return Codes	347	Description	362
Examples	347	Return Codes	362
HILITE—Enhanced Edit Coloring	347	Example	362
Macro Command Syntax	348	MACRO—Identify an Edit Macro	362
Description	350	Macro Command Syntax	362
Return Codes	350	Description	363
IMACRO—Set or Query an Initial Macro	351	Return Codes	363
Macro Command Syntax	351	Examples	363
Assignment Statement Syntax	351	MACRO_LEVEL—Query the Macro Nesting Level	364
Return Codes	351	Assignment Statement Syntax	364
Examples	351	Description	364
INSERT—Prepare Display for Data Insertion.	351	Return Codes	364
Macro Command Syntax	352	Example	364
Description	352	MASKLINE—Set or Query the Mask Line	364
Return Codes	352	Assignment Statement Syntax	364
Example	352	Description	365
LABEL—Set or Query a Line Label	352	Return Codes	365
Assignment Statement Syntax	352	Examples	365
Description	353	MEMBER—Query the Current Member Name	365
Return Codes	353	Assignment Statement Syntax	365
Example	353	Return Codes	365
LEFT—Scroll Left.	353	Example	366
Macro Command Syntax	354	MEND—End a Macro in the Batch Environment	366
Description	354	Macro Command Syntax	366
Return Codes	354	Description	366
Example	354	Return Codes	366
LEVEL—Set or Query the Modification Level	354	Example	366
Number	354	MODEL—Copy a Model into the Current Data Set	366
Macro Command Syntax	355	Macro Command Model Name Syntax	367
Assignment Statement Syntax	355	Macro Command Class Name Syntax	367
Return Codes	355	Return Codes	368
Examples	355	Example	368
LINE—Set or Query a Line from the Data Set	355	MOVE— Move a Data Set or a Data Set Member	368
Assignment Statement Syntax	355	Macro Command Syntax	368
Description	356	Description	368
Return Codes	356	Return Codes	369
Examples	356	Examples	369
LINE_AFTER—Add a Line to the Current Data Set.	356	NONUMBER—Turn Off Number Mode	369
Assignment Statement Syntax	356	Syntax	369
Description	357	Description	369
Return Codes	357	Return Codes	369
Examples	358	Example	369
LINE_BEFORE—Add a Line to the Current Data Set.	358	NOTES—Set or Query Note Mode	370
Assignment Statement Syntax	358	Macro Command Syntax	370
Description	359	Assignment Statement Syntax	370
		Return Codes	370

Examples	370	Examples	384
NULLS—Set or Query Nulls Mode	370	RENUM—Renumber Data Set Lines	384
Macro Command Syntax	371	Macro Command Syntax	384
Assignment Statement Syntax	371	Return Codes	385
Description	371	Examples	385
Return Codes	371	REPLACE—Replace a Data Set Member	386
Examples	372	Macro Command Syntax	386
NUMBER—Set or Query Number Mode	372	Return Codes	386
Macro Command Syntax	372	Example	386
Assignment Statement Syntax	373	RESET—Reset the Data Display	386
Description	374	Macro Command Syntax	387
Return Codes	374	Description	387
Example	374	Return Codes	387
PACK—Set or Query Pack Mode	374	Examples	388
Macro Command Syntax	374	RFIND—Repeat Find	388
Assignment Statement Syntax	375	Macro Command Syntax	388
Return Codes	375	Return Codes	388
Example	375	Example	388
PASTE—Move or Copy Lines from Clipboard	375	RIGHT—Scroll Right	389
Syntax	375	Macro Command Syntax	389
Description	376	Description	389
Return Codes	376	Return Codes	389
Examples	376	Example	390
PRESERVE	376	RMACRO—Set or Query the Recovery Macro	390
Macro Command Syntax	376	Macro Command Syntax	390
Assignment Statement Syntax	376	Assignment Statement Syntax	390
Description	377	Return Codes	390
Return Codes	377	Example	390
Examples	377	SAVE—Save the Current Data	390
PROCESS—Process Line Commands	377	Macro Command Syntax	391
Macro Command Syntax	377	Description	391
Description	378	Return Codes	391
Return Codes	378	Example	391
Examples	378	SAVE_LENGTH—Set or Query Length for Variable Length Data	391
PROFILE—Set or Query the Current Profile	379	Assignment Statement Syntax	391
Macro Command Profile Control Syntax	379	Description	391
Macro Command Profile Lock Syntax	379	Return Codes	392
Macro Command Profile Reset Syntax	380	Examples	392
Assignment Statement Syntax	380	SCAN—Set Command Scan Mode	392
Description	380	Macro Command Syntax	392
Return Codes	380	Assignment Statement Syntax	393
Example	380	Return Codes	393
RANGE_CMD—Query a Command That You Entered	381	Example	393
Assignment Statement Syntax	381	SEEK—Seek a Data String, Positioning the Cursor	393
Description	381	Macro Command Syntax	393
Return Codes	381	Description	394
Example	381	Return Codes	395
RCHANGE—Repeat a Change	381	Examples	395
Macro Command Syntax	381	SEEK_COUNTS—Query Seek Counts	395
Description	382	Assignment Statement Syntax	395
Return Codes	382	Return Codes	396
Example	382	Example	396
RECFM—Query the Record Format.	382	SESSION	396
Assignment Statement Syntax	382	Assignment Statement Syntax	396
Return Codes	383	Return Codes	396
Example	383	SETUNDO—Set UNDO Mode	396
RECOVERY—Set or Query Recovery Mode	383	Macro Command Syntax	396
Macro Command Syntax	383	Assignment Statement Syntax	397
Assignment Statement Syntax	384	Description	397
Return Codes	384	Return Codes	397

Examples	398	Example	410
SHIFT (—Shift Columns Left	398	UNNUMBER—Remove Sequence Numbers	410
Macro Command Syntax	398	Macro Command Syntax	410
Description	398	Description	410
Return Codes	398	Return Codes	410
Examples	398	Example	411
SHIFT)—Shift Columns Right	399	UP—Scroll Up	411
Macro Command Syntax	399	Macro Command Syntax	411
Description	399	Description	411
Return Codes	399	Return Codes	412
Examples	399	Examples	412
SHIFT <—Shift Data Left	399	USER_STATE—Save or Restore User State	412
Macro Command Syntax	399	Assignment Statement Syntax	412
Description	400	Description	412
Return Codes	400	Return Codes	413
Examples	400	Examples	413
SHIFT >—Shift Data Right.	400	VERSION—Set or Query Version Number	413
Macro Command Syntax	400	Macro Command Syntax	413
Description	400	Assignment Statement Syntax	413
Return Codes	400	Return Codes	413
Examples	400	Examples	414
SORT—Sort Data	401	VIEW—View from within an Edit Session	414
Macro Command Syntax	401	Macro Command Syntax	414
Description	401	Description	414
Sorting Data Without Operands	401	Return Codes	414
Limiting the SORT Command	402	Examples	414
Sorting DBCS Data	402	VOLUME—Query Volume Information	414
Return Codes	402	Assignment Statement Syntax	415
Examples	402	Return Codes	415
STATS—Set or Query Stats Mode	403	Examples	415
Macro Command Syntax	403	XSTATUS—Set or Query Exclude Status of a Line	415
Assignment Statement Syntax	403	Assignment Statement Syntax	415
Return Codes	403	Description	415
Examples	403	Return Codes	416
SUBMIT—Submit Data for Batch Processing	404	Examples	416
Macro Command Syntax	404		
Description	404		
Return Codes	404		
Examples	404		
TABS—Set or Query Tabs Mode	404		
Macro Command Syntax	405		
Assignment Statement Syntax	406		
Return Codes	406		
Examples	406		
TABSLINE—Set or Query Tabs Line	406		
Assignment Statement Syntax	406		
Return Codes	407		
Examples	407		
TENTER—Set Up Panel for Text Entry.	407		
Macro Command Syntax	407		
Description	408		
Return Codes	409		
Example	409		
TFLOW—Text Flow a Paragraph	409		
Macro Command Syntax	409		
Return Codes	409		
Example	409		
TSPLIT—Text Split a Line	409		
Macro Command Syntax	409		
Description	410		
Return Codes	410		

Chapter 9. Edit Line Commands

Edit line commands affect only a single line or block of lines. You enter line commands by typing over the 6-digit number in the line command area on one or more lines and pressing Enter. Most command definitions in this book consist of the following information:

Syntax	A syntax diagram is how you type the command. It includes a description of any required or optional operands.
Description	A description explains the function and operation of the command. This description may also refer to other commands that can be used with this command.
Example	An example gives a sample usage of the line command.

Rules for Entering Line Commands

Enter a line command by one of the following:

- Type the command in the line command area and press Enter.
- Place the cursor in the data or line command field and press a function key to which the command is assigned.

The following rules apply to all line commands:

- You can type several line commands and make multiple data changes before you press Enter. The editor displays an error message if the line command is ambiguous. Because the line commands are processed from top to bottom, it is possible to have one error message appear that masks a later error condition. Only the first error condition found is displayed. After you have corrected that error condition, processing can continue and the next error condition, if any, is displayed. If you type a line command incorrectly, you can replace it before you press Enter by retyping it, blanking it out, or entering RESET.
- Generally, you need to type over only the first 1 or 2 characters of the line number to enter a line command. Sometimes, however, typing a single character can be ambiguous. In the following example, it is unclear whether the intended line command is R to repeat line 31700, or R3 to repeat the line three times:

```
031600  
R31700  
031800
```

In such cases, the ISPF editor assumes that you have not typed a number following the line command. If you want to repeat the line three times, you can use any of the following procedures:

- Leave the cursor on the character that immediately follows the R3:
R31700
- Type one or more blanks following the R3:
R3 700
- Type one or more blanks following the R but before the number, leaving the cursor on the character that immediately follows the 3:

Rules for Entering Line Commands

R 3700

- Type R3 and press the Erase EOF key to clear the rest of the Line Command field, or press the Erase EOF key and then type R3.
- You can type the following line commands on the TOP OF DATA line by typing over the asterisks that appear in its line command field:
 - I[n]** Insert one or *n* lines ahead of the data.
 - A[n]** Move or copy a line or lines one or *n* times ahead of the data.
 - TE[n]** Type one or *n* text lines ahead of the data.
- You can type the following line command on the BOTTOM OF DATA line by typing over the asterisks:
 - B[n]** Move or copy a line or lines one or *n* times following the data.

Edit Line Command Notation Conventions

The syntax of the PDF line commands uses the following notation conventions:

Uppercase

Uppercase commands or operands must be spelled as shown (in either uppercase or lowercase). (See “Appendix A. Abbreviations for Commands and Other Values” on page 419.)

Lowercase

Lowercase operands are variables; substitute your own values.

Underscore

Underscored operands are the system defaults.

Brackets ([])

Operands in brackets are optional.

Stacked operands

Stacked operands show two or more operands from which you can select. If you do not choose any, the default operand is used.

Braces ({ })

Braces show two or more operands from which you must select one. .

OR (|)

The OR (|) symbol shows two or more operands from which you must select one.

Table 4 summarizes line commands.

Line Command Summary

Table 4. Summary of the Line Commands

Command	Page	Description
([n] [2] ([n] [2]	“(—Column Shift Left” on page 156	Shifts columns left two positions or the specified number of positions.

Line Command Summary

Table 4. Summary of the Line Commands (continued)

Command	Page	Description
) [n] [2])) [n] [2]	") —Column Shift Right" on page 158	Shifts columns right two positions or the specified number of positions.
< [n] [2] << [n] [2]	" < —Data Shift Left" on page 159	Shifts data left two positions or the specified number of positions.
> [n] [2] >> [n] [2]	" > —Data Shift Right" on page 162	Shifts data right two positions or the specified number of positions.
A [n]	" A —Specify an "After" Destination" on page 163	Identifies the line after which copied, moved, or model lines are to be inserted.
B [n]	" B —Specify a "Before" Destination" on page 166	Identifies the line before which copied, moved, or model lines are to be inserted.
BOUNDS	" BOUNDS —Define Boundary Columns" on page 168	Displays the column boundary definition line.
C [n] CC	" C —Copy Lines" on page 170	Copies one or more lines from one location to another.
COLS	" COLS —Identify Columns" on page 172	Displays a position identification line.
D [n] DD	" D —Delete Lines" on page 174	Deletes one or more lines.
F [n]	" F —Show the First Line" on page 175	Redisplays one or more lines at the beginning of a block of excluded lines.
I [n]	" I —Insert Lines" on page 177	Inserts one or more blank data entry lines.
L [n]	" L —Show the Last Line(s)" on page 179	Redisplays one or more lines at the end of a block of excluded lines.
LC [n] LCC LCLC	" LC —Convert Characters to Lowercase" on page 180	Converts all uppercase alphabetic characters in one or more lines to lowercase.
M [n] MM	" M —Move Lines" on page 182	Moves one or more lines from one location to another.
MASK	" MASK —Define Masks" on page 184	Displays the contents of the mask when used with the I (insert), TE (text entry), and TS (text split) line commands.
MD [n] MDD MDMD	" MD —Make Dataline" on page 186	Converts one or more ==MSG>, =NOTE=, =COLS>, and ===== (information) lines to data so that they can be saved as part of your data set.
O [n] OO	" O —Overlay Lines" on page 188	Identifies the lines over which data is to be moved or copied.
R [n] RR [n]	" R —Repeat Lines" on page 190	Repeats one or more lines.
S [n]	" S —Show Lines" on page 193	Redisplays one or more lines with the leftmost indentation in a block of excluded lines.

Line Command Summary

Table 4. Summary of the Line Commands (continued)

Command	Page	Description
TABS	"TABS—Control Tabs" on page 195	Displays the tab definition line.
TE[n]	"TE—Text Entry" on page 197	Inserts blank lines to allow power typing for text entry.
TF[n]	"TF—Text Flow" on page 200	Restructures paragraphs following deletions, insertions, splitting, and so forth.
TS[n]	"TS—Text Split" on page 202	Divides one or more lines so that data can be added.
UC[n] UCC UCUC	"UC—Convert Characters to Uppercase" on page 203	Converts all lowercase alphabetic characters in one or more lines to uppercase.
X[n] XX	"X—Exclude Lines" on page 205	Excludes one or more lines from a panel.

(—Column Shift Left

The ((column shift left) line command moves characters on a line to the left without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See "Shifting Data" on page 51 for more information.

Syntax

[n]
[2]
([n]
[2]

n A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To column shift one line toward the left side of your display:

1. Type (in the line command area of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the line other than 2 columns.
2. Press Enter.

To column shift a block of lines toward the left side of your display:

1. Type ((in the line command area of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.
2. Type ((in the line command area of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first ((and the second ((, if necessary.
3. Press Enter. The lines that contain the two ((commands and all of the lines between them are column shifted to the left.

The BOUNDS setting limits column shifting. If you shift columns beyond the current BOUNDS setting, the editor deletes the text beyond the BOUNDS without displaying a warning message.

Example

To shift a group of lines to the left three column positions, specify the number of columns and the range in the line command area, as shown in Figure 66. Press Enter and the editor shifts the specified lines three columns to the right. See

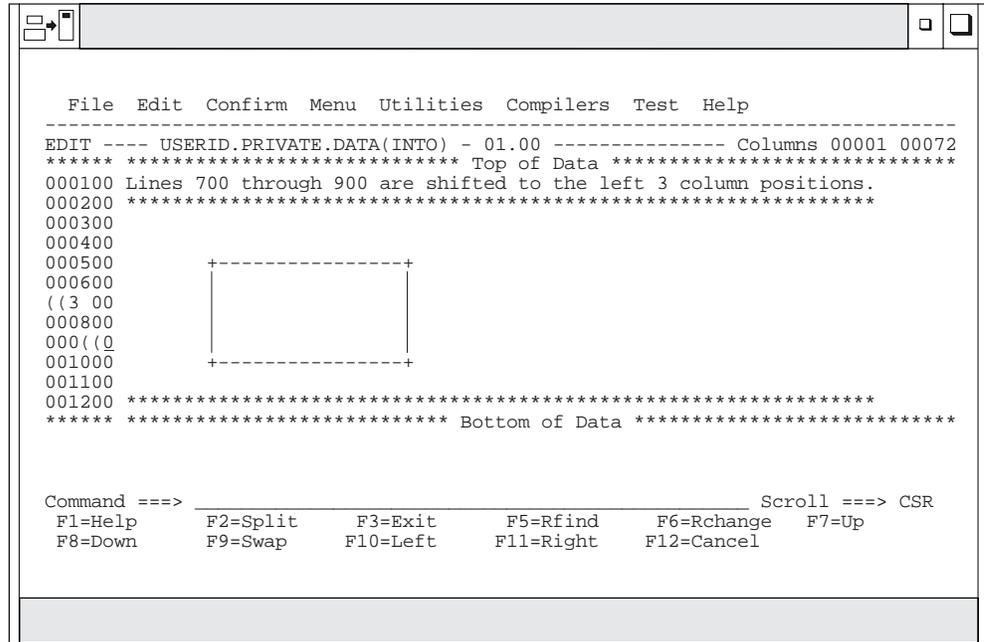


Figure 66. Before the ((Column Shift Left) Line Command

Figure 67.

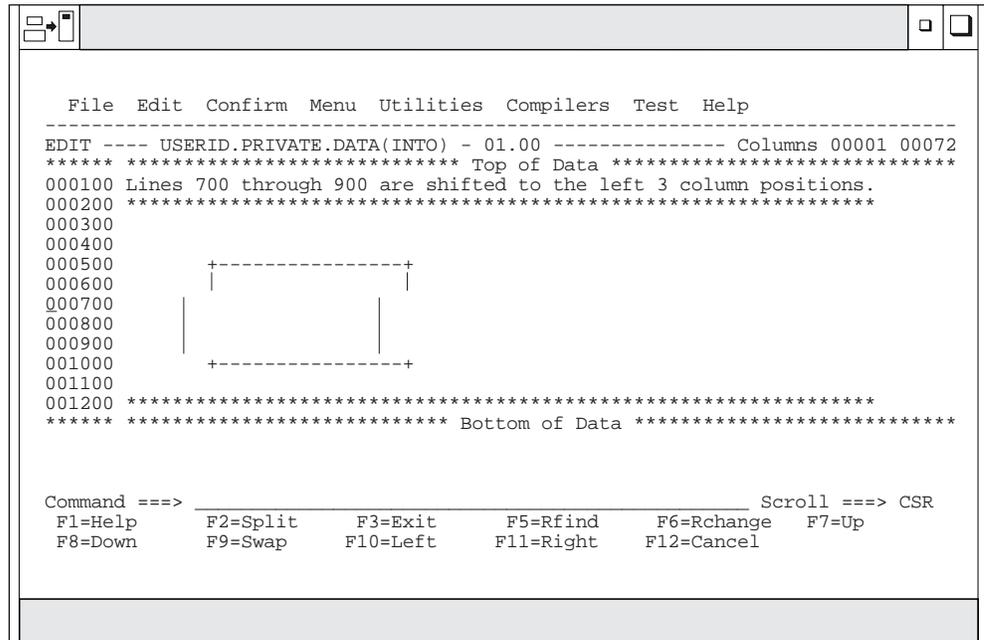


Figure 67. After the ((Column Shift Left) Line Command

)—Column Shift Right

The) (column shift right) line command moves characters on a line to the right without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting Data” on page 51 for more information.

Syntax

```
) [n]  
  [2]  
) ) [n]  
    [2]
```

n A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To column shift one line toward the right side of your display:

1. Type) in the line command area of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the data other than 2 columns.
2. Press Enter.

To column shift a block of lines toward the right side of your display:

1. Type)) in the line command area of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.
2. Type)) in the line command area of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first)) and the second)), if necessary.
3. Press Enter. The lines that contain the two)) commands and all of the lines between them are column shifted to the right.

The BOUNDS setting limits column shifting. If you shift columns beyond the current BOUNDS setting, the editor deletes the text beyond the BOUNDS without displaying a warning message.

Example

To shift a group of lines to the right 3 column positions, specify the number of columns and the range in the line command area, as shown in Figure 68 on page 159.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 Lines 700 through 900 are shifted to the right 3 column positions.
000200 *****
000300
000400
000500
000600      +-----+
           |         |
           |         |
           |         |
           +-----+
000800
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap      F10=Left    F11=Right    F12=Cancel

```

Figure 68. Before the) (Column Shift Right) Line Command

Figure 69 shows that when you press Enter, the editor shifts the specified lines to the right 3 columns.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 Lines 700 through 900 are shifted to the right 3 column positions.
000200 *****
000300
000400
000500
000600      |         |
           |         |
           |         |
           |         |
           +-----+
000700
000800
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap      F10=Left    F11=Right    F12=Cancel

```

Figure 69. After the) (Column Shift Right) Line Command

<—Data Shift Left

The < (data shift left) line command moves the body of a program statement to the left without shifting the label or comments. This command attempts to prevent loss of data. See “Shifting Data” on page 51 for more information.

<—Data Shift Left

Syntax

<[n]
[2]
<<[n]
[2]

- n** A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To data shift one line toward the left side of your display:

1. Type < in the line command area of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the data other than 2 columns.
2. Press Enter.

To data shift a block of lines toward the left side of your display:

1. Type << in the line command area of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.
2. Type << in the line command area of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first << and the second <<, if necessary.
3. Press Enter. The lines that contain the two << commands and all of the lines between them are data shifted to the left.

The BOUNDS setting limits data shifting. If you shift data beyond the current BOUNDS setting, the text stops at the left bound and the shifted lines are marked with ==ERR> flags. If an error occurs in an excluded line, you can find the error with LOCATE, and remove the error flag by using RESET.

Example

To use a data shift to delete 5 blanks before a segment of three lines, specify the shift and the range in the line command area, as shown in Figure 70 on page 161.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INT0) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 The first bar on lines 600 through 800 shift 5 spaces to the left.
000200 *****
000300
000400
000500
000600
000700
000800
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 70. Before the < (Data Shift Left) Line Command

When you press Enter, the editor deletes 5 blanks on the specified lines. Notice that the editor does not shift data within the BOUNDS setting, as shown in Figure 71.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INT0) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 The first bar on lines 600 through 800 shift 5 spaces to the left.
000200 *****
000300
000400
000500
000600
000700
000800
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 71. After the < (Data Shift Left) Line Command

>—Data Shift Right

The > (data shift right) line command moves the body of a program statement to the right without shifting the label or comments. This command attempts to prevent loss of data. See “Shifting Data” on page 51 for more information.

Syntax

```
>[n]  
[2]  
>>[n]  
[2]
```

n A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To data shift one line toward the right side of your display:

1. Type > in the line command area of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the line other than 2 columns.
2. Press Enter.

To data shift a block of lines toward the right side of your display:

1. Type >> in the line command area of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.
2. Type >> in the line command area of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first >> and the second >>, if necessary.
3. Press Enter. The lines that contain the two >> commands and all of the lines between them are data shifted to the right.

The BOUNDS setting limits data shifting. If you shift data beyond the current BOUNDS setting, the text stops at the right bound and the shifted lines are marked with ==ERR> flags. If an error occurs in an excluded line, you can find the error with the LOCATE command, and remove the error flag by using RESET.

Example

To use a data shift to insert 5 blanks before a segment of three lines, specify the shift and the range in the line command area, as shown in Figure 72 on page 163.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 The first bar on lines 600 through 800 shift 5 spaces to the left.
000200 *****
000300
000400
000500
>>5 00
000700
000>>0
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==>> _____ Scroll ==>> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 72. Before the > (Data Shift Right) Line Command

When you press Enter, the editor inserts 5 blanks on the specified lines. See Figure 73. Notice that the editor does not shift the data within the BOUNDS setting.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(INTO) - 01.01 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 The first bar on lines 600 through 800 shift 5 spaces to the right.
000200 *****
000300
000400
000500
000600
000700
000800
000900
001000
001100
001200 *****
***** ***** Bottom of Data *****

Command ==>> _____ Scroll ==>> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 73. After the > (Data Shift Right) Line Command

A—Specify an “After” Destination

The A (after) line command specifies the destination for data is to be moved, copied, or inserted.

A—Specify an "After" Destination

Syntax

A[n]

- n** A number that tells the ISPF editor to repeat the associated line command a specified number of times. If you do not type a number, or if the number you type is 1, the editor performs the command only once. The number does not affect associated primary commands.

Description

To specify that data is to be moved, copied, or inserted after a specific line:

1. Type one of the commands that are listed in the following table. Line commands are typed in the line command area. Primary commands are typed on the Command line.

Line Commands	See topic	Primary Commands	See topic
C	"C—Copy Lines" on page 170	COPY	"COPY—Copy Data" on page 227
M	"M—Move Lines" on page 182	MODEL	"MODEL—Copy a Model into the Current Data Set" on page 257
		MOVE	"MOVE—Move Data" on page 260

2. Type A in the line command area of the line that the moved, copied, or inserted data is to follow. If you are specifying the destination for a line command, a number after the A line command specifies the number of times the other line command is performed. However, a number after the A command has no effect on a primary command.
3. Press Enter.
4. Some of the commands in the preceding table can cause another panel to be displayed if more information is needed. If so, fill in the required information and press Enter to move, copy, or insert the data. Refer to information about the specified command if you need help.

If no panel is displayed, the data is moved, copied, or inserted when you press Enter in step 3.

You must always specify a destination except when you are using a primary command to move, copy, or insert data into a member or data set that is empty.

Two other line commands that are used to specify a destination are the B (before) command and the O (overlay) command. See "B—Specify a "Before" Destination" on page 166 and "O—Overlay Lines" on page 188 for more information.

Example

Figure 74 shows how you can move data with the M and A line commands, or copy data with the C and A line commands. Type M in the line command area of the line you want to move. Type A in the line command area of the line that you want the moved line to follow.

A—Specify an "After" Destination

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
M 0400 This is the line to be moved.
000500
000600      +-----+
A 0700      |           |
000800      |           |
000900      |           |
001000      |           |
001100      +-----+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel

```

Figure 74. Before the A (After) Line Command

When you press Enter, the line where you typed the M command is moved after the line where you typed the A command. See Figure 75.

Note: If you press Enter before specifying where you want the data to go, the editor displays a MOVE/COPY pending message at the top of the panel. The line does not move until you specify a destination.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400
000500      +-----+
000600      |           |
000700 This is the line to be moved.
000800      |           |
000900      |           |
001000      |           |
001100      +-----+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel

```

Figure 75. After the A (After) Line Command

B—Specify a "Before" Destination

The B (before) line command specifies the destination for data to be moved, copied, or inserted.

Syntax

B[n]

- n** A number that tells the ISPF editor to repeat the associated line command a specified number of times. If you do not type a number, or if the number you type is 1, the command is not repeated. For associated primary commands, this number has no effect.

Description

To specify that data is to be moved, copied, or inserted before a specific line:

1. Type one of the commands that are listed in the following table. Line commands are typed in the line command area. Primary commands are typed on the Command line.

Line Commands	See topic	Primary Commands	See topic
C	"C—Copy Lines" on page 170	COPY	"COPY—Copy Data" on page 227
M	"M—Move Lines" on page 182	MODEL	"MODEL—Copy a Model into the Current Data Set" on page 257
		MOVE	"MOVE—Move Data" on page 260

2. Type B in the line command area of the line that the moved, copied, or inserted data is to precede. If you are specifying the destination for a line command, a number after the B line command to specifies the number of times that the other line command is performed. However, a number that you type after the B command has no effect on a primary command.
3. Press Enter.
4. Some of the commands in the preceding table can cause another panel to be displayed if more information is needed. If so, fill in the required information and press Enter to move, copy, or insert the data. Refer to information about the specified command if you need help.
If no panel is displayed, the data is moved, copied, or inserted when you press Enter in step 3.

You must always specify a destination except when you are using a primary command to move, copy, or insert data into a member or data set that is empty.

Two other line commands that are used to specify a destination are the A (after) command and the O (overlay) command. See "A—Specify an "After" Destination" on page 163 and "O—Overlay Lines" on page 188 for more information.

Example

Figure 76 shows how you can copy data with the C and B line commands, or move data with the M and B line commands. Type C in the line command area of the line you want to copy. Type B in the line command area of the line that the copied line precedes.

When you press Enter, the line where you typed the C command is moved before

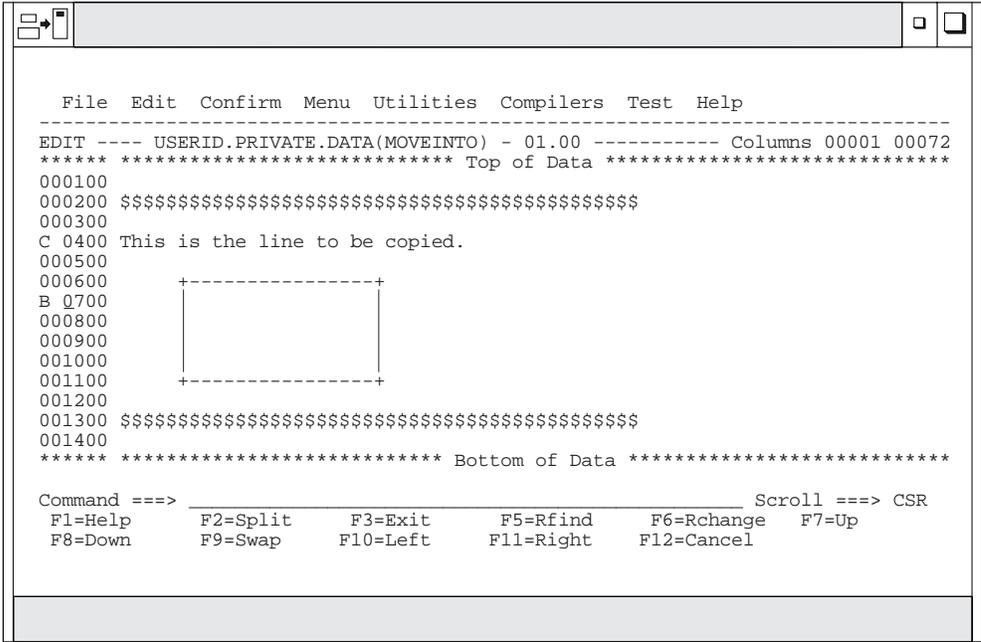


Figure 76. Before the B (Before) Line Command

the line where you typed the B command, as shown in Figure 77.

Note: If you press Enter before specifying where you want the data to go, the editor displays a MOVE/COPY pending message at the top of the panel. The line is not copied until you specify a destination.

BOUNDS—Define Boundary Columns

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 This is the line to be copied.
000500
000600      +-----+
000700 This is the line to be copied.
000800      |
000900      |
001000      |
001100      |
001200      +-----+
001300
001400 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001500
***** ***** Bottom of Data *****
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel
```

Figure 77. After the B (Before) Line Command

BOUNDS—Define Boundary Columns

The BOUNDS line command displays the boundary definition line.

Syntax

BOUNDS

Description

The BOUNDS line command provides an alternative to setting the boundaries with the BOUNDS primary command or macro command; the effect on the member or data set is the same. However, if you use both the BOUNDS primary command and the BOUNDS line command in the same interaction, the line command overrides the primary command.

To display the boundary definition (=BNDS>) line:

1. Type BOUNDS in the line command area of any unflagged line.
2. Press Enter. The boundary definition line is inserted in the data set or member.

To change the BOUNDS settings:

1. Delete a < or > character. The < character shows the left BOUNDS setting and the > character shows the right BOUNDS setting.
2. Move the cursor to a different location on the =BNDS> line.

Note: You can use the COLS line command with the BOUNDS line command to help check and reposition the BOUNDS settings. The COLS line command displays the column identification line.

3. Retype the deleted character or characters.

Note: The < character must be typed to the left of the > character.

BOUNDS—Define Boundary Columns

4. Press Enter. The new BOUNDS settings are now in effect.

To revert to the default settings:

1. Display the boundary definition line.
2. Blank out its contents with the Erase EOF key, the cursor, or the Del (delete) key.
3. Press Enter.

Note: See “Edit Boundaries” on page 28 for a table that shows the default bounds settings for various types of data sets.

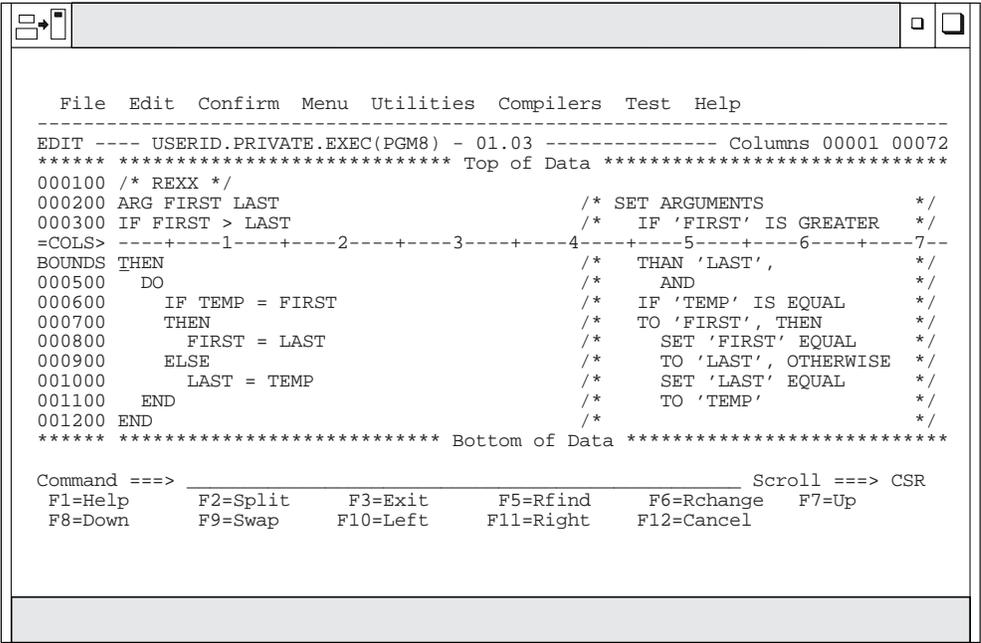
To remove the boundary definition line from the panel:

1. You can either type D in the line command area that contains the =BNDS> flag or type one of the following on the Command line:
 - RESET (to reset all flagged lines), or
 - RESET SPECIAL (to reset only the special lines).
2. Press Enter. The =BNDS> line is removed from the display.

See “Edit Boundaries” on page 28 for more information, including tables that show commands affected by BOUNDS settings and default bounds settings for various types of data sets.

Example

Figure 78 shows the boundary definition line displayed with the column identification line. Type BOUNDS in the line command area.



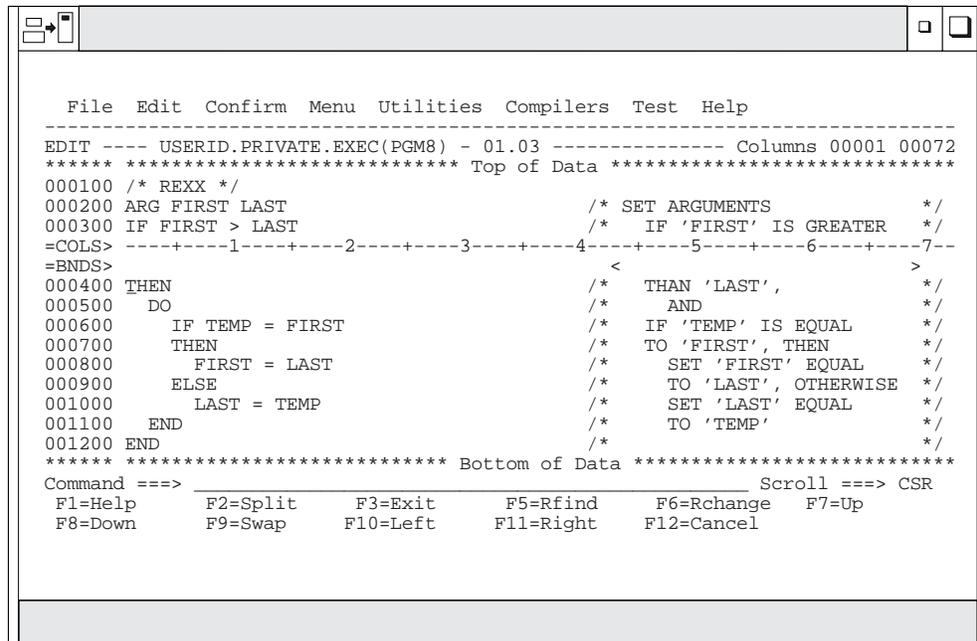
```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
=COLS> -----1-----2-----3-----4-----5-----6-----7--
BOUNDS THEN /* THAN 'LAST', */
000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST', THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
000900 ELSE /* TO 'LAST', OTHERWISE */
001000 LAST = TEMP /* SET 'LAST' EQUAL */
001100 END /* TO 'TEMP' */
001200 END /*
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 78. Before the BOUNDS Line Command

Figure 79 shows that when you press Enter, the editor inserts the BOUNDS line and sets the left bound at column 43 and the right bound at column 69.

C—Copy Lines



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
=COLS> -----1-----2-----3-----4-----5-----6-----7--
=BNDS> <
000400 THEN /* THAN 'LAST', */
000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST', THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
000900 ELSE /* TO 'LAST', OTHERWISE */
001000 LAST = TEMP /* SET 'LAST' EQUAL */
001100 END /* TO 'TEMP' */
001200 END /*
***** ***** Bottom of Data *****
Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 79. After the BOUNDS Line Command

C—Copy Lines

The C (copy) line command copies lines from one location to another.

Syntax

```
C[n]  
CC
```

n The number of lines to be copied. If you do not type a number, or if the number you type is 1, only the line on which you type C is copied.

Description

To copy one or more lines within the same data set or member:

1. Type C in the line command area of the line to be copied. If you also want to copy one or more lines that immediately follow this line, type a number greater than 1 after the C command.
2. Next, specify the destination of the line to be copied by using either the A (after), B (before), or O (overlay) line command.
3. Press Enter. The line or lines are copied to the new location.

To copy a block of lines within the same data set or member:

1. Type CC in the line command area of both the first and last lines to be copied. You can scroll (or use FIND or LOCATE) between typing the first CC and the second CC, if necessary.
2. Use the A (after), B (before), or OO (overlay) command to show where the copied lines are to be placed. Notice that when you use the block form of the C command (CC) to copy and overlay lines, you should also use the block form of the O command (OO).

3. Press Enter. The lines that contain the two CC commands and all of the lines between them are copied to the new location.

To copy lines to another data set or member:

Note: To copy lines into an existing data set or member without replacing that data set or member, edit the existing data set or member and use the COPY primary or macro command.

1. Type either CREATE or REPLACE on the Command line.
2. Use one of the forms of the C command described previously.
3. Press Enter.
4. On the next panel that PDF displays, type the name of the data set or member that you want to create or replace.
5. Press Enter. The lines are copied to the data set or member that you specified.

Example

The example in Figure 80 shows how to copy data by using the C with the B (Before), A (After), or O (Overlay) line commands. Type C in the line command area of the line you want to copy. Type B in the line command area of the line that you want the copied line to precede.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ---- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
C 0400 This is the line to be copied.
000500
000600      +-----+
B 0700      |           |
000800      |           |
000900      |           |
001000      |           |
001100      +-----+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right     F12=Cancel

```

Figure 80. Before the C (Copy) Line Command

When you press Enter, the line where you typed the C command is copied preceding the line where you typed the B command, as shown in Figure 81 on page 172.

Note: If you press Enter before specifying where you want the data to go, the editor displays a MOVE/COPY pending message at the top of the panel. The line is not copied until you specify a destination.

COLS—Identify Columns

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 This is the line to be copied.
000500
000600
000700 This is the line to be copied.
000800
000900
001000
001100
001200
001300
001400 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001500
***** ***** Bottom of Data *****
Command ==>
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
Scroll ==> CSR
```

Figure 81. After the C (Copy) Line Command

COLS—Identify Columns

The COLS line command displays a column identification line.

Syntax

COLS

Description

To display the column identification (=COLS>) line:

1. Type COLS in the line command area of any line.
2. Press Enter.

The column identification line is inserted in the data set or member.

Note: You can use the COLS line command with the BOUNDS line command to help check and reposition the bounds settings.

To remove the column identification line from the panel:

1. You can either type D in the line command area that contains the =COLS> flag or type one of the following on the Command line:
 - RESET (to reset all flagged lines), or
 - RESET SPECIAL (to reset only the special lines).
2. Press Enter.

The =COLS> line is removed from the display.

Example

The example in Figure 82 shows the column identification line displayed with the boundary definition line. The COLS command is typed in the line command area.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
=BNDS> < >
COLS00 THEN /* THAN 'LAST', */

000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST', THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
000900 ELSE /* TO 'LAST', OTHERWISE */
001000 LAST = TEMP /* SET 'LAST' EQUAL */
001100 END /* TO 'TEMP' */
001200 END /* */
***** ***** Bottom of Data *****

Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 82. Before the COLS Line Command

When you press Enter, the editor inserts the COLS line, as shown in Figure 83.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
=BNDS> < >
=COLS> -----1-----2-----3-----4-----5-----6-----7--
000400 THEN /* THAN 'LAST', */

000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST', THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
000900 ELSE /* TO 'LAST', OTHERWISE */
001000 LAST = TEMP /* SET 'LAST' EQUAL */
001100 END /* TO 'TEMP' */
001200 END /* */
***** ***** Bottom of Data *****

Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 83. After the COLS Line Command

D—Delete Lines

The D (delete) line command deletes lines from your display.

Syntax

D[n]
DD

n The number of lines to be deleted. If you do not type a number, or if the number you type is 1, only the line on which you type D is deleted.

Description

To delete one or more lines:

1. Type D in the line command area of the line to be deleted. If you also want to delete one or more lines that immediately follow this line, type a number greater than 1 after the D command.
2. Press Enter.
The line or lines are deleted.

To delete a block of lines:

1. Type DD in the line command area of both the first and last lines to be deleted. You can scroll (or use FIND or LOCATE) between typing the first DD and the second DD, if necessary.
2. Press Enter.
The lines that contain the two DD commands and all of the lines between them are deleted.

Example

To delete two lines, type D2 in the Command line area of the first line you want to delete. See Figure 84.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
D2 400 This is the line to be deleted.
000500
000600      +-----+
000700      |           |
000800      |           |
000900      |           |
001000      |           |
001100      +-----+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel
    
```

Figure 84. Before the D (Delete) Line Command

When you press Enter, the editor deletes the two lines specified. See Figure 85.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400      +-----+
000500      |           |
000600      |           |
000700      |           |
000800      |           |
000900      +-----+
001000
001100 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001200
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel
    
```

Figure 85. After the D (Delete) Line Command

F—Show the First Line

The F (show first line) line command redisplay one or more lines at the beginning of a block of excluded lines. See “Redisplaying Excluded Lines” on page 65 for more information about excluding lines.


```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ----- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
000300 TEST-#
000400 TEST-#
000500 TEST-#
-----
- - - - - 3 LINE(S) NOT DISPLAYED
-----
000900 TEST-#
001000 TEST-#
001100 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap    F10=Left   F11=Right   F12=Cancel

```

Figure 87. After the F (Show First Line) Line Command

I—Insert Lines

The I (insert) line command inserts one or more lines in your data set or member. The inserted lines are blank unless you have defined a mask. See “MASK—Define Masks” on page 184 for more information about defining a mask.

Syntax

I[n]

n The number of blank lines to insert. If you do not type a number, or if the number you type is 1, only one line is inserted.

Description

To insert one or more lines in a data set or member:

1. Type I in the line command area of the line that the inserted line is to follow. If you want to insert more than one line, type a number greater than 1 after the I command.
2. Press Enter. The line or lines are inserted.

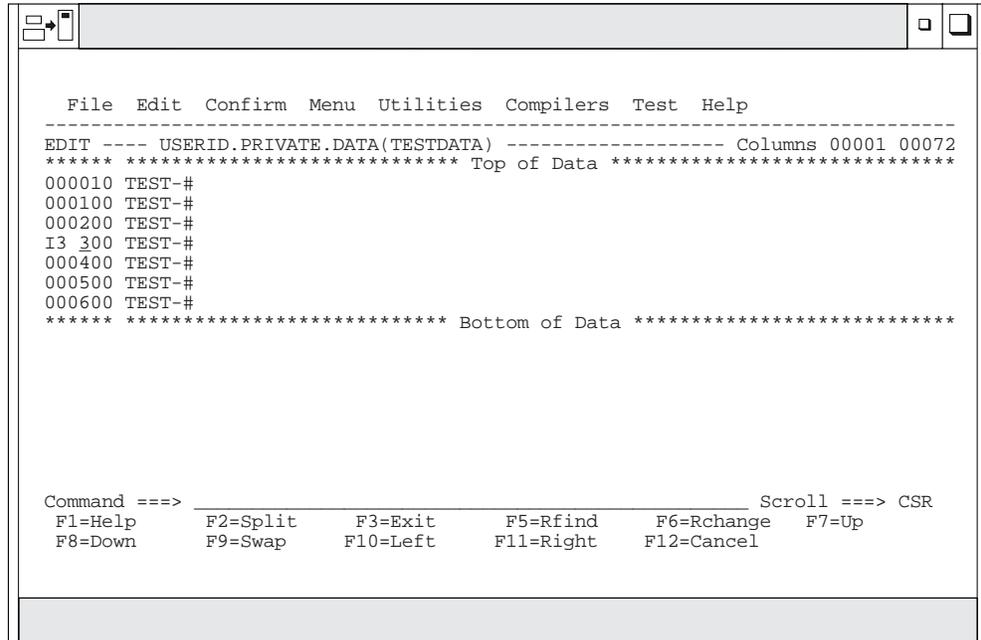
If you type any information, even a blank character in the inserted line, the line becomes part of the source data and is assigned a line number the next time you press Enter. However, if you do not type any information, the space for the new line is automatically deleted the next time you press Enter.

If you type information on the last, or only, inserted line and the cursor is still in the data portion of that line, the editor automatically inserts another line when you press Enter or a scroll function key, but only if the new inserted line remains on the panel. If the new line is at the bottom of the panel, the editor automatically scrolls down so that the new line is displayed at the bottom of the screen.

I—Insert Lines

Example

Figure 88 shows how to insert lines in a member. To insert three lines, type I3 in the line command area.

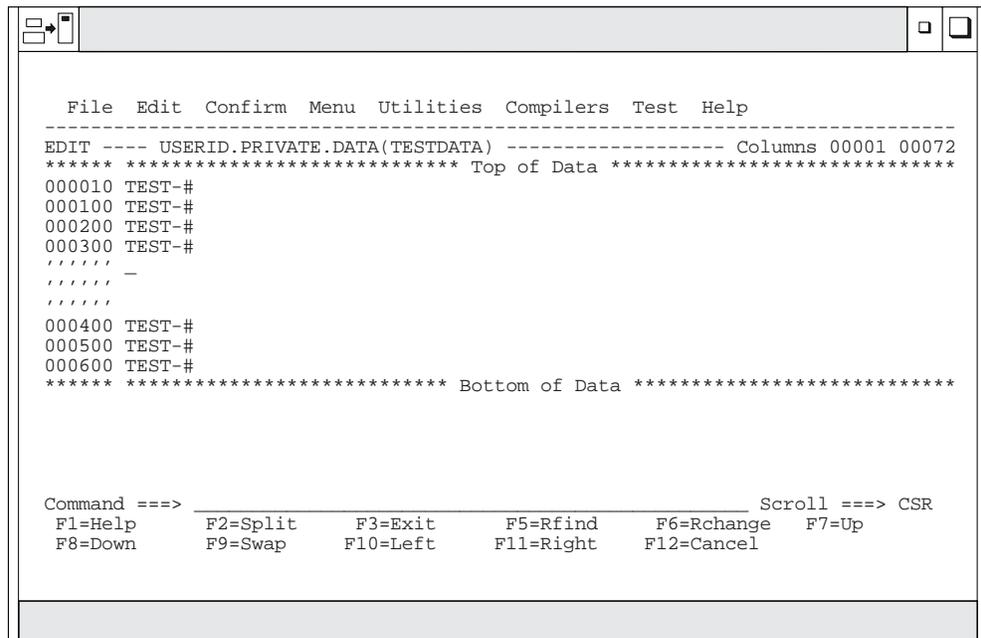


```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ----- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
I3 300 TEST-#
000400 TEST-#
000500 TEST-#
000600 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 88. Before the I (Insert) Line Command

When you press Enter, the editor inserts three lines. See Figure 89.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ----- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
000300 TEST-#
//////// -
//////// -
//////// -
000400 TEST-#
000500 TEST-#
000600 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 89. After the I (Insert) Line Command

L—Show the Last Line(s)

The L (show last line) line command redisplay one or more lines at the end of a block of excluded lines. See “Redisplaying Excluded Lines” on page 65 for more information about excluding lines.

Syntax

L[n]

n The number of lines to be redisplayed. If you do not type a number, or if the number you type is 1, only one line is redisplayed.

Description

To redisplay the last line or lines of a block of excluded lines:

1. Type L in the line command area next to the dashed line that shows where lines have been excluded. The message in the dashed line tells you how many lines are excluded. If you want to redisplay more than one line, type a number greater than 1 after the L command.
2. Press Enter. The last line or lines are redisplayed.

Example

Figure 90 shows how to redisplay the last three excluded lines. To redisplay the last three lines, type L3 in the line command area of the excluded lines.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ---- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
L3_ - - - - - 6 LINE(S) NOT DISPLAYED
000900 TEST-#
001000 TEST-#
001100 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap    F10=Left   F11=Right   F12=Cancel

```

Figure 90. Before the L (Show Last Line) Line Command

When you press Enter, the editor redisplay the last three lines. See Figure 91 on page 180.

L—Show the Last Line(s)

Note: Excluded lines do not need to be displayed again before saving the data. The excluded lines message line is never saved.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ---- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
- - - - - 3 LINE(S) NOT DISPLAYED
000600 TEST-#
000700 TEST-#
000800 TEST-#
000900 TEST-#
001000 TEST-#
001100 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 91. After the L (Show Last Line) Line Command

LC—Convert Characters to Lowercase

The LC (lowercase) line command converts characters in a data set or member from uppercase to lowercase. However, it does not affect the caps mode of the data that you are editing.

Syntax

```
LC[n]
LCC
LCLC
```

n The number of lines to be converted to lowercase. If you do not type a number, or if the number you type is 1, only the line on which you type LC is converted to lowercase.

Description

To convert characters on one or more lines to lowercase:

1. Type LC in the line command area of the source code line that contains the characters you want to convert. If you also want to convert characters on one or more lines that immediately follow this line, type a number greater than 1 after the LC command.
2. Press Enter. The characters on the source code lines are converted to lowercase.

To convert characters in a block of lines to lowercase:

1. Type LCC in the line command area of both the first and last source code lines that contain characters that are to be converted. You can scroll (or use FIND or LOCATE) between typing the first LCC and the second LCC, if necessary.

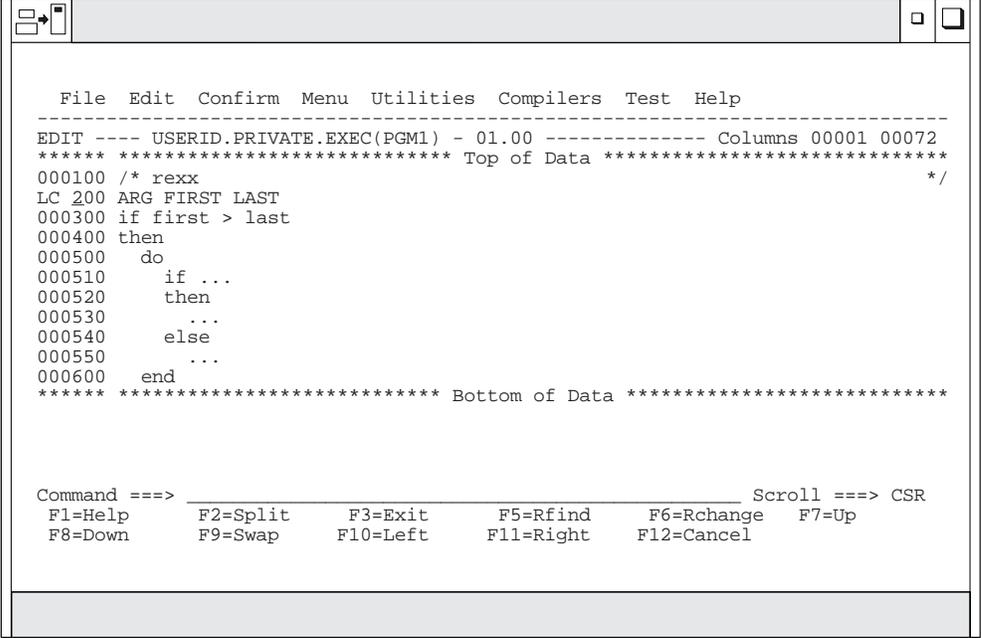
LC—Convert Characters to Lowercase

2. Press Enter. The characters in the source code lines that contain the two LCC commands and in all of the source code lines between them are converted to lowercase.

See the UC (uppercase) line command and the CAPS primary and macro commands, which are related, for information about converting characters from uppercase to lowercase and vice versa.

Example

Figure 92 shows how to use the LC command without any operands. To convert a line, type LC in the line command area of the line you want to convert.



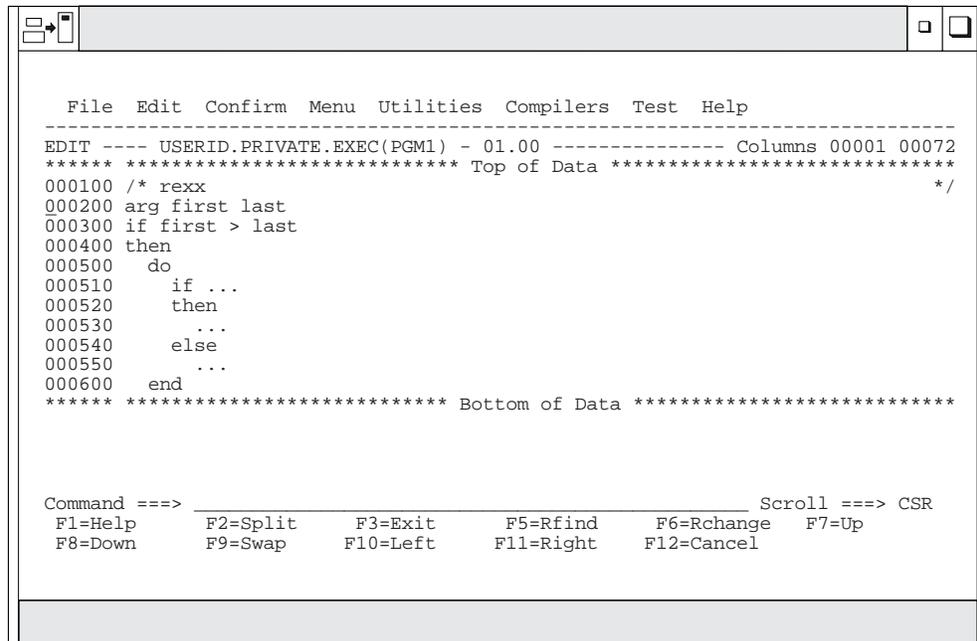
```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM1) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* rexx
LC 200 ARG FIRST LAST
000300 if first > last
000400 then
000500 do
000510     if ...
000520     then
000530     ...
000540     else
000550     ...
000600 end
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 92. Before the LC (Lowercase) Line Command

When you press Enter, the editor converts the characters in the line to lowercase. See Figure 93 on page 182.

M—Move Lines



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM1) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* rexx
000200 arg first last
000300 if first > last
000400 then
000500 do
000510 if ...
000520 then
000530 ...
000540 else
000550 ...
000600 end
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 93. After the LC (Lowercase) Line Command

M—Move Lines

The M (move) line command moves lines from one location to another.

Syntax

M[n]
MM

n The number of lines to be moved. If you do not type a number, or if the number you type is 1, only the line on which you type M is moved.

Description

To move one or more lines within the same data set or member:

1. Type M in the line command area of the line to be moved. If you want to move one or more lines that immediately follow this line, type a number greater than 1 after the M command.
2. Next, specify the destination of the line to be moved by using either the A (after), B (before), or O (overlay) line command. See the descriptions of those commands if you need more information about them.
3. Press Enter. The line or lines are moved to the new location.

To move a block of lines within the same data set or member:

1. Type MM in the line command area of both the first and last lines to be moved. You can scroll (or use FIND or LOCATE) between typing the first MM and the second MM, if necessary.
2. Use the A (after), B (before), or OO (overlay) command to show where the moved lines are to be placed. Notice that when you use the block form of the M command (MM) to move and overlay lines, you should also use the block form of the O command (OO).

3. Press Enter. The lines that contain the two MM commands and all of the lines between them are moved to the new location.

To move lines to another data set or member:

Note: To move lines into an existing data set or member without replacing that data set or member, use the MOVE primary or macro command.

1. Type either CREATE or REPLACE on the Command line.
2. Use one of the forms of the M command described previously.
3. Press Enter.
4. On the next panel, type the name of the data set or member that you want to create or replace.
5. Press Enter. The lines are moved to the data set or member that you specified.

Example

Figure 94 shows how you can move data by using the M with the B (Before) line command. To move a line, type M in the line command area of the line you want to move. Type a B in the line command area of the line you want the moved line to precede.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
M 0400 This is the line to be moved.
000500
000600
B 0700
000800
000900
001000
001100
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel

```

Figure 94. Before the M (Move) Line Command

When you press Enter, the editor moves the line where you typed the M command to a position immediately before the line where you typed the B command, as shown in Figure 95. If you press Enter before specifying a destination, the editor displays a MOVE/COPY pending message at the top of the panel. The line is not moved until you specify a destination.

MASK—Define Masks

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400
000500
000600 This is the line to be moved.
000700
000800
000900
001000
001100
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F4=Right    F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 95. After the M (MOVE) Line Command

MASK—Define Masks

The MASK line command displays the =MASK> line. On this line, you can type characters that you want to insert into an unformatted data set or member. These characters, which are called the *mask*, are inserted whenever you use the I (insert), TE (text entry), or TS (text split) line commands, or when you edit an empty data set.

Syntax

MASK

Description

To display the =MASK> line:

1. Type MASK in the line command area of any line.
2. Press Enter. The =MASK> line is displayed.

Initially, the mask contains all blanks. To define a mask:

1. Add characters to or delete characters from the =MASK> line while it is displayed.
2. Press Enter. The mask is now defined.

Once a mask is defined, the contents of the =MASK> line are displayed whenever a new line is inserted. This occurs when you use the I (insert), TE (text entry), and TS (text split) line commands, and when you edit an empty data set. You can change the mask definition whenever you need to by repeating the preceding steps.

To remove the =MASK> line from the panel, do one of the following:

- Type D in the line command field that contains the =MASK> flag and press Enter.

- Type RESET on the Command line and press Enter.
- End the edit session by:
 - Pressing F3 (if it is defined as the END command), or
 - Typing END on the Command line and pressing Enter.

The mask line is never saved as part of the data. However, the mask remains in effect, even if it is not displayed, until you change it. The contents of the mask are retained in the current edit profile, and are automatically used the next time you edit the same kind of data.

The MASK command is ignored in *formatted edit mode*. You enter formatted edit mode when you type the name of a previously defined format in the **Format Name** field on the Edit Entry panel when beginning an edit session. If you have defined a mask before entering formatted edit mode, the mask is not retained in the current edit profile.

Example

In Figure 96, the mask is displayed and the characters /* and */ are typed on the mask line.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PLI(ED@21PM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
001300          DO;
=COLS> -----1-----2-----3-----4-----5-----6-----7--
=MASK>                               /*                               */
001400          LP = 8;
001500          DO WHILE (ED@MPROJ(LP) = ' ');
001600              LP = LP-1;
001700          END;
I5 800          DO K=1 TO 4;
001900              IF ED@MLIB(K,1) &not.= ' ' THEN
002000                  DO;
002100                      LL(K) = 8;
002200                      DO WHILE
002300                          (ED@MLIB(K,LL(K)) = ' ');
002400                          LL(K) = LL(K)-1;
002500                      END;
002600                  END;
Command ==>> _____ Scroll ==>> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
    
```

Figure 96. Before the MASK Line Command

When you insert five lines, the new lines contain the contents of the mask. See Figure 97 on page 186.

MD—Make Dataline

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PLI(ED@21PM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
001300      DO;
=COLS>  -----1-----2-----3-----4-----5-----6-----7--
=MASK>                               /*                               */
001400          LP = 8;
001500      DO WHILE (ED@MPROJ(LP) = ' ');
001600          LP = LP-1;
001700      END;
001800      -                               /*                               */
001900      -                               /*                               */
002000      -                               /*                               */
002100      -                               /*                               */
002200      -                               /*                               */
002300      DO K=1 TO 4;
002400          IF ED@MLIB(K,1) = ' ' THEN
002500      DO;
002600          LL(K) = 8;
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange    F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel
```

Figure 97. After the MASK Line Command

MD—Make Dataline

The MD (make dataline) line command converts one or more ==MSG>, =NOTE=, =COLS>, or ===== (information) lines to data so they can be saved as part of your data set.

Syntax

```
MD[n]
MDD
MDMD
```

- n** The number of lines to be converted to data. If you do not type a number, or if the number you type is 1, only the line on which you type MD is converted.

Description

If you enter the MD line command on:

- Any line except a ==MSG>, =NOTE=, =COLS>, or ===== line, it is ignored.
- The TOP OF DATA and BOTTOM OF DATA lines, it is not allowed.
- An excluded line, any converted lines remain excluded and are converted.
- A line that contains a label, the label remains after the line is converted.

For best results, you should set your edit profile to NUMBER OFF and make sure that the record length of your data set or member is at least 80 before entering the MD line command. Otherwise, data on the right may be truncated.

To convert one or more lines to data:

1. Type MD in the line command area next to the line that is to be converted. If you also want to convert one or more lines that immediately follow this line, type a number greater than 1 after the MD command.

2. Press Enter. The lines are converted to data.

To convert a block of lines to data:

1. Type MDD in the line command area of both the first and last lines to be converted. You can scroll (or use the FIND or LOCATE command) between typing the first MDD and the second MDD, if necessary.
2. Press Enter. The lines that contain the two MDD commands and all eligible lines between them are converted to data.

Example

Figure 98 shows how you can convert a block of temporary lines to data by using the block form of the MD line command. The CLIST model of the DISPLAY service is inserted into member DEMO1, along with the notes for that model. Type MDD over the =NOTE= line flags in the line command area of the first and last lines of the block of lines that you want to convert to data.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      USERID.PRIVATE.CLIST(SCREEN) - 01.01          Columns 00001 00072
*****  ***** Top of Data *****
000010  ISPEXEC DISPLAY PANEL(PANELNAM)  MSG(MSG-ID)      +
000020                                CURSOR(FIELDNAM)  CSRPOS(POS#)     +
000030                                COMMAND(COMMANDS)  RETBUFFR(BUF-NAME) +
000040                                RETLGTH(LNG-NAME)  MSGLOC(MSG-FIELD)
MDD
=NOTE=   PANELNAM  - OPTIONAL, NAME OF THE PANEL TO BE DISPLAYED.
=NOTE=   MSG-ID    - OPTIONAL, IDENTIFIER OF A MESSAGE TO BE DISPLAYED ON
=NOTE=   THE PANEL.
=NOTE=   FIELDNAM  - OPTIONAL, NAME OF THE FIELD WHERE THE CURSOR IS TO BE
=NOTE=   POSITIONED.
=NOTE=   POS#      - OPTIONAL, POSITION OF CURSOR IN FIELD. DEFAULT IS 1.
=NOTE=   COMMANDS  - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE CHAIN
=NOTE=   OF COMMANDS.
=NOTE=   BUF-NAME  - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE
=NOTE=   REMAINING PORTION OF THE COMMAND CHAIN TO BE STORED
=NOTE=   IF AN ERROR OCCURS.
MDD
Command ==> _____ Scroll ==> CSR
F1=Help   F2=Split   F3=Exit   F5=Rfind   F6=Rchange  F7=Up
F8=Down   F9=Swap    F10=Left  F12=Right  F12=Cancel

```

Figure 98. Before the MD (Make Dataline) Line Command

When you press Enter, the lines on which the MDD commands are typed and all of the lines between them are converted to data. See Figure 99 on page 188.

O—Overlay Lines

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.CLIST(DEMO1) - 01.01 ----- Columns 00009 00080
***** ***** Top of Data *****
000010 ISPEXEC DISPLAY PANEL(PANELNAM) MSG(MSG-ID) +
000020 CURSOR(FIELDNAM) CSRPOS(POS#) +
000030 COMMAND(COMMANDS) RETBUFFR(BUF-NAME) +
000040 RETLGTH(LNG-NAME) MSGLOC(MSG-FIELD)
000050
000060 PANELNAM - OPTIONAL, NAME OF THE PANEL TO BE DISPLAYED.
000070 MSG-ID - OPTIONAL, IDENTIFIER OF A MESSAGE TO BE DISPLAYED ON
000080 THE PANEL.
000090 FIELDNAM - OPTIONAL, NAME OF THE FIELD WHERE THE CURSOR IS TO BE
000100 POSITIONED.
000110 POS# - OPTIONAL, POSITION OF CURSOR IN FIELD. DEFAULT IS 1.
000120 COMMANDS - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE CHAIN
000130 OF COMMANDS.
000140 BUF-NAME - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE
000150 REMAINING PORTION OF THE COMMAND CHAIN TO BE STORED
000160 IF AN ERROR OCCURS.
Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F4=Scroll F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 99. After the MD (Make Dateline) Line Command

O—Overlay Lines

The O (overlay) line command specifies the destination of data that is to be copied or moved by the C (copy) or M (move) line commands. The data that is copied or moved overlays blanks in an existing line of data. This allows you to rearrange a single-column list of items into multiple column, or tabular, format.

Syntax

O[n]
OO

n The number of lines to be overlaid. If you do not type a number, or if the number you type is 1, only one line is overlaid.

Description

To overlay one or more lines:

1. Type either M or C in the line command area of the line that is to be moved or copied.
2. Type O in the line command area of the line that the moved or copied line is to overlay. You can type a number after the O line command to specify the number of times that the M or C line command is to be performed.
3. Press Enter. The data being moved or copied overlays the specified line or lines.

To overlay a block of lines:

1. Type either MM or CC in the line command area of the first and last lines of a block of lines that is to be moved or copied. You can scroll (or use FIND or LOCATE) between typing the first command and the second command, if necessary.

O—Overlay Lines

2. Type 00 in the line command area of the first and last lines that the block of lines being moved or copied is to overlay. Again, you can scroll (or use FIND or LOCATE) between typing the first 00 and the second 00, if necessary.
3. Press Enter. The lines that contain the two CC or MM commands and all of the lines between them overlay the lines that contain the two OO commands and all of the lines between them.

Only blank characters in the lines specified with O or OO are overlaid with corresponding characters from the source lines. Characters that are not blank are not overlaid. The overlap affects only those characters within the current column boundaries.

The number of source and receiving lines need not be the same. If there are more receiving lines, the source lines are repeated until the receiving lines are gone. If there are more source lines than receiving lines, the extra source lines are ignored. The overlay operation involves only data lines. Special lines such as MASK, TABS, BNDS, and COLS are ignored as either source or receiving lines.

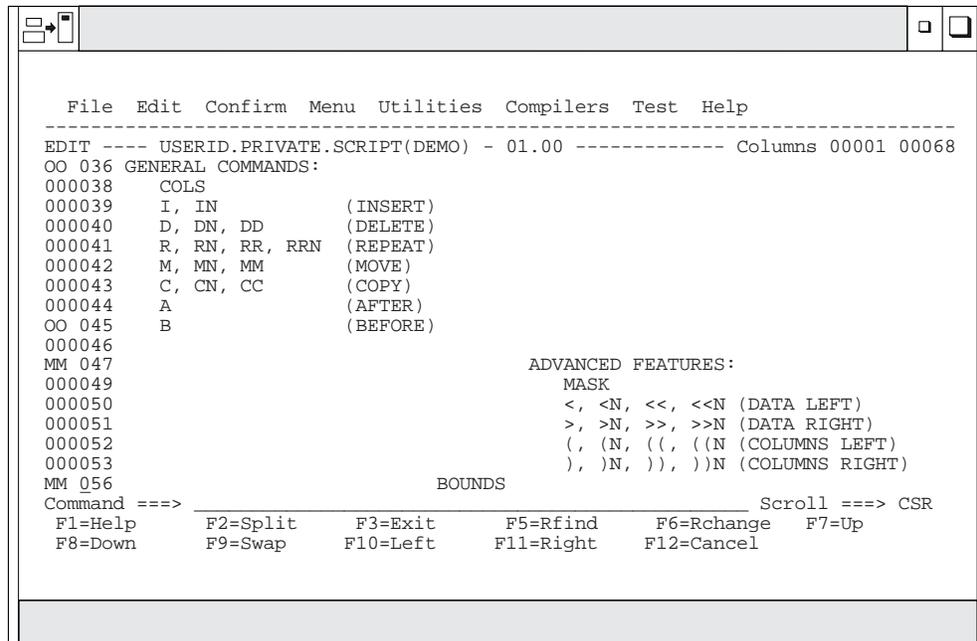
Note: There is no special support for DBCS data handling. You are responsible for DBCS data integrity when overlaying lines.

Two other line commands that allow you to specify a destination are the A (after) command and the B (before) command. See “A—Specify an “After” Destination” on page 163 and “B—Specify a “Before” Destination” on page 166 for more information.

Example

Figure 100 illustrates the O (overlay) line command. Suppose you were editing a list in a single left-adjusted column and wanted to place portions of the list side-by-side. First, using the) (column shift right) command, shift a portion of the list the appropriate amount to the right to overlay in a multiple column format. Type MM in the line command area to mark the beginning and end of the block of lines you want to move. Then type 00 in the line command area to mark the destination of the lines you want to move.

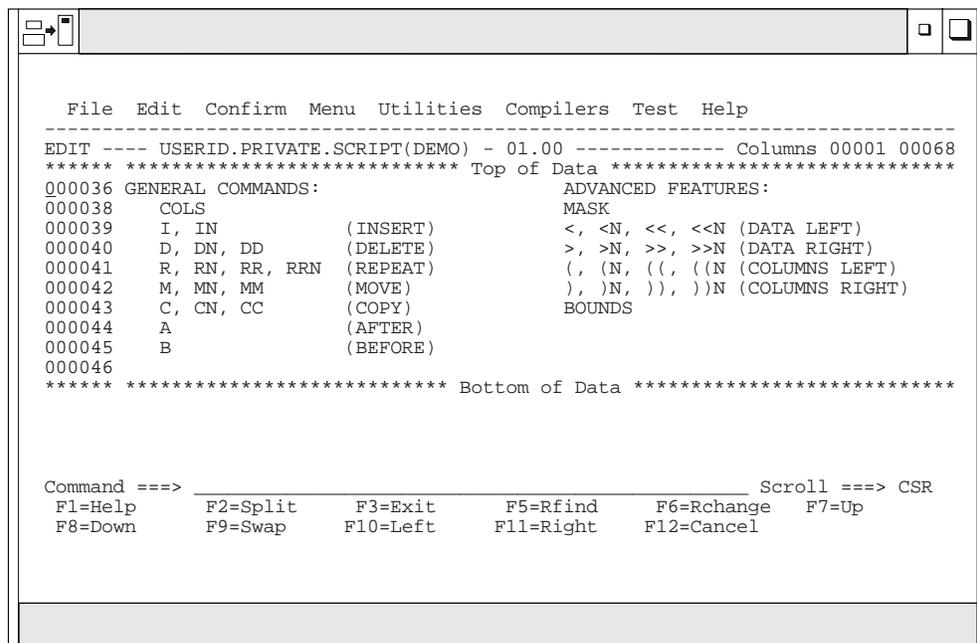
O—Overlay Lines



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(DEMO) - 01.00 ----- Columns 00001 00068
OO 036 GENERAL COMMANDS:
000038 COLS
000039 I, IN (INSERT)
000040 D, DN, DD (DELETE)
000041 R, RN, RR, RRN (REPEAT)
000042 M, MN, MM (MOVE)
000043 C, CN, CC (COPY)
000044 A (AFTER)
OO 045 B (BEFORE)
000046
MM 047 ADVANCED FEATURES:
000049 MASK
000050 <, <N, <<, <<N (DATA LEFT)
000051 >, >N, >>, >>N (DATA RIGHT)
000052 (, (N, ((, ((N (COLUMNS LEFT)
000053 ), )N, ))), ))N (COLUMNS RIGHT)
MM 056 BOUNDS
Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 100. Before the O (Overlay) Line Command

When you press Enter, the editor overlays the lines you marked to move on the destination block. See Figure 101.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(DEMO) - 01.00 ----- Columns 00001 00068
***** ***** Top of Data *****
000036 GENERAL COMMANDS:
000038 COLS
000039 I, IN (INSERT)
000040 D, DN, DD (DELETE)
000041 R, RN, RR, RRN (REPEAT)
000042 M, MN, MM (MOVE)
000043 C, CN, CC (COPY)
000044 A (AFTER)
000045 B (BEFORE)
000046
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 101. After the O (Overlay) Line Command

R—Repeat Lines

The R (repeat) line command repeats one or more lines in your data set or member immediately after the line on which the R command is entered.

Syntax

```
R[n]
RR[n]
```

- n** The number of lines to be repeated. If you do not type a number, or the number you type is 1, only the line on which you type R is repeated.

Description

To repeat one or more lines:

1. Type R in the line command area of the line that is to be repeated. If you want to repeat the line more than once, type a number that is greater than 1 immediately after the R command.
2. Press Enter. The editor inserts a duplicate copy or copies of the line immediately after the line that contains the R command.

To repeat a block of lines:

1. Type RR in the line command area of both the first and last lines to be repeated. You can scroll (or use FIND or LOCATE) between typing the first RR and the second RR, if necessary.
2. Press Enter. The lines that contain the two RR commands and all of the lines between them are repeated immediately after the line that contains the second RR command.

Example

As an example of the R command, assume that the following code is to be generated:

```
000200 DECLARE
000300 I  FIXED BINARY(31),      /* WORK COUNTER */
000400 J  FIXED BINARY(31),      /* WORK COUNTER */
000500 K  FIXED BINARY(31),      /* WORK COUNTER */
000600 L  FIXED BINARY(31);      /* WORK COUNTER */
```

The easiest way to generate this code is to type lines 000200 and 000300, repeat line 000300 three times, then move the cursor to each of the repeated lines and make the necessary changes. Figure 102 on page 192 shows how this is done.

Type lines 000200 and 000300 from the preceding code. Type the R3 command in the line command area of line 000300 to repeat it three times, as shown in Figure 102 on page 192.

R—Repeat Lines

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PLI(DCLS) - 01.01 ---- Columns 00001 00072
***** Top of Data *****
000100 /* SEGMENT 'DCLS' -- INCLUDED FROM SEGMENT 'MAIN' */
000200 DECLARE
R3 300 I FIXED BINARY(31), /* WORK COUNTER */
***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 102. Before the R (repeat) Line Command

When you press Enter, the editor repeats line 000300 three times. See Figure 103 .

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PLI(DCLS) - 01.01 ---- Columns 00001 00072
***** Top of Data *****
000100 /* SEGMENT 'DCLS' -- INCLUDED FROM SEGMENT 'MAIN' */
000200 DECLARE
000300 I FIXED BINARY(31), /* WORK COUNTER */
000400 I FIXED BINARY(31), /* WORK COUNTER */
000500 I FIXED BINARY(31), /* WORK COUNTER */
000600 I FIXED BINARY(31), /* WORK COUNTER */
***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 103. After the R (Repeat) Line Command

You can then edit the repeated lines to match the desired code, as shown in Figure 104.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.PLI(DCLS) - 01.01 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* SEGMENT 'DCLS' -- INCLUDED FROM SEGMENT 'MAIN' */
000200 DECLARE
000300     I    FIXED BINARY(31),      /* WORK COUNTER */
000400     J    FIXED BINARY(31),      /* WORK COUNTER */
000500     K    FIXED BINARY(31),      /* WORK COUNTER */
000600     L    FIXED BINARY(31),_     /* WORK COUNTER */
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 104. Editing after the R (Repeat) Line Command

S—Show Lines

The S (show line) line command causes one or more lines in a block of excluded lines to be redisplayed. The redisplayed lines have the leftmost indentation levels; they contain the fewest leading blanks. See “Redisplaying Excluded Lines” on page 65 for more information about redisplaying excluding lines.

Syntax

S[n]

- n** The number of lines to be redisplayed. If there are only 2 excluded lines, and you do not type a number, or if the number you type is 1, both lines are redisplayed. If more than 2 lines are excluded, only one line is redisplayed if you do not type a number, or if the number you type is 1.

Description

To redisplay a line or lines of a block of excluded lines:

1. Type S in the line command area next to the dashed line that shows where a line or lines has been excluded. The message in the dashed line tells you how many lines are excluded.

If you want to redisplay more than one line, type a number greater than 1 after the S command. If you type S3, for example, the three lines with the leftmost indentation level are displayed again. If more than three lines exist at this indentation level, only the first three are displayed.

2. Press Enter. The line or lines with the fewest leading blanks are redisplayed.


```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      USERID.PRIVATE.PLI(DCLS) - 01.01          Columns 00001 00072
*****  ***** Top of Data *****
000100 /* SEGMENT 'DCLS'  -- INCLUDED FROM SEGMENT 'MAIN'  */
000200 DECLARE
000300     I      FIXED BINARY(31),      /* WORK COUNTER
000400 DECLARE
000500     COUNT   FIXED BINARY (31) AUTOMATIC INIT (1),
000600     HALVES  FIXED BINARY (31),
000700     QUARTERS FIXED BINARY (31),
000800     DIMES   FIXED BINARY (31),
000900     NICKELS FIXED BINARY (31),
001000     SYSPRINT DATA SET STREAM OUTPUT PRINT;
001100     DO HALVES = 100 TO 0 BY -50;
001200     DO QUARTERS = (100 - HALVES) TO 0 BY -25;
001300     DO DIMES = ((100 - HALVES - QUARTERS)/10)*10 TO 0 BY -10;
----- 3 LINES(S) NOT DISPLAYED
001700     END;
001800     END;
001900     END;
Command ==>>> _____ Scroll ==>>> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down      F9=Swap       F10=Left    F12=Right    F12=Cancel

```

Figure 106. After the S (Show) Line Command

TABS—Control Tabs

The TABS line command:

- Displays the =TABS> (tab-definition) line
- Defines tab positions for software, hardware, and logical tabs.

Use PROFILE to check the setting of tabs mode and the logical tab character. See “Using Tabs” on page 71 if you need more information about using tabs.

Syntax

TABS

Description

When you type TABS in the line command area, =TABS> is displayed along with any previously defined tab positions. To remove the =TABS> line, use the D (delete) line command or the RESET primary command, or end the edit session. The =TABS> line is never saved as part of the data.

The tab definitions remain in effect, even if they are not displayed, until you change them. Tab definitions are retained in the current edit profile, and are automatically used the next time you edit the same kind of data.

Examples

This section contains two examples: one using software and hardware tabs, and one using software tab fields.

TABS—Control Tabs

Using Software and Hardware Tabs

Edit a data set, type TABS ALL on the Command line, and press Enter:

```
Command ==> TABS ALL
```

Now, type COLS in the line command area and press Enter again. A partial =COLS> line with positions 9 through 45 is shown in the following example:

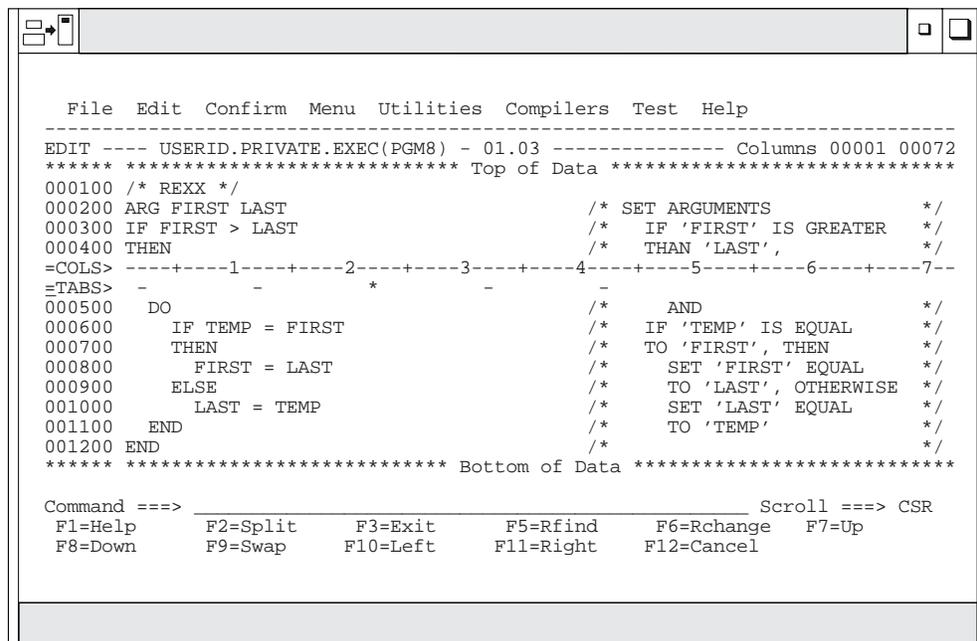
```
=COLS> -1----+----2----+----3----+----4----+
```

Next use the TABS line command to define software and hardware tabs. Type TABS in the line command area beneath the =COLS> line and press Enter.

When the =TABS> line appears, type hyphens in columns 15, 25, and 35, and asterisks in columns 20, 30, and 40, using the =COLS> line to find these columns:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>      -      *      -      *      -      *
```

With the preceding =TABS> line, you can move the cursor to a software tab position (hyphen) by pressing Enter, even if another character already occupies that position. To move the cursor to a hardware tab position (one space to the right of an asterisk), press either the Tab Forward or Tab Backward key. See Figure 107.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ---- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
=TABS> -1----+----2----+----3----+----4----+5----+6----+7--
=TABS>      -      *      -      *      -      *
000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST', THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
000900 ELSE /* TO 'LAST', OTHERWISE */
001000 LAST = TEMP /* SET 'LAST' EQUAL */
001100 END /* TO 'TEMP' */
001200 END /*
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 107. TAB Line Command Example. A =TABS> line with four software tabs and one hardware tab defined.

Using Software Tab Fields

You can define a *software tab field* by typing underscores or hyphens in two or more consecutive columns. This moves the cursor to the first non-blank character in the field. If the field contains all blanks, the cursor moves to the beginning of the field.

Using the example in the preceding section, create a software tab field by typing hyphens in columns 10 through 14. Then type some data inside the field and at each of the other tab positions, but below the =TABS> line:

```
=COLS> -1-----+-----2-----+-----3-----+-----4-----+
=TABS>  ----- *   -   *   -   *
          123         456         789_
```

Notice in the preceding example that the cursor is positioned to the right of data string 789. With the cursor in this position, press Enter. The cursor moves under the 1 in the 123 data string, not to column 10, which is the beginning of the field.

TE—Text Entry

The TE (text entry) line command provides one very long line wrapped around many lines of the display to allow power typing for text entry. The editor does the formatting for you.

The TE line command is different from the I (insert) line command. The I command inserts a specified number of separate, blank lines as well as the mask, if there is one, as you typed it. With the TE command, the input data is formatted, only mask line characters outside the current boundaries are added to the formatted lines.

Syntax

TE[n]

n The number of blank lines to be added. If you do not type a number, the display is filled with blanks from the line following the TE to the bottom of the screen.

Description

Before you enter text entry mode, consider the following:

- If you are going to be typing text in paragraph form, make sure caps mode is off. Otherwise, when you press Enter, your text changes to all caps.
- You may want to turn off number mode to prevent sequence numbers from writing over any of your text.
- Make sure the bounds setting is where you want it so that the text will flow correctly when you end text entry mode.

To enter text entry mode:

1. Type TE in the line command area. If you want to specify several blank lines, type a number greater than 1 immediately after the TE command. If the number that you type is greater than the number of lines remaining on the display, the vertical bar that shows where you will run out of room is not displayed and the keyboard does not lock at the last character position on the display. You can scroll down to bring the additional blank text entry space into view.
2. Press Enter. The editor inserts a single continuous blank area for the specified number of lines or to the bottom of the display.

To begin a new paragraph:

1. Use the return (Enter), cursor movement, or Tab keys to advance the cursor enough spaces to leave one blank line on the display.

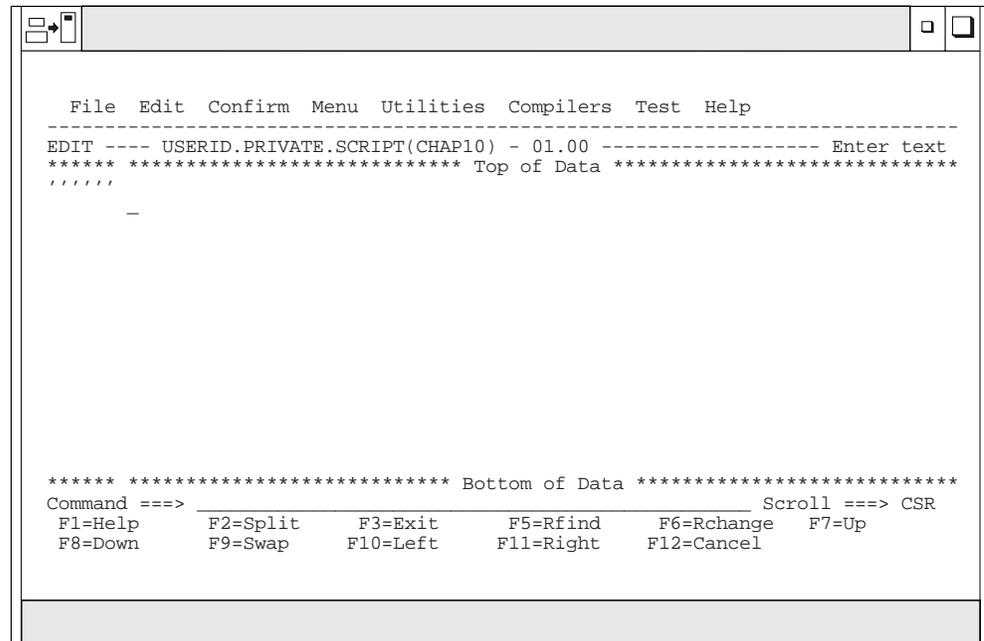


Figure 109. After the TE (Text Entry) Line Command

When you enter text, some of the words are split between lines, with part of the word at the right end of a line and the remainder of the word at the beginning of the next line. See Figure 110.

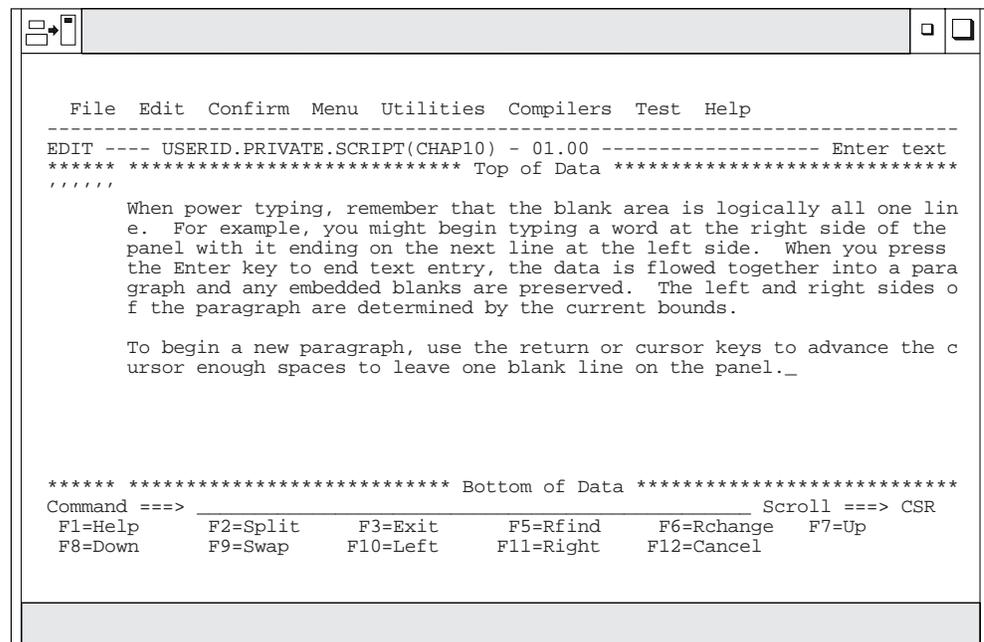


Figure 110. Sample Text During Text Entry Mode.

When you press Enter, the editor exits text entry mode. As shown in Figure 111 on page 200, the text flows between the bounds settings and the line numbers are displayed in the line command area.

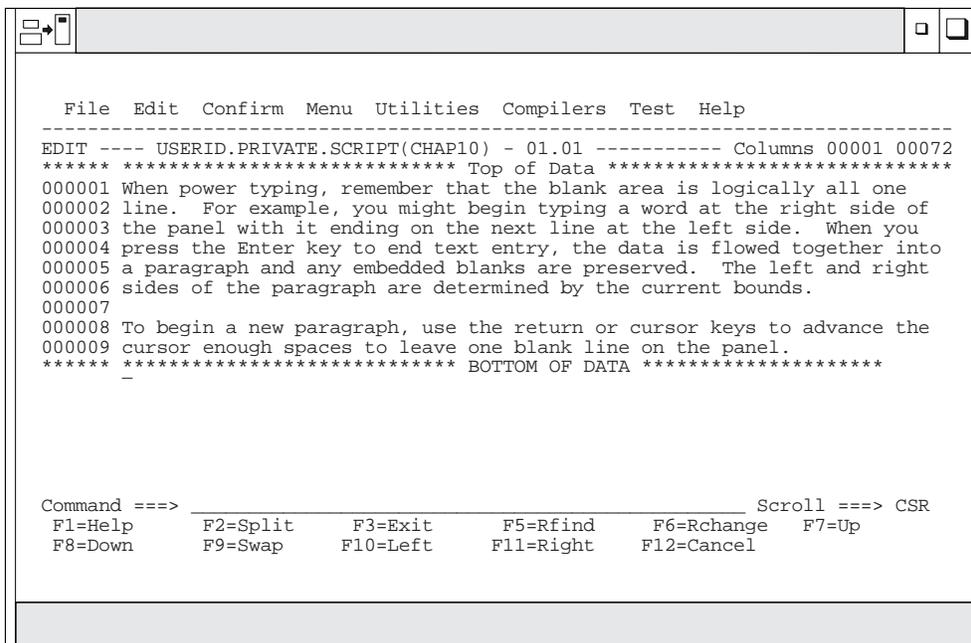


Figure 111. Sample Text After Text Entry Mode.

TF—Text Flow

The TF (text flow) line command restructures paragraphs. This is sometimes necessary after deletions, insertions, or splitting.

Syntax

TF[n]

n The column number to which the text should be flowed. The default is the panel width when default boundaries are in effect. If you are using nondefault bounds, the right boundary is used. This is different from the TFLOW macro command, which always defaults to the right boundary.

If a number greater than the right boundary is specified, the right boundary is used.

Description

To flow text:

1. Type TF in the line command area of the line at which you want the text to begin flowing. If you want to specify the rightmost column position for the restructured text, type a number greater than 1 immediately after the TF command.
2. Press Enter. The text is flowed from the beginning of that line to the end of the paragraph.

See “Word Processing” on page 68 and “Formatting Paragraphs” on page 68 for more information.

Example

Figure 112 demonstrates text restructuring. The bounds are set at columns 1 and 72. A TF50 command is typed on line 000041.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT      USERID.PRIVATE.SCRIPT(DEMO) - 01.01          Columns 00001 00072
*****  ***** Top of Data *****
000036  When the Enter key is pressed, the new
000037  information is automatically reflowed to fit within the currently
000038  defined boundaries (see BOUNDS) line command), and any unused blank
000039  lines at the end of the new text are deleted.
000040
000041  TF50 1 If insufficient blank lines have been generated, the keyboard
000042  will lock when you attempt to type beyond the last character
000043  position of the last blank line. A vertical bar (|) will be
000044  displayed above the cursor at the locked position. To generate
000045  more blank lines, reset the keyboard to unlock it and then press
000046  the Enter key.
000047
000048  Multiple paragraphs can be typed by using either of the following
000049  techniques:
000050
000051
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right     F12=Cancel

```

Figure 112. Before the TF (Text Flow) Line Command

When you press Enter, the editor takes all text in that paragraph between columns 1 and 72 and reformats it between columns 1 and 50. See Figure 113.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(DEMO) - 01.01 ----- Columns 00001 00068
*****  ***** Top of Data *****
000036  When the Enter key is pressed, the new
000037  information is automatically reflowed to fit within the currently
000038  defined boundaries (see BOUNDS line command), and any unused blank
000039  lines at the end of the new text are deleted.
000040
000041  If insufficient blank lines have been generated,
000042  the keyboard will lock when you attempt to
000043  type beyond the last character position of the
000044  last blank line. A vertical bar (|) will be
000045  displayed above the cursor at the locked position.
000046  To generate more blank lines, reset the keyboard
000047  to unlock it and then press the Enter key.
000048
000049  Multiple paragraphs can be typed by using either of the following
000050  techniques:
000051
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right     F12=Cancel

```

Figure 113. After the TF (Text Flow) Line Command

TS—Text Split

The TS (text split) line command moves part or all of a line of text to the following line. This makes it easier for you to add new material to existing text.

Syntax

TS[n]

- n** The number of blank lines to be inserted between the split lines. If you do not type a number, or if the number that you type is 1, the editor inserts only one blank line.

Description

To split a line:

1. Type TS in the line command area of the line you would like to split. If you want to insert more than one blank line between the split lines, type a number greater than 1 immediately after the TS command.
2. Move the cursor to the desired split point.
3. Press Enter.

To rejoin lines, use the TF (text flow) line command. See “TF—Text Flow” on page 200 for more information.

For more information about splitting lines and other word processing commands, see “Word Processing” on page 68 and “Splitting Lines” on page 70.

Examples

Figure 114 shows how to split text and to insert blank lines. To split the text and insert three lines, type TS3 in the line command area of the line you want to split and place the cursor where you want the line split.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(DEMO) - 01.02 ----- Columns 00001 00068
***** ***** Top of Data *****
000059 The TS command is intended for insertion of new material into
TS3 60 existing text. To split a line, type the characters TS in a line
000061 command area and then move the cursor to the desired split point (on
000062 the same line) before pressing the Enter key. A new line is
000063 inserted following the line that contains the TS, and all text to
000064 the right of the cursor is moved to the beginning of the next line
000065 (following the inserted line). A number may follow the TS to
000066 cause additional lines to be inserted.
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel

```

Figure 114. Before TS (Text Split) Line Command

When you press Enter, the line is split at the cursor position and the editor inserts the number of blank lines specified, as shown in Figure 115.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.SCRIPT(DEMO) - 01.02 ----- Columns 00001 00068
***** ***** Top of Data *****
000059 The TS command is intended for insertion of new material into
000060 existing text. _
////////
////////
////////
000061 To split a line, type the characters TS in a line
000062 command area and then move the cursor to the desired split point (on
000063 the same line) before pressing Enter. A new line is
000064 inserted following the line that contains the TS, and all text to
000065 the right of the cursor is moved to the beginning of the next line
000066 (following the inserted line). A number may follow the TS to
000067 cause additional lines to be inserted.
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel

```

Figure 115. After TS (Text Split) Line Command

UC—Convert Characters to Uppercase

The UC (uppercase) line command converts characters in a data set or member from lowercase to uppercase. However, it does not affect the caps mode of the data that you are editing.

UC—Convert Characters to Uppercase

Syntax

UC[n]
UCC
UCUC

- n** The number of lines to be converted to uppercase. If you do not type a number, or if the number you type is 1, only the line on which you type UC is converted to uppercase.

Description

To convert characters on one or more lines to uppercase:

1. Type UC in the line command area of the source code line that contains the characters that you want to convert. To convert characters on lines following this one, type a number greater than 1 after the UC command.
2. Press Enter. The characters on the source code line or lines are converted to uppercase.

To convert characters in a block of lines to uppercase:

1. Type UCC in the line command area of both the first and last source code lines that contain characters that are to be converted. You can scroll (or use FIND or LOCATE) between typing the first UCC and the second UCC, if necessary.
2. Press Enter. The characters in the source code lines that contain the two UCC commands and in all of the source code lines between them are converted to uppercase.

See the LC (lowercase) line command and the CAPS primary and macro commands on pages 157, 202, and 298 for information about converting characters from uppercase to lowercase and vice versa.

Example

Figure 116 shows how to convert lines of text to uppercase. To convert lines of text to uppercase, place the UC command and the number of lines you want to convert in the line command area where you want the conversion to start.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM1) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
UC 200 arg first last
000300 IF FIRST > LAST
000400 THEN
000500 DO
000510 IF ...
000520 THEN
000530 ...
000540 ELSE
000550 ...
000600 END
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 116. Before the UC (Uppercase) Line Command

When you press Enter, the editor converts the lines specified to uppercase. See Figure 117.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM1) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST
000300 IF FIRST > LAST
000400 THEN
000500 DO
000510 IF ...
000520 THEN
000530 ...
000540 ELSE
000550 ...
000600 END
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 117. After the UC (Uppercase) Line Command

X—Exclude Lines

The X (exclude) line command replaces one or more lines on the panel with a dotted line. The dotted line contains a message that specifies how many lines have been excluded.

X—Exclude Lines

The excluded lines are not erased. They are simply hidden from view and can still be affected by edit line, primary, and macro commands.

Syntax

X[n]
XX

n The number of lines to be excluded. If you do not type a number, or if the number that you type is 1, PDF excludes only the line on which you type the X command.

Description

To exclude one or more lines:

1. Type X in the line command area of the line that you want to exclude. If you want to exclude one or more lines that immediately follow this line, type a number greater than 1 immediately after the X command.
2. Press Enter. The lines are excluded from the panel.

To exclude a block of lines:

1. Type XX in the line command area of both the first and last lines that you want to exclude. You can scroll (or use FIND or LOCATE) between typing the first XX and the second XX, if necessary.
2. Press Enter. The lines that contain the two XX commands and all of the lines between them are excluded.

See “Excluding Lines” on page 65 for more information on using this command.

Example

Figure 118 shows how lines are excluded from a member. To exclude six lines, type X6 in the line command area.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ----- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
X60300 TEST-#-----+
000400 TEST-#         |
000500 TEST-#         |
000600 TEST-#         |
000700 TEST-#         |
000710 TEST-#-----+
000800 TEST-#
000900 TEST-#
001000 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 118. Before the X (Exclude) Line Command

When you press Enter, the editor excludes the specified lines. See Figure 119 .

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(TESTDATA) ----- Columns 00001 00072
***** ***** Top of Data *****
000010 TEST-#
000100 TEST-#
000200 TEST-#
- - - - - 6 LINE(S) NOT DISPLAYED
000800 TEST-#
000900 TEST-#
001000 TEST-#
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 119. After the X (Exclude) Line Command

X—Exclude Lines

Chapter 10. Edit Primary Commands

Primary commands affect the entire data set being edited, whereas line commands usually affect only a single line or block of lines. To enter a primary command, do either of the following:

- Type the command on the Command line and press Enter
- Press the function key to which the command is assigned.

Most primary commands can be abbreviated. In fact, many can be typed as a single letter, such as L for LOCATE or F for FIND. For a list of command abbreviations, see Appendix A. Abbreviations for Commands and Other Values.

Each command description consists of the following information:

Syntax

A syntax diagram for coding the command, including a description of any required or optional operands.

Description

A summary of the function and operation of the command. This definition also refers to other commands that can be used with this command.

Example

Sample usage of the command.

Edit Primary Command Notation Conventions

The syntax of the edit primary commands uses the following notation conventions:

Uppercase

Uppercase commands or operands must be spelled as shown (in either uppercase or lowercase).

Lowercase

Lowercase operands are variables; substitute your own values.

Underscore

Underscored operands are the system defaults.

Brackets ([])

Operands in brackets are optional.

Stacked operands

Stacked operands show two or more operands from which you can select. If you do not choose any, the Editor uses the default operand.

Braces ({ })

Braces show two or more operands from which you must select one. .

OR (|)

The OR (|) symbol shows two or more operands from which you must select one.

Edit Primary Command Summary

The following table summarizes the edit primary commands. See the complete description of the commands on the referenced page.

Edit Primary Command Summary

Table 5. Summary of the Primary Commands

Command Syntax	topic	Description
AUTOLIST [ON] [OFF]	“AUTOLIST—Create a Source Listing Automatically” on page 213	Controls the automatic printing of data to the ISPF list data set.
AUTONUM [ON] [OFF]	“AUTONUM—Number Lines Automatically” on page 215	Controls the automatic renumbering of data when it is saved.
AUTOSAVE [ON] [OFF PROMPT] [OFF NOPROMPT]	“AUTOSAVE—Save Data Automatically” on page 217	If the data is changed, automatically saves it when you issue an END command.
BOUNDS [left-col right-col]	“BOUNDS—Control the Edit Boundaries” on page 218	Sets the left and right boundaries.
BROWSE [member]	“BROWSE—Browse from within an Edit Session” on page 220	Browse a data set or member without leaving your current edit session.
BUILTIN cmdname	“BUILTIN—Process a Built-In Command” on page 219	Processes a built-in command even if a macro with the same name has been defined.
CANCEL	“CANCEL—Cancel Edit Changes” on page 221	Ends the edit session without saving any of the changes.
CAPS [ON] [OFF]	“CAPS—Control Automatic Character Conversion” on page 221	Sets caps mode.
CHANGE string-1 string-2 [range] [NEXT] [CHARS] [X] [col-1 [col-2]] [ALL] [PREFIX] [NX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]	“CHANGE—Change a Data String” on page 222	Changes a data string into another string.
COMPARE {dsname} {EXCLUDE} {SAVE} {SYSIN} {NEXT}	“COMPARE—Edit Compare” on page 224	Compares library member or data set with the data being edited.
COPY [member] [AFTER label] [(member)][BEFORE label] [data set name (member)][linenum range] [data set name]	“COPY—Copy Data” on page 227	Copies a library member or data set into the data being edited.
CREATE [member] [range] (member) [range] [data_set(member)] [range] [data_set name]	“CREATE—Create Data” on page 231	Writes the data you are editing into a library member or data set only if it does not already exist.
CUT [lptr-range] [DEFAULT clipboardname] [REPLACE] [DISPLAY]	“CUT—Cut and Save Lines” on page 235	Saves lines to a clipboard for later retrieval by PASTE command.
DEFINE name {MACRO CMD } {MACRO PGM } {ALIAS name-2} {NOP } {RESET } {DISABLED }	“DEFINE—Define a Name” on page 236	<ul style="list-style-type: none"> • Assigns an alias to a macro or built-in command. • Disables the use of a macro or built-in command. • Identifies a macro that replaces a built-in command of the same name. • Identifies programs that are edit macros.

Edit Primary Command Summary

Table 5. Summary of the Primary Commands (continued)

Command Syntax	topic	Description
DELETE {ALL X NX} {range X NX} {ALL range }	“DELETE—Delete Lines” on page 238	Deletes lines from the data you are editing.
EDIT [member]	“EDIT—Edit from within an Edit Session” on page 239	Edits a data set or member without leaving your current edit session (recursive edit).
END	“END—End the Edit Session” on page 241	Ends the current edit session.
EXCLUDE string [range] [NEXT] [CHARS] [col-1 [col-2]] [ALL] [PREFIX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]	“EXCLUDE—Exclude Lines from the Display” on page 242	Excludes lines from the panel.
FIND string [range] [NEXT] [CHARS] [X] [col-1 [col-2]] [ALL] [PREFIX] [NX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]	“FIND—Find a Data String” on page 243	Finds a data string.
FLIP [label-range]	“FLIP—Reverse Exclude Status of Lines” on page 245	Reverses the exclude status of a specified range of lines in a file or all the lines in the file.
HEX [ON DATA] [ON VERT] [OFF]	“HEX—Display Hexadecimal Characters” on page 247	Specifies whether the hexadecimal form of the data should be displayed.
HILITE [ON] [AUTO] [RESET] [PAREN] [FIND] [CURSOR] [SEARCH] [DISABLED] [OFF] [DEFAULT] [LOGIC] [OTHER] [IFLOGIC] [ASM] [DOLOGIC] [BOOK] [NOLOGIC] [C] [COBOL] [DTL] [JCL] [PANEL] [PASCAL] [PLI] [REXX] [SKEL]	“HILITE—Enhanced Edit Coloring” on page 250	Highlights, in user-specified colors, numerous language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. Can also be used to set default colors for the data area in non-program files and for any characters typed since the previous Enter or function key entry.
IMACRO {name NONE}	“IMACRO—Specify an Initial Macro” on page 253	Saves the name of an initial macro in the edit profile.
LEVEL num	“LEVEL—Specify the Modification Level Number” on page 254	Sets the modification level number to be kept as part of the PDF library statistics.

Edit Primary Command Summary

Table 5. Summary of the Primary Commands (continued)

Command Syntax	topic	Description
LOCATE {label line-number} LOCATE [FIRST] {CHANGE } [range] [LAST] {COMMAND } [NEXT] {ERROR } [PREV] {EXCLUDED} {LABEL } {SPECIAL }	“LOCATE—Locate a Line” on page 255	Locates a line.
MODEL [model-name [qualifier...]] {AFTER label} [NOTES] {BEFORE label} [NONOTES] MODEL [CLASS [class-name]]	“MODEL—Copy a Model into the Current Data Set” on page 257	Copies a model into the data you are editing or defines the current model class.
MOVE [member] [AFTER label] (member) [BEFORE label] [data set name (member)] [data set name]	“MOVE—Move Data” on page 260	Moves a library member or data set into the data you are editing.
NONUMBER	“NONUMBER—Turn Off Number Mode” on page 264	Turns off number mode.
NOTES [ON] [OFF]	“NOTES—Display Model Notes” on page 264	Specifies whether the MODEL command is to display notes.
NULLS [ON STD] [ON ALL] [OFF]	“NULLS—Control Null Spaces” on page 265	Controls null spaces.
NUMBER [ON] [STD] [DISPLAY] [OFF] [COBOL] [STD COBOL] [NOSTD] [NOCOBOL] [NOSTD NOCOBOL]	“NUMBER—Generate Sequence Numbers” on page 266	Generates sequence numbers.
PACK [ON] [OFF]	“PACK—Compress Data” on page 268	Specifies whether data is to be stored normally or compressed.
PASTE [clipboardname] [AFTER label] [BEFORE label] [KEEP]	“PASTE—Move or Copy Lines from Clipboard” on page 268	Moves or copies lines from a clipboard into an edit session.
PRESERVE [ON] [OFF]	“PRESERVE - Enable Saving of Trailing Blanks” on page 269	Specifies whether trailing blanks should be saved when data is stored.
PROFILE [name] [number] PROFILE {LOCK UNLOCK} PROFILE RESET	“PROFILE—Control and Display Your Profile” on page 270	Controls and displays your profile.
RCHANGE	“RCHANGE—Repeat a Change” on page 273	Repeats the most recently processed CHANGE command.
RECOVERY [ON OFF] [WARN NOWARN SUSP]	“RECOVERY—Control Edit Recovery” on page 273	Controls edit recovery.
RENUM [ON] [STD] [DISPLAY] [COBOL] [STD COBOL]	“RENUM—Renumber Data Set Lines” on page 274	Renumbers data set lines.

Table 5. Summary of the Primary Commands (continued)

Command Syntax	topic	Description
REPLACE [member] [range] REPLACE [data set name (member)] [range] REPLACE [data set (member)] [range] REPLACE [data set] [range]	“REPLACE—Replace Data” on page 277	Writes the data you are editing into a library member even if it already exists.
RESET [CHANGE] [range] [COMMAND] [ERROR] [EXCLUDED] [FIND] [LABEL] [SPECIAL]	“RESET—Reset the Data Display” on page 280	Resets the data display.
RFIND	“RFIND—Repeat Find” on page 282	Locates the data string defined by the most recently processed SEEK, FIND, or CHANGE command, or excludes a line that contains the data string from the previous EXCLUDE command.
RMACRO {name NONE}	“RMACRO—Specify a Recovery Macro” on page 282	Saves the name of a recovery macro in the edit profile.
SAVE	“SAVE—Save the Current Data” on page 283	Saves the current data without ending the edit session.
SETUNDO [STORAGE RECOVER] [OFF]	“SETUNDO—Set the UNDO Mode” on page 284	Sets the UNDO mode.
SORT [range] [X] [sort-field1 ... sort-field5] [NX]	“SORT—Sort Data” on page 286	Puts data in a specified order.
STATS [ON] [OFF]	“STATS—Generate Library Statistics” on page 288	Specifies whether PDF library statistics are to be created when this member is saved.
SUBMIT [range]	“SUBMIT—Submit Data for Batch Processing” on page 288	Submits the data you are editing for batch processing.
TABS [ON] [STD] [OFF] [ALL] [tab-character]	“TABS—Define Tabs” on page 289	Defines tab positions for software, hardware, and logical tabs.
UNDO	“UNDO—Reverse Last Edit Interaction” on page 290	Removes the data modifications of a previous interaction.
UNNUMBER	“UNNUMBER—Remove Sequence Numbers” on page 293	Removes sequence numbers.
VERSION num	“VERSION—Control the Version Number” on page 295	Sets the version number to be kept as part of the PDF library statistics.
VIEW [member]	“VIEW—View from within an Edit Session” on page 296	View a data set or member without leaving your current edit session.

AUTOLIST—Create a Source Listing Automatically

The AUTOLIST primary command sets autolist mode, which controls the automatic printing of data to the ISPF list data set.

AUTOLIST

Syntax

AUTOLIST [ON]
 [OFF]

- ON** Generates a source listing in the ISPF list data set for eventual printing when you end an edit session in which you changed and saved data.
- OFF** No source listing is generated.

Description

Autolist mode is saved in the edit profile. To check the current setting of autolist mode:

1. On the Command line, type:
 Command ==> PROFILE 3
2. Press Enter. The third line of the edit profile shows the autolist mode setting.

To turn on autolist mode:

1. On the Command line, type:
 Command ==> AUTOLIST ON
2. Press Enter.

To turn off autolist mode:

1. On the Command line, type:
 Command ==> AUTOLIST OFF
2. Press Enter.

Example

This example shows how to use the AUTOLIST command to save a copy of a source code listing in the ISPF list data set and to print the list data set.

1. As you edit a data set, you decide to store a listing of the source code in the ISPF list data set so that you can print it later. Enter the PROFILE 3 command to display the first 3 lines of the edit profile. This shows you whether autolist mode is on or off.

Command ==> PROFILE 3

2. You can see from the edit profile that autolist mode is off:

```
=PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....  
=PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.....  
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON.....
```

3. Enter the AUTOLIST ON command to turn on autolist mode:

Command ==> AUTOLIST ON

The edit profile changes accordingly:

```
=PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....  
=PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.....  
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST ON....STATS ON.....
```

4. After editing the data set, save your changes by entering the END command. The changes are saved because, as you can see in the preceding partial edit profile, autosave mode is on.

Command ==> END

The PDF component creates an ISPF list data set with the contents of the data set member that you were editing. The name of the list data set is:
`prefix.user-id.SPFn.LIST`

Note: Refer to *ISPF User's Guide* for information about list data sets.

- Before leaving the PDF component, use the jump function to go to option 0.2 and check the log/list defaults:

Command `===> =0.2`

The Log and List Defaults panel shows the current default settings for the handling of log and list data sets.

- Because you want to print the list data set, make sure that the PD option is entered in the **Process Option** field under the List Data Set Default Options heading:

Process option `===> PD`

Note: Also, make sure that the appropriate JCL information is entered at the bottom of the Log and List Defaults panel so that the print job is submitted.

- You can now end the session, knowing that the list data set will be printed:

Command `===> =X`

- When the session ends, TSO displays a message that says the print job has been submitted.

AUTONUM—Number Lines Automatically

The AUTONUM primary command sets autonum mode, which controls the automatic renumbering of data when it is saved.

Syntax

AUTONUM `[ON]`
`[OFF]`

ON Turns on automatic renumbering. When number mode is also on, the data is automatically renumbered when it is saved.

OFF Turns off automatic renumbering. Data is not renumbered.

Description

When number mode is on, the first line of a data set or member is normally line number 000100, the second number is 000200, and so forth. However, as lines are inserted and deleted, the increment between line numbers can change.

For example, you might think that when a line is inserted between 000100 and 000200, line 000200 would be given the number 000300 and the new line would become 000200. Instead, the existing lines retain their numbers and the new line is given line number 000110.

Therefore, if the original line number increments are important to you, the AUTONUM command renumbers your lines automatically so that the original increments are maintained.

AUTONUM

Autonum mode is saved in the edit profile. To check the current settings of number mode and autonum mode:

1. On the Command line, type:
Command ==> PROFILE 3
2. Press Enter. The first line of the edit profile shows the number mode setting and the third line shows the autonum mode setting.

To turn on autonum mode:

1. On the Command line, type:
Command ==> AUTONUM ON
2. Press Enter.

To turn off autonum mode:

1. On the Command line, type:
Command ==> AUTONUM OFF
2. Press Enter.

Example

This example shows a practical application of AUTONUM command usage. You have been editing a data set with number mode on.

Note: If you are editing a data set or member with number mode off and then decide to turn number mode on, make sure that columns 1 through 6 of your data set are blank. Otherwise, the sequence numbers created by the NUMBER command can overlay any of your data in columns 1 through 6. Use either the COLUMN SHIFT or DATA SHIFT line command to indent the data.

You now want to end the edit session. However, since you had to insert and delete many lines, your line numbering is no longer uniform. Therefore, you decide to use autonum mode so that the next time you edit this data set the line numbers will be correct.

1. First, check the edit profile to see whether autonum mode is already on by entering the PROFILE 3 command to display the first 3 lines of the edit profile.

```
Command ==> PROFILE 3
```

2. You can see from the edit profile that autonum mode is off:

```
=PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....  
=PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.....  
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON.....
```

3. Enter the AUTONUM ON command to turn on autonum mode:

```
Command ==> AUTONUM ON
```

The edit profile changes accordingly:

```
=PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....  
=PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.....  
=PROF> ....AUTOSAVE ON....AUTONUM ON....AUTOLIST ON....STATS ON.....
```

4. After editing the data set, save your changes by entering the END command. The changes will be saved because, as you can see in the preceding partial edit profile, autosave mode is on.

```
Command ==> END
```

The PDF component saves the data set that you were editing, along with any changes. The next time you edit the data set, the line numbers will have the proper increments.

AUTOSAVE—Save Data Automatically

The AUTOSAVE primary command sets autosave mode, which controls whether changed data is saved when you enter END.

Syntax

```
AUTOSAVE [ON          ]
          [OFF [PROMPT]]
          [OFF NOPROMPT]
```

ON Turns autosave mode on. When you enter END, any changed data is saved.

OFF PROMPT

Turns autosave mode off with the PROMPT operand. You are notified that changes have been made and that either the SAVE command (followed by END) or CANCEL must be used. When you use AUTOSAVE PROMPT by itself, it implies the OFF command.

OFF NOPROMPT

Turns autosave mode off with the NOPROMPT operand. You are not notified and the data is not saved when you issue an END command. END becomes an equivalent to CANCEL. Use the NOPROMPT operand with caution.

Description

Data is considered changed if you have operated on it in any way that could cause a change. Shifting a blank line or changing a word to the same word does not actually alter the data, but the editor considers this data changed. When you enter SAVE, the editor resets the change status.

Autosave mode, along with the PROMPT operand, is saved in the edit profile. To check the current setting of autosave mode:

1. On the Command line, type:
Command ==> PROFILE 3
2. Press Enter. The third line of the edit profile shows the autosave mode setting.

To turn on autosave mode:

1. On the Command line, type:
Command ==> AUTOSAVE

Note: This is the equivalent of entering AUTOSAVE ON.

2. Press Enter. The next time you enter END, any changes that you made to the data set or member that you were editing are saved.

To turn off autosave mode:

1. On the Command line, type:
Command ==> AUTOSAVE OFF

Note: This is the equivalent of entering AUTOSAVE OFF PROMPT.

AUTOSAVE

2. Press Enter. The next time you enter END when a data set or member has been changed, the editor prompts you to specify whether you want changes to the data set or member saved (SAVE) or not saved (CANCEL). However, if no changes have been made to the data set or member, the edit session ends without a prompt.

To turn off autosave mode and specify that you do not want to be prompted when data has changed:

1. On the Command line, type:
Command ==> AUTOSAVE OFF NOPROMPT
2. Press Enter. The next time you enter END when a data set or member has been changed, the edit session ends without saving your changes, just as if you had entered CANCEL. You are not prompted to save the changes.

For more information on saving data, see the CANCEL and END primary commands, and the DATA_CHANGED, CANCEL, and END macro commands.

Example

This example shows a practical application of AUTOSAVE usage.

1. You have been editing a data set member and now want to end the edit session. Enter END:
Command ==> END
2. The member that you were editing remains with the following message in the upper-right corner:
DATA CHANGED-SAVE/CANCEL

This message implies that autosave mode in the edit profile is set to AUTOSAVE OFF PROMPT. You are prompted to enter either SAVE to save your changes, or CANCEL to end the edit session without saving your changes.

You also have the option to change autosave mode in the edit profile to AUTOSAVE ON. By doing so, the next time you enter END, your changes will be saved and the edit session will end.

3. You decide to turn on autosave mode:
Command ==> AUTOSAVE ON
4. Then you enter END again to save your changes and end the edit session.
Command ==> END

BOUNDS—Control the Edit Boundaries

The BOUNDS primary command sets the left and right boundaries and saves them in the edit profile.

Syntax

BOUNDS [left-col right-col]

left-col

The left boundary column to be set.

right-col

The right boundary column to be set.

You cannot specify the same column for both boundaries. An asterisk (*) can be used to represent the current value of the boundary.

Description

The BOUNDS primary command provides an alternative to setting the boundaries with the BOUNDS line command or macro command; the effect on the member or data set is the same. However, if you use both the BOUNDS primary command and the BOUNDS line command in the same interaction, the line command overrides the primary command.

To reset the boundaries to the default columns:

1. On the Command line, type:

```
Command ==> BOUNDS
```

2. Press Enter. The boundaries are reset to the default columns.

See “Edit Boundaries” on page 28 for more information, including tables that show commands affected by bounds settings and default bounds settings for various types of data sets.

Examples

To set the left boundary to 1 and the right boundary to 72, type:

```
Command ==> BOUNDS 1 72
```

To set the left boundary to 10 and leave the right as is, type:

```
Command ==> BOUNDS 10 *
```

BUILTIN—Process a Built-In Command

You can use the BUILTIN primary command with edit macros and the DEFINE command to process a built-in edit primary command, even if a macro has been defined with the same name.

Syntax

```
BUILTIN cmdname
```

cmdname The built-in command to be processed.

Description

To process a built-in primary command instead of a command with the same name that has been defined as an alias:

1. On the Command line, type:

```
Command ==> BUILTIN cmdname
```

where *cmdname* is the name of a primary command.

2. Press Enter. The edit primary command is processed.

Example

This example shows a practical application of BUILTIN command usage.

BUILTIN

1. You have a macro named MACEND that you have created. You want to run your MACEND macro instead of the PDF component's built-in END command. Enter the following:

```
Command ===> DEFINE END ALIAS MACEND
```

Note: If the END command is issued in your MACEND macro without being preceded by the BUILTIN macro command, the MACEND macro would be run again, resulting in a loop.

2. Enter the following to run your MACEND macro:

```
Command ===> END
```

3. To end the edit session without redefining END, use BUILTIN, as follows:

```
Command ===> BUILTIN END
```

This command issues the PDF component's built-in END command instead of your MACEND macro.

BROWSE—Browse from within an Edit Session

The BROWSE primary command allows you to browse a sequential data set or partitioned data set member during your current edit session.

Syntax

```
BROWSE [member]
```

member

A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

To browse a data set or member during your current edit session:

1. On the Command line, type:

```
Command ===> BROWSE member
```

Here, *member* represents the name of a member of the partitioned data set you are editing. The member operand is optional.

2. Press Enter. If you specified a member name, the current library concatenation sequence finds the member. The member displays for browsing. If you do not specify a member name, the Browse Command Entry panel, which is similar to the regular Browse Entry panel, appears. You can enter the name of any sequential or partitioned data set to which you have access. When you press Enter, the data set or member displays for browsing. The editor suspends your initial edit session until the browse session is complete.
3. To exit from the browse session, enter the END command. The current session resumes.

Example

To browse member YYY of the current library concatenation:

1. On the command line, type:

```
Command ===> BROWSE YYY
```

2. Press Enter.

CANCEL—Cancel Edit Changes

The CANCEL primary command ends your edit session without saving any of the changes you have made.

Syntax

CANCEL

Description

CANCEL is especially useful if you have changed the wrong data, or if the changes themselves are incorrect. To cancel changes to a data set:

1. On the Command line, type:
Command ==> CANCEL
2. Press Enter. The edit session ends without saving your changes.

Note: If you issue SAVE and later issue CANCEL, the changes you made before issuing SAVE are not canceled.

See the DATA_CHANGED, AUTOSAVE, and END commands for more information about saving data.

CANCEL does not cause automatic recording in the ISPF list data set, regardless of the setting of the autolist mode.

Example

After editing the data, you decide that you want the data set the way it was before editing. Enter the following:

Command ==> CANCEL

The edit session ends with the data set in its original state.

CAPS—Control Automatic Character Conversion

The CAPS primary command sets the caps mode, which controls whether alphabetic data that you type at the terminal is automatically converted to uppercase during the edit session.

Syntax

CAPS [ON]
[OFF]

ON Turns caps mode on.

OFF Turns caps mode off.

Description

The editor sets the caps mode according to the data in the file retrieved for editing. If caps mode has been on and the data contains lowercase letters, the mode

CAPS

switches and the editor displays a message indicating the change. Likewise, if caps mode is off and the editor contains all uppercase letters, the mode switches and the editor displays a message.

Caps mode is saved in the edit profile. To override the automatic setting of caps mode, you can include the CAPS command in an initial macro.

Caps mode is usually on during program development work. When caps mode is on, any alphabetic data that you type, plus any other alphabetic data that already exists on that line, is converted to uppercase when you press Enter or a function key.

To set caps mode on:

1. On the Command line, type:
Command ==> CAPS
2. Press Enter. Caps mode is set to on in the edit profile.

Caps mode is usually off when you edit text documentation. When caps mode is set to off, any alphabetic data that you type remains just as you typed it. If you typed it in uppercase, it stays in uppercase; if you typed it in lowercase, it stays in lowercase. Alphabetic data already typed on a line is not affected. To set caps mode off:

1. On the Command line, type:
Command ==> CAPS OFF
2. Press Enter. Caps mode is set to off in the edit profile.

The CAPS command does not apply to DBCS fields in formatted data or to DBCS fields in mixed fields. If you specify CAPS, the DBCS fields remain unchanged.

See the LC (lowercase) and UC (uppercase) line commands and the CAPS macro command for more information about changing case.

Example

This example shows a practical application of CAPS command usage.

1. You are editing a data set that contains all uppercase letters, with caps mode off. The data you are typing contains both uppercase and lowercase letters, but you want all of the letters to be uppercase. On the Command line, type:
COMMAND ==> CAPS
2. Press Enter.
3. Move the cursor back to the line on which you were typing.
4. Finish typing the line or type over one or more of the existing letters.
5. Press Enter. All of the letters on the line are converted to uppercase.

CHANGE—Change a Data String

The CHANGE primary command changes one search string into another.

Syntax

```
CHANGE string-1 string-2 [range] [NEXT ] [CHARS ] [X ] [col-1 [col-2]]
                               [FIRST] [SUFFIX]
                               [LAST ] [WORD ]
                               [PREV ]
```

string-1

The search string you want to change.

string-2

The string you want to replace *string-1*.

range Two labels that identify the range of lines the CHANGE command is to search.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string-1*. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of *string-1*.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of *string-1*.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of *string-1*.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of *string-1*.

CHARS

Locates *string-1* anywhere the characters match. CHARS is the default.

PREFIX

Locates *string-1* at the beginning of a word.

SUFFIX

Locates *string-1* at the end of a word.

WORD

Locates *string-1* when it is delimited on both sides by blanks or other non-alphanumeric characters.

X Scans only lines that are excluded from the display.

NX Scans only lines that are not excluded from the display.

col-1 and col-2

Numbers that identify the columns the CHANGE command is to search.

Description

You can use the CHANGE command with the FIND and EXCLUDE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To change the next occurrence of ME to YOU without specifying any other qualifications:

1. On the Command line, type:

```
Command ==> CHANGE ME YOU
```

2. Press Enter. This command changes only the next occurrence of the letters ME to YOU. Since no other qualifications were specified, the letters ME can be:

CHANGE

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In an excluded line or a nonexcluded line
- Anywhere within the current boundaries.

To change the next occurrence of ME to YOU, but only if the letters are uppercase:

1. On the Command line, type:

```
Command ==> CHANGE C'ME' YOU
```

2. Press Enter. This type of change is called a character string change (note the C that precedes the search string) because it changes the next occurrence of the letters ME to YOU only if the letters are found in uppercase. However, since no other qualifications were specified, the change occurs no matter where the letters are found, as outlined in the preceding list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

Examples

The following example changes the first plus in the data set to a minus. However, the plus must occur on or between lines labeled .E and .S and it must be the first character of a word:

```
CHANGE '+' '-' .E .S FIRST PREFIX
```

The following example changes the last plus in the data set to a minus. However, the plus must occur on or between lines labeled .E and .S; it must be the last character of a word; and it must be found on an excluded line:

```
CHANGE '+' '-' .E .S LAST SUFFIX X
```

The following example changes the plus that immediately precedes the cursor position to a minus. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the plus must occur on or between the labeled lines; it must be a stand alone character (not part of any other word); it must be on a nonexcluded line; and it must exist within columns 1 and 5:

```
CHANGE '+' '-' .E .S PREV WORD NX 1 5
```

COMPARE—Edit Compare

The COMPARE command compares the file you are editing with an external sequential data set or member of a partitioned data set. Lines that exist only in the file being edited are marked, and lines that exist only in the file being compared are inserted as information lines in the file being edited. The command operates as a primary command or an edit macro command.

You can use the Delete and Make Data line commands to merge changes between files that are being compared.

The COMPARE function supports all line lengths, but some SuperC options are ignored for line lengths greater than 256 characters long.

Data sets being compared must be cataloged. Compare operates by allocating data sets by name and then calling the SuperC function. If you are editing an

uncataloged data set, and a data set with the same name is cataloged, the Compare command uses the cataloged version of the data set. This can cause incorrect results.

Note: COMPARE is not available in edit sessions controlled by the EDIF, EDREC, or EDIREC services.

Command Syntax

```
COMPARE {dsname} {EXCLUDE} [SAVE]{SYSIN} {NEXT}
```

no operand

The *Edit Compare Settings* panel is displayed. This panel enables you to customize the comparison by selecting the relevant SuperC options to use. The comparison is always a LINE compare with the options UPDLDEL, NOLISTL, LINECMP, and CKPACKL specified.

The SEQ, NOSEQ, or COBOL keywords are automatically specified depending on the NUMBER state in the edit profile. Mixed data can be enabled, and is always assumed to be specified when you are in an edit session with MIXED specified in the profile. Each field in the Edit Compare Settings panel has field level help.

Note: When *don't process* (DP) options are used, the resulting display shows DP lines in the current file as unlabeled and does not show DP lines from the comparison file. This can be misleading. Because comparisons which ignore parts of the file might show data in one file and not in the other, use caution when using DP options. When you use options that ignore programming language comments, the *don't process reformatted lines* option is recommended.

dsname

The name of a member or data set to which the current file is compared. This variable can be specified as a fully qualified data set name (in quotation marks), a partially qualified data set name, or a member name.

If you specify only a member name, it must be preceded by a left parenthesis symbol. The right parenthesis is allowed but not required. The current edit session must be of a member of a partitioned data set. The current edit concatenation is searched for the member to compare.

If you specify only a data set name and the current file is a member of a PDS, then the specified data set is searched for a member of the same name as the member being edited.

EXCLUDE

Specifies that all matching lines in the compared data sets are excluded from the display *except* for a specified number of lines above and below the differences. The differences themselves are also shown in the display. The specified number of lines that are shown is set on the Edit Compare Settings panel. If you do not respecify the number for this edit session, then whatever was the last number set is still valid. To change this number, issue the COMPARE command with no operand and change the EXCLUDE field on the Edit Compare Settings panel. Valid numbers are 0 through 12, inclusive.

You can also use the **COMPARE EXCLUDE** command at any time to exclude all lines in a file except lines with line labels and information lines, and the lines above and below those lines. When you specify EXCLUDE

COMPARE

without a data set name or NEXT, no comparison is done. Instead the labels and information lines that already exist in the file are used to exclude functions.

NEXT Specifies to do a comparison between the currently edited member and the next member of the same name found at a higher level of the hierarchy (or next level of the edit concatenation) than the current member. For example, if the current member is found in the third level of the concatenation, and a like-named member exists at the fourth level, then the third and fourth level members are compared. After data is saved in the lowest level, compares are done from that level upward. If you specify *dsname*, the NEXT keyword cannot be used.

SAVE Specifies that SuperC (which performs the actual compare function) create a listing. The listing is saved in a data set named *prefix.ISPFEDIT.COMPARE.LIST*. The save function is intended for debugging purposes, but it also provides a way to create a SuperC listing. The listing produced is a Change listing (option CHNGL). No notification is given regarding successful creation of the listing, and errors allocating the listing do not cause the comparison to end.

Note: Because of the way the SuperC comparison is done, the file currently being edited is shown in the SuperC listing as the *old* file, and the file to which the current file is being compared is listed as the *new* file. Therefore, insertions refer to lines that are *not* in the current file, and deletions refer to lines that are only in the current file.

SYSIN

Specifies not to free the DD name SYSIN before calling SuperC to compare files. This enables you to pass SuperC Process Statements to alter the comparison. No validation is done on the type of SYSIN allocation or the contents of the data set.

Examples

To display the Edit Compare Settings panel
COMPARE

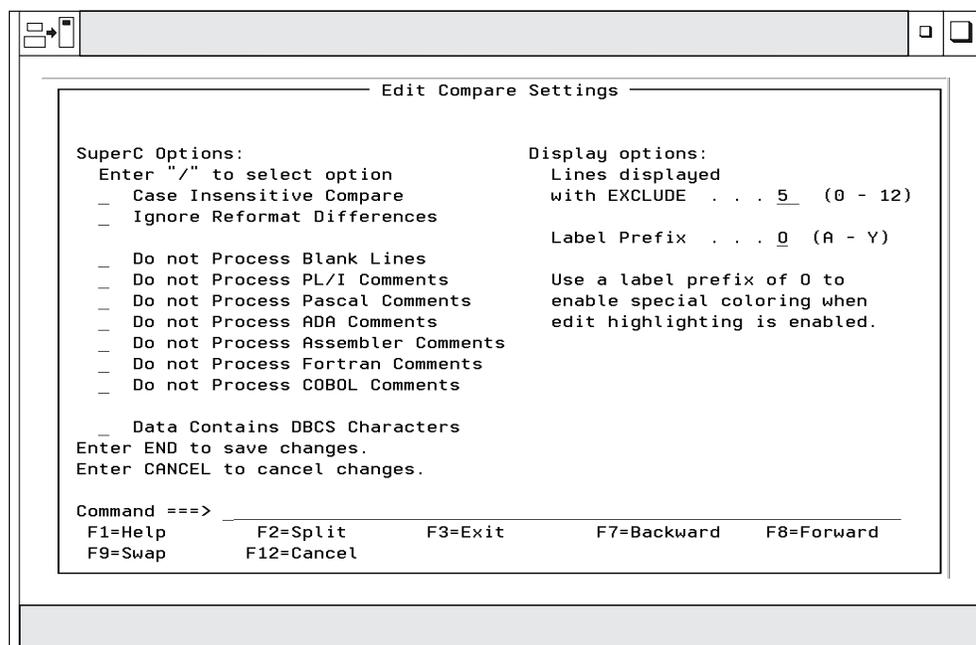


Figure 120. Edit Compare Settings Panel

To compare the data to a member in the current data set or concatenation
 COMPARE (member

COPY—Copy Data

The COPY primary command copies a sequential data set or a member of a partitioned data set into the data being edited.

Syntax

```
COPY [member][AFTER label ][linenum range]
      [(member)][BEFORE label]
      [data set name]
```

member

A member of the ISPF library or partitioned data set that you are editing.

data set name

A partially qualified or fully qualified data set name. If the data set is partitioned you can include a member name in parentheses or select a member from a member list.

AFTER label

The destination for the data being copied. AFTER label copies the data after the specified label.

BEFORE label

The destination for the data that is being copied. BEFORE label copies the data before the specified label.

linenum range

Two numbers that specify the relative line numbers of the member or data set to be copied. To specify standard, ISPF, or Cobol line numbers omit the member name or data set name to use the Extended Edit Copy panel.

COPY

The label can be either a label that you define or one of the PDF editor-defined labels, such as .ZF and .ZL.

If you have not defined a label and the ISPF editor-defined labels are not appropriate for your purpose, use the A (after) or B (before) line command to specify where the data is to be copied.

If the data set or member that you are editing is empty, you do not need to specify a destination for the data being copied.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned a like-named member is copied. If a like-named member does not exist the name is considered to be a partially qualified data set name.

Description

COPY adds a copy of data that already exists to the data set or member that you are editing. Use MOVE if you want to move data from one data set or member to another, rather than just copy it.

To copy data into an *empty* data set or member:

1. On the Command line, type:

```
Command ==> COPY member
```

The member or data set name operand is optional. If you do not specify the name of a member or of a data set to be copied, the Edit Copy panel appears. Enter the data set or member name on this panel.

Also, if you are copying a member of a partitioned data set, you can specify the numbers of the first and last lines to be copied, along with the kind of line numbers (standard, ISPFSTD, COBOL, or relative) on the Edit Copy panel. This allows you to copy only part of the data set or member.

Note: When you select ISPFSTD line numbers and the STATS mode is ON, the editor uses the first 6 digits and ignores the 2 digit modification number. When the STATS mode is OFF, the editor uses all 8 digits.

2. Press Enter. The data is copied.

To copy data into a data set or member that is *not empty*:

1. On the Command line, type:

```
Command ==> COPY member AFTER | BEFORE label  linenum range  
                COPY data set name
```

The member or data set name operand is optional. You should omit the member name only if you do not know the member name, or if you are going to copy a sequential data set or a member of a different partitioned data set.

The AFTER label and BEFORE label operands are also optional. However, if the data set or member that is to receive the copied data is not empty, you must specify a destination for the copied data. Therefore, if you do not want to use a label, you can substitute either the A (after) or B (before) line command as the destination of the copied data. However, a number indicating that the A or B command should be repeated cannot follow the line command. See the descriptions of these commands for information about them.

If the data set or member is not empty and you do not specify a destination, a MOVE/COPY Pending message appears in the upper-right corner of the panel and the data is not copied. When you type a destination and press Enter, the data is copied.

2. Press Enter. If you entered a member name or data set name, the member or data set is copied. Otherwise, the Edit Copy panel appears. If a range of line numbers is specified, only those lines are copied. See the previous example for more information.

See “Copying and Moving Data” on page 50 if you need more information.

Example

The following steps show how you can copy data when you omit the member name and the ISPF editor panels appear.

1. Type COPY on the Command line and specify the destination of the operation. The panel in Figure 121 shows you that the data is to be copied after line 000700, as specified by the A (after) line command.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(COPYINTO) - 01.00 ----- Columns 00001 00072
***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 This is the member into which the lines are to be copied.
000500
000600
A00700
000800
000900
001000
001100
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** Bottom of Data *****

Command ==> copy
F1=Help    F2=Split   F3=Exit    F5=Rfind   F6=Rchange F7=Up
F8=Down    F9=Swap    F10=Left   F11=Right  F12=Cancel

Scroll ==> CSR

```

Figure 121. Member Before Data is Copied

2. When you press Enter, the Edit Copy panel appears. Specify the data you want copied.

The example in Figure 122 copies the data set member named COPYFROM. Since you are using the Edit Copy panel, you can also specify the number of lines you want copied.


```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(COPYINTO) - 01.01 ---- Requested line(s) copied
***** ***** Top of Data *****
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 This is the member into which the lines are to be copied.
000500
000600      +-----+
000700      |           |
000710 These are the lines that are to be copied.
000720 These are the lines that are to be copied.
000800      |           |
000900      +-----+
001000
001100
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
***** ***** Bottom of Data *****
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange    F7=Up
F8=Down      F9=Swap       F10=Left     F11=Right     F12=Cancel

```

Figure 124. Member After Data Has Been Copied

CREATE—Create Data

The CREATE primary command creates a member of a partitioned data set, or a sequential data set, from the data you are editing.

Syntax

```

CREATE [member] [range]
      (member) [range]
      [data_set(member)] [range]
      [data_set] [range]

```

member

The name of the new member added to the partitioned data set currently being edited. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

range Two labels that specify the group of lines, from beginning to end, which are added to the new member.

data_set(member)

The name of a different partitioned data set and new member name to be added to the partitioned data set. The data set name can be fully qualified or partially qualified.

data_set

The name of a different sequential data set to be added. The data set name can be fully qualified or partially qualified.

Description

CREATE adds a new member to a partitioned data set only if a member of the same name does not already exist. Use REPLACE if the member already exists.

To create a member of a partitioned data set or a sequential data set:

CREATE

1. On the Command line, type:

```
Command ==> CREATE member range
Command ==> CREATE (member) range
Command ==> CREATE data_set(member) range
Command ==> CREATE data_set range
```

The member operand is optional unless you specify a data set name. It represents the name of the member you want to create.

The range operand is also optional. It represents a pair of labels that specify the first and last lines in a group of lines used to create the new member or sequential data set.

If you omit the range operand, you must specify the lines by using either the C (copy) or M (move) line command. See the descriptions of these commands if you need more information about them.

If you omit the range operand and do not enter one of the preceding line commands, a CREATE Pending message is displayed in the upper-right corner of the panel.

2. Press Enter. If you did not specify the name of the member or the name of another partitioned data set along with the member name to be created, the Edit Create panel appears. Enter the member name on this panel and press Enter again. If you used either a pair of labels or a C line command, the data is copied from the member that you are editing into the member that you are creating. If you used the M line command, however, the data is removed from the member that you are editing and placed in the member that you are creating.

If the data set specified does not exist, ISPF prompts you to see if the data set should be created. You can create the data set using the characteristics of the source data set as a model, or specify the characteristics for the new data set. You can suppress this function through the ISPF configuration table, causing any CREATE request for a non-existent data set to fail.

Refer to “Creating and Replacing Data” on page 49 if you need more information about the CREATE command.

Example

The following steps show how you can create a new member when you omit the member name.

1. Type CREATE on the Command line and specify which lines you want to copy or move into the new data set or member. The example in Figure 125 uses the MM (block move) line command to move a block of lines from the data.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(CREATE) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 This line will be left in the member CREATE.
000200 This line will be left in the member CREATE.
MM0300 +-----+
000400 | This is the
000500 | material to
000600 | be created in
000700 | another member
MM0800 +-----+
000900 This line will be left in the member CREATE.
001000 This line will be left in the member CREATE.
***** ***** Bottom of Data *****

Command ==> create
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 125. Member Before New Member Is Created

- When you press Enter, the Edit Create panel (Figure 126) appears. Type the name of a new member and press Enter. If you type the name of a member that already exists, an error message appears and the CREATE fails. The name of the member created for this example is NEWMEM.

```

Menu RefList Utilities Help
-----
Edit - Create
More: +
"Current" Data Set: USERID.PRIVATE.CLIST(SCREEN)

To ISPF Library:
Project . . . USERID
Group . . . PRIVATE
Type . . . CLIST
Member . . . NEWMEM__

To Other Partitioned Data Set Member:
Data Set Name . . _____
Volume Serial . . _____ (If not cataloged)

Data Set Password . . (If password protected)

Enter "/" to select option
_ Specify pack option for "CREATE" Data Set

Command ==>
F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F10=Actions F12=Cancel

```

Figure 126. Edit Create Panel (ISRECRA1)

- Figure 127 shows the lines remaining in the original member after the specified lines were moved to the new member.

CREATE

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(CREATE) - 01.00 ----- Member NEWMEM created
***** ***** Top of Data *****
000100 This line will be left in the member CREATE.
000200 This line will be left in the member CREATE.
000900 This line will be left in the member CREATE.
001000 This line will be left in the member CREATE.
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 127. Member After New Member Has Been Created

- Figure 128 shows the contents of the new member. Notice that the data is renumbered if both number mode and autonum mode are on. A source listing of the data is also recorded in the ISPF list data set for eventual printing if autolist mode is on.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(NEWMEM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100
000200 | This is the
000300 | material to
000400 | be created in
000500 | another member
000600 |
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 128. New Member Created

CUT—Cut and Save Lines

The CUT primary command saves lines to one of eleven named clipboards for later retrieval by the PASTE command. The lines can be appended to lines already saved by a previous CUT command.

Syntax

```
CUT [lptr-range] [DEFAULT | clipboardname]
    [REPLACE] [DISPLAY]
```

lptr-range	Two line pointers that specify the range of lines in the current member that are to be added to or replace data in the clipboard. A line pointer can be a label or relative line number. You must specify both a starting and ending line pointer.
clipboardname	The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.
REPLACE	Specify REPLACE to replace existing data in the clipboard. If you do not specify REPLACE, the lines in the current CUT are added to the end of the existing data within the clipboard.
DISPLAY	Show a list of existing clipboards. From this list you can browse, edit, clear, or rename the clipboards.

Description

CUT saves copies of lines from an edit session to a clipboard for later retrieval by the PASTE command. The lines are moved or copied from the session to the named clipboard. Lines are specified by either the C (Copy) or M (Move) line commands, CC or MM block line commands, or label names. If the C or CC line commands or labels are used to identify the lines, the lines are *copied* to the clipboard. If the M or MM line commands are used to identify the lines, the lines are copied to the clipboard and deleted from the edit session (in effect, *moving* them).

If you specify a clipboard name, lines are copied to that clipboard. If the specified clipboard does not yet exist, it is created. ISPF provides a default clipboard named DEFAULT. You can use up to 10 other clipboards that you define. The defined clipboards exist as long as you are logged on to TSO and are deleted when you log off.

You can view the contents of clipboards and rename existing clipboards using the DISPLAY keyword of the CUT command. If you specify the DISPLAY, other keywords are ignored.

Example

To save all the lines in the current file to the default clipboard, appending them to lines already in the clipboard:

```
CUT .ZFIRST .ZLAST
```

To save all the lines in the current file to a clipboard named USERC1, replacing any lines already in the clipboard:

```
CUT .ZFIRST .ZLAST USERC1 REPLACE
```

DEFINE—Define a Name

The DEFINE primary command is used to:

- Identify a macro that replaces a built-in command of the same name
- Identify programs that are edit macros
- Assign an alias to a macro or built-in command
- Make a macro or built-in command inoperable
- Reset an inoperable macro or built-in command
- Disable a macro or built-in command.

DEFINE is often used with the BUILTIN command.

Syntax

```
DEFINE name {MACRO CMD }
           {MACRO PGM }
           {ALIAS name-2}
           {NOP }
           {RESET }
           {DISABLED }
```

name The name for the command.

MACRO CMD

Identifies the name you are defining as a command language (CLIST or REXX EXEC) macro, which is called in the same way as using the SELECT service CMD keyword with a percent symbol (%) preceding the command. That means that you can specify only CLISTs or REXX EXECs. This operand is the default.

MACRO PGM

Identifies the name that you are defining as a program (load module) macro.

ALIAS name-2

Identifies the name you are defining as an alias of another name, with the same characteristics. If name-2 is already an alias, the editor replaces it with the command for which it is an alias. Therefore, it is not possible to have an alias of an alias.

NOP Makes the name that you are defining and all of its aliases inoperable until you reset them with RESET. Therefore, when the name or an alias of the name is called, nothing is processed. NOP is similar to DISABLED, except that disabled names cannot be reset by the RESET operand.

RESET

Resets the most recent definition of the name that you are defining to the status in effect before that definition. For example, RESET makes inoperable names operable again.

DISABLED

Disables the name you are defining and all of its aliases until you completely exit the editor and return to the ISPF Primary Option Menu. Therefore, when the name or an alias of the name is entered, nothing is processed. A disabled command or macro cannot be restored by the RESET operand. To disable RESET, use delimiters around 'RESET' to distinguish it from the keyword.

Description

The effects of a DEFINE command remain until you either issue DEFINE RESET or exit from the editor. You enter the editor when you select option 2, and you do not exit the editor until you return to the ISPF Primary Option Menu. Therefore, if you edit several members of a partitioned data set, one DEFINE at the beginning affects them all.

To temporarily override DEFINE, BUILTIN.

Stacking DEFINE Commands

Except for the DISABLED operand, the DEFINE operations are stacked. The RESET operand unstacks them. For example:

```
DEFINE A alias FIND
DEFINE A alias COPY
DEFINE A alias SAVE
```

stacks three definitions of A. Only the last one is effective. Here, A would be defined as SAVE.

The following operation:

```
DEFINE A RESET
```

removes one command from the stack, making the previous command effective. In the preceding example, A would now be defined as COPY.

Examples

To define the name IJKDOIT as a CLIST or REXX macro, enter:

```
Command ==> DEFINE IJKDOIT MACRO
```

To define the name SETITUP as a program macro, enter:

```
Command ==> DEFINE SETITUP MACRO PGM
```

To define the name DOIT as an alias of the macro IJKDOIT, enter:

```
Command ==> DEFINE DOIT ALIAS IJKDOIT
```

To define the name SAVE to have no effect, enter:

```
Command ==> DEFINE SAVE NOP
```

To reset the definition of the name SAVE, enter:

```
Command ==> DEFINE SAVE RESET
```

To define the name FINDIT as disabled, enter:

```
Command ==> DEFINE FINDIT DISABLED
```

DELETE—Delete Lines

The DELETE primary command deletes lines from the data you are editing.

Syntax

```
DELETE {ALL X | NX }
       {range X | NX}
       {ALL range }
```

ALL Specifies that all selected lines are deleted. The DELETE command, unlike FIND, CHANGE, and EXCLUDE, does not accept NEXT, FIRST, PREV, or LAST. ALL is required to emphasize that NEXT is not the default.

X | NX

Restricts the lines deleted to those that are excluded or not excluded, respectively.

range Two labels that limit the lines deleted to a range within and including those labels. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

Description

There is no DELETE ALL command, as a precaution against error. To delete all lines, do one of the following:

- To delete all lines by using the editor-defined labels:
Command ==> DELETE ALL .ZFIRST .ZLAST
- To delete all lines by first resetting any excluded lines to make them not excluded, and then deleting all lines that are not excluded:
Command ==> RESET; DELETE ALL NX

Here are other uses of the DELETE command:

- To delete all excluded lines:
Command ==> DELETE ALL X
- To delete all not excluded lines:
Command ==> DELETE ALL NX
- To delete all excluded lines within a range:
Command ==> DELETE .label1 .label2 X

Here, and in the commands that follow, *label1* and *label2* represent the two labels that show the range of lines to be deleted.

- To delete all not excluded lines within a range:
Command ==> DELETE .label1 .label2 NX
- To delete all lines within a range:
Command ==> DELETE .label1 .label2

Examples

You can more easily determine which lines to delete in a large data set by excluding lines that meet some criterion, or by leaving all lines that meet the criterion nonexcluded. Then, with DELETE you can delete many lines. For example, to delete all blank lines in a data set, type the following commands on the Command line and press Enter after each one:

1. First, reset all excluded lines:
RESET X
2. Then, exclude lines containing characters that are not blanks:
EXCLUDE ALL P'~'
3. Finally, delete the nonexcluded lines, which contain only blanks:
DEL ALL NX

Another way to do the same thing is this:

1. First, exclude all lines:
EXCLUDE ALL
2. Then, find all lines containing a character that is not a blank:
FIND ALL P'~'
3. Finally, delete the remaining excluded lines, which contain only blanks:
DEL ALL X

EDIT—Edit from within an Edit Session

The EDIT primary command allows you to edit another sequential data set or partitioned data set member during your current edit session.

Syntax

EDIT [member]

member

A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Editing one data set or member while you are already editing another is called *recursive editing*. To edit another data set or member during your current edit session:

1. On the Command line, type:

Command ==> EDIT member

Here, member represents the name of a member of the partitioned data set you are editing. The member operand is optional.

2. Press Enter.

If you specified a member name, the current library concatenation sequence finds the member. The member is displayed for editing.

If you do not specify a member name, the Edit Command Entry panel, which is identical to the regular Edit Entry panel, appears. You can enter the name of any sequential or partitioned data set to which you have access. When you press Enter, the data set or member is displayed for editing.

The editor suspends your initial edit session until the second-level edit session is complete. Editing sessions can be nested until you run out of storage.

3. To exit from a nested edit session, enter an END or CANCEL command. The current edit session resumes.

Example

The following steps show the use of the EDIT primary command:

1. Assume that you are editing a member named PGM8 and you need to edit a member in another data set. So, you enter the EDIT command on the Command line, omitting the member operand, as shown in Figure 129.

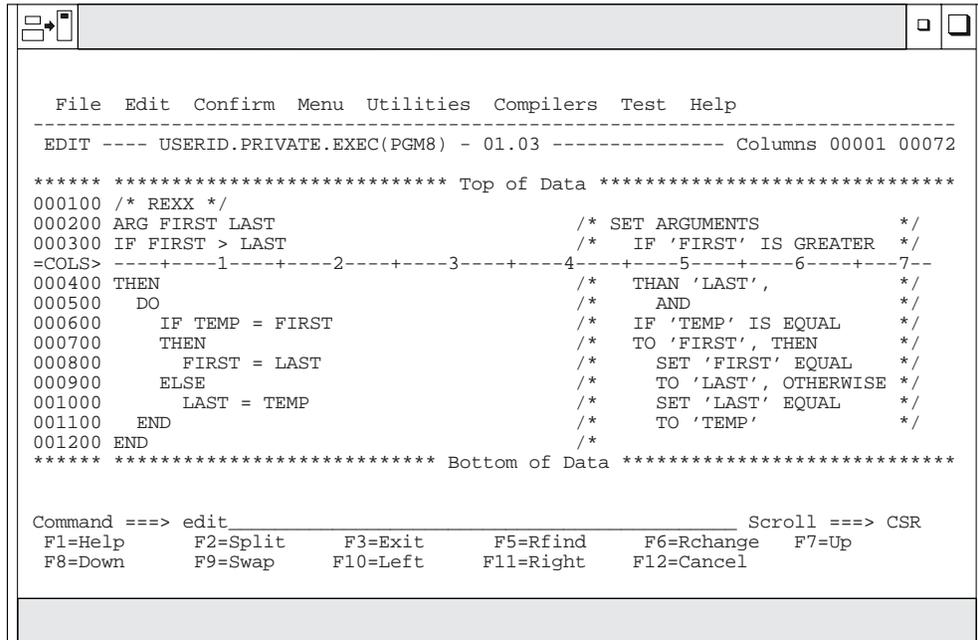


Figure 129. EDIT Primary Command Example

2. When you press Enter, the Edit Command Entry panel (Figure 130) appears. On this panel, you enter the name of the partitioned data set and member that you want to edit:

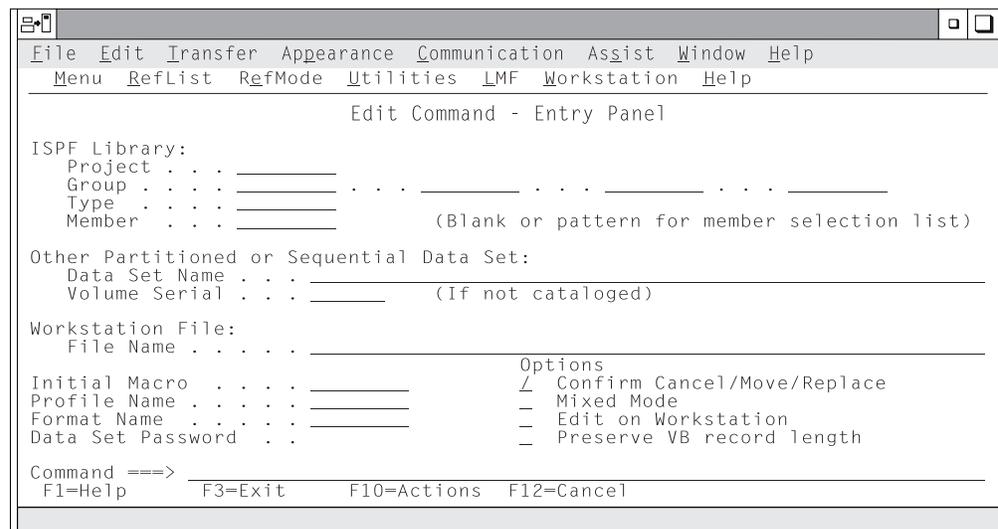


Figure 130. Edit Command Entry Panel (ISREDM03)

3. When you press Enter again, the member is displayed for editing, as shown in Figure 131:

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(FORMAT) - 1.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG First Last /*
000300 IF First > Last /*
000400 THEN /*
000500 DO /*
000600 IF First > 0 /*
000700 THEN /*
000800 Scale = First /*
000900 ELSE /*
001000 Scale = 0 /*
001100 Temp = First /*
001200 First = Last /*
001300 Last = Temp /*
001400 END /*
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 131. Nested Member Editing Example

END—End the Edit Session

The END primary command ends the editing of the current sequential data set or partitioned data set member.

Syntax

```
END
```

Description

To end an edit session by using END, do one of the following:

- Enter END on the Command line, or
- Press a function key to which END is assigned. The default setting is F3.

If no aliases have been defined for END, the editor's response to END depends on:

- Whether changes were made to the data during your current edit session
- If changes were made, whether SAVE was entered after the last change
- The setting of number mode, autonum mode, stats mode, autolist mode, and autosave mode in the edit profile
- Whether you were editing a member that was an alias of another member.

For additional explanation, see "Ending an Edit Session" on page 15.

END

Example

To end the current edit session:

1. On the Command line, type:
Command ==> END
2. Press Enter.

EXCLUDE—Exclude Lines from the Display

The EXCLUDE primary command hides lines that contain a search string from view and replaces them with a dashed line. To see the lines again, you enter either the FLIP, RESET or RESET EXCLUDED command.

Syntax

```
EXCLUDE string [range] [NEXT ] [CHARS ] [col-1 [col-2]]  
                        [ALL ] [PREFIX]  
                        [FIRST] [SUFFIX]  
                        [LAST ] [WORD ]  
                        [PREV ]
```

string The search string you want to exclude.

range Two labels that identify the lines which the EXCLUDE command is to search.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS

Locates string anywhere the characters match. CHARS is the default.

PREFIX

Locates string at the beginning of a word.

SUFFIX

Locates string at the end of a word.

WORD

String is delimited on both sides by blanks or other non-alphanumeric characters.

col-1 and col-2

Numbers that identify the columns the EXCLUDE command is to search.

Description

You can use the EXCLUDE command with the FIND and CHANGE commands to find a search string, change it, and exclude the line that contains the string from the panel.

To exclude the next nonexcluded line that contains the letters ELSE without specifying any other qualifications:

1. On the Command line, type:

```
Command ==> EXCLUDE ELSE
```
2. Press Enter. Since no other qualifications were specified, the letters ELSE can be:
 - Uppercase or a mixture of uppercase and lowercase
 - At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
 - Anywhere within the current boundaries.

To exclude the next line that contains the letters ELSE, but only if the letters are uppercase:

1. On the Command line, type:

```
Command ==> EXCLUDE C'ELSE'
```
2. Press Enter. This type of exclusion is called a character string exclusion (note the C that precedes the search string) because it excludes the next line that contains the letters ELSE only if the letters are found in uppercase. However, since no other qualifications were specified, the exclusion occurs no matter where the letters are found on a nonexcluded line, as outlined in the previous list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

Examples

The following example excludes the first nonexcluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
Command ==> EXCLUDE ELSE .E .S FIRST PREFIX
```

The following example excludes the last nonexcluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the last four letters of a word.

```
Command ==> EXCLUDE ELSE .E .S LAST SUFFIX
```

The following example excludes the first nonexcluded line that immediately precedes the cursor position and that contains the letters ELSE. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be stand alone characters (not part of any other word); and they must exist within columns 1 and 5:

```
Command ==> EXCLUDE ELSE .E .S PREV WORD 1 5
```

FIND—Find a Data String

The FIND primary command locates one or more occurrences of a search string.

FIND

Syntax

```
FIND string [range] [NEXT ] [CHARS ] [X ] [col-1[col-2]]  
                [ALL ] [PREFIX] [NX]  
                [FIRST] [SUFFIX]  
                [LAST ] [WORD ]  
                [PREV ]
```

string The search string you want to find.

range Two labels that identify the lines which FIND is to search.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS

Locates string anywhere the characters match. CHARS is the default.

PREFIX

Locates string at the beginning of a word.

SUFFIX

Locates string at the end of a word.

WORD

String is delimited on both sides by blanks or other non-alphanumeric characters.

X Scans only lines that are excluded from the display.

NX Scans only lines that are not excluded from the display.

col-1 and col-2

Numbers that identify the columns the FIND command is to search.

Description

You can use the FIND command with the EXCLUDE and CHANGE commands to find a search string, change it, and exclude the line that contains the string from the panel.

To find the next occurrence of the letters ELSE without specifying any other qualifications:

1. On the Command line, type:

```
Command ==> FIND ELSE
```

2. Press Enter. Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In either an excluded or a nonexcluded line

- Anywhere within the current boundaries.

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

1. On the Command line, type:

```
Command ==> FIND C'ELSE'
```

2. Press Enter. This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

Examples

The following example finds the first occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
Command ==> FIND ELSE .E .S FIRST PREFIX
```

The following example finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
Command ==> FIND ELSE .E .S LAST SUFFIX X
```

The following example finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines labeled .E and .S. The letters must occur on or between lines labeled .E and .S; they must be stand alone characters (not part of any other word); they must be found in a nonexcluded line; and they must exist within columns 1 and 5:

```
Command ==> FIND ELSE .E .S PREV WORD NX 1 5
```

FLIP—Reverse Exclude Status of Lines

The FLIP primary command reverses the exclude status of a specified group of lines or of all the lines in a file, including data, information, message, and note lines.

Syntax

```
FLIP [label-range]
```

Description

The FLIP primary command reverses the exclude status of a range of lines you specify with labels. It can also reverse the exclude status of all the lines in a file. For example, if you have used the 'X ALL;FIND ALL xyz' command to find lines containing a string (xyz), you can use FLIP to see the lines which do not contain the string.

The range is optional. If no range is specified, the exclude status is reversed for all of the lines in the file.

FLIP

To reverse the exclude status of all the lines in a file:

1. Enter the following on the Command line:

```
Command ==> flip
```

2. Press Enter.

All the excluded lines in the file are displayed, and all the previously displayed lines are excluded.

To reverse the exclude status of a range of lines:

1. Enter the following on the Command line:

```
Command ==> flip .a .b
```

Actual values are substituted for .a and .b and can be defined by an edit macro or by the user.

2. Press Enter.

All the lines with the specified range that were previously excluded are displayed, and all the lines within the specified range that were displayed are excluded.

Example

In the example shown in Figure 132, the edit session contains 10 lines:

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USER1.CLIST(FLIPEXAM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 FLIP example line number 000001
000002 FLIP example line number 000002
000003 FLIP example line number 000003
000004 FLIP example line number 000004
000005 FLIP example line number 000005
000006 FLIP example line number 000006
000007 FLIP example line number 000007
000008 FLIP example line number 000008
000009 FLIP example line number 000009
000010 FLIP example line number 000010
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F4=Left     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 132. Example of Data Set

After excluding lines 4 through 7, the data set looks like Figure 133:

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USER1.CLIST(FLIPEXAM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 FLIP example line number 000001
000002 FLIP example line number 000002
000003 FLIP example line number 000003
----- 4 LINE(S) NOT DISPLAYED -----
000008 FLIP example line number 000008
000009 FLIP example line number 000009
000010 FLIP example line number 000010

***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left    F11=Right   F12=Cancel

```

Figure 133. Example of Data Set with Excluded Lines

After executing FLIP, all previously excluded lines are shown. All previously visible lines are excluded, as shown in Figure 134.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USER1.CLIST(FLIPEXAM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
----- 3 LINE(S) NOT DISPLAYED -----
000004 FLIP example line number 000004
000005 FLIP example line number 000005
000006 FLIP example line number 000006
000007 FLIP example line number 000007
----- 3 LINE(S) NOT DISPLAYED -----

***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down     F9=Swap     F10=Left    F11=Right   F12=Cancel

```

Figure 134. Example of Data Set using FLIP on Excluded Lines

HEX—Display Hexadecimal Characters

The HEX primary command sets hexadecimal mode, which determines whether data is displayed in hexadecimal format.

Syntax

```
HEX [ON VERT]  
    [ON DATA]  
    [OFF   ]
```

ON VERT

Displays the hexadecimal representation of the data vertically (two rows per byte) under each character.

ON DATA

Displays the hexadecimal representation of the data as a string of hexadecimal characters (two per byte) under the characters.

OFF Does not display hexadecimal representation of the data.

Description

The HEX command determines whether the editor displays hexadecimal representation in a vertical or data string format. See Figure 136 on page 249 and Figure 137 on page 250 for examples of these two formats.

When the editor is operating in hexadecimal mode, three lines are displayed for each source line. The first line shows the data in standard character form, while the next two lines show the same data in hexadecimal representation.

Besides normal editing on the first of the three lines, you can change any characters by typing over the hexadecimal representations.

You can also use the FIND, CHANGE, and EXCLUDE commands to find, change, or exclude invalid characters or any specific hexadecimal character, regardless of the setting of hexadecimal mode. See the discussion of picture strings and hexadecimal strings under “Finding, Seeking, Changing, and Excluding Data” on page 53.

Examples

Suppose you are editing the data set member shown in Figure 135:

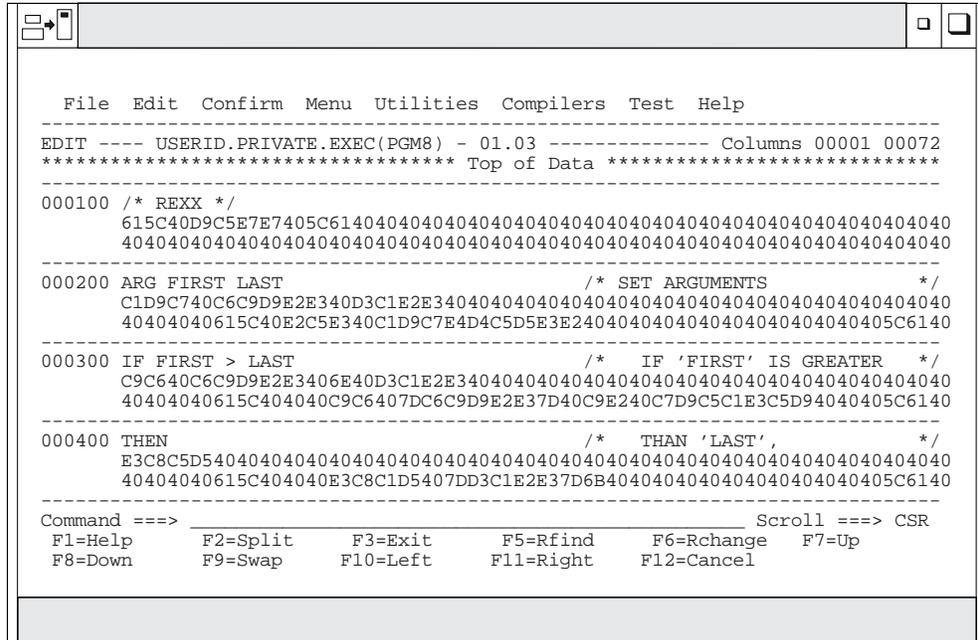


Figure 137. Hexadecimal Display, Data Representation

HILITE—Enhanced Edit Coloring

HILITE is used to control the use of color in the editor by changing the settings for the enhanced color and language-sensitive editing features.

Note: Language-sensitive and enhanced coloring of the edit session is only available when enabled by the installer or the person who maintains the ISPF product. For information on enabling the enhanced color function, see *ISPF Planning and Customizing*

HILITE with *no* operands presents a dialog (see “The HILITE Dialog” on page 40) that allows you to change coloring options, and to see which keywords are supported for each language.

Syntax

```
HILITE [ON      ] [AUTO  ] [RESET] [PAREN] [FIND] [CURSOR] [SEARCH] [DISABLED]
        [OFF    ] [DEFAULT]
        [LOGIC  ] [OTHER  ]
        [IFLOGIC] [ASM    ]
        [DOLOGIC] [BOOK   ]
        [NOLOGIC] [C      ]
                [COBOL  ]
                [DTL    ]
                [JCL    ]
                [PANEL  ]
                [PASCAL ]
                [PLI    ]
                [REXX   ]
                [SKEL   ]
                [IDL    ]
```

ON Sets program coloring ON and turns LOGIC coloring off.

OFF Sets coloring OFF, with the exception of cursor, find, and parenthesis highlighting.

LOGIC

LOGIC highlighting matches logical language-specific keywords in the same color. If an unmatched *closing* keyword is found, such as END for PL/I or :eol. for BookMaster, it is highlighted in reverse video pink *only* if HILITE LOGIC is active. When logic is being highlighted, only comments are highlighted along with it.

Logic highlighting is available for PL/I, PL/X, Rexx, OTHER, C, SKELS, Pascal and BookMaster only. HILITE LOGIC turns on both IFLOGIC and DOLOGIC.

Note: LOGIC highlighting can be turned off by issuing HILITE ON, HILITE NOLOGIC, or HILITE RESET commands. Changing the HILITE language does not change the LOGIC setting.

IFLOGIC

Turns on IF/ELSE logic matching. IFLOGIC matches IF and ELSE statements. When IFLOGIC is enabled, unmatched ELSE keywords are highlighted in reverse video pink.

DOLOGIC

Turns on DO/END logic matching. DOLOGIC matches logical blocks such as DO/END in PL/I or :ol/:eol in BookMaster. For the C language, DOLOGIC matches curly braces ({ and }). C trigraphs for curly braces are not recognized and are not supported by DOLOGIC highlighting. When DOLOGIC is enabled, unmatched logical block terminators, (such as END keywords in PL/I, :e tags in BookMaster or right braces (}) in C) are highlighted in reverse video pink.

NOLOGIC

Same as ON.

AUTO

Allows the PDF component to determine the language.

DEFAULT

Highlights the data in a single color.

OTHER

Highlight the data as a pseudo-PL/I language. Limited CLIST support is also provided by OTHER.

ASM Highlights the data as Assembler.

BOOK

Highlights the data as BookMaster.

C Highlights the data as C.

COBOL

Highlights the data as COBOL

DTL Highlights the data as Dialog Tag Language.

JCL Highlights the data as MVS Job Control Language.

PANEL

Highlights the data as ISPF Panel Language.

PASCAL

Highlights the data as Pascal.

HILITE

PLI Highlights the data as PL/I.

REXX Highlights the data as REXX.

SKEL Highlights the data as ISPF Skeleton Language.

IDL Highlights the data as IDL.

RESET

Resets defaults (AUTO, ON, Find and Cursor on).

PAREN

Toggles parenthesis matching. When parenthesis matching is active, only comments are specially colored. All other code appears in the default color. Note that extra parenthesis highlighting is always active when highlighting is active.

FIND The HILITE FIND command toggles the highlighting color of any string that would be found by an RFIND. The user can select the highlight color. The default is reverse video white.

Only non-picture strings are supported, and the only additional qualifiers recognized are hex strings (X'...'), character strings (C'...'), text strings (T'...'), WORD, PREFIX and SUFFIX, and boundaries specified in the FIND command. Hex strings may be highlighted, but non-displayable characters are not highlighted. Labels are ignored when FIND strings are highlighted.

Because FIND highlighting is not quite as robust as the FIND command itself, the editor may highlight more occurrences of the FIND string than FIND would actually locate. The FIND operand toggles the display of search strings. If HILITE FIND is issued when FIND highlighting is in effect, FIND highlighting is disabled. Similarly, if FIND highlighting is disabled, the HILITE FIND command enables it.

Note:

RESET has been enhanced, through the addition of a FIND operand, to temporarily disable the highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command is issued. RESET with the FIND operand (or no operands at all), temporarily disables the highlighting of FIND strings.

CURSOR

The CURSOR operand toggles the highlighting of the phrase that contains the cursor in a user selectable color. The default is white.

Cursor highlighting in Edit is performed in a manner similar to the way it is done in Browse. The entire phrase from the previous blank to the next blank is highlighted. The CURSOR operand toggles cursor highlighting. If HILITE CURSOR is issued when CURSOR highlighting is in effect, CURSOR highlighting is disabled. Similarly, if CURSOR highlighting is disabled, the HILITE CURSOR command enables it.

SEARCH

HILITE SEARCH finds the first unmatched END, ELSE, }, or) above the last displayed line on the screen. If a mismatched item is found, the file is scrolled so that the mismatch is at the top of the screen. The search for mismatches only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file before issuing the HI SEARCH command.

Search is not available when the DEFAULT language operand is used. Search for language keywords is only supported for languages which supported by the logic option.

DISABLED

Turns off all HILITE features and removes all action bars. This benefits performance at the expense of function. Since DISABLED status is not stored in the edit profile, you need to reenter this operand each time you enter the editor. When DISABLED is in effect, keylists are unavailable for that edit session.

Description

The HILITE primary command can be used to highlight, in user-specified colors, numerous language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. In addition, when HILITE is entered with no operands, a dialog appears that allows you to set default colors for the data area in non-program files, for any characters typed since the previous Enter or PF key entry, and for strings located by FIND.

Both HI and HILIGHT are valid synonyms for HILITE.

Note: Highlighting is *not* available for edit sessions that involve the following:

- Data sets with record lengths greater than 255
- Mixed mode edit sessions (normally used when editing DBCS data)
- Formatted data.

IMACRO—Specify an Initial Macro

The IMACRO primary command saves the name of an initial macro in the current edit profile.

See “Initial Macros” on page 29 for more information on creating and using initial macros.

Syntax

IMACRO {name | NONE}

name The name of the initial macro to be run when you are editing the data set type that matches the current edit profile. This macro is run before any data appears.

For more information about displaying and defining a profile, see “Displaying or Defining an Edit Profile” on page 21.

NONE

Indicates that no macro is to be run at the beginning of each edit session. The edit profile shows a value of NONE is shown in the edit profile when no initial macro has been specified.

Examples

To save STARTUP as the initial macro, type:

```
IMACRO STARTUP
```

IMACRO

To reset the profile with no initial macro, type:
IMACRO NONE

LEVEL—Specify the Modification Level Number

The LEVEL primary command allows you to control the modification level that is assigned to a member of an ISPF library.

See “Version and Modification Level Numbers” on page 31 for more information about level numbers.

Syntax

LEVEL num

num The modification level. It can be any number from 0 to 99.

Description

To specify the modification level number:

1. On the Command line, type:

```
COMMAND ==> LEVEL num
```

where num is the new level number.

2. Press Enter.

Example

In Figure 138, the version and modification level numbers on line 1 show that this is Version 1, Modification 3 (01.03). Type LEVEL 0 on the Command line to reset the modification level number to 00.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> level 0 Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 138. Member With Modification Level of 03

After you press Enter, the editor resets the modification level, as shown in Figure 139.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 139. Member With Modification Level Reset to 00

LOCATE—Locate a Line

The LOCATE primary command allows you to scroll up or down to a specified line. The line then appears as the first line on the panel. There are two forms of LOCATE: specific and generic.

LOCATE

Specific Locate Syntax

The specific form of the LOCATE command positions a particular line at the top of the panel. You must specify either a line number or a label.

```
LOCATE {label | line-number}
```

label	A previously assigned label. An error message appears if the label is not currently assigned.
line-number	An edit line number. If that line number exists, it appears at the top. If the line number does not exist, the line with the next lower number appears at the top of the data area. The line-number operand is a numeric value of up to 8 digits. You do not need to type leading zeros. If the operand contains 6 or fewer digits, it refers to the number in the line command field to the left of each line. If the line-number operand contains 7 or 8 digits, it refers to the sequence numbers in the data records. For NUMBER ON STD, the editor refers to the <i>modification flag</i> . For NUMBER OFF, it refers to the <i>ordinal line number</i> (first=1, fifth=5, and so on). For NUMBER ON COBOL, it refers to the number in the line command field, which is the data sequence number. See "Sequence Number Format and Modification Level" on page 32 for more information.

Generic Locate Syntax

The generic LOCATE command positions the panel to the first, last, next, or previous occurrence of a particular kind of line.

```
LOCATE [FIRST] {CHANGE } [range]
        [LAST ] {COMMAND }
        [NEXT ] {ERROR   }
        [PREV ] {EXCLUDED}
                {LABEL   }
                {SPECIAL }
```

FIRST Searches from the first line, proceeding forward.

LAST Searches from the last line, proceeding backward.

NEXT Searches from the first line of the page displayed, proceeding forward.

PREV Searches from the first line of the page displayed, proceeding backward.

CHANGE

Searches for a line with a change flag (==CHG>).

COMMAND

Searches for a line with a pending line command.

ERROR

Searches for a line with an error flag (==ERR>).

EXCLUDED

Searches for an excluded line.

LABEL

Searches for a line with a label.

SPECIAL

Searches for a special non-data (temporary) line:

- Bounds line flagged as =BNDS>
- Column identification lines flagged as =COLS>
- Information lines flagged as =====
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged as =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>.

range Two labels that define the group of lines to be searched.

Examples

To find the next special line, type:

```
LOCATE SPE
```

To find the first error line (=ERR>), type:

```
LOCATE ERR FIRST
```

To find the next line with a label, type:

```
LOC NEXT LABEL
```

To find the next excluded line between .START and .END, type:

```
LOC X .START .END
```

To find the first excluded line between .E and .S, type:

```
L FIRST .E .S X
```

MODEL—Copy a Model into the Current Data Set

The model name form of the MODEL primary command copies a specified dialog development model before or after a specified line.

The class name form of the MODEL primary command changes the model class that the editor uses to determine which model you want. For more information on edit models, see Chapter 4. Using Edit Models.

Model Name Syntax

```
MODEL [model-name [qualifier...]] {AFTER label} [NOTES ]
                                     {BEFORE label} [NONOTES]
```

If you omit the model name or a required qualifier, or if there is a validation error, the editor displays a series of selection panels from which you can select the desired information.

model-name

The name of the model to be copied, such as VGET for the VGET service model. This operand can also be one of the options listed on a model selection panel, such as V1 for the VGET service model. Refer to *ISPF Planning and Customizing* for a list of models and model names.

qualifier

The name of a model on a secondary model selection panel, such as

MODEL

TBCREATE for the TBCREATE service model. This operand can also be one of the options listed on a model selection panel, such as G1 for the TBCREATE service model.

For example, a model selection panel allows you to enter T1 to choose table models. Another model selection panel then appears for choosing table models, such as G1 for the TBCREATE service model. Therefore, your MODEL primary command could use either TABLES or T1 as the model-name operand and either TBCREATE or G1 at the qualifier operand. The simplest way would be to use TBCREATE or G1 as the model-name operand and omit the qualifier operand. Refer to *ISPF Planning and Customizing* for a list of models and model names.

AFTER label

Identifies the line after which the model is to be copied. If you have not defined a label, use the A or B line command to specify the destination. The only time this operand or the BEFORE label operand is not required is when the data set or member is empty.

BEFORE label

Identifies the line before which the model is to be copied. If you have not defined a label, use the A or B line command to specify the destination. The only time this operand or the AFTER label operand is not required is when the data set or member is empty.

NOTES

Overrides the current edit profile setting for note mode, to include any notes that are part of the model.

NONOTES

Overrides the current edit profile setting for note mode, to exclude any notes that are part of the model.

Class Name Syntax

```
MODEL [CLASS [class-name]]
```

If you omit the class-name, or if there is a validation error, the editor displays a series of selection panels from which you can select the desired information.

CLASS

When entered without the optional class-name operand, the editor displays the Model Classes panel, from which you can select a model class. When entered with the class-name operand, the macro specifies that the current model class is to be replaced by class-name. In both cases, the new class name is used for all models from that point on, until you change the model class again or end the edit session.

class-name

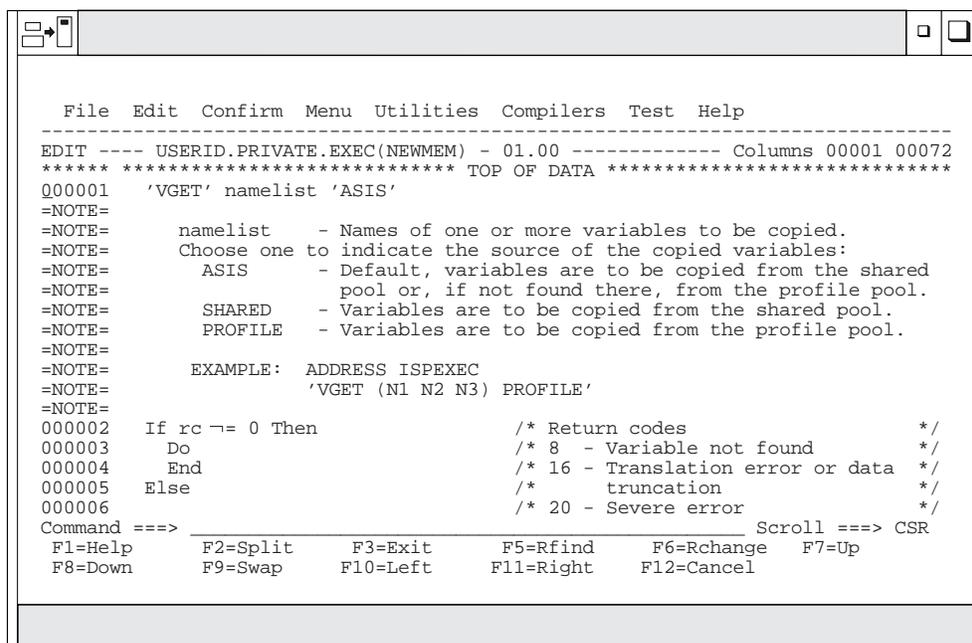
Specifies a new class for the current edit session. It must be a name on the Model Classes panel or an allowable abbreviation. The model class coincides with the type of model, such as REXX, COBOL, or FORTRAN.

Example

You are editing a new member named NEWMEM and have not decided which service to use first. Figure 140 shows the display screen for NEWMEM. Type MODEL on the Command line without any operands. Here, the model name form

MODEL

The editor inserts the VGET service model into the NEWMEM member, as shown in Figure 142. Because the edit profile is set to NOTE ON, the model's notes are also included.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(NEWMEM) - 01.00 ----- Columns 00001 00072
***** ***** TOP OF DATA *****
Q000001 'VGET' namelist 'ASIS'
=NOTE=
=NOTE=      namelist      - Names of one or more variables to be copied.
=NOTE=      Choose one to indicate the source of the copied variables:
=NOTE=      ASIS         - Default, variables are to be copied from the shared
=NOTE=                pool or, if not found there, from the profile pool.
=NOTE=      SHARED      - Variables are to be copied from the shared pool.
=NOTE=      PROFILE     - Variables are to be copied from the profile pool.
=NOTE=
=NOTE=      EXAMPLE:   ADDRESS ISPEXEC
=NOTE=                'VGET (N1 N2 N3) PROFILE'
=NOTE=
000002  If rc ^= 0 Then          /* Return codes          */
000003  Do                      /* 8 - Variable not found */
000004  End                      /* 16 - Translation error or data */
000005  Else                    /* truncation            */
000006                          /* 20 - Severe error     */
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange   F7=Up
F8=Down      F9=Swap       F10=Left    F11=Right    F12=Cancel
```

Figure 142. REXX Model of VGET Service

MOVE—Move Data

The MOVE primary command moves a sequential data set or a member of a partitioned data set into the data being edited.

Syntax

```
MOVE [member] [AFTER label ]
      (member) [BEFORE label]
      [data set name]
```

member

A member of the ISPF library or partitioned data set you are editing.

data set name

A partially qualified or fully qualified data set name. If the data set is partitioned you can include a member name in parentheses or select a member from a member list.

AFTER label

The destination of the data that is being moved. AFTER label causes the data to be moved after the specified label.

BEFORE label

The destination of the data that is being moved. BEFORE label causes the data to be moved before the specified label.

The label can be either a label you define or one of the editor-defined labels, such as .ZF and .ZL. If you have not defined a label and the editor-defined labels are not appropriate for your purpose, use the A (after) or B (before) line command to specify the data's destination.

If the data set or member that you are editing is empty, you do not need to specify a destination for the data being moved.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned a like-named member is copied. If a like-named member does not exist the name is considered to be a partially qualified data set name.

Description

MOVE adds data that already exists to the data set or member that you are editing. Use MOVE if you want to move data rather than copy it from one data set or member to another.

The member or sequential data set is deleted after the move. For a concatenated sequence of ISPF libraries, the deletion occurs only if the member was in the first library.

To move data into an empty data set or member:

1. On the Command line, type:

```
Command ==> MOVE member
              (member)
              data set name
```

The member operand is optional. If you do not specify the name of a member or a data set to be moved, the Edit Move panel appears. Enter the data set or member name on this panel.

2. Press Enter. The data is moved.

To move data into a data set or member that is not empty:

1. On the Command line, type:

```
Command ==> MOVE member AFTER | BEFORE label
              (member)
              data set name
```

The member operand is optional.

The AFTER label and BEFORE label operands are optional, also. However, if the data set or member that is to receive the moved data is not empty, you must specify a destination for the moved data. Therefore, if you do not use a label, substitute either the A (after) or B (before) line command as the destination of the moved data. However, a number indicating that the A or B command should be repeated cannot follow the line command.

If the data set or member is not empty and you do not specify a destination, a MOVE/COPY Pending message appears in the upper right-hand corner of the panel and the data is not moved. When you type a destination and press Enter, the data is moved.

2. Press Enter. If you entered a member name or a data set name, the member or data set is moved. Otherwise, the Edit Move panel appears. See the previous example for more information.

See “Copying and Moving Data” on page 50 if you need more information.

MOVE

Example

The following steps show how you can move data when you omit the member name and the editor panels appear.

1. Type MOVE on the Command line and specify the destination of the operation. In Figure 143, the data is to be moved after line 000700, as specified by the A (after) line command.

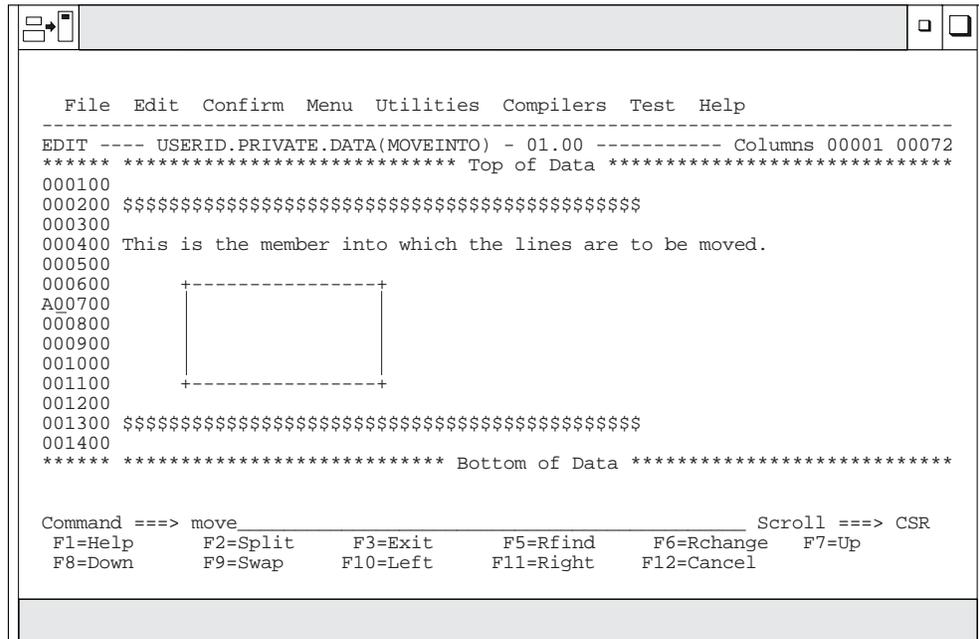


Figure 143. Member Before Data is Moved

2. When you press Enter, the Edit Move panel appears. Specify the data you want moved.

This example (Figure 144) moves the data set member named MOVEFROM.

```

Menu  RefList  Utilities  Help
-----
                          Edit/View Move

"Current" Data Set: _____

From ISPF Library:
Project . . . PROJ1
Group . . . PRIVATE . . . _____ . . . _____ . . . _____
Type . . . DATA
Member . . . MOVEFROM (Blank or pattern for member selection list)

From Other Partitioned or Sequential Data Set:
Data Set Name . . _____
Volume Serial . . _____ (If not cataloged)

Data Set Password . . (If password protected)

Press ENTER key to move. (Member or sequential data set may be deleted)
Enter END command to cancel move.

Command ==> _____
F1=Help      F2=Split      F3=Exit      F7=Backward  F8=Forward  F9=Swap
F10=Actions  F12=Cancel

```

Figure 144. Edit Move Panel (ISREMOV1)

- Figure 145 shows the contents of the MOVEFROM member which is moved into the original data set. This panel is shown only for this example, so you can see the data that is being moved. It is not displayed during a move sequence.

```

File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEFROM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000610 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000620 These are the lines that are to be moved.
000630 These are the lines that are to be moved.
000640 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split      F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down     F9=Swap      F10=Left    F11=Right   F12=Cancel

```

Figure 145. Data Set to be Moved

- When you press Enter, the editor moves the data and displays a short message in the upper-right hand side of the panel. Figure 146 shows the result of using MOVE.

NONUMBER

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(MOVEINTO) - 01.01 ---- Requested line(s) copied
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 This is the member into which the lines are to be moved.
000500
000600      +-----+
000700      |         |
000710 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000720 These are the lines that are to be moved.
000730 These are the lines that are to be moved.
000740 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000800      |         |
000900      +-----+
001000
001100
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit      F5=Rfind     F6=Rchange   F7=Up
F8=Down     F9=Swap     F10=Left    F11=Right    F12=Cancel
```

Figure 146. Member After Data Has Been Moved

NONUMBER—Turn Off Number Mode

The NONUMBER primary command turns off number mode, which controls the numbering of lines in the current data.

Syntax

NONUMBER

The NONUMBER primary command has no operands.

Description

You can also use NUMBER OFF to turn off number mode.

When number mode is off, NONUMBER prevents any verification of valid line numbers, generation of sequence numbers, and the renumbering of lines that normally occurs when autonum mode is on.

Example

To turn number mode off by using NONUMBER, enter the following:

Command ==> NONUMBER

NOTES—Display Model Notes

The NOTES primary command sets note mode, which controls whether notes are displayed when a dialog development model is inserted into the data.

Syntax

NOTES [ON]
[OFF]

ON Displays explanatory notes when a model is copied into the data being edited or when notes are added to the edit session by an edit macro.

OFF Does not display explanatory notes.

Description

Note mode is saved in the edit profile. To check the setting of note mode:

1. On the Command line, type:
Command ==> PROFILE 4
2. Press Enter. The note mode setting appears as either NOTE ON or NOTE OFF on the fourth line of the edit profile.

You can set the note mode with a primary command and then use the NOTES or NONOTES operand on the MODEL command to override the default mode for a particular model.

See “MODEL—Copy a Model into the Current Data Set” on page 257 for information about copying dialog development models.

Examples

To set note mode on:

1. On the Command line, type:
Command ==> NOTES ON
2. Press Enter. The next time you insert a model, the explanatory notes appear along with the model.

To set note mode off:

1. On the Command line, type:
Command ==> NOTES OFF
2. Press Enter. The next time you insert a model, the explanatory notes are not displayed along with the model.

NULLS—Control Null Spaces

The NULLS primary command sets nulls mode, which determines whether trailing spaces in each data field are written to the panel as blanks or nulls.

Syntax

NULLS [ON STD]
[ON ALL]
[OFF]

ON STD

Specifies that in fields containing any blank trailing space, the space is written as one blank followed by nulls. If the field is entirely empty, it is written as all blanks.

NULLS

ON ALL

Specifies that all trailing blanks and all-blank fields are written as nulls.

OFF

Specifies that trailing blanks in each data field are written as blanks.

Description

Blank characters (X'40') and null characters (X'00') both appear as blanks. When you use the I (insert) line command, the data entry area appears as blanks for NULLS ON STD and as nulls for NULLS ON ALL.

Trailing nulls simplify use of the Ins (insert) key on the IBM 3270 keyboard. You can use this key to insert characters on a line if the line contains trailing nulls.

Besides using the NULLS command, you can create nulls at the end of a line by using the Erase EOF or Del (delete) key. Null characters are never stored in the data; they are always converted to blanks.

Note: When you swap screens in split screen mode, the nulls are replaced by spaces until you press an interrupt key, such as Enter, or a function key.

Examples

To set nulls mode on with all trailing blanks and all-blank fields written as nulls, enter the following:

```
Command ==> NULLS ON ALL
```

To set nulls mode on with blank trailing space written as one blank followed by nulls and empty fields written as all blanks, enter the following:

```
Command ==> NULLS ON STD
```

To set nulls mode off and thus have trailing blanks in each data field, enter the following:

```
Command ==> NULLS OFF
```

NUMBER—Generate Sequence Numbers

The NUMBER primary command sets number mode, which controls the numbering of lines in the current data.

Syntax

```
NUMBER [ON ] [STD      ] [DISPLAY]
        [OFF] [COBOL   ]
           [STD COBOL]
           [NOSTD]
           [NOCOBOL]
           [NOSTD NOCOBOL]
```

ON

Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. You can also use RENUM to turn number mode on and renumber lines.

The editor interprets the STD, COBOL, and DISPLAY operands only when number mode is turned on.

OFF Turns number mode off. You can also use NONUMBER to turn number mode off. If you alter or delete sequence numbers and enter NONUMBER on the Command line at the same time, the editor issues the message Some input data ignored and discards the data typed over the sequence numbers. The editor converts the original sequence numbers to data.

STD Numbers the data in the standard sequence field. This is the default for all non-COBOL data set types.

COBOL Numbers the data in the COBOL field. This is the default for all COBOL data set types.

STD COBOL Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

NOSTD Turns standard number mode off.

NOCOBOL Turns COBOL number mode off.

NOSTD NOCOBOL Turns both the standard number mode and COBOL number mode off.

DISPLAY Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. Automatic left or right scrolling is performed, if required, so that the leftmost column of the data window is the first column displayed.

Description

Attention: If number mode is off, make sure the first 6 columns of your data set are blank before turning COBOL number mode on. Otherwise, the data in these columns is replaced by sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. You can also use CANCEL at any time to end the edit session without saving the data.

When number mode is on, NUMBER verifies that all lines have valid numbers in ascending sequence. It renumbers any lines that are either unnumbered or out of sequence, but it does not otherwise change existing numbers.

In number mode, the editor automatically generates sequence numbers in the data for new lines created when data is copied or inserted. The editor also automatically renumbers the data when it is saved if autonum mode is in effect.

If the number overlays the shift-in (SI) or shift-out (SO) characters, the double-byte characters appear incorrectly and results are unpredictable.

NUMBER

Examples

To number data in the standard sequence field, enter the following:

```
Command ==> NUMBER ON STD
```

To number data in both the standard and COBOL fields and include sequence numbers in the display, enter the following:

```
COMMAND ==> NUMBER ON STD COBOL DISPLAY
```

PACK—Compress Data

The PACK primary command sets pack mode, which controls whether the data is to be stored in packed format.

The PACK command saves the pack mode setting in the edit profile. See “Packing Data” on page 19 for more information about packing data.

Syntax

```
PACK [ON ]  
      [OFF]
```

ON Saves data in packed format.

OFF Saves data in unpacked (standard) format.

Examples

To set pack mode on, enter the following:

```
Command ==> PACK ON
```

To set pack mode off, enter the following:

```
Command ==> PACK OFF
```

PASTE—Move or Copy Lines from Clipboard

The PASTE primary command moves or copies lines from a clipboard into an edit session.

Syntax

```
PASTE [clipboardname] [AFTER label]  
      [BEFORE label][KEEP]
```

clipboardname

The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

BEFORE label

The destination of the data that is being transferred from the clipboard. BEFORE copies the data *before* the specified label.

AFTER label

The destination of the data that is being transferred from the clipboard. AFTER copies the data *after* the specified label.

KEEP Records are copied and not removed from the clipboard. If you omit this keyword, the records are removed from the clipboard.

Description

PASTE copies or moves lines from a specified clipboard to the current edit session. If lines in the clipboard are longer than the lines in the edit session, they are truncated.

The portion of the line that is saved in the clipboard is only the data portion of the line. Line numbers are *not* saved. If the data was CUT from a data set that had sequence numbers and is PASTEd into an edit session without sequence numbers, or if it was CUT from a data set without sequence numbers and PASTEd into a session with sequence numbers, some shifting of data is likely to occur.

Example

To paste data from the default clipboard to the line after the last line in the edit session:

```
PASTE AFTER .ZLAST
```

To paste data from the default clipboard to the line after the first line in the edit session, without clearing the contents of the clipboard:

```
PASTE AFTER .ZFIRST KEEP
```

PRESERVE - Enable Saving of Trailing Blanks

The PRESERVE primary command enables or disables the saving of trailing blanks in the editor. This gives you the ability to override the setting for the **Preserve VB record length** field on the edit entry panel.

Syntax

```
PRESERVE [ON ]
          [OFF]
```

ON The editor preserves the record length of the record when the data is saved.

OFF Turns truncation on. ISPF removes trailing blanks when saving variable length files.

Regardless of the PRESERVE setting, if a line has a length of zero, ISPF saves 1 blank. If the line is eight characters long, ISPF pads the record to 9 characters by adding a blank to the end.

Description

PRESERVE ON causes the editor to save trailing blanks for variable length files. The number of blanks saved for a particular record is determined by one of the following:

- the original length of the record when it was read in to the editor

PRESERVE

- the number of blanks required to pad the record length specified by the SAVE_LENGTH edit macro command
- the length of the record that was saved on disk during a previous SAVE request in the same edit session.

PRESERVE OFF cause the editor to truncate trailing blanks. If a line is empty ISPF saves 1 blank. If the line is eight characters long, ISPF pads the record to 9 characters by appending a blank to the end.

Use of the PRESERVE command does not prevent the editor from working on data past the specified record length. The length set and returned by the PRESERVE command is only used when the data is written and does not affect the operation of other edit functions.

Examples

To enable the editor to remove trailing blanks when data is saved, enter the following:

```
Command ==>> PRESERVE OFF
```

To save the trailing blanks, enter the following:

```
Command ==>> PRESERVE ON
```

PROFILE—Control and Display Your Profile

The control form of the PROFILE primary command appears your current edit profile, defines a new edit profile, or switches to a different edit profile.

The lock form of the PROFILE primary command locks or unlocks the current edit profile.

Profile Control Syntax

```
PROFILE [name] [number]
```

name The profile name. It can consist of up to 8 alphanumeric characters, the first of which must be alphabetic. The edit profile table is searched for an existing entry with the same name. That profile is then read and used. If one is not found, a new entry is created in the profile table.

If you omit this operand, the current edit profile is used.

number

The number of lines, from 0 through 9, of profile data to be displayed. When you type 0 as the number, no profile data is displayed. When no operands are entered, the first five lines, which contain the =PROF> flags, always appear. However, the =MASK> and =TABS> lines are not displayed if they contain all blanks; if the =MASK> and/or =TABS> lines do contain data, they appears, followed by the =COLS> line.

For more information about displaying and defining a profile, see “Displaying or Defining an Edit Profile” on page 21.

Profile Lock Syntax

```
PROFILE {LOCK | UNLOCK}
```

LOCK Specifies that the current values in the profile are saved in the edit profile table and are not modified until the profile is unlocked. The current copy of the profile can be changed, either because of commands you enter that modify profile values (BOUNDS and NUMBER, for example) or because of differences in the data from the current profile settings. However, unless you unlock the edit profile, the saved values replace the changes when you end the edit session.

Caps, number, stats, and pack mode are automatically changed to fit the data. These changes occur when the data is first read or when data is copied into the data set. Message lines (==MSG>) are inserted in the data set to show you which changes occurred.

Note: To force caps, number, stats, or pack mode to a particular setting, use an initial macro. Be aware, however, that if you set number mode on, data may be overlaid.

UNLOCK

Specifies that the editor saves changes to profile values.

See “Locking an Edit Profile” on page 23 for more information about locking and unlocking the profile.

Profile Reset Syntax

PROFILE RESET

RESET

Specifies that the ZDEFAULT profile is to be removed and the site-wide configuration for new edit profiles is to be used.

See “Locking an Edit Profile” on page 23 for more information about locking and unlocking the profile.

Description

To display the current edit profile:

1. On the Command line, type:
Command ==> PROFILE number
2. Press Enter. The current edit profile appears.

To switch edit profiles or define a new edit profile without displaying the new profile:

1. On the Command line, type:
Command ==> PROFILE name 0

where name is the name of the edit profile to which you want to switch. This also specifies that no lines are to be displayed. If you want to display the new profile, you can omit the number or enter a number from 1 to 9.

2. Press Enter. The profile specified by the name operand becomes the active edit profile, but is not displayed if you entered 0. If the profile does not exist, an entry is created for it in the edit profile table, using the values of the current edit profile.

To lock the current edit profile:

1. On the Command line, type:

PROFILE

Command ==> PROFILE LOCK

2. Press Enter. The values in the current edit profile are saved in the edit profile table. From this point on, any changes you make to the current edit profile affect only the current edit session. Values that were saved when the current profile was locked are used the next time you begin an edit session with this profile.

To unlock an edit profile:

1. On the Command line, type:

Command ==> PROFILE UNLOCK

2. Press Enter. From this point on, any changes that you make to the current edit profile replace any values that may have been saved for this profile in the edit profile table. Also, these changes are saved when you end the current edit session.

Example

Figure 147 shows a typical edit profile for a REXX data set. The display results from entering PROFILE with no operands. The =TABS> and =MASK> lines appear because they contained data. If they had been empty, they would not have appeared.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 -----COLUMNS 00001 00072
***** ***** Top of Data *****
=PROF> ....EXEC (FIXED - 80)...RECOVERY ON...NUMBER ON STD.....
=PROF> ....CAPS ON...HEX OFF...NULLS OFF...TABS OFF.....
=PROF> ....AUTOSAVE ON...AUTONUM OFF...AUTOLIST OFF...STATS ON.....
=PROF> ....PROFILE UNLOCK...IMACRO NONE...PACK OFF...NOTE ON.....
=PROF> ....HILITE OFF.....
=TABS> - - - - * - - - -
=MASK>
=BNDS>
=COLS> -----1-----2-----3-----4-----5-----6-----7-----
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
000500 DO /* AND */
000600 IF TEMP = FIRST /* IF 'TEMP' IS EQUAL */
000700 THEN /* TO 'FIRST' THEN */
000800 FIRST = LAST /* SET 'FIRST' EQUAL */
Command ==> Scroll ==>
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
  
```

Figure 147. Edit Profile Display

The sample profile contains the following information:

- The first profile line (=PROF>) shows the profile name (EXEC), the data set record format and length (FIXED - 80), and the settings for edit recovery mode (RECOVERY ON) and number mode (NUMBER ON STD).
- The second profile line shows the settings for caps mode (CAPS ON), hexadecimal mode (HEX OFF), nulls mode (NULLS OFF), tabs mode (TABS OFF), and UNDO mode (SETUNDO STG).

- The third profile line shows the settings for the auto modes: autosave (AUTOSAVE ON), autonum (AUTONUM OFF), and autolist (AUTOLIST OFF). It also shows the setting for stats mode (STATS ON).
- The fourth profile line shows the lock status of the EXEC profile (PROFILE UNLOCK), the name, if any, of the initial macro called at the beginning of the edit session (IMACRO NONE), and the settings for pack mode (PACK OFF) and note mode (NOTE ON).
- The fifth profile line shows the current hilite status (HILITE OFF).
- The last four lines of the edit profile show the tabs settings (=TABS>), edit mask (=MASK>), bounds settings (=BNDS>), and the column position line (=COLS>).

RCHANGE—Repeat a Change

RCHANGE repeats the change requested by the most recent CHANGE command.

Syntax

RCHANGE

Description

You can use this command to repeatedly change other occurrences of the search string. After a *string* NOT FOUND message appears, the next RCHANGE issued starts at the first line of the current range for a forward search (FIRST or NEXT specified) or the last line of the current range for a backward search (LAST or PREV specified).

Note: RCHANGE is normally assigned to a program function key, although you can issue it directly from the Command line.

RECOVERY—Control Edit Recovery

RECOVERY sets edit recovery mode, which allows you to recover data after a system failure or power outage.

Syntax

RECOVERY [*ON* | *OFF*]
 [*WARN* | *NOWARN* | *SUSP*]

ON The system creates and updates a recovery data set for each change.

OFF The system does not create and update a recovery data set.

WARN

This operand no longer has a practical function due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN

This operand no longer has a practical function due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP This operand functions the same as the ON operand.

RECOVERY

Note: When SETUNDO is enabled during installation, both the RECOVERY primary command and edit macro command continue to accept the NOWARN and WARN keywords for compatibility reasons, but the value is ignored. NOWARN will always be in effect.

Description

You cannot edit data recursively while you are in recovery.

Attention:

If the data set to be recovered was edited by another user before edit recovery, the changes made by the other

See “Undoing Edit Interactions” on page 74 for more information.

To turn on edit recovery mode:

1. On the Command line, type:

```
Command ==> RECOVERY ON
```

RECOVERY can be abbreviated REC. This command can also ensure that your edit session is not lost due to a system failure.

2. Press Enter. The editor begins recording an audit trail of your interactions. After a system failure, the editor uses that record to reestablish the edit session at the time of failure.

Note: For edit recovery to work properly, the data set to be recovered, the edit recovery data set, and the edit recovery table all must exist, be cataloged, and be intact. For example, with RECOVERY on, uncataloging a data set and then trying to recover it fails.

To turn off edit recovery mode:

1. On the Command line, type:

```
COMMAND ==> RECOVERY OFF
```

2. Press Enter. The editor stops recording your interactions. Edit recovery is not available following a system failure. When an edit session is recovered, the data is scrolled all the way to the left when the recovery edit session begins.

See “Edit Recovery” on page 47 for more information about edit recovery.

RENUM—Renumber Data Set Lines

RENUM immediately turns on number mode and rennumbers all lines, starting with number 100 and incrementing by 100. For members exceeding 10 000, the increment would be less than 100.

Syntax

```
RENUM [ON ] [STD      ] [DISPLAY]
          [COBOL    ]
          [STD COBOL]
```

ON Automatically verifies that all lines have valid numbers in ascending

sequence and renumbers any lines that are either unnumbered or out of sequence. It also turns number mode on and renumbers lines.

The STD, COBOL, and DISPLAY operands are interpreted only when number mode is turned on.

STD Numbers the data in the standard sequence field. This is the default for all non-COBOL data set types.

COBOL

Numbers the data in the COBOL field. This is the default for all COBOL data set types. **Attention:**

If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. Or, you can use CANCEL at any time to end the edit session without saving the data.

STD COBOL

Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields are not synchronized. Use RENUM to synchronize them.

DISPLAY

Causes the width of the data window to include the sequence number fields. Otherwise the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. The editor automatically scrolls left or right, if required, so that the leftmost column of the data window is the first column to appear.

Description

To renumber all lines using the standard sequence fields only:

Command ==> RENUM STD

To renumber all lines using both the standard and COBOL sequence fields:

Command ==> RENUM STD COBOL

To renumber all lines using the COBOL sequence fields only:

Command ==> RENUM COBOL

To renumber all lines using both the standard and COBOL sequence fields and specifying that the data window is to include the sequence number fields:

Command ==> RENUM STD COBOL DISPLAY

To renumber all lines by using the standard sequence fields only and specifying that the data window is to include the sequence number fields:

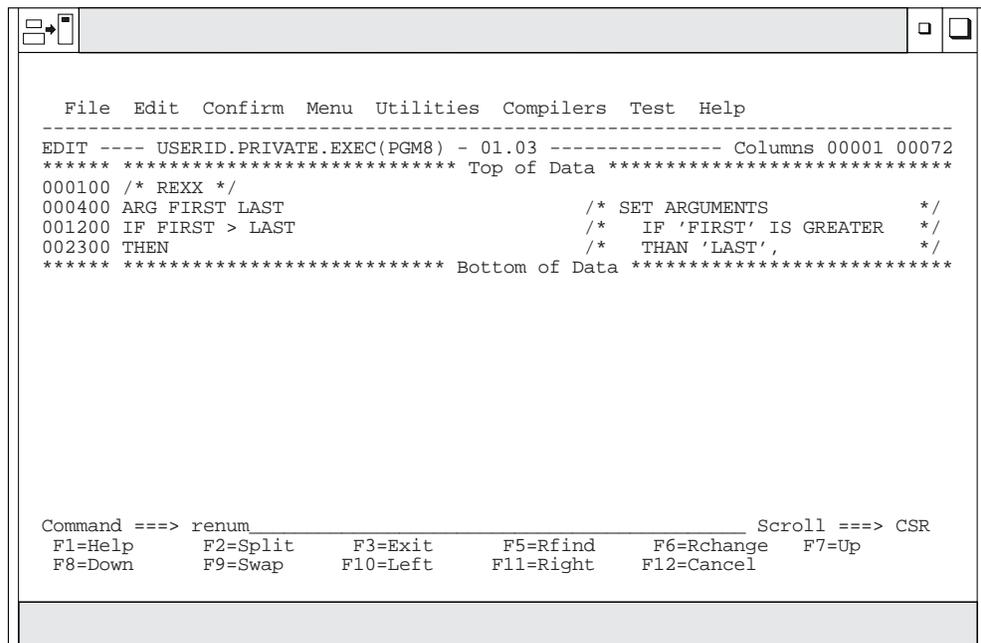
Command ==> RENUM DISPLAY

RENUM

Here, the DISPLAY operand is the only operand needed because STD is the default.

Example

In Figure 148, the line numbers are not incremented uniformly. Type RENUM on the Command line. Figure 149 shows how the lines are renumbered after you press Enter.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000400 ARG FIRST LAST /* SET ARGUMENTS */
001200 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
002300 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> renum
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 148. Member Before Lines Are Renumbered

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 149. Member After Lines Are Renumbered

REPLACE—Replace Data

The REPLACE primary command replaces a sequential data set or a member of a partitioned data set with data you are editing. If the member you want to replace does not exist, the editor creates it.

Syntax

```

REPLACE [member] [range]
REPLACE (member) [range]
REPLACE [data_set]
REPLACE [data_set(member)]

```

member

The name of the member to be replaced in the partitioned data set currently being edited. If the member does not exist, the editor creates it. If you are using a concatenated sequence of libraries, the editor writes the member to the first library in the sequence. This operand is optional.

To replace a sequential data set or a member of a different partitioned data set, enter REPLACE without a member operand. The editor displays the Edit Replace panel, from which you can enter the data set name.

data_set

A partially qualified or fully qualified sequential data set you want to replace.

data_set(member)

A partially qualified or fully qualified partitioned data set and member you want to replace.

range Two labels that show which lines replace the member or data set. Specify a pair of labels that show the beginning and end of the group of lines.

REPLACE

Description

To replace a member of a partitioned data set or to replace a sequential data set:

1. On the Command line, type:

```
Command ==> REPLACE member range
Command ==> REPLACE (member) range
Command ==> REPLACE data_set range
Command ==> REPLACE data_set(member) range
```

The member operand is optional unless you specify the name of a partitioned data set. It represents the name of the member that you want to replace. If you specify a data set name only, it must be a sequential data set.

The range operand is optional, also. It represents a pair of labels that show the first and last lines in a group of lines used to replace the member.

If you omit the range operand, you must specify the lines by using either the C (copy) or M (move) line command. See the descriptions of these commands if you need more information about them.

If you omit the range operand and do not enter one of the preceding line commands, a REPLACE Pending message is displayed in the upper-right corner of the panel.

2. Press Enter. If you did not specify a member name or a data set name, the Edit Replace panel is displayed. Enter the member name on this panel and press Enter again. If you used either a pair of labels or a C line command, the data is copied from the member that you are editing into the member that you are replacing. If you used the M line command, however, the data is removed from the member that you are editing and placed in the member that you are replacing.

If the data set specified does not exist, ISPF prompts you to see if the data set should be created. You can create the data set using the characteristics of the source data set as a model, or specify the characteristics for the new data set. You can suppress this function through the ISPF configuration table, causing any CREATE request for a non-existent data set to fail.

See “Creating and Replacing Data” on page 49 for more information about the REPLACE command.

Example

The following steps show how you can replace a member when you omit the member name. These same steps apply when you create data.

1. Type REPLACE and specify which lines you want to copy or move into the data set or member. The example in Figure 150 uses the MM (block move) line command to move a block of lines from the data.

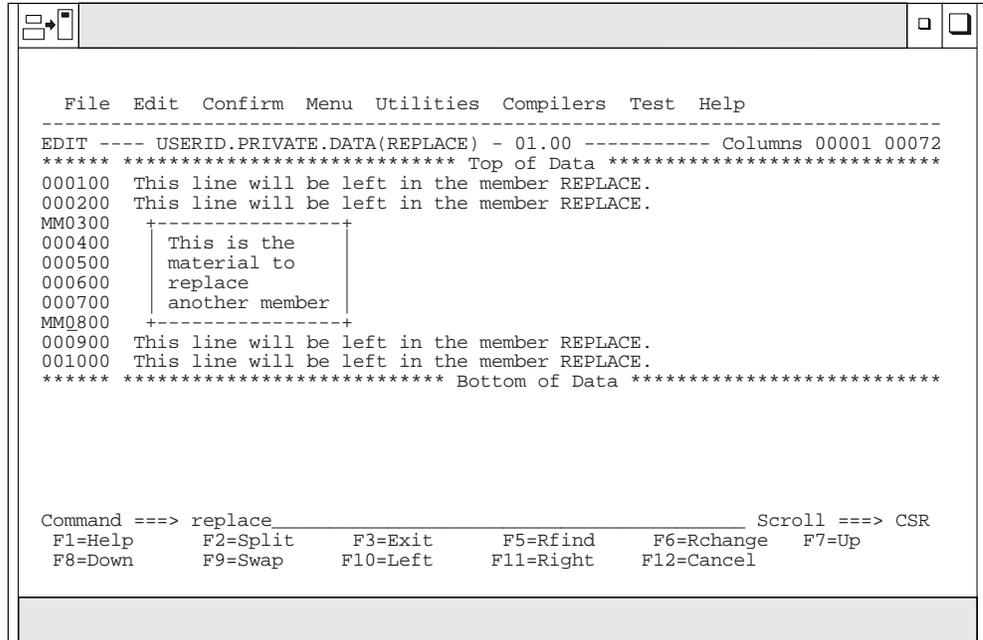


Figure 150. Member Before Other Member Is Replaced

- When you press Enter, the Edit Replace panel (Figure 151) appears. Type the name of the member to be replaced and press Enter. A member is created when you type the name of a member that does not already exist. The name of the member replaced in this example is REPMEM.

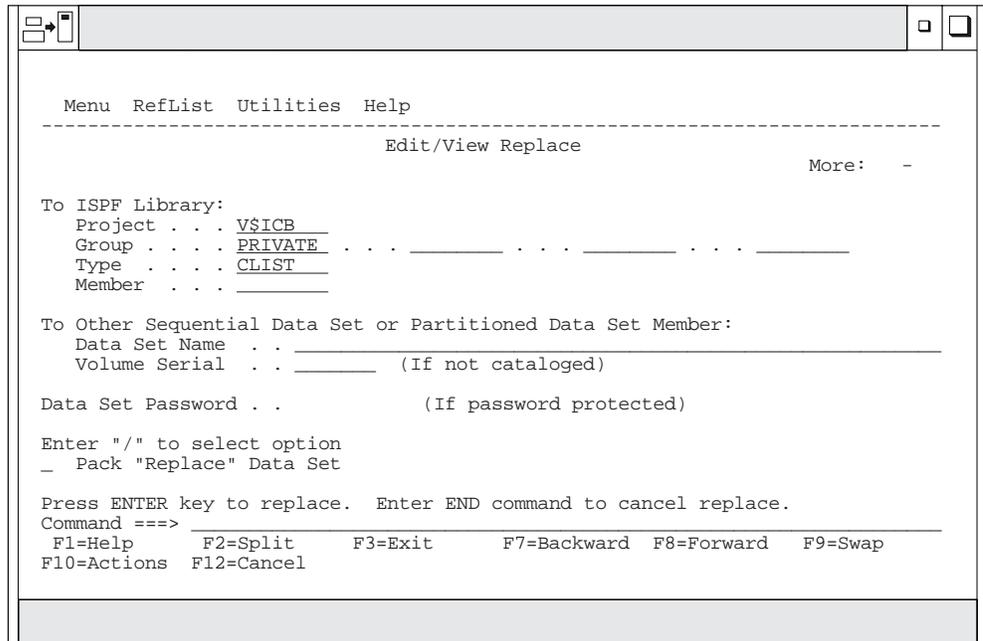


Figure 151. Edit - Replace Panel (ISRERPL1)

- Figure 152 shows the lines remaining in the data being edited after the specified lines were moved.

REPLACE

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(REPLACE) - 01.00 ----- MEMBER REPMEM REPLACED
***** ***** TOP OF DATA *****
000100 This line will be left in the member REPLACE.
000200 This line will be left in the member REPLACE.
000900 This line will be left in the member REPLACE.
001000 This line will be left in the member REPLACE.
***** ***** BOTTOM OF DATA *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 152. Member After the Other Member Has Been Replaced

4. Figure 153 shows the contents of the replaced member.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.DATA(REPMEM) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 +-----+
000200 | This is the |
000300 | material to |
000400 | replace |
000500 | another member |
000600 +-----+
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left    F11=Right   F12=Cancel
```

Figure 153. Other Member Replaced

RESET—Reset the Data Display

The RESET primary command can restore line numbers in the line command area when those line numbers have been replaced by labels, pending line commands, error flags, and change flags. RESET can also delete special lines from the display, redisplay excluded lines, and temporarily disable the highlighting of FIND strings.

Syntax

```
RESET [CHANGE ] [range]
      [COMMAND ]
      [ERROR ]
      [EXCLUDED]
      [FIND ]
      [LABEL ]
      [SPECIAL ]
```

You can type the operands in any order. If you do not specify any operands, RESET processes all operands except LABEL.

CHANGE

Removes ==CHG> flags from the line command area.

COMMAND

Removes any pending line commands from the line command area.

ERROR

Removes ==ERR> flags from the line command area.

EXCLUDED

Redisplays any excluded line.

FIND Turns off highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command. SEEK and EXCLUDE do not return the highlighting of FIND strings in this manner.

The resetting of FIND highlighting does not honor the range specified on the RESET command.

LABEL

Removes labels from the line command area.

SPECIAL

Deletes any temporary line from the panel:

- Bounds line flagged as =BNDS>
- Column identification lines flagged with =COLS>
- Information lines flagged with =====
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged with =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>.

range Specifies the range of lines to be reset. The labels can be labels that the PDF component has defined or labels that you have defined. The range operand is useful when you do not want to reset lines in the complete data set. You can specify the range operand with any other operand on the command.

Description

RESET scans every line of data. If you want to delete a small number of special lines, you can get faster response time if you use the D (delete) line command.

Examples

To reset all lines except those that contain labels:

```
Command ==>> RESET
```

RESET

To reset only the lines that contain labels:

Command ==> RESET LABEL

To reset only the lines that contain pending line commands:

Command ==> RESET COMMAND

To reset only the lines that contain ==ERR> flags:

Command ==> RESET ERROR

To reset only the lines that contain ==CHG> flags:

Command ==> RESET CHANGE

To reset only the special (temporary) lines:

Command ==> RESET SPECIAL

To reset only the excluded lines:

Command ==> RESET EXCLUDED

To reset all lines between and including the .START and .STOP labels, except those that contain labels:

Command ==> RESET .START .STOP

RFIND—Repeat Find

RFIND locates the search string defined by the most recent SEEK, FIND, or CHANGE command, or excludes a line containing the search string defined by the previous EXCLUDE command.

RFIND can be used repeatedly to find other occurrences of the search string. After a *string* NOT FOUND message is displayed, the next RFIND issued starts at the first line of the current range for a forward search (FIRST or NEXT specified), or the last line of the current range for a backward search (LAST or PREV specified).

Syntax

RFIND

Note: RFIND is normally assigned to a program function key, although you can issue it directly from the Command line.

RMACRO—Specify a Recovery Macro

RMACRO saves the name of a recovery macro in the edit profile.

Syntax

RMACRO {name | NONE}

name The name of the recovery macro to be run. The name can be preceded by an exclamation point (!) to show that it is a program macro.

NONE

The name to prevent a recovery macro from being run.

Description

To specify the name of a recovery macro:

1. On the Command line, type:

```
Command ==> RMACRO name
```

where name is the name of the recovery macro that you want to run.

2. Press Enter.

See “Recovery Macros” on page 118 for more information.

Example

To define RESTART as the recovery macro, type:

```
Command ==> RMACRO RESTART
```

To reset the profile with no recovery macro, type:

```
Command ==> RMACRO NONE
```

SAVE—Save the Current Data

SAVE saves edited data without ending your edit session. Generally, you do not need to use SAVE if recovery mode is on. See AUTOSAVE, CANCEL, and END for more information about saving data.

Syntax

```
SAVE
```

Description

SAVE writes the data to the same data set from which it was retrieved unless you specified a concatenated sequence of partitioned data sets on the Edit Entry panel. In that case, the data is saved in the first library in the concatenation sequence, regardless of from which library it came. For a sequential data set, the complete data set is rewritten. For a partitioned data set, the member is rewritten with the same member name. If stats mode is on, the library statistics for the member are automatically updated.

If both number mode and autonum mode are on, the data is automatically renumbered before it is saved.

If SAVE cannot successfully rewrite the data because of I/O errors or insufficient space, the system displays a message in the upper-right corner of the panel, accompanied by an audible alarm, if installed. You can then try to save the data in another data set by taking the following steps:

1. Enter CREATE or REPLACE with no operand on the Command line. Use CREATE only if the destination is a member of a partitioned data set, such as an ISPF library member.
2. Type CC on the first and last data lines to specify that all lines are to be copied. Then press Enter.
3. Fill in the data set and member name of the alternate library on the Edit Create or Edit Replace panel, and press Enter.

SAVE

When a space ABEND such as D37 occurs, ISPF unallocates the data set so that you can swap to another screen or user ID and reallocate the data set. This does not occur for data sets that were edited using the DDNAME parameter of the EDIT service.

See “Creating and Replacing Data” on page 49 for more information.

Example

To save the data in the data set or member that you are editing:

1. On the Command line, type:

```
Command====> SAVE
```

2. Press Enter.

SETUNDO—Set the UNDO Mode

The SETUNDO primary command determines whether or not the UNDO command is available and how the history of changes should be managed.

Note: The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *ISPF Planning and Customizing*

Syntax

```
SETUNDO [STORAGE | RECOVER | ON | OFF]
```

STORAGE

Enables the saving of edit changes in storage. If the setting is changed, and the profile lines are displayed, the profile lines reflect the new value after the change (SETUNDO STG).

RECOVER

Enables the saving of edit changes through the recovery file only. If recovery is off, it is turned on by this command. If the setting is changed and the profile lines are displayed, the profile lines reflect the new value after the change (SETUNDO REC).

ON Enables edit changes to be saved in STORAGE

OFF Disables the saving of edit changes in storage. If SETUNDO OFF is specified and recovery is on, then a state of SETUNDO RECOVER is set and UNDO is available from the recovery file. All transactions on the storage UNDO chain are removed, and no changes before SETUNDO OFF can be undone (unless RECOVERY ON is specified). If the setting is changed and the profile lines are displayed, the profile lines reflect the new value after the change (SETUNDO OFF or SETUNDO REC).

Description

SETUNDO allows you to specify how changes you make during your edit session are to be recorded and used by the UNDO command. UNDO can be run when either SETUNDO or RECOVERY is on. Changes can be recorded in storage, in the recovery file, or in both places. Saving the changes in storage only is the fastest method.

To enable recording in storage:

1. On the Command line, type either of the following:

```
Command ==> SETUNDO STORAGE
OR
Command ==> SETUNDO
```

2. Press Enter.

Valid abbreviations for STORAGE are STO, STG, STOR and STORE. SETUNDO may be abbreviated SETU. The value of ON is accepted to compliment the OFF state.

To use the recovery file:

1. On the Command line, type:

```
Command ==> SETUNDO RECOVER
```

2. Press Enter.

If RECOVERY is off, it is turned on by this command. REC is a valid abbreviation for RECOVER.

To turn off recording and disable the UNDO command, enter:

```
Command ==> SETUNDO OFF
```

Note: If recovery is on, setting SETUNDO OFF is the same as specifying SETUNDO REC, and the recovery file is used for UNDO.

Example

The edit profile shown in Figure 154 shows SETUNDO set to STORAGE and RECOVERY OFF.

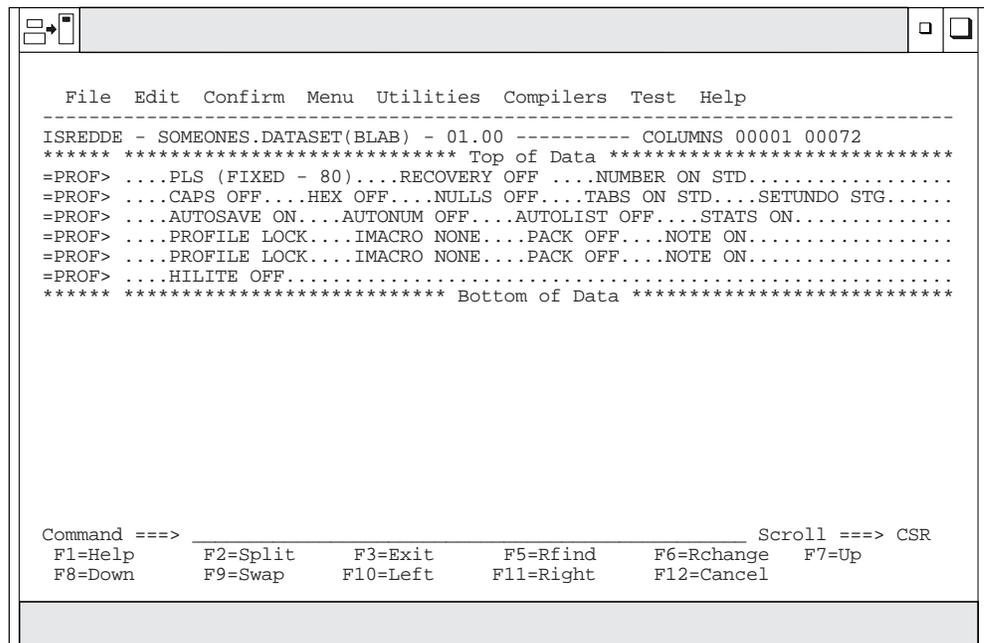


Figure 154. SETUNDO STORAGE and RECOVERY OFF

SORT—Sort Data

The SORT primary command puts data in a specified order.

Syntax

```
SORT [range] [X ] [sort-field1 ... sort-field5]
      [NX]
```

range Two labels that define the first and last lines to be sorted.

X Sorts only lines that are excluded.

NX Sorts only lines that are not excluded.

sort-field1 ... sort-field5

Specifies the fields to be used in sorting data. You can specify up to five sort fields as follows:

```
[A] [start-col [end-col]]
[D]
```

where:

A Specifies ascending order. It can either precede or follow the column specification. A is the default.

D Specifies descending order. It can either precede or follow the column specification.

start-col

Defines the starting column of the field that is to be compared. It must be within the current boundaries.

end-col

Defines the ending column of the field that is to be compared. It must be within the current boundaries.

If you specify several fields, you must specify both the starting and ending columns of each field. The fields cannot overlap. If you specify A or D for one field, you must specify it for all fields.

Description

SORT operates in two different modes, based on the hexadecimal mode status. If hexadecimal mode is on, the data is ordered according to its hexadecimal representation. If hexadecimal mode is off, data is sorted in the collating sequence defined for the national language being used.

Sorting Data Without Operands

For SORT with no operands, the editor compares the data within the current boundaries character by character, and then orders it line by line in the proper collating sequence. It ignores data outside the current boundaries during both operations. Therefore only the data inside the current boundaries is changed. Labels, excluded lines, line numbers, and change, error, and special line flags are considered associated with the data, and therefore point to the same data fields after the sort as they did before the sort.

For example, if you issue a CHANGE ALL that changes the first, third, and sixth lines in a data set, these lines are flagged with the change flag, ==CHG>. If you then

issue a SORT command that results in the former lines 1, 3 and 6 becoming the first, second and third lines of the sorted file, the changed line flags would now exist on the first, second and third lines of the sorted data set.

It is important to properly set the boundaries before issuing SORT. SORT is a powerful tool for editing data that may be formatted in multiple columns. You can set the boundaries, for example, to the first half of a record and sort one column of data. Then you can set the boundaries to the last half of the record and sort a second column of data.

Limiting the SORT Command

Sorting is limited to data within the current boundaries. You can specify up to five sort fields by labelling starting and ending columns. You can also identify each field as having data sorted in either ascending or descending order.

Optionally, you can limit sorting to a range of lines by specifying the labels of the first and last lines of the range. You can also limit sorting to either excluded or nonexcluded lines.

If you have labels or line ranges that are between the labels or line ranges specified with SORT, you can keep SORT from rearranging them by:

- Excluding them before you enter SORT
- Using the NX operand to sort only lines that are not excluded.

For more information, see the definition of the NX operand and “EXCLUDE—Exclude Lines from the Display” on page 242.

Sorting DBCS Data

When sorting data that contains DBCS character strings, you must ensure that no DBCS string crosses the boundaries. Also, all records must have the same format at the boundaries, although the format of the left and right boundaries can differ.

If a boundary divides a DBCS character, or if all records do not have the same format at the boundaries, the result is unpredictable.

Examples

The following form of the SORT command sorts in ascending order. The start-column is the left boundary and the end-column is the right boundary:

```
SORT
```

The following form of the SORT command sorts in descending order. The start-column is the left boundary and the end-column is the right boundary:

```
SORT D
```

The following form of the SORT command sorts in ascending order. The start-column is column 5 and the end-column is the right boundary:

```
SORT 5
```

The following form of the SORT command sorts in descending order. The start-column is column 5 and the end-column is the right boundary:

```
SORT 5 D
```

STATS—Generate Library Statistics

The STATS primary command sets stats mode, which creates and maintains statistics for a member of a partitioned data set.

Syntax

```
STATS [ON ]  
      [OFF]
```

ON Creates or updates library statistics when the data is saved.

OFF Does not create or update library statistics.

See “Statistics for PDS Members” on page 30 for more information.

Examples

To set stats mode on:

```
Command ==>> STATS ON
```

To set stats mode off:

```
Command ==>> STATS OFF
```

SUBMIT—Submit Data for Batch Processing

The SUBMIT primary command submits the member or data set you are editing (or the part of the member or data set defined by the range of line pointers) to be processed as a batch job.

Syntax

```
SUBMIT [range]
```

range Two labels that define the first and last lines to be submitted.

Description

The editor does not supply a job statement when you enter the SUBMIT command. You can supply job statements as part of the data being submitted. When you supply a job statement, only the job name is logged to the ISPF log data set to ensure the protection of sensitive data.

The PDF component uses the TSO SUBMIT command to submit the job.

Examples

To submit lines between labels .START and .END as a batch job:

```
Command ==>> SUBMIT .START .END
```

To submit all of the data as a batch job:

```
Command ==>> SUBMIT
```

TABS—Define Tabs

The TABS primary command:

- Turns tabs mode on and off
- Defines the logical tab character
- Controls the insertion of attribute bytes at hardware tab positions defined with TABS.

Use PROFILE to check the setting of tabs mode and the logical tab character. See “Using Tabs” on page 71 if you need more information about using tabs.

Syntax

```
TABS [ON ] [STD]
      [OFF] [ALL]
      [tab-character]
```

ON Turns tabs mode on, which means that logical tabs can be used to break up strings of data. This is the default operand. If no other operands are included, all hardware tab positions (asterisks) that contain a blank or null character are activated because STD is also a default operand. The TABS ON STD message appears in the profile display.

OFF Turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab positions, causing the Tab Forward and Tab Backward keys to ignore hardware tabs defined on the =TABS> line. Blanked-out characters occupying these positions reappear. The TABS OFF message appears in the profile display.

STD Activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. STD is the default operand. You can use the Tab Forward and Tab Backward keys to move the cursor one space to the right of the attribute bytes. The TABS ON STD message appears in the profile display.

ALL Causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over. The Tab Forward and Tab Backward keys can be used to move the cursor one space to the right of these attribute bytes. The TABS ON ALL message appears in the profile display.

tab-character

Defines a single character that is not a number, letter, or command delimiter as the logical tab character. This character is used with hardware tab definitions. The TABS ON tab-character message appears in the profile display.

You can enclose the character in quotes (' or "), although this is not necessary unless a quote or a comma (,) is used as the tab character.

The tab-character operand causes the data string that follows the logical tab character to align itself one space to the right of the first available hardware tab position when you press Enter. No attribute bytes are inserted.

If no hardware tabs are defined, the editor aligns the data vertically. If software tabs are defined, the first data string is aligned under the first software tab position and the remaining data strings are aligned at the left

TABS

boundary. If neither software nor hardware tabs are defined, the editor aligns all the data strings at the left boundary.

With the tab-character operand, the Tab Forward and Tab Backward keys ignore hardware tab positions because no attribute bytes are inserted.

You can type the operands in any order, but keep the following rules in mind:

- The tab-character and ALL operands cannot be used together, because the tab-character operand does not allow the PDF component to insert attribute bytes at tab positions, while the ALL operand does.
- The TABS primary command has no effect on software tabs. Whenever software tabs are defined, you can always use the Enter key to move the cursor to a software tab position in the data, even if tabs mode is off. Attribute bytes are not inserted at software tab positions.

Example

Define the pound sign (#) as a logical tab character by typing the following and pressing Enter:

```
Command ==> TAB #
```

Now, enter the COLS line command by typing COLS in the line command area and pressing Enter. A partial =COLS> line with positions 9 through 45 is shown in the following example.

To use the logical tab character you have defined (#), you also need at least one hardware tab. For this example, we will assume that three hardware tabs have already been defined in columns 20, 30, and 40:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>          *          *          *
```

If you then type the following information on a line:

```
#$4237#$ 596#$ 81
```

the data \$4237 is repositioned after the first tab column, defined by an * in the =TABS line, when you press Enter. The \$ 596 is repositioned after the next tab column and so forth, as follows:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>          *          *          *
              $4237    $ 596    $ 81
```

UNDO—Reverse Last Edit Interaction

The UNDO primary command allows you to remove the data modifications of a previous interaction.

Note: The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *ISPF Planning and Customizing*

Syntax

UNDO

Description

Each time you enter UNDO, it reverses edit interactions, one at a time, in the order in which they have been entered. To use UNDO, you must have either RECOVERY on or SETUNDO on. You can undo only those changes made after RECOVERY or SETUNDO was turned on. SETUNDO and RECOVERY can be specified in your edit profile. You can also use the edit macro command ISREDIT SETUNDO to turn UNDO processing on and off. See “SETUNDO—Set UNDO Mode” on page 396 for more information.

RECOVERY is now optional and is not required to run UNDO. Performance improves if the editor is run with SETUNDO STORAGE and RECOVERY OFF. In this mode, non-data changes, such as setting line labels, adding note lines, and inserting blank lines, can be undone by UNDO even if no data changes have been made. With RECOVERY ON, only changes made after (and including) the first change to edit data can be undone.

Note: Changes made by initial edit macros cannot be undone.

See “Understanding Differences in SETUNDO Processing” on page 76 for more information on the differences between SETUNDO RECOVER and SETUNDO STORAGE processing.

Each time you press Enter, an interaction occurs between you and the PDF component. If you combine line and primary commands in one entry, the PDF component considers this one interaction. Therefore, UNDO would cause all of the commands to be reversed. The PDF component also considers running edit macros that contain a combination of macro commands and assignment statements, while entering a combination of edit line and primary commands at the same time, as one interaction.

Profile changes, such as HEX ON, LEVEL, and CAPS, cannot be undone separately. Profile changes are associated with the data change that came before them, and can be undone only when preceded by a data change. The data change and the profile change are undone at the same time. For example, if you make a change to the data, change the version number, set caps off, turn hex on, and then enter UNDO, the version number, caps setting, and hex mode all revert to the way they existed before the data change. The data change is also undone.

Note: UNDO is not accepted if any line commands or data changes are also specified since it would be unclear what is to be undone.

To undo the last changes:

1. Type on the Command line:
 Command ==> UNDO
2. Press Enter.

Note: UNDO is reset by SAVE. Once you save your data for the current edit session, you can no longer recover any interactions made before the data was saved.

Failures in recovery processing due to I/O errors no longer terminate the UNDO function if SETUNDO STORAGE is active. When UNDO is processed, the editor scrolls the data all the way to the left.

UNDO

See "Undoing Edit Interactions" on page 74 for more information.

Example

You are editing the member shown in Figure 155 and decide to delete all of the lines. You have type the block form of the D (DELETE) command in the line command area.

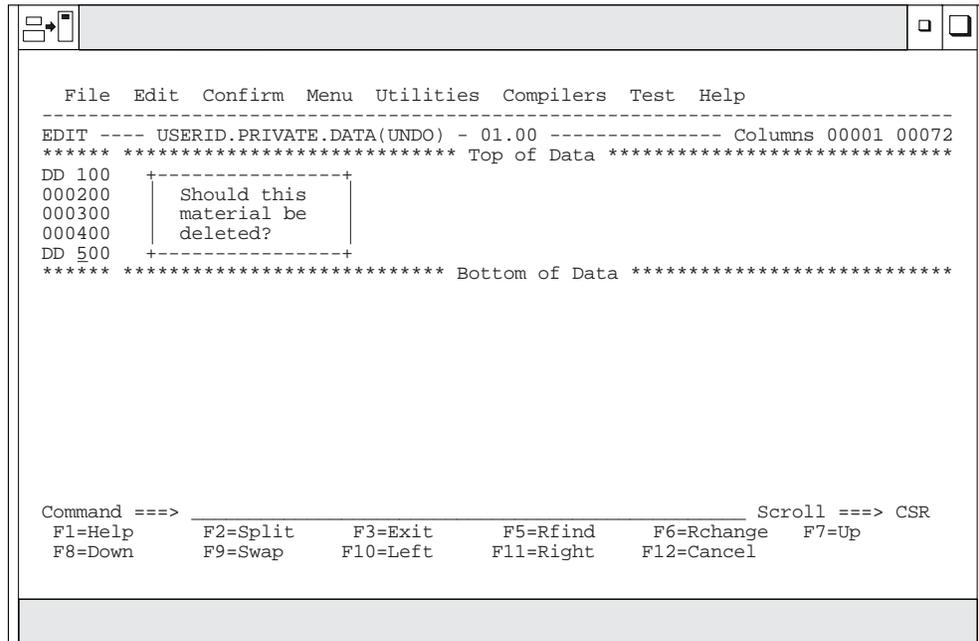


Figure 155. Member Before Lines Are Deleted

Figure 156 shows the member after the lines have been deleted. However, you have changed your mind and want to put the lines back again. Therefore, type UNDO on the Command line.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT --- USERID.PRIVATE.DATA(UNDO) - 01.01 ----- Columns 00001 00072
***** ***** Top of Data *****
***** ***** Bottom of Data *****

Command ==> undo_____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 156. Member After Lines Are Deleted

Figure 157 shows the member after UNDO has been entered and the deleted lines have been restored.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT --- USERID.PRIVATE.DATA(UNDO) - 01.00 ----- Columns 00001 00072
***** ***** Top of Data *****
000100  +-----+
000200  |      Should this      |
000300  |      material be      |
000400  |      deleted?      |
000500  +-----+
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help      F2=Split    F3=Exit     F5=Rfind    F6=Rchange  F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel

```

Figure 157. Member After Lines Have Been Restored

UNNUMBER—Remove Sequence Numbers

The UNNUMBER primary command sets all sequence fields to blanks, turns off number mode, and positions the data so that column 1 is the first column displayed.

UNNUMBER

Syntax

UNNUMBER

Description

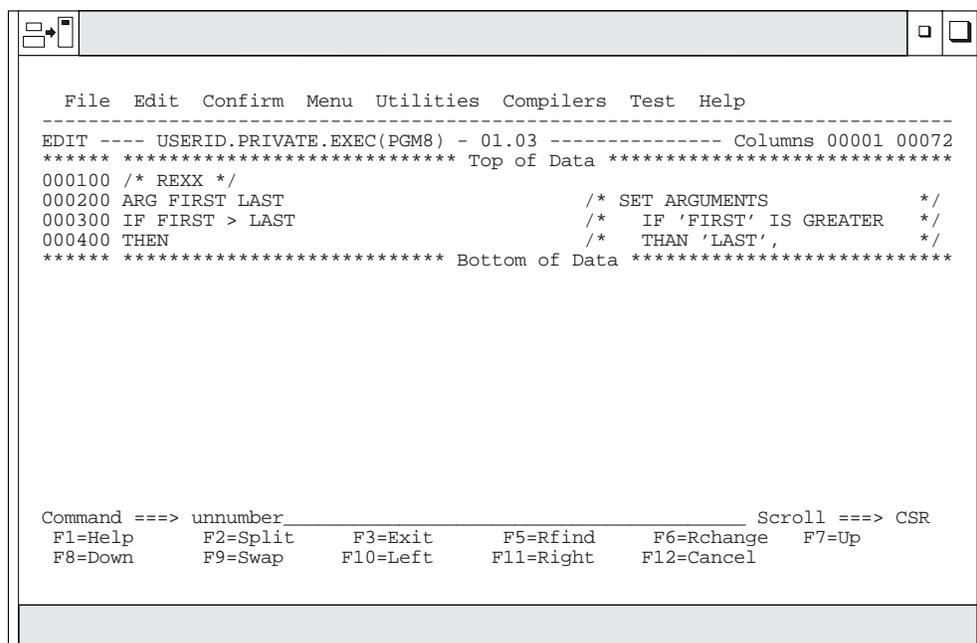
UNNUMBER is valid only when number mode is also on. The standard sequence field, the COBOL sequence field, or both, are blanked out. If you alter or delete sequence numbers and enter UNNUMBER on the Command line at the same time, the editor issues the message Some input data ignored and discards the data you typed over the sequence numbers.

To set all sequence fields to blanks, turn number mode off, and position the panel so that column 1 is the first column to appear:

Command ==> UNNUMBER

Example

You are editing the member in Figure 158 and you want to turn off the sequence numbers. Enter UNNUMBER on the Command line.



```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> unnumber_____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 158. Member Before Lines Are Unnumbered

Figure 159 shows the member after the sequence numbers have been turned off.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.03 ----- Columns 00001 00072
***** ***** Top of Data *****
000001 /* REXX */
000002 ARG FIRST LAST /* SET ARGUMENTS */
000003 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000004 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> _____ Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 159. Member After Lines Are Unnumbered

VERSION—Control the Version Number

The VERSION primary command allows you to change the version number assigned to a member of an ISPF library.

Syntax

```
VERSION num
```

num The version number. It can be any number from 1 to 99.

Description

To change the version number of the member that you are editing:

1. On the Command line, type:

```
Command ==> VERSION num
```

where num is the new version number.

2. Press Enter.

See “Version and Modification Level Numbers” on page 31, for more information about version numbers.

Example

Version and modification level numbers are shown on the first line of an edit data display in the format VV.MM, where VV is the version number and MM is the modification level number.

VERSION

You are editing the member shown in Figure 160 and you want to change the version number from 01 to 02. Enter VERSION on the Command line.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 01.00 ---- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==> version
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 160. Member Before Version Number is Changed

Figure 161 shows the member with the changed version number.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT ---- USERID.PRIVATE.EXEC(PGM8) - 02.00 ---- Columns 00001 00072
***** ***** Top of Data *****
000100 /* REXX */
000200 ARG FIRST LAST /* SET ARGUMENTS */
000300 IF FIRST > LAST /* IF 'FIRST' IS GREATER */
000400 THEN /* THAN 'LAST', */
***** ***** Bottom of Data *****

Command ==>
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
```

Figure 161. Member After Version Number is Changed

VIEW—View from within an Edit Session

The VIEW primary command allows you to view a sequential data set or partitioned data set member during your current edit session.

Syntax

VIEW [member]

member

A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

To view a data set or member during your current edit session:

1. On the Command line, type:

```
Command ==> VIEW member
```

Here, member represents the name of the partitioned data set you are editing. The member operand is optional.

2. Press Enter. If you specified a member name, the current library concatenation sequence finds the member. The member is displayed for viewing. If you do not specify a member name, the View Command Entry panel, which is similar to the regular View Entry panel, appears. You can enter the name of any sequential or partitioned data set to which you have access. When you press Enter, the data set or member is displayed for viewing. The editor suspends your initial edit session until the view session is complete. Viewing sessions can be nested until you run out of storage.
3. To exit from the view session, enter the END command. The current edit session resumes.

Example

To view member YYY of the current library concatenation:

1. On the Command line, type:

```
Command ==> VIEW YYY
```

2. Press enter.

VIEW

Chapter 11. Edit Macro Commands and Assignment Statements

This chapter documents general-use programming interfaces and associated guidance information.

This chapter describes the edit macro commands and assignment statements available for the PDF component. Edit macro commands and assignment statements must be included in edit macros that you create.

Macro commands and assignment statements cannot be entered individually from the edit command line. However, once you have created an edit macro, you can use the macro just like any other Edit primary command. You can run an edit macro by:

- Typing the macro name on the Command line and pressing Enter
- Pressing a function key to which the macro has been assigned, if any.

Note: Edit macro commands should not be confused with TSO commands. Although both are programs, edit macros must not be prefixed with the word 'TSO' when they are invoked.

All edit macros must have an ISREDIT MACRO statement as the first edit command. For more information see "Macro Command Syntax" on page 362.

Each command description in this book consists of the following information:

Syntax

A syntax diagram for coding the macro command, including a description of any required or optional operands.

Description

An explanation of the function and operation of the command. This description also refers to other commands that can be used with this command.

Return Codes

A description of codes returned by the macro command. For all commands, a return code of 20 or higher implies a severe error. See "Return Codes from User-Written Edit Macros" on page 118 and "Return Codes from PDF Edit Macro Commands" on page 119 for more information.

Examples

Sample usage of the macro command.

Edit Macro Command Notation Conventions

The descriptions of the syntax of the the PDF component macro commands and assignment statements use the following notation conventions:

Uppercase

Uppercase commands or operands must be spelled as shown (in either uppercase or lowercase).

Edit Macro Command Notation Conventions

Lowercase

Lowercase operands are variables; substitute your own values.

Underscore

Underscored operands are the system defaults.

Brackets ([])

Operands in brackets are optional.

Stacked operands

Stacked operands show two or more operands from which you can select. If you do not choose any, the PDF component uses the default operand.

Braces ({ })

Braces show two or more operands from which you must select one.

OR (|)

The OR (|) symbol shows two or more operands from which you must select one.

Edit Macro Command Summary

The following table summarizes the edit macro commands. See the complete description of the commands on the referenced page.

Table 6. Summary of the Macro Commands

Command Syntax	page	Description
ISREDIT AUTOLIST [ON] [OFF] ISREDIT (varname) = AUTOLIST ISREDIT AUTOLIST = [ON] [OFF]	“AUTOLIST—Set or Query Autolist Mode” on page 307	Sets the current autolist mode or retrieves the value and places it in a variable.
ISREDIT AUTONUM [ON] [OFF] ISREDIT (varname) = AUTONUM ISREDIT AUTONUM = [ON] [OFF]	“AUTONUM—Set or Query Autonum Mode” on page 308	Sets the current autonum mode or retrieves the value and places it in a variable.
ISREDIT AUTOSAVE [ON] [OFF PROMPT] [OFF NOPROMPT] ISREDIT (var1,var2) = AUTOSAVE ISREDIT AUTOSAVE = [ON] [OFF PROMPT] [OFF NOPROMPT]	“AUTOSAVE—Set or Query Autosave Mode” on page 309	Sets the current autosave mode or retrieves the value and places it in a variable.
ISREDIT (varname) = BLKSIZE	“BLKSIZE—Query the Block Size” on page 311	Returns the block size of the data set being edited in a specified variable.
ISREDIT BOUNDS [left-col right-col] ISREDIT (var1,var2) = BOUNDS ISREDIT BOUNDS = [left-col right-col]	“BOUNDS—Set or Query the Edit Boundaries” on page 311	Sets the left and right boundaries or retrieves the values and places them in variables.
ISREDIT BROWSE member	“BROWSE—Browse from within an Edit Session” on page 313	Browses another member in the data set.
ISREDIT BUILTIN cmdname	“BUILTIN—Process a Built-In Command” on page 314	Processes a built-in command even if a macro or macro statement with the same name has been defined.

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
ISREDIT CANCEL	"CANCEL—Cancel Edit Changes" on page 315	Ends the edit session without saving any changes.
ISREDIT CAPS [ON] [OFF] ISREDIT (varname) = CAPS ISREDIT CAPS = [ON] [OFF]	"CAPS—Set or Query Caps Mode" on page 315	Sets caps mode.
ISREDIT CHANGE string-1 string-2 [label-range] <small>[NEXT] [CHARS] [X] [col-1 [col-2]] [ALL] [PREFIX] [NX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]</small>	"CHANGE—Change a Search String" on page 316	Changes a data string to another string.
ISREDIT (var1,var2) = CHANGE_COUNTS	"CHANGE_COUNTS—Query Change Counts" on page 319	Retrieves the values set by the most recently processed CHANGE command and places these values in variables.
ISREDIT COMPARE {dsname} {NEXT} {SAVE} {SYSIN} {EXCLUDE}	"COMPARE—Edit Compare" on page 224	Compares a library member or data set with the data being edited.
ISREDIT COPY member {AFTER } lptr [linenum-range] (member) {BEFORE} dataset name	"COPY—Copy Data" on page 322	Copies a member of the library into the member being edited.
ISREDIT CREATE member lptr-range (member) {range } dataset(member) {range }	"CREATE—Create a Data Set Member" on page 323	Creates a new member from the data that is being edited.
ISREDIT (var1,var2) = CTL_LIBRARY	"CTL_LIBRARY—Query Controlled Library Status" on page 324	Retrieves the status of a controlled library and places the status in variables.
ISREDIT (var1,var2) = CURSOR ISREDIT CURSOR = lptr [col]	"CURSOR—Set or Query the Cursor Position" on page 326	Sets the relative line and column number of the cursor or retrieves the values and places them in variables.
ISREDIT CUT [lptr-range] [DEFAULT clipboard] <small>[name]</small> [REPLACE]	"CUT—Cut and Save Lines" on page 328	Cut and save lines.
ISREDIT (varname) = DATA_CHANGED	"DATA_CHANGED—Query the Data Changed Status" on page 329	Retrieves the data changed status and places it in a variable.
ISREDIT (varname) = DATA_WIDTH	"DATA_WIDTH—Query Data Width" on page 330	Retrieves the logical data width and places it in a variable.
ISREDIT (varname) = DATAID	"DATAID—Query Data ID" on page 331	Retrieves the data ID for the data set being edited and places it in a variable.
ISREDIT (varname) = DATASET	"DATASET—Query the Current Data Set Name" on page 331	Retrieves the name of a data set and places it in a variable.
ISREDIT DEFINE name {MACRO CMD } {MACRO PGM } {ALIAS name-2} {NOP } {RESET } {DISABLED }	"DEFINE—Define a Name" on page 332	<ul style="list-style-type: none"> • Assigns an alias to a macro or built-in command. • Disables the use of a macro or built-in command. • Identifies a macro that replaces a built-in command of the same name. • Identifies programs that are edit macros.

Edit Macro Command Summary

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
ISREDIT DELETE { ALL X NX [lptr-range] } { [ALL] X NX lptr-range } { lptr { lptr-range }	"DELETE—Delete Lines" on page 334	Deletes lines from the data.
ISREDIT (var1,var2) = DISPLAY_COLS	"DISPLAY_COLS—Query Display Columns" on page 335	Retrieves the column numbers for the first and last data columns on the panel and places them in variables.
ISREDIT (var1,var2) = DISPLAY_LINES	"DISPLAY_LINES—Query Display Lines" on page 336	Retrieves the relative line numbers of the first and last data lines that would appear if the macro ended and places them in variables.
ISREDIT DOWN amt	"DOWN—Scroll Down" on page 336	Scrolls data down from the current panel position.
ISREDIT EDIT member	"EDIT—Edit from within an Edit Session" on page 338	Edits another member in the data set (recursive editing).
ISREDIT END	"END—End the Edit Session" on page 338	Ends the edit session.
ISREDIT EXCLUDE string [label-range] [NEXT] [CHARS] [col-1 [col-2]] [ALL] [PREFIX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]	"EXCLUDE—Exclude Lines from the Display" on page 339	Marks lines in the data that should not appear.
ISREDIT (var1,var2) = EXCLUDE_COUNTS	"EXCLUDE_COUNTS—Query Exclude Counts" on page 341	Retrieves the values set by the most recently processed EXCLUDE command and places them in variables.
ISREDIT FIND string [label-range] [NEXT] [CHARS] [X] [col-1 [col-2]] [ALL] [PREFIX] [NX] [FIRST] [SUFFIX] [LAST] [WORD] [PREV]	"FIND—Find a Search String" on page 342	Locates a search string. It is recommended that you do not use FIND in a macro because any excluded data string found is shown on the panel. Use SEEK to perform the identical function without changing the lines' exclude status.
ISREDIT (var1,var2) = FIND_COUNTS	"FIND_COUNTS—Query Find Counts" on page 344	Retrieves values set by the most recently processed FIND or RFIND command and places them in variables.
ISREDIT FLIP [label-range]	"FLIP—Reverse Exclude Status of Lines" on page 345	Reverses the exclude status of a specified group of lines in a file or of all the lines in a file.
ISREDIT (var1,var2) = FLOW_COUNTS	"FLOW_COUNTS—Query Flow Counts" on page 345	Retrieves values set by the most recently processed TFLOW command and places them in variables.
ISREDIT HEX [ON DATA] [ON VERT] [OFF] ISREDIT (var1,var2) = HEX ISREDIT HEX = [ON DATA] [ON VERT] [OFF]	"HEX—Set or Query Hexadecimal Mode" on page 346	Sets the hexadecimal mode or retrieves the value and places it in a variable.

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
<pre> ISREDIT HILITE [ON] [AUTO] [RESET] [PAREN] [FIND] [CURSOR] [SEARCH] [DISABLED] [OFF] [DEFAULT] [LOGIC] [OTHER] [IFLOGIC] [ASM] [DOLOGIC] [BOOK] [NOLOGIC] [C] [COBOL] [DTL] [JCL] [PANEL] [PASCAL] [PLI] [REXX] [SKEL] </pre>	<p>“HILITE—Enhanced Edit Coloring” on page 347</p>	<p>Highlights, in user-specified colors, numerous language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFINDD or RCHANGE request. Can also be used to set default colors for the data area in non-program files and for any characters typed since the previous Enter or function key entry.</p>
<pre> ISREDIT IMACRO {name NONE} ISREDIT (varname) = IMACRO ISREDIT IMACRO = {name NONE} </pre>	<p>“IMACRO—Set or Query an Initial Macro” on page 351</p>	<p>Sets or retrieves the value for the initial macro in the profile and places it in a variable.</p>
<pre> ISREDIT INSERT lptr [numlines] </pre>	<p>“INSERT—Prepare Display for Data Insertion” on page 351</p>	<p>Displays one or more lines for data entry.</p>
<pre> ISREDIT (var1,var2) = LABEL lptr ISREDIT LABEL lptr = labelname [level] </pre>	<p>“LABEL—Set or Query a Line Label” on page 352</p>	<p>Sets or retrieves the values for the label on the specified line and places them in variables.</p>
<pre> ISREDIT LEFT amt </pre>	<p>“LEFT—Scroll Left” on page 353</p>	<p>Scrolls data left from the current panel position.</p>
<pre> ISREDIT LEVEL num ISREDIT (varname) = LEVEL ISREDIT LEVEL = num </pre>	<p>“LEVEL—Set or Query the Modification Level Number” on page 354</p>	<p>Sets the modification level number or retrieves the value and places it in a variable.</p>
<pre> ISREDIT (varname) = LINE lptr ISREDIT LINE lptr = data </pre>	<p>“LINE—Set or Query a Line from the Data Set” on page 355</p>	<p>Sets or retrieves the data from the data line and places it in a variable.</p>
<pre> ISREDIT LINE_AFTER lptr = [DATA LINE] data [INFO LINE] [MSG LINE] [NOTE LINE] </pre>	<p>“LINE_AFTER—Add a Line to the Current Data Set” on page 356</p>	<p>Adds a line after the specified line.</p>
<pre> ISREDIT LINE_BEFORE lptr = [DATA LINE] data [INFO LINE] [MSG LINE] [NOTE LINE] </pre>	<p>“LINE_BEFORE—Add a Line to the Current Data Set” on page 358</p>	<p>Adds a line before the specified line.</p>
<pre> ISREDIT (varname) = LINENUM label </pre>	<p>“LINENUM—Query the Line Number of a Labeled Line” on page 359</p>	<p>Retrieves the relative line number of a specified label and places it in a variable.</p>
<pre> ISREDIT LOCATE lptr ISREDIT LOCATE [FIRST] {CHANGE } [lptr-range] [LAST] {COMMAND } [NEXT] {ERROR } [PREV] {EXCLUDED} {LABEL } {SPECIAL } </pre>	<p>“LOCATE—Locate a Line” on page 360</p>	<p>Locates a line.</p>
<pre> ISREDIT (varname) = LRECL </pre>	<p>“LRECL—Query the Logical Record Length” on page 362</p>	<p>Returns the logical record length of the data being edited in a variable.</p>

Edit Macro Command Summary

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
ISREDIT MACRO [(var1 [,var2,...])] [PROCESS] [NOPROCESS]	"MACRO—Identify an Edit Macro" on page 362	Identifies a command as a macro. MACRO is required for all macros and must be the first command in a CLIST or REXX EXEC macro that is not a CLIST or REXX EXEC statement or the first edit command in a program macro.
ISREDIT (varname) = MACRO_LEVEL	"MACRO_LEVEL—Query the Macro Nesting Level" on page 364	Retrieves the nesting level of the macro being run and places it in a variable.
ISREDIT (varname) = MASKLINE ISREDIT MASKLINE = data	"MASKLINE—Set or Query the Mask Line" on page 364	Sets or retrieves the value of the mask line, which controls the display formatting of input.
ISREDIT (varname) = MEMBER	"MEMBER—Query the Current Member Name" on page 365	Retrieves the name of the ISPF library member currently being edited and places it in a variable.
ISREDIT MEND	"MEND—End a Macro in the Batch Environment" on page 366	Ends a macro that is running in the batch environment.
ISREDIT MODEL model-name [qualifier] {AFTER } {BEFORE} lptr [NOTES][NONOTES] ISREDIT MODEL CLASS class-name	"MODEL—Copy a Model into the Current Data Set" on page 366	Copies a specified dialog development model before or after a specified line.
ISREDIT MOVE member {AFTER } lptr (member){BEFORE} data set name data.set.name(member)	"MOVE— Move a Data Set or a Data Set Member" on page 368	Moves a member of a data set and places it after or before the line specified.
ISREDIT NONUMBER	"NONUMBER—Turn Off Number Mode" on page 369	Turns off number mode.
ISREDIT NOTES [ON] [OFF] ISREDIT (varname) = NOTES ISREDIT NOTES = [ON] [OFF]	"NOTES—Set or Query Note Mode" on page 370	Sets the current note mode or retrieves the value and places it in a variable.
ISREDIT NULLS [ON STD] [ON ALL] [OFF] ISREDIT (var1,var2) = NULLS ISREDIT NULLS = [ON STD] [ON ALL] [OFF]	"NULLS—Set or Query Nulls Mode" on page 370	Sets the current nulls mode or retrieves the value and places it in a variable.
ISREDIT NUMBER [ON] [STD] [DISPLAY] [OFF] [COBOL] [STD COBOL] [NOSTD] [NOCOBOL] [NOSTD NOCOBOL] ISREDIT (var1,var2) = NUMBER ISREDIT NUMBER = [ON] [STD] [DISPLAY] [OFF] [COBOL] [STD COBOL] [NOSTD] [NOCOBOL] [NOSTD NOCOBOL]	"NUMBER—Set or Query Number Mode" on page 372	Sets the current number mode or retrieves the value and places it in a variable.

Edit Macro Command Summary

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
ISREDIT PACK [ON] [OFF] ISREDIT (varname) = PACK ISREDIT PACK = [ON] [OFF]	“PACK—Set or Query Pack Mode” on page 374	Sets the current pack mode or retrieves the value and places it in a variable.
ISREDIT PASTE [AFTER] lptr [clipboardname] [BEFORE] [KEEP]	“PASTE—Move or Copy Lines from Clipboard” on page 375	Move or copy lines from a clipboard.
ISREDIT PRESERVE [ON] [OFF] ISREDIT (varname) = PRESERVE ISREDIT PRESERVE = [ON] [OFF]	“PRESERVE” on page 376	Sets the current pack mode or retrieves the value and places it in a variable.
ISREDIT PROCESS [DEST] [RANGE cmd1 [cmd2]]	“PROCESS—Process Line Commands” on page 377	Controls when the line commands or data changes typed at the keyboard are to be processed.
ISREDIT PROFILE [name] [number] ISREDIT PROFILE {LOCK UNLOCK} ISREDIT RESET ISREDIT (var1,var2) = PROFILE	“PROFILE—Set or Query the Current Profile” on page 379	Allows you to view or change the default modes for your edit session.
ISREDIT (varname) = RANGE_CMD	“RANGE_CMD—Query a Command That You Entered” on page 381	Identifies the name of a line command typed at the keyboard and processed by a macro.
ISREDIT RCHANGE	“RCHANGE—Repeat a Change” on page 381	Repeats the most recently processed CHANGE command.
ISREDIT (varname) = RECFM	“RECFM—Query the Record Format” on page 382	Retrieves the record format of the data set being edited and places the value in variables.
ISREDIT RECOVERY [ON] [OFF [WARN]] [OFF NOWARN] ISREDIT (varname) = RECOVERY ISREDIT RECOVERY = [ON [SUSP]] [OFF [WARN]] [OFF NOWARN]	“RECOVERY—Set or Query Recovery Mode” on page 383	Sets the recovery mode or retrieves the value and places it in a variable.
ISREDIT RENUM [ON] [STD] [DISPLAY] [COBOL] [STD COBOL]	“RENUM—Renum Data Set Lines” on page 384	Sets number mode on and rennumbers all data lines.
ISREDIT REPLACE member lptr-range ISREDIT REPLACE (member) lptr-range ISREDIT REPLACE dataset lptr-range ISREDIT REPLACE dataset(member) lptr-range	“REPLACE—Replace a Data Set Member” on page 386	Replaces the specified member in the library with the data specified in the member being edited.
ISREDIT RESET [CHANGE] [lptr-range] [COMMAND] [ERROR] [EXCLUDED] [FIND] [LABEL] [SPECIAL]	“RESET—Reset the Data Display” on page 386	Restores the status of lines or deletes special temporary lines.

Edit Macro Command Summary

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description															
ISREDIT RFIND	"RFIND—Repeat Find" on page 388	Locates the data string defined by the most recently processed SEEK, FIND, or CHANGE command, or excludes a line that contains the data string from the previous EXCLUDE command.															
ISREDIT RIGHT amt	"RIGHT—Scroll Right" on page 389	Scrolls data to the right of the current panel position.															
ISREDIT RMACRO {name NONE} ISREDIT (varname) = RMACRO ISREDIT RMACRO = {name NONE}	"RMACRO—Set or Query the Recovery Macro" on page 390	Sets or retrieves the name of the macro set in this edit session.															
ISREDIT SAVE	"SAVE—Save the Current Data" on page 390	Saves the data.															
ISREDIT SCAN [ON] [OFF] ISREDIT (varname) = SCAN ISREDIT SCAN = [ON] [OFF]	"SCAN—Set Command Scan Mode" on page 392	Sets the current value of scan mode (for variable substitution) or retrieves the value and places it in a variable.															
ISREDIT SEEK string [label-range] <table style="display: inline-table; vertical-align: middle; border: none;"><tr><td>[NEXT]</td><td>[CHARS]</td><td>[X]</td></tr><tr><td>[ALL]</td><td>[PREFIX]</td><td>[NX]</td></tr><tr><td>[FIRST]</td><td>[SUFFIX]</td><td></td></tr><tr><td>[LAST]</td><td>[WORD]</td><td></td></tr><tr><td>[PREV]</td><td></td><td></td></tr></table> [col-1 [col-2]]	[NEXT]	[CHARS]	[X]	[ALL]	[PREFIX]	[NX]	[FIRST]	[SUFFIX]		[LAST]	[WORD]		[PREV]			"SEEK—Seek a Data String, Positioning the Cursor" on page 393	Finds one or more occurrences of a data string. SEEK is similar to FIND; however, when a string is found, the exclude status of the line is not affected.
[NEXT]	[CHARS]	[X]															
[ALL]	[PREFIX]	[NX]															
[FIRST]	[SUFFIX]																
[LAST]	[WORD]																
[PREV]																	
ISREDIT (var1,var2) = SEEK_COUNTS	"SEEK_COUNTS—Query Seek Counts" on page 395	Retrieves the values set by the most recently processed SEEK command and places them in variables.															
ISREDIT (varname) = SETUNDO ISREDIT SETUNDO = [STORAGE] [RECOVER] [ON] [OFF]	"SETUNDO—Set UNDO Mode" on page 396	Sets the UNDO function on or disables the UNDO function, and retrieves the current UNDO status.															
ISREDIT SHIFT (lptr [n] [2]	"SHIFT (—Shift Columns Left" on page 398	Moves columns of data to the left.															
ISREDIT SHIFT) lptr [n] [2]	"SHIFT)—Shift Columns Right" on page 399	Moves columns of data to the right.															
ISREDIT SHIFT < lptr [n] [2]	"SHIFT <—Shift Data Left" on page 399	Moves data to the left.															
ISREDIT SHIFT > lptr [n] [2]	"SHIFT >—Shift Data Right" on page 400	Moves data to the right.															
ISREDIT SORT [label-range] <table style="display: inline-table; vertical-align: middle; border: none;"><tr><td>[X]</td></tr><tr><td>[NX]</td></tr></table> [sort-field1 ... sort-field5]	[X]	[NX]	"SORT—Sort Data" on page 401	Puts data in a specified order.													
[X]																	
[NX]																	
ISREDIT STATS [ON] [OFF] ISREDIT (varname) = STATS ISREDIT STATS = [ON] [OFF]	"STATS—Set or Query Stats Mode" on page 403	Sets the current stats mode or retrieves the value and places it in a variable.															
ISREDIT SUBMIT [lptr-range]	"SUBMIT—Submit Data for Batch Processing" on page 404	Submits data that is to be processed as a batch job.															

Table 6. Summary of the Macro Commands (continued)

Command Syntax	page	Description
ISREDIT TABS [ON] [STD] [OFF] [ALL] [tab-character] ISREDIT (var1,var2) = TABS ISREDIT TABS = [ON] [STD] [OFF] [ALL] [tab-character]	“TABS—Set or Query Tabs Mode” on page 404	Sets the tabs mode or retrieves the mode and places it in a variable.
ISREDIT (varname) = TABSLINE ISREDIT TABSLINE = data	“TABSLINE—Set or Query Tabs Line” on page 406	Sets the tabs line or retrieves the tabs line and places it in a variable.
ISREDIT TENTER lptr [numlines]	“TENTER—Set Up Panel for Text Entry” on page 407	Prepares the panel for power typing.
ISREDIT TFLOW lptr [col]	“TFLOW—Text Flow a Paragraph” on page 409	Restructures paragraphs.
ISREDIT TSPLIT [lptr col]	“TSPLIT—Text Split a Line” on page 409	Divides a line so data can be added.
ISREDIT UNNUMBER	“UNNUMBER—Remove Sequence Numbers” on page 410	Removes the numbers from the data set and turns number mode off.
ISREDIT UP amt	“UP—Scroll Up” on page 411	Scrolls data up from the current panel position.
ISREDIT (varname) = USER_STATE ISREDIT USER_STATE = (varname)	“USER_STATE—Save or Restore User State” on page 412	Saves or restores the state of the edit profile values, FIND and CHANGE values, and panel and cursor values.
ISREDIT (varname) = VERSION ISREDIT VERSION = num ISREDIT VERSION num	“VERSION—Set or Query Version Number” on page 413	Sets the version number or retrieves the value and places it in a variable.
ISREDIT VIEW member	“VIEW—View from within an Edit Session” on page 414	Views another member in the data set.
ISREDIT (varname) = XSTATUS lptr ISREDIT XSTATUS lptr = X NX	“XSTATUS—Set or Query Exclude Status of a Line” on page 415	Sets the exclude status of the specified data line or retrieves the value and places it in a variable.

AUTOLIST—Set or Query Autolist Mode

The AUTOLIST macro command sets autolist mode, which controls the automatic printing of data to the ISPF list data set.

The AUTOLIST assignment statement either sets autolist mode or retrieves the current setting of autolist mode and places it in a variable.

Autolist mode is saved in the edit profile.

Macro Command Syntax

```
ISREDIT AUTOLIST [ON ]  
                  [OFF]
```

ON Specifies that when you end an edit session and save changed data, the editor generates a source listing in the ISPF list data set for eventual printing.

AUTOLIST

OFF Does not generate a source listing.

Assignment Statement Syntax

```
ISREDIT (varname) = AUTOLIST
ISREDIT AUTOLIST = [ON ]
                  [OFF]
```

varname

The name of a variable that contains the setting of autolist mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Examples

To turn autolist mode on:

```
ISREDIT AUTOLIST ON
```

or

```
ISREDIT AUTOLIST = ON
```

To turn autolist mode off:

```
ISREDIT AUTOLIST OFF
```

or

```
ISREDIT AUTOLIST = OFF
```

AUTONUM—Set or Query Autonom Mode

The AUTONUM macro command sets autonom mode, which controls the automatic renumbering of data when it is saved.

The AUTONUM assignment statement either sets autonom mode or retrieves the current setting of autonom mode and places it in a variable.

Macro Command Syntax

```
ISREDIT AUTONUM [ON ]
                [OFF]
```

ON Turns on automatic renumbering. When number mode is also on, the data is automatically renumbered when it is saved.

OFF Turns off automatic renumbering. Data is not renumbered.

Assignment Statement Syntax

```
ISREDIT (varname) = AUTONUM
ISREDIT AUTONUM = [ON ]
                  [OFF]
```

varname

The name of a variable containing the setting of autonum mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Description

When number mode is on, the first line of a data set or member is normally line number 000100, the second number is 000200, and so on. However, as lines are inserted and deleted, the increments between line numbers can change.

For example, you might think that when a line is inserted between 000100 and 000200, line 000200 would be given the number 000300 and the new line would become 000200. Instead, the existing lines retain their numbers and the new line is given line number 000110.

Therefore, if the original line number increments are important to you, AUTONUM rennumbers your lines automatically so that the original increments are maintained.

Autonum mode is saved in the edit profile.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Examples

To turn autonum mode on:

```
ISREDIT AUTONUM ON
```

or

```
ISREDIT AUTONUM = ON
```

To turn autonum mode off:

```
ISREDIT AUTONUM OFF
```

or

```
ISREDIT AUTONUM = OFF
```

AUTOSAVE—Set or Query Autosave Mode

The AUTOSAVE macro command sets autosave mode, which controls whether changed data is saved when you issue the END command.

The AUTOSAVE assignment statement either sets autosave mode, or retrieves the current setting of autosave mode and places it in variables.

AUTOSAVE

Macro Command Syntax

```
ISREDIT AUTOSAVE [ON      ]  
                  [OFF PROMPT ]  
                  [OFF NOPROMPT]
```

ON Turns autosave mode on. When you enter END, any changed data is saved.

OFF PROMPT

Turns autosave mode off with the PROMPT operand. You are notified that changes have been made and to use either SAVE (followed by END) or CANCEL. If you specify only the PROMPT keyword, OFF is implied.

OFF NOPROMPT

Turns autosave mode off with the NOPROMPT operand. You are not notified and the data is not saved when you issue an END command. END becomes an equivalent to CANCEL. Use the NOPROMPT operand with caution.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = AUTOSAVE  
ISREDIT AUTOSAVE = [ON      ]  
                   [OFF PROMPT ]  
                   [OFF NOPROMPT]
```

var1 The name of a variable to contain the setting of autosave mode, either ON or OFF.

var2 The name of a variable to contain the prompt value, PROMPT or NOPROMPT.

ON Same as macro command syntax.

OFF PROMPT

Same as macro command syntax.

OFF NOPROMPT

Same as macro command syntax.

Description

Data is considered changed if you have operated on it in any way that could cause a change. Shifting a blank line or changing a name to the same name does not actually alter the data, but the editor considers this data changed. When you enter SAVE, the editor resets the change status.

Autosave mode, along with the PROMPT operand, is saved in the edit profile.

See the DATA_CHANGED, CANCEL, and END macro commands, and the CANCEL and END primary commands for more information on saving data.

Return Codes

The following return codes can be issued:

0	Normal completion
4	OFF NOPROMPT specified
20	Severe error.

Examples

To turn autosave mode on:

```
ISREDIT AUTOSAVE ON
```

or

```
ISREDIT AUTOSAVE = ON
```

To turn autosave mode off and have the editor prompt you to use the SAVE or CANCEL command:

```
ISREDIT AUTOSAVE OFF
```

or

```
ISREDIT AUTOSAVE = OFF
```

To turn autosave mode off and not have the editor prompt you to use SAVE or CANCEL::

```
ISREDIT AUTOSAVE OFF NOPROMPT
```

or

```
ISREDIT AUTOSAVE = OFF NOPROMPT
```

BLKSIZE—Query the Block Size

The BLKSIZE assignment statement returns the block size of the data being edited in a specified variable.

Assignment Statement Syntax

```
ISREDIT (varname) = BLKSIZE
```

varname

The name of a variable to contain the block size of the data being edited. The block size is a 6-digit value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Syntax Error
20	Severe error.

Example

To find the block size and continue processing if the block size is greater than 800:

```
ISREDIT (BSIZE) = BLKSIZE
IF &BSIZE > 000800 THEN -
...

```

BOUNDS—Set or Query the Edit Boundaries

The BOUNDS macro command sets the left and right boundaries and saves them in the edit profile.

BOUNDS

The BOUNDS assignment statement sets or retrieves the left and right boundaries and places the values in variables.

Macro Command Syntax

```
ISREDIT BOUNDS [left-col right-col]
```

left-col

The left boundary column to be set.

right-col

The right boundary column to be set.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = BOUNDS
```

```
ISREDIT BOUNDS = [left-col right-col]
```

var1 A variable containing the left boundary. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, A length of 3 or 4 is allowed in cases where no data loss will occur.

var2 A variable containing the right boundary. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, A length of 3 or 4 is allowed in cases where no data loss will occur.

left-col

Same as macro command syntax.

right-col

Same as macro command syntax.

Description

The BOUNDS macro command provides an alternative to setting the boundaries with the BOUNDS line command or primary command; the effect on the member or data set is the same.

The column numbers are always data column numbers. Thus, for a variable format data set with number mode on, data column 1 is column 9 in the record.

See "Edit Boundaries" on page 28 for more information, including tables that show commands affected by bounds settings and default bounds settings for various types of data sets.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Right boundary greater than default, default right boundary used
12	Invalid boundaries specified
20	Severe error.

Examples

To set the boundaries to their default values, type:

```
ISREDIT BOUNDS
```

To set one boundary while leaving the other value unchanged, type an asterisk (*) for the boundary to be unchanged. For example, to set the left boundary from the variable &LEFT, and leave the right boundary unchanged, type:

```
ISREDIT BOUNDS &LEFT *
```

To set the left boundary to 1, leaving the right boundary unchanged:

```
ISREDIT BOUNDS = 1 *
```

To save the value of the left boundary in the variable &LEFT:

```
ISREDIT (LEFT) = BOUNDS
```

To save the value of the right boundary in the variable &RIGHT:

```
ISREDIT (,RIGHT) = BOUNDS
```

To evaluate numbers for bounds when NUMBER COBOL is on, or NUMBER is on for a variable blocked data set:

```
/* Rexx - Set physical bounds in a macro.  Input is 2 column      */
/*      numbers and result is bounds set on that physical column */
/*      regardless of number setting.  Bounds will not be set   */
/*      within line number areas.  This sample has minimal     */
/*      error checking.                                         */
Address isredit
'MACRO (LEFT,RIGHT)'          /* Take left and right bounds*/
'(NUMBER,COBOL) = NUMBER'    /* Get number status          */
Parse Var cobol . cobol .    /* Get just left status       */
'(RECFM) = RECFM'            /* Get record format          */
'(DW) = DATA_WIDTH'         /* Get data width             */
If left='' Then left = 1     /* Assume col 1 for left      */
If right='' Then right = dw  /* Assume datawidth for right*/
shift = 0                    /* Assume no left seq numbers*/
If cobol='COBOL' Then        /* If numbered as cobol      */
  shift = 6                  /* Account for sequence num*/
Else If number='ON' & recfm='V' Then /* If numbered variable block*/
  shift = 8                  /* Account for sequence num*/
right = max(1,right - shift) /* Adjust right column       */
right = min(right,dw)       /* Adjust right column       */
left = max(1,left - shift)  /* Adjust left column        */
left = min(left ,dw)        /* Adjust left column        */

'BOUNDS 'min(left,right) max(left,right) /* Issue bounds command */
'PROFILE'
```

BROWSE—Browse from within an Edit Session

The BROWSE macro command allows you to browse a member of the same partitioned data set during your current edit session.

Macro Command Syntax

```
ISREDIT BROWSE member
```

member

A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Your initial edit session is suspended until the browse session is complete.

BROWSE

To exit from the browse session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

For more information on using the BROWSE service, refer to *ISPF Services Guide*

Return Codes

The following return codes can be issued:

0	Normal completion
12	Your error (invalid member name, recovery pending)
20	Severe error.

Examples

To browse the member OLDMEM in your current ISPF library:

```
ISREDIT BROWSE OLDMEM
```

BUILTIN—Process a Built-In Command

The BUILTIN macro command is used within an edit macro to process a built-in edit command, even if a macro or macro statement with the same name has been defined.

Macro Command Syntax

```
ISREDIT BUILTIN cmdname
```

cmdname

The built-in command to be processed.

Description

If you create a macro named MACEND and enter a DEFINE END ALIAS MACEND command, your MACEND macro runs when you enter END. Within the MACEND macro you can perform logic and use a built-in END command to actually end the edit session.

Note that if END is issued in your MACEND macro without being preceded by BUILTIN, the MACEND macro would run again, resulting in an infinite loop.

Return Codes

The following return codes can be issued:

n	Return code from the built-in command
20	Severe error.

Examples

To process the built-in END command:

```
ISREDIT BUILTIN END
```

To process the built-in CHANGE command:

```
ISREDIT BUILTIN CHANGE ALL " " "-"
```

CANCEL—Cancel Edit Changes

The CANCEL macro command ends your edit session without saving any of the changes you have made.

Macro Command Syntax

```
ISREDIT CANCEL
```

Description

CANCEL is especially useful if you have changed the wrong data, or if the changes themselves are incorrect. See the DATA_CHANGED, AUTOSAVE, and END commands for more information about saving data.

Note: If you issue SAVE and later issue CANCEL, the changes you made before issuing SAVE are not canceled.

CANCEL does not cause automatic recording in the ISPF list data set, regardless of the setting of the autolist mode.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Example

To cancel the current edit session:

```
ISREDIT CANCEL
```

CAPS—Set or Query Caps Mode

The CAPS macro command sets caps mode, which controls whether alphabetic data that you type at the terminal is automatically converted to uppercase during edit operations.

The CAPS assignment statement either sets caps mode or retrieves the setting of caps mode and places it in a variable.

Macro Command Syntax

```
ISREDIT CAPS [ON ]  
[OFF]
```

ON Turns caps mode on.

OFF Turns caps mode off.

Assignment Statement Syntax

```
ISREDIT (varname) = CAPS  
ISREDIT CAPS = [ON ]  
[OFF]
```

CAPS

varname

The name of a variable containing the setting of caps mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Description

When the editor retrieves data, it sets the caps mode on if the data contains all uppercase letters, or off if the data contains lowercase letters. The editor displays a message when the caps mode changes.

Caps mode is saved in the edit profile. To override the automatic setting of caps mode, you can include the CAPS command in an initial macro.

Caps mode is normally on for program development work. When caps mode is set to on, any alphabetic data that you type, plus any other alphabetic data that already exists on that line, is converted to uppercase when you press Enter or a function key.

Caps mode is normally off when you edit text documentation. When caps mode is set to off, any alphabetic data that you type remains just as you typed it. If you typed it in uppercase, it stays in uppercase; if you typed it in lowercase, it stays in lowercase. Also, alphabetic data that is already typed on that line is not affected.

CAPS does not apply to DBCS fields in formatted data or to DBCS fields in mixed fields. If you specify CAPS, the DBCS fields remain unchanged. See the LC (lowercase) and UC (uppercase) line commands and the CAPS primary command for more information about changing cases.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Examples

To save the value of caps mode in variable &CAPMODE:

```
ISREDIT (CAPMODE) = CAPS
```

To turn caps mode OFF:

```
ISREDIT CAPS = OFF
```

To set the value of caps mode from variable &CAPMODE:

```
ISREDIT CAPS &CAPMODE
```

CHANGE—Change a Search String

The CHANGE macro command changes one search string into another.

Macro Command Syntax

```
ISREDIT CHANGE string-1 string-2 [label-range] [NEXT ] [CHARS ] [X ] [col-1 [col-2]]
                [ALL ] [PREFIX] [NX]
                [FIRST] [SUFFIX]
                [LAST ] [WORD ]
                [PREV ]
```

string-1

The search string you want to change.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within the &STR() delimiters such as &STR(/*XXX). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

string-2

The string you want to replace *string-1*. The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

label-range

Two labels that identify the range of lines CHANGE searches. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

When using a macro that uses NEXT or PREV with a label-range, be careful concerning cursor placement. If the cursor is currently placed below the label-range, and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

If the cursor is currently placed above the label-range, and the PREV occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string-1*. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of *string-1*.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of *string-1*.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of *string-1*.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of *string-1*.

CHARS

Locates *string-1* anywhere the characters match. CHARS is the default.

PREFIX

Locates *string-1* at the beginning of a word.

SUFFIX

Locates *string-1* at the end of a word.

WORD

Locates *string-1* when it is delimited on both sides by blanks or other non-alphanumeric characters.

CHANGE

- X Scans only lines that are excluded from the display.
- NX Scans only lines that are not excluded from the display.
- col-1 and col-2**
Numbers that identify the columns CHANGE is to search.

Description

CHANGE is often used with FIND, EXCLUDE, and SEEK, and the CHANGE_COUNTS assignment statement.

To change the next occurrence of ME to YOU without specifying any other qualifications, include the following command in an edit macro:

```
ISREDIT CHANGE ME YOU
```

This command changes only the next occurrence of the letters ME to YOU. Since no other qualifications were specified, the letters ME can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In an excluded line or a nonexcluded line
- Anywhere within the current boundaries.

To change the next occurrence of ME to YOU, but only if the letters are uppercase:

```
ISREDIT CHANGE C'ME' YOU
```

This type of change is called a character string change (note the C that precedes the search string) because it changes the next occurrence of the letters ME to YOU only if the letters are found in uppercase. However, since no other qualifications were specified, the change occurs no matter where the letters are found, as outlined in the preceding list.

When you would like to issue CHANGE, but you are unsure of the exclude status of a line, you can use the XSTATUS assignment statement with SEEK. First, find the particular line with SEEK. Then, determine the exclude status with the XSTATUS assignment statement. Use CHANGE to change the string; and finally, reset the exclude status with another XSTATUS assignment statement. For example:

```
ISREDIT SEEK ABC
DO WHILE &LASTCC=0
  ISREDIT (X) = XSTATUS .ZCSR
  ISREDIT CHANGE ABC DEF .ZCSR .ZCSR
  ISREDIT XSTATUS .ZCSR = &X
  ISREDIT SEEK ABC
END
```

For more information, including other types of search strings, see "Finding, Seeking, Changing, and Excluding Data" on page 53.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 String not found

- 8 Change error. String-2 is longer than string-1 and substitution was not performed on at least one change.
- 12 Inconsistent parameters. The string to be found does not fit between the specified columns.
- 20 Severe error.

Example

Before changing the current member name, put it into a variable name such as MEMNAME. To add an identifier to that name, if it is in columns 1 to 10 and lies within the first line and the line labeled .XLAB:

```
ISREDIT (MEMNAME) = MEMBER
ISREDIT CHANGE WORD &MEMNAME "MEMBER:&MEMNAME" 1 10 .ZFIRST .XLAB
```

CHANGE_COUNTS—Query Change Counts

The CHANGE_COUNTS assignment statement retrieves values set by the most recently processed CHANGE command and places these values in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = CHANGE_COUNTS
```

- var1** The name of a variable to contain the number of strings changed. It must be an 8-character value that is left-padded with zeros.
- var2** The name of a variable to contain the number of strings that could not be changed. It also must be an 8-character value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 20 Severe error.

Examples

To put the number of changes resulting from the most recent CHANGE command into the variable &CHGED:

```
ISREDIT (CHGED) = CHANGE_COUNTS
```

To put the number of change errors into variable &ERRS:

```
ISREDIT (,ERRS) = CHANGE_COUNTS
```

To put the number of changes and change errors into variables &CHG and &ERR:

```
ISREDIT (CHG,ERR) = CHANGE_COUNTS
```

COMPARE—Edit Compare

The COMPARE command compares the file you are editing with an external sequential data set or member of a partitioned data set. Lines that exist only in the file being edited are marked, and lines that exist only in the file being compared are inserted as information lines in the file being edited. The command operates as a primary command or an edit macro.

You can use the Delete and Make Data line commands to merge changes between files that are being compared.

The COMPARE function supports all line lengths, but some SuperC options are ignored for line lengths greater than 256 characters long.

Data sets being compared must be cataloged. Compare operates by allocating data sets by name and then calling the SuperC function. If you are editing an uncataloged data set, and a data set with the same name is cataloged, the Compare command uses the cataloged version of the data set. This can cause incorrect results.

Note: COMPARE is not available in edit sessions controlled by the EDIF, EDREC, or EDIREC services.

Macro Command Syntax

```
ISREDIT COMPARE {dsname| NEXT } [SAVE ] [SYSIN ] [EXCLUDE]
```

dsname

The name of a member or data set to which the current file is compared. This variable can be specified as a fully qualified data set name (in quotation marks), a partially qualified data set name, or a member name.

If you specify only a member name, it must be preceded by a left parenthesis symbol. The right parenthesis is allowed but not required. The current edit session must be of a member of a partitioned data set. The current edit concatenation is searched for the member to compare.

If you specify only a data set name and the current file is a member of a PDS, then the specified data set is searched for a member of the same name as the member being edited.

EXCLUDE

Specifies that all matching lines in the compared data sets are excluded from the display *except* for a specified number of lines above and below the differences. The differences themselves are also shown in the display. The specified number of lines that are shown is set on the Edit Compare Settings panel. If you do not respecify the number for this edit session, then whatever was the last number set is still valid. To change this number, issue the COMPARE command with no operand and change the EXCLUDE field on the Edit Compare Settings panel. Valid numbers are 0 through 12, inclusive.

You can also use the **COMPARE EXCLUDE** command at any time to exclude all lines in a file except lines with line labels and information lines, and the lines above and below those lines. When you specify EXCLUDE without a data set name or NEXT, no comparison is done. Instead the

labels and information lines that already exist in the file are used to exclude functions. See “Compare Examples” for a macro that uses this technique.

- NEXT** Specifies to do a comparison between the currently edited member and the next member of the same name found at a higher level of the hierarchy (or next level of the edit concatenation) than the current member. For example, if the current member is found in the third level of the concatenation, and a like-named member exists at the fourth level, then the third and fourth level members are compared. After data is saved in the lowest level, compares are done from that level upward. If you specify *dsname*, the NEXT keyword cannot be used.
- SAVE** Specifies that SuperC (which performs the actual compare function) create a listing. The listing is saved in a data set named *prefix.ISPFEDIT.COMPARE.LIST*. The save function is intended for debugging purposes, but it also provides a way to create a SuperC listing. The listing produced is a Change listing (option CHNGL). No notification is given regarding successful creation of the listing, and errors allocating the listing do not cause the comparison to end.

Note: Because of the way the SuperC comparison is done, the file currently being edited is shown in the SuperC listing as the *old* file, and the file to which the current file is being compared is listed as the *new* file. Therefore, insertions refer to lines that are *not* in the current file, and deletions refer to lines that are only in the current file.

SYSIN

Specifies not to free the DD name SYSIN before calling SuperC to compare files. This enables you to pass SuperC Process Statements to alter the comparison. No validation is done on the type of SYSIN allocation or the contents of the data set.

Return Codes

The following return codes can be issued:

- | | |
|----|---|
| 0 | Normal completion |
| 8 | Member or data set not found, or an error opening the member or data set occurred. |
| 12 | No parameters specified, or another parameter error such as not valid NEXT or member specification. |
| 20 | Severe error. SuperC, allocation, or delta file error occurred. |

Compare Examples

To compare the current file to another file called X.Y.Z and to save the SuperC output file in ISPFEDIT.COMPARE.LIST:

```
ISREDIT COMPARE X.Y.Z SAVE
```

To compare the current file to a member in the same partitioned data set, and exclude everything but the context in which changes exist:

```
ISREDIT COMPARE (memname) EXCLUDE
```

COMPARE

To find all of the occurrences of a string in a file and exclude lines to show the context in which the strings were found, you can use the following macro:

```
/* Rexx - Edit macro to find a string, show only lines with the */
/*      string and a few lines above and below found strings.  */
/*      This uses the COMPARE EXCLUDE command to perform the  */
/*      line exclude function.                                */
/* ----- */
Address isredit          /*
'MACRO (PARM)'          /* Accept input string
If parm ^= '' Then     /* Do nothing if no parameters
  Do                    /*
    'RESET LABEL'      /* Remove all existing labels
    'F FIRST 'parm     /* Find first string occurrence
  Do While(rc=0)       /* For each occurrence
    'LABEL .ZCSR = 'label()' 0' /* Assign a label to line
    'RFIND'            /* Find next occurrence
  End                  /*
  'COMPARE X'         /* Exclude everything except
                    /* Labels and above/below lines
    'RESET LABEL'      /* Remove all labels
    '(XSTAT) = XSTATUS .ZFIRST' /* Save exclude status of line 1
    'LOCATE .ZFIRST'   /* Move display to line 1
    'XSTATUS .ZFIRST = 'xstat /* Restore line 1 exclude status
  End                  /*
Exit 0                /* Always return a zero
/* ----- */
label:Procedure Expose labelnum /* Routine to generate a unique
If datatype(labelnum,'N')=0 Then /* Edit line label
  labelnum=0           /*
Else                  /*
  labelnum=labelnum+1 /*
Return '.'translate(right(labelnum,4,'0'),'ABCDEFGHIJ','0123456789')
```

COPY—Copy Data

The COPY macro command copies any member of the ISPF library or partitioned data set you are editing into the member you are editing.

Macro Command Syntax

```
ISREDIT COPY member {AFTER } lptr [lینum-range]
              (member) {BEFORE}
              data set name
```

member

A member of the ISPF library or partitioned data set that you are editing.

data set name

A partially or fully qualified data set name. If the data set is partitioned, you must include a member name in parentheses.

AFTER

The destination of the data that is being copied. AFTER copies the data after lptr.

BEFORE

The destination of the data that is being copied. BEFORE copies the data before lptr.

lptr

Indicates where the data is to be copied. A line pointer can be a label or a relative line number. If you use a label, the label can be either a label that you define or one of the editor-defined labels, such as .ZF or .ZL.

linenum-range

Two numbers that specify the line numbers of the member being copied.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned a like-named member is copied. If a like-named member does not exist the name is considered to be a partially qualified data set name.

Return Codes

The following return codes can be issued:

0	Normal completion
8	End of data reached before last record read
12	Invalid line pointer (lptr); member not found or BLDL error
16	End of data reached before first record of specified range was reached
20	Syntax error (invalid name, incomplete range), or I/O error.

Examples

To copy all of the member MEM1 at the end of the data:

```
ISREDIT COPY MEM1 AFTER .ZLAST
```

To copy all of data set MOVECOPY.DATA before the first line of data:

```
ISREDIT COPY MOVECOPY.DATA BEFORE .ZFIRST
```

To copy the first three lines of the member MEM1 before the first line of data:

```
ISREDIT COPY MEM1 BEFORE .ZF 1 3
```

CREATE—Create a Data Set Member

The CREATE macro command creates a member of a partitioned data set from the data you are editing. This command cannot be used to create a sequential data set. Use the Data Set Utility (option 3.2) to allocate a sequential data set.

Macro Command Syntax

```
ISREDIT CREATE member lptr-range
              (member) [range]
              dataset(member) [range]
```

member

The name of the new member added to the partitioned data set currently being edited. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

dataset(member)

The name of a different partitioned data set and new member to be added to the partitioned data set. The data set name can be fully or partially qualified.

lptr-range

Two line pointers that specify the range of lines used to create the new member. A line pointer can be a label or a relative line number. Specifying one line pointer is incorrect.

CREATE

Description

CREATE adds a member to a partitioned data set only if a member with the same name does not already exist. Use REPLACE if the member already exists.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 8 Member already exists, member not created
- 12 Invalid line pointer (lptr). The referenced line does not exist in the file.
- 20 Syntax error (invalid name or incomplete lptr range), or I/O error.

Example

To create a new 10-line member from the first 10 lines of the member being edited:

```
ISREDIT CREATE MEM1 1 10
```

CTL_LIBRARY—Query Controlled Library Status

The CTL_LIBRARY assignment statement retrieves the status of a controlled library and places the status in variables. CTL_LIBRARY is used in initial macros to define the use of controlled library members.

Note: The CTL_LIBRARY assignment statement applies to LMF only. You cannot use it to query the status of a library that is controlled by SCLM. Refer to *ISPF/PDF Software Configuration and Library Manager (SCLM) Guide and Reference* for information about querying the status of libraries that are controlled by SCLM.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = CTL_LIBRARY
```

var1 The name of a variable to contain the lock status of the member.

var2 The name of a variable to contain additional information about the status.

Table 7 summarizes the information contained in *var1* and *var2*. The table entries are defined following the table.

Table 7. *var1* and *var2* Contents

<i>var1</i>	<i>var2</i>
OBTAINED	User ID that obtained the member
UNAVAILABLE	{User ID} {DEACTIVATED} {*LOCKED*}
ERROR	blanks
NOCHECK	{FIRSTLIB} {blanks}

The value placed in *var1* is one of the following:

OBTAINED

Specifies that the lock has been obtained for the member being edited. The

member was found in a controlled library. If the member is modified and saved in your library, the next time this statement is processed, NOCHECK is returned as the lock status. If the member is not saved in your library, the lock for this member is freed.

If OBTAINED is placed in *var1*, the value of *var2* contains your user ID (the user ID that locked the member).

UNAVAILABLE

Specifies that the lock could not be obtained for the member being edited. The member was found in a controlled library.

If UNAVAILABLE is placed in *var1*, indicating the lock is not available, the value placed in *var2* is one of the following:

User ID

The user ID that has locked the member.

DEACTIVATED

Library controls have been deactivated.

LOCKED

The member is available, but exists in a lower level of the library structure that did not precede the library in which the member was found in the Edit concatenation sequence. This is called a *pseudo-lock*.

ERROR

Specifies that the library access service was unable to determine whether the member was locked because of an error or unusual condition.

blanks

If ERROR is placed in *var1*, the value of *var2* is **blanks**.

NOCHECK

Specifies that no check was done to determine the status of the member. NOCHECK is returned in the following cases:

- NO or NEVER was typed in the **Lock** field of the Edit - Entry panel.
- The member is new.
- The member was obtained from the first library in the concatenation sequence.
- An ISRCFIL data set name is not allocated to you.

If NOCHECK is placed in *var1*, indicating that no checking was done, the value placed in *var2* is the reason no checking was done (which is one of the following):

FIRSTLIB

The member was found in the first library of the concatenation sequence used for editing, or the member is new.

blanks

The data set allocation shows that library management should not be called; no ISRCFIL DD is allocated or LOCK=NO was specified.

Return Codes

The following return codes can be issued:

- | | |
|----|-------------------|
| 0 | Normal completion |
| 20 | Severe error. |

Example

To get the control and lock status of the current member:
 ISREDIT (CSTATUS,LSTATUS) = CTL_LIBRARY

CURSOR—Set or Query the Cursor Position

The CURSOR assignment statement sets or retrieves the column number of the cursor location within the data and either the relative line number or label. These values are placed in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = CURSOR
ISREDIT CURSOR = lptr [col]
```

- var1** The name of a variable containing the line number. The line number is a 6-digit value that is left-padded with zeros. It is the ordinal number (not the sequence number) of the line.
- var2** The name of a variable containing the data column number. The data column number is a 3-digit number that is left-padded with zeros. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, a length of 3 or 4 is allowed in cases where no data loss will occur. The columns are numbered starting with 1 at the first data column. If the cursor is in the command area, the cursor value is column 1 of the first data line on the panel; the value is column 0, if the cursor is in the line command area. When you retrieve the cursor position in an empty member, the line number and column number are both set to 0.
- lptr** The relative line number or label of the line on which the cursor is to be located. Make sure when you set the cursor to a line number that the line number exists.

Note: If you try use a label that has not been assigned, you receive a return code of 20. To avoid this, use the LINENUM assignment statement. When using the LINENUM statement, a return code of 8 is issued if the label does not exist.

```
ISREDIT X = LINENUM .LABEL
```

- col** The data column number where the cursor is to be located.
- If the column number is beyond the end of the data area when setting the cursor, the cursor is positioned to the next line, which is equivalent to the first position of the line command area.

Description

The position of the cursor shows the starting or ending location for the SEEK, FIND, CHANGE, and EXCLUDE commands. It is also used as the text split point for TSPLIT. See “Referring to Column Positions” on page 115 for more information on how the column number is determined.

When you run a macro, the cursor value is the cursor position on the panel at run time.

CURSOR

Note: To position the cursor on the Command line, issue a return code of 1 from the macro. For example, in CLIST code EXIT CODE(1) as the last statement in your EDIT MACRO to position the cursor on the command line.

The following statements can change the cursor position:

CHANGE	SEEK
CURSOR	TSPLIT
EXCLUDE	USER_STATE
FIND	

Table 8 shows the line and column numbers returned, depending on the location of the cursor.

Table 8. Cursor Position

If the CURSOR location is:	The LINE number is:	And the COLUMN number is:
Command area	1st display area	0
Line number field	Line by the cursor	0
Left sequence number (the sequence number is on the left of the data when number mode is on)	Line by cursor	0
Right sequence number	Line by the cursor	Column by the cursor
Left or right of the bounds	Line by the cursor	Column by the cursor
Data within the bounds	Line by the cursor	Column by the cursor
Insert blank space	Line above the cursor. If the cursor is at the top of the panel, then the line number returned is the line below the cursor and the column number is column 0.	Column by the cursor
Non-data line and its line command area	Line below the non-data line. If the non-data line is at the bottom of the panel, then the line number returned is the line above and the column is the data width plus 1.	0

Return Codes

The following return codes can be issued:

0	Normal completion
4	Column number beyond data, line number incremented
12	Invalid line number
20	Severe error.

Examples

To put the line number of the current cursor position into variable &LINE:

```
ISREDIT (LINE) = CURSOR
```

To set the cursor position to data line 1, column 1:

```
ISREDIT CURSOR = 1 1
```

CURSOR

To set the cursor position to column 1 of the last data line:

```
ISREDIT CURSOR = .ZLAST 1
```

To set the cursor position to the line with the label .LAB, without changing the column position:

```
ISREDIT CURSOR = .LAB
```

CUT—Cut and Save Lines

The CUT macro command saves lines to one of eleven named clipboards for later retrieval by the PASTE command. The lines can be appended to lines already saved by a previous CUT command.

Syntax

```
ISREDIT CUT [lptr-range] [DEFAULT | clipboardname]
            [REPLACE]
```

lptr-range

Two line pointers that specify the range of lines in the current member that are to be added to or replace data in the clipboard. A line pointer can be a label or relative line number. You must specify both a starting and ending line pointer.

clipboardname

The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

REPLACE

Specify REPLACE to replace existing data in the clipboard. If you do not specify REPLACE, the lines in the current CUT are added to the end of the existing data within the clipboard.

Description

CUT saves copies of lines from an edit session to a clipboard for later retrieval by the PASTE command. The lines are copied from the session to the named clipboard. Lines are specified by label names on the CUT command. The edit macro CUT command always copies lines to the clipboard and does not delete them from the edit session.

If you specify a clipboard name, lines are copied to that clipboard. If the specified clipboard does not yet exist, it is created. ISPF provides a default clipboard named DEFAULT. You can use up to 10 other clipboards that you define. The defined clipboards exist as long as you are logged on to TSO and are deleted when you log off.

You can view the contents of clipboards and rename existing clipboards using the DISPLAY keyword of the CUT command. If you specify the DISPLAY, other keywords are ignored.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Parameter error. Insufficient storage, or no more clipboards available.
20	Severe error.

Examples

To save all the lines in the current file to the default clipboard, appending them to lines already in the clipboard:

```
ISREDIT CUT .ZFIRST .ZLAST
```

To save all the lines in the current file to a clipboard named USERC1, replacing any lines already in the clipboard:

```
ISREDIT CUT .ZFIRST .ZLAST USERC1 REPLACE
```

DATA_CHANGED—Query the Data Changed Status

The DATA_CHANGED assignment statement retrieves the current data-changed status and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = DATA_CHANGED
```

varname

The name of a variable containing the data-changed status, either YES or NO. The data-changed status is initially set to NO at the beginning of an edit session, and is reset to NO whenever a save is done. If you change data on your screen, but issue the END command, the data_changed status is still NO. When data is changed, or if a command is issued which might have changed the data, the changed status is set to YES.

Description

This command returns information about whether the data might have changed. However, it does not specify whether data is saved when the END command is issued. Data can be saved without being changed if there is a change to the version, number, stats, or pack mode. When DATA_CHANGED returns a value of NO, an 8 character variable called ZEDSAVE is set to indicate whether the data is saved. ZEDSAVE will contain either "SAVE " or "NOSAVE". See AUTOSAVE, CANCEL, SAVE and END for more information about saving data.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Example

To determine whether data has been changed and, if it has, to issue the built-in SAVE command:

```
ISREDIT (CHGST) = DATA_CHANGED
IF &CHGST = YES THEN ISREDIT BUILTIN SAVE
```

DATA_WIDTH—Query Data Width

The DATA_WIDTH assignment statement retrieves the current logical data width and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = DATA_WIDTH
```

varname

The name of the variable to contain the logical data width. The logical data width is a 3-digit value that is left-padded with zeros. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, a length of 3 or 4 is allowed in cases where no data loss occurs.

Description

The logical data width is the maximum space, in bytes, that is available for data only. It does not include any COBOL or sequence number fields or, for variable-length records, the 4-byte record descriptor word (RDW).

The value returned by the DATA_WIDTH assignment statement depends on the record format (fixed or variable) and the setting of number mode, as shown in Table 9. See “NUMBER—Generate Sequence Numbers” on page 266 if you need more information about number mode.

Table 9. Data Width Return Value

Number Mode Setting	Logical Data Width for Fixed-Length Records	Logical Data Width for Variable-Length Records
OFF	LRECL	LRECL - 4
ON STD	LRECL - 8	LRECL - 12
ON COB	LRECL - 6	N/A ¹
ON STD COB	LRECL - 14	N/A ¹

Use the LRECL assignment statement to get the maximum space, in bytes, that is available for data, COBOL number fields, and sequence number fields.

Return Codes

The following return codes can be issued:

0 Normal completion
 12 Invalid command format
 20 Severe error.

Example

To put the data width in variable &MAXCOL and override the boundary setting for SEEK:

1. COBOL numbering is invalid for variable-length records.

```
ISREDIT (MAXCOL) = DATA_WIDTH
ISREDIT SEEK 1 &MAXCOL &ARGSTR
```

DATAID—Query Data ID

The DATAID assignment statement retrieves the data ID for the data set currently being edited and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = DATAID
```

varname

The name of a variable containing the data ID of the data set currently allocated for editing.

Description

The data ID is created by the LMINIT service to identify a data set.

If you begin an edit session with a data ID, the data ID is returned when you issue this command. If you begin an edit session without a data ID, then an LMINIT service obtains a data ID and returns it. On return from a top-level macro, the editor releases any data ID it has obtained.

For further information about the use of library access services, refer to *ISPF User's Guide*.

Return Codes

The following return codes can be issued:

0	The data ID returned was passed to the editor
4	Data ID was generated by and is freed by the editor
8	A previously generated data ID was returned
20	Severe error.

Example

To store the data ID in variable &DID, and then find the member MEM1 of that data set by using the LMMFIND library access service:

```
ISREDIT (DID) = DATAID
ISPEXEC LMMFIND DATAID(DID) MEMBER(MEM1)
IF &LASTCC = 0 THEN ...
```

DATASET—Query the Current Data Set Name

The DATASET assignment statement retrieves the name of the data set into which the data currently being edited will be stored, and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = DATASET
```

varname

The name of a variable to contain the name of the data set currently being edited. The data set name is fully qualified without quotation marks (').

DATASET

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Example

To determine if you are editing a data set with a prefix of PROJ:

```
ISREDIT (DSNAME) = DATASET
IF &SUBSTR(1:4,&DSNAME ) = PROJ THEN -
...
```

DEFINE—Define a Name

The DEFINE macro command is used to:

- Identify a macro that replaces a built-in command of the same name
- Identify programs that are edit macros
- Assign an alias to a macro or built-in command
- Make a macro or built-in command inoperable
- Reset an inoperable macro or built-in command
- Disable a macro or built-in command.

DEFINE is often used with the BUILTIN command.

Macro Command Syntax

```
ISREDIT DEFINE name {MACRO CMD }
                   {MACRO PGM }
                   {ALIAS name-2}
                   {NOP }
                   {RESET }
                   {DISABLED }
```

name The name with which you process the command.

MACRO CMD

Identifies the name that you are defining as a command language (CLIST or REXX EXEC) macro, which is called in the same way as using the SELECT service CMD keyword with a percent symbol (%) preceding the command. That means that you can specify only CLISTs or REXX EXECs. This operand is the default.

MACRO PGM

Identifies the name that you are defining as a program (load module) macro, which is called by the SELECT PGM service.

ALIAS name-2

Identifies the name that you are defining as an alias of another name, with the same characteristics. If *name-2* is already an alias, the editor replaces it with the command it names. Therefore, it is not possible to have an alias of an alias.

NOP Makes the name you are defining and all of its aliases inoperable until you reset them with the RESET operand. Therefore, when the name or an alias of the name is called, nothing is processed. NOP is similar to DISABLED, except that disabled names cannot be reset by the RESET operand.

RESET

Resets the most recent definition of the name that you are defining to the status in effect before that definition. For example, RESET makes inoperable names operable again.

DISABLED

Makes the name that you are defining and all of its aliases disabled until you end the edit session. Therefore, when the name or an alias of the name is called, nothing is processed. A disabled command or macro cannot be restored by RESET.

Description

The effects of the DEFINE macro command apply only to the edit session of the member or sequential data set being edited when the macro is run. This effect is different from the DEFINE primary command.

To temporarily override DEFINE, use BUILTIN.

Note: To define RESET as disabled, enclose it in quotes ('RESET'). If you do not use quotes, the editor interprets RESET as a keyword.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 8 RESET was attempted for a name not currently defined, or DEFINE name ALIAS *name-2* requested and *name-2* is an NOP
- 12 DEFINE was attempted for a name not currently defined
- 20 Severe error (unknown command).

Examples

To define the name IJKDOIT as a CLIST or REXX macro:

```
ISREDIT DEFINE IJKDOIT MACRO
```

To define the name SETITUP as a program macro:

```
ISREDIT DEFINE SETITUP MACRO PGM
```

To define the name DOIT as an alias of the macro IJKDOIT:

```
ISREDIT DEFINE DOIT ALIAS IJKDOIT
```

To define the name SAVE to have no effect:

```
ISREDIT DEFINE SAVE NOP
```

To reset the definition of the name SAVE:

```
ISREDIT DEFINE SAVE RESET
```

To define the name FINDIT as disabled:

```
ISREDIT DEFINE FINDIT DISABLED
```

DEFINE

To create and update library statistics when data is saved, first set the stats mode on. Then make it impossible to turn off by defining it as disabled. Note that none of the commands that are defined as disabled can be called while you are editing a member.

```
ISREDIT MACRO
ISREDIT STATS ON
ISREDIT DEFINE STATS DISABLED
```

DELETE—Delete Lines

The DELETE macro command deletes lines from the data you are editing.

Macro Command Syntax

```
ISREDIT DELETE { ALL X | NX [lptr-range] }
                { [ALL] X | NX lptr-range }
                { lptr }
                { lptr-range }
```

ALL Specifies that all selected lines are deleted. The DELETE command, unlike FIND, CHANGE, and EXCLUDE, does not use NEXT, FIRST, PREV, or LAST. ALL is required to emphasize that NEXT is not the default.

X | NX

Restricts the lines deleted to those that are excluded or not excluded, respectively.

lptr Specifies that a line pointer must be used to identify a line to be deleted. A line pointer can be a label or a relative line number.

lptr-range

Specifies with two line pointers a range of lines to be deleted. The range must consist of two labels or two relative line numbers. When specifying a range, providing only one line pointer is incorrect. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

Description

DELETE can specify a single line or a range of lines. It can limit the lines to be deleted to all excluded or nonexcluded lines in the data, or to all excluded or nonexcluded lines within a line pointer range.

Return Codes

The following return codes can be issued:

0	Normal (lines deleted successfully)
4	No lines deleted
8	No standard records exist
12	Invalid line number
20	Severe error.

Examples

To delete all nonexcluded lines:

```
ISREDIT DELETE ALL NX
```

To delete all lines between labels .A and .B with a blank in column 1:

```
ISREDIT RESET X .A .B
ISREDIT EXCLUDE ALL " " 1 .A .B
ISREDIT DELETE ALL X .A .B
```

To delete the last line of data in the current data set:

```
ISREDIT DELETE .ZLAST
```

To delete the first 10 lines of data in the current data set:

```
ISREDIT DELETE 1 10
```

DISPLAY_COLS—Query Display Columns

The DISPLAY_COLS assignment statement retrieves the column numbers of the first and last data columns that you are seeing, and places them in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = DISPLAY_COLS
```

var1 The name of a variable containing the column number of the first data column visible to you. The column number is a 3-digit value that is left-padded with zeros. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, a length of 3 or 4 is allowed in cases where no data loss will occur.

var2 The name of a variable containing the column number of the last data column visible to you. The column number is a 3-digit value that is left-padded with zeros. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, a length of 3 or 4 is allowed in cases where no data loss will occur.

Description

Columns that contain sequence numbers are not considered data columns. Do not use this assignment statement in initial macros because the columns displayed are not known until the data first appears. See “Referring to Column Positions” on page 115 for more information.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid command format
20	Severe error.

Example

To put the leftmost and rightmost column values displayed to you in variables &LEFT and &RIGHT:

```
ISREDIT (LEFT,RIGHT) = DISPLAY_COLS
```

DISPLAY_LINES—Query Display Lines

The DISPLAY_LINES assignment statement retrieves the relative line numbers of the first and last data lines that would appear at this point if the macro ended, and places them in variables. Other non-data lines might be on the display. Do not use this assignment statement in an initial macro because the lines displayed are not known until the data is first displayed.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = DISPLAY_LINES
```

- var1** The name of a variable containing the relative line number of either the first visible data line or block of excluded lines if the macro ended at this point. The relative line number is a 6-digit value that is left-padded with zeros.
- var2** The name of a variable containing the relative line number of either the last visible data line or block of excluded lines. The relative line number is a 6-digit value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

- | | |
|----|------------------------|
| 0 | Normal completion |
| 4 | No visible data lines |
| 8 | No existing data lines |
| 12 | Invalid command format |
| 20 | Severe error. |

Example

To place the top and bottom line numbers in variables &TOP and &BOT:

```
ISREDIT (TOP,BOT) = DISPLAY_LINES
```

DOWN—Scroll Down

The DOWN macro command scrolls data down from the current panel position.

Macro Command Syntax

```
ISREDIT DOWN amt
```

amt The number of lines (0 - 9999) to scroll, or one of the following operands:

MAX Scrolls to the end of data in the specified direction.

HALF Displays the next sequential half panel of data.

PAGE Displays the next sequential full panel of data.

CURSOR

Scrolls until the line on which the cursor is located becomes the first data line on the panel.

DATA Scrolls until the last data line on the current panel of data becomes the first data line on the next panel of data.

Description

To scroll down using the panel position when the macro was first issued, use `USER_STATE` assignment statements to save and then restore the panel position operands.

When you issue `DOWN`, the non-data lines on the panel affect the number of lines scrolled. However, if you define a macro named `DOWN`, it only overrides the `DOWN` command when used from another macro. `DOWN` does not change the cursor position and cannot be used in an initial macro.

The actual number of lines appearing on the panel is determined by:

- The number of lines excluded from the display
- The terminal display size and split-panel line
- The number of special temporary lines appearing, such as the `==ERR>`, `==CHG>`, `=COLS>`, `=====`, `=PROF>`, `==MSG>`, `=NOTE=`, `=BNDS>`, `=TABS>` or `=MASK>` lines.

The first line appearing is determined in one of two ways: (1) a `LOCATE` command can set the line first on the panel, and (2) the first line to appear depends on whether the cursor was set explicitly by a `CURSOR` assignment statement or implicitly by a `SEEK`, `FIND`, `CHANGE`, or `TSPLIT` command. Since the cursor must be on the panel, the line that is the first line on the panel may be different from the line that was first when you called the macro.

Return Codes

The following return codes can be issued:

0	Normal completion
2	No more data <code>DOWN</code>
4	No visible lines
8	No data to display
12	Amount not specified
20	Severe error.

Examples

To scroll down to the end of the data set:

```
ISREDIT DOWN MAX
```

To display the next half panel of data:

```
ISREDIT DOWN HALF
```

To display the next full panel of data:

```
ISREDIT DOWN PAGE
```

To make the line where the cursor is placed the first one on the display:

```
ISREDIT DOWN CURSOR
```

To display the next page less one line:

```
ISREDIT DOWN DATA
```

EDIT—Edit from within an Edit Session

The EDIT macro command allows you to edit a member of the same partitioned data set during your current edit session.

Macro Command Syntax

```
ISREDIT EDIT member
```

member

A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Editing one data set or member while you are already editing another is called *recursive editing*. Your initial edit session is suspended until the second-level edit session is complete. Editing sessions can be nested until you run out of storage.

To exit from a nested edit session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

The EDIT service call, ISPEXEC EDIT, is an alternate method of recursively starting the editor. It offers the option of editing another data set and specifying an initial macro.

For more information on using the EDIT service for recursive editing, refer to *ISPF Services Guide*

Return Codes

The following return codes can be issued:

0	Normal completion, data was saved
4	Normal completion, data was <i>not</i> saved
12	Your error (invalid member name, recovery pending)
14	Member in use
20	Severe error.
28	No ISREDIT MACRO statement preceded this call, or BROWSE was substituted because of the size of the member being edited.

Example

To recursively edit the member OLDMEM in your current ISPF library:

```
ISREDIT EDIT OLDMEM
```

END—End the Edit Session

The END macro command ends the editing of the current sequential data set or partitioned data set member.

Macro Command Syntax

```
ISREDIT END
```

Description

If an edit macro contains an ISREDIT END statement, there can be no other ISREDIT or ISPEXEC statements following it. If one of these kinds of statements does follow an ISREDIT END, the edit macro ends with an error when that statement occurs. However, any other CLIST, REXX EXEC, or program statements can follow an ISREDIT END statement and process normally.

If no aliases have been defined for END, the response of the editor to the END command depends on:

- Whether changes were made to the data during your current edit session
- If changes were made, whether a SAVE command was entered after the last change
- The setting of number mode, autonum mode, stats mode, autolist mode, and autosave mode in the edit profile
- Whether you were editing a member that was an alias of another member.

See “Ending an Edit Session” on page 15 for more information.

Return Codes

The following return codes can be issued:

0	Normal completion
4	New member saved
12	END not done, AUTOSAVE OFF PROMPT set, or Data not saved (insufficient space)
20	Severe error.

Example

To end the current edit session:

```
ISREDIT END
```

EXCLUDE—Exclude Lines from the Display

The EXCLUDE macro command hides lines that contain a search string from view, and replaces them with a dashed line. To see the lines again, you enter either the RESET or RESET EXCLUDED command.

Macro Command Syntax

```
ISREDIT EXCLUDE string [label-range] [NEXT ] [CHARS ] [col-1 [col-2]]
                                     [ALL ] [PREFIX]
                                     [FIRST] [SUFFIX]
                                     [LAST ] [WORD ]
                                     [PREV ]
```

string The search string you want to exclude.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within the &STR() delimiters

EXCLUDE

such as &STR(/*XXX). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

label-range

Two labels that identify the lines within which the EXCLUDE command is to search. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

When using a macro that uses NEXT or PREV with a label-range, be careful concerning cursor placement. If the cursor is currently placed below the label-range, and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

If the cursor is currently placed above the label-range, and the PREV occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string*. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of *string*.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of *string*.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of *string*.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of *string*.

CHARS

Locates *string* anywhere the characters match. CHARS is the default.

PREFIX

Locates *string* at the beginning of a word.

SUFFIX

Locates *string* at the end of a word.

WORD

Locates *string* when it is delimited on both sides by blanks or other non-alphanumeric characters.

col-1 and col-2

Numbers that identify the columns the EXCLUDE command is to search.

Description

You can use the EXCLUDE command with the FIND and CHANGE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To exclude the next nonexcluded line that contains the letters ELSE without specifying any other qualifications, include the following command in an edit macro:

```
ISREDIT EXCLUDE ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase

EXCLUDE

- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- Anywhere within the current boundaries.

To exclude the next line that contains the letters ELSE, but only if the letters are uppercase, include the following command in an edit macro:

```
ISREDIT EXCLUDE C'ELSE'
```

This type of exclusion is called a character string exclusion (note the C that precedes the search string) because it excludes the next line that contains the letters ELSE only if the letters are found in uppercase. However, since no other qualifications were specified, the exclusion occurs no matter where the letters are found on a nonexcluded line, as outlined in the previous list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

Return Codes

The following return codes can be issued:

0	Normal completion
4	String not found
8	Lines not excluded
12	Inconsistent parameters
20	Severe error.

Examples

This example excludes the first nonexcluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
ISREDIT EXCLUDE ELSE .E .S FIRST PREFIX
```

This example excludes the last nonexcluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the last four letters of a word.

```
ISREDIT EXCLUDE ELSE .E .S LAST SUFFIX
```

This example excludes the first nonexcluded line that immediately precedes the cursor position and that contains the letters ELSE. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between the labeled lines; they must be standalone characters (not part of any other word); and they must exist within columns 1 and 5:

```
ISREDIT EXCLUDE ELSE .E .S PREV WORD 1 5
```

EXCLUDE_COUNTS—Query Exclude Counts

The EXCLUDE_COUNTS assignment statement retrieves values set by the most recently processed EXCLUDE command and places them in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = EXCLUDE_COUNTS
```

EXCLUDE_COUNTS

- var1** The name of a variable to contain the number of strings found. The number of strings is an 8-digit value that is left-padded with zeros.
- var2** The name of a variable to contain the number of lines excluded. The number of lines excluded is an 8-digit value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

- 0** Normal completion
- 12** Invalid command format
- 20** Severe error.

Example

To determine the number of lines that contain the word BOX:

```
ISREDIT EXCLUDE ALL BOX
ISREDIT (,BOXLINES) = EXCLUDE_COUNTS
```

FIND—Find a Search String

The FIND macro command locates one or more occurrences of a search string.

Macro Command Syntax

```
ISREDIT FIND string [label-range] [NEXT ] [CHARS ] [X ] [col-1 [col-2]]
                                     [ALL ] [PREFIX] [NX]
                                     [FIRST] [SUFFIX]
                                     [LAST ] [WORD ]
                                     [PREV ]
```

string The search string you want to find.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within the &STR() delimiters such as &STR(/*XXX). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

label-range

Two labels that identify the lines within which the FIND command is to search. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

When using a macro that uses NEXT or PREV with a label-range, be careful concerning cursor placement. If the cursor is currently placed below the label-range, and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

If the cursor is currently placed above the label-range, and the PREV occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string*. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of *string*.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of *string*.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of *string*.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of *string*.

CHARS

Locates *string* anywhere the characters match. CHARS is the default.

PREFIX

Locates *string* at the beginning of a word.

SUFFIX

Locates *string* at the end of a word.

WORD

Locates *string* when it is delimited on both sides by blanks or other non-alphanumeric characters.

X Scans only lines that are excluded from the display.

NX Scans only lines that are not excluded from the display.

col-1 and col-2

Numbers that identify the columns FIND is to search.

Description

Use the SEEK macro command instead of FIND if you want to locate a string without changing the exclude status of the line that contains the string.

You can use FIND with the EXCLUDE and CHANGE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To find the next occurrence of the letters ELSE without specifying any other qualifications, include the following line in an edit macro:

```
ISREDIT FIND ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In either an excluded or a nonexcluded line
- Anywhere within the current boundaries.

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

```
ISREDIT FIND C'ELSE'
```

This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

FIND

Return Codes

The following return codes can be issued:

0	Normal completion
4	String not found
12	Syntax error
20	Severe error.

Examples

The following example finds the first occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
ISREDIT FIND ELSE .E .S FIRST PREFIX
```

The following example finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
ISREDIT FIND ELSE .E .S LAST SUFFIX X
```

The following example finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be standalone characters (not part of any other word); they must be found in a nonexcluded line; and they must exist within columns 1 and 5:

```
ISREDIT FIND ELSE .E .S PREV WORD NX 1 5
```

FIND_COUNTS—Query Find Counts

The FIND_COUNTS assignment statement retrieves values that were set by the most recently entered FIND or RFIND command, and places these values in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = FIND_COUNTS
```

var1 The name of a variable to contain the number of strings found. The number of strings is an 8-digit value that is left-padded with zeros.

var2 The name of a variable to contain the number of lines on which strings were found. The number of lines on which strings were found is an 8-digit value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid command format
20	Severe error.

Example

To find all occurrences of && in the line labeled .A and loop through and process them:

```
ISREDIT FIND .A .A && ALL
ISREDIT (FINDS) = FIND_COUNTS
DO WHILE &FINDS > 0
    ...
END
```

FLIP—Reverse Exclude Status of Lines

The FLIP macro command lets you reverse the exclude status of a specified range of lines or of all the lines in a file, including data, information, message, and note lines.

Assignment Statement Syntax

```
ISREDIT FLIP [label-range]
```

label-range

Two labels that identify the lines within which the FLIP command is to reverse the exclude status.

If one label is specified, only that labeled line is reversed. This is optional.

Return Codes

The following return codes can be issued:

- 0 Successful completion. The excluded status of the requested lines was reversed.
- 20 Severe error.

Examples

The following are examples of statements using the FLIP commands from an Edit macro. The actual values for .a and .b can be defined by edit macro or by the user.

```
ISREDIT FLIP          /* Flip all lines          */
ISREDIT FLIP .ZL .ZF /* Flip all lines          */
ISREDIT FLIP .ZF      /* Flip first line in file */
ISREDIT FLIP .a .b   /* Flip lines between and including .a and .b */
ISREDIT FLIP .a      /* Flip line labeled .a    */
```

FLOW_COUNTS—Query Flow Counts

The FLOW_COUNTS assignment statement retrieves values that were set by the most recently entered TFLOW command, and places these values in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = FLOW_COUNTS
```

- var1** The name of a variable to contain the number of original lines that participated in the text flow operation. The number of original lines is an 8-digit value that is left-padded with zeros.

FLOW_COUNTS

var2 The name of a variable to contain the number of lines that were generated by the text flow operation. The number of lines is an 8-digit value that is left-padded with zeros.

If the value in *var1* is larger than the value in *var2*, the difference is the number of lines that were deleted from the current data because of the text flow operation. If the value in *var1* is less than the value in *var2*, the difference is the number of lines that were added to the current data because of the text flow operation.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Example

To retrieve the value of the rightmost column displayed, allow a margin of 8 for the text flow, and then take action if lines were added because of the text flow operation:

```
ISREDIT (,MAXCOL) = DISPLAY_COLS
ISREDIT TFLOW .ZCSR &EVAL(MAXCOL - 8)
ISREDIT (INLINE,OUTLIN) = FLOW_COUNTS
IF &OUTLIN > &INLINE THEN DO
...

```

HEX—Set or Query Hexadecimal Mode

The HEX macro command sets hexadecimal mode, which determines whether data appears in hexadecimal format.

The HEX assignment statement either sets hexadecimal mode or retrieves the current values of hexadecimal mode, and places them in variables.

Macro Command Syntax

```
ISREDIT HEX [ON DATA]
             [ON VERT]
             [OFF ]
```

ON DATA

Displays the hexadecimal representation of the data as a string of hexadecimal characters (two per byte) under the characters.

ON VERT

Displays the hexadecimal representation of the data vertically (two rows per byte) under each character.

OFF Does not display hexadecimal representation of the data.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = HEX
ISREDIT HEX = [ON DATA]
              [ON VERT]
              [OFF ]
```

var1 The name of a variable to contain ON or OFF.

var2 The name of a variable to contain DATA, VERT, or blanks.

ON DATA

Same as macro command syntax.

ON VERT

Same as macro command syntax.

OFF Same as macro command syntax.

Description

The HEX macro command and assignment statement determines whether the editor displays hexadecimal representation in a vertical or data string format.

When the editor is operating in hexadecimal mode, three lines are displayed for each source line. The first line shows the data in standard character form, while the next two lines show the same data in hexadecimal representation.

Besides normal editing on the first of the three lines, you can change any characters by typing over the hexadecimal representations.

You can also use the FIND, CHANGE, and EXCLUDE commands to find, change, or exclude invalid characters or any specific hexadecimal character, regardless of the setting of hexadecimal mode. See the discussion of picture strings and hexadecimal strings under “Finding, Seeking, Changing, and Excluding Data” on page 53.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Examples

To put the value of hexadecimal mode (on or off) in variable &HEXMODE and to process if hexadecimal mode is on:

```
ISREDIT (HEXMODE) = HEX
IF &HEXMODE = ON THEN -
...
```

To turn hexadecimal mode off:

```
ISREDIT HEX OFF
```

HILITE—Enhanced Edit Coloring

HILITE is used to control the use of color in the editor by changing the settings for the enhanced color and language-sensitive editing features.

The HILITE dialog is not available in the Edit Macro environment.

Note: Language sensitive and enhanced coloring of the edit session is only available if it is enabled by the installer or person who maintains the ISPF product. For information on enabling the enhanced color functions, see *ISPF Planning and Customizing*

Macro Command Syntax

```

ISREDIT HILITE [ON      ] [AUTO  ] [RESET] [PAREN] [FIND] [CURSOR] [SEARCH] [DISABLED]
               [OFF     ] [DEFAULT]
               [LOGIC  ] [OTHER  ]
               [IFLOGIC] [ASM   ]
               [DOLOGIC] [BOOK  ]
               [NOLOGIC] [C     ]
                               [COBOL ]
                               [DTL   ]
                               [JCL   ]
                               [PANEL ]
                               [PASCAL]
                               [PLI   ]
                               [REXX  ]
                               [SKEL  ]
                               [IDL   ]

```

ON Sets program coloring ON and turns LOGIC coloring off.

OFF Sets coloring OFF, with the exception of cursor highlighting.

LOGIC

LOGIC highlighting matches logical language-specific keywords in the same color. If an unmatched *closing* keyword is found, such as END for PL/I or :eul. for BookMaster, it is highlighted in reverse video pink *only* if HILITE LOGIC is active. When logic is being highlighted, only comments are highlighted along with it.

Logic highlighting is available for PL/I, PL/X, Rexx, OTHER, C, SKELS, Pascal and BookMaster only. HILITE LOGIC turns on both IFLOGIC and DOLOGIC.

Note: LOGIC highlighting can be turned off by issuing HILITE ON, HILITE NOLOGIC, or HILITE RESET commands. Changing the HILITE language does not change the LOGIC setting.

IFLOGIC

Turns on IF/ELSE logic matching. IFLOGIC matches IF and ELSE statements. When IFLOGIC is enabled, unmatched ELSE keywords are highlighted in reverse video pink.

DOLOGIC

Turns on DO/END logic matching. DOLOGIC matches logical blocks such as DO/END in PL/I or :ol/:eol in BookMaster. For the C language, DOLOGIC matches curly braces ({ and }). C trigraphs for curly braces are not recognized and are not supported by DOLOGIC highlighting. When DOLOGIC is enabled unmatched logical block terminators, (such as END keywords in PL/I, :e tags in BookMaster or right braces (}) in C are highlighted in reverse video pink.

NOLOGIC

Same as ON.

AUTO

Allows ISPF to determine the language.

DEFAULT

Highlights the data in a single color.

OTHER

Highlight the data as a pseudo-PL/I language.

ASM Highlights the data as Assembler.

BOOK

Highlights the data as BookMaster.

C Highlights the data as C.

COBOL

Highlights the data as COBOL.

DTL Highlights the data as Dialog Tag Language.

JCL Highlights the data as MVS Job Control Language.

PANEL

Highlights the data as ISPF Panel Language.

PASCAL

Highlights the data as Pascal.

PLI Highlights the data as PL/I.

REXX Highlights the data as REXX.

SKEL Highlights the data as ISPF Skeleton Language.

IDL Highlights the data as IDL.

RESET

Resets defaults (AUTO, ON, Find and Cursor on).

PAREN

Toggles parenthesis matching. When parenthesis matching is active, only comments and quoted strings are specially colored. All other code appears in the default color. Note that extra parenthesis highlighting is always active when highlighting is active.

Parentheses within quoted strings and comments are not checked or highlighted by the parenthesis matching function.

FIND The HILITE FIND command toggles the highlighting color of any string that would be found by an RFIND. The user can select the highlight color. The default is reverse video white.

Only non-picture strings are supported, and the only additional qualifiers recognized are hex strings (X'...'), character strings (C'...'), text strings (T'...'), WORD, PREFIX and SUFFIX, and boundaries specified in the FIND command. Hex strings may be highlighted, but non-displayable characters are not highlighted. Default bounds and labels are ignored when FIND strings are highlighted.

Because FIND highlighting is not quite as robust at the FIND command itself, the editor may highlight more occurrences of the FIND string than FIND would actually locate.

RESET has been enhanced, through the addition of a FIND operand, to temporarily disable the highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command is issued. RESET with the FIND operand (or no operands at all), temporarily disables the highlighting of FIND strings.

CURSOR

The CURSOR operand toggles the highlighting of the phrase that contains the cursor in a user-selectable color. The default is white.

HILITE

Cursor highlighting in Edit is performed in a manner similar to the way it is done in Browse. The entire phrase from the previous blank to the next blank is highlighted.

SEARCH

HILITE SEARCH finds the first unmatched END, ELSE, }, or) above the last displayed line on the panel. If a mismatched item is found, the file is scrolled so that the mismatch is at the top of the panel. The search for mismatches only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file before issuing the HI SEARCH command.

Search is not available for the when the DEFAULT language operand is used.

DISABLED

Turns off all HILITE features and removes all action bars. This benefits performance at the expense of function. Since DISABLED status is not stored in the edit profile, you need to reenter this operand each time you enter the editor. If ISREDIT HILITE DISABLED is issued by a macro, any attempts to restore highlighting within the same macro invocation are ignored.

Description

The HILITE macro command can be used to highlight, in user-specified colors, numerous language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. In addition, when HILITE is entered with no operands, a dialog appears that allows you to set default colors for the data area in non-program files, for any characters typed since the previous Enter or function key entry, and for strings located by the FIND command.

Both HI and HILIGHT are valid synonyms for HILITE.

Note: Highlighting is *not* available for edit sessions that involve the following:

- Data sets with record lengths greater than 255
- Mixed mode edit sessions (normally used when editing DBCS data)
- Formatted data.

If a macro issues HILITE in any of these situations, a return code of 12 is set.

Return Codes

The following return codes can be issued:

- | | |
|----|--|
| 0 | Normal completion. |
| 8 | Logic or search not supported in the current environment. Invalid language. |
| 12 | Hilite dialog is invalid from an edit macro or Hilite not available because of the installation defaults or because the edit panel in use is not enabled for enhanced color. |
| 20 | Severe error. Possibly extra parameters. |

IMACRO—Set or Query an Initial Macro

The IMACRO macro command saves the name of an initial macro in the current edit profile.

The IMACRO assignment statement sets or retrieves the value for the initial macro in the current profile, and places it in a variable.

See “Initial Macros” on page 29 for more information on creating and using initial macros.

Macro Command Syntax

```
ISREDIT IMACRO {name | NONE}
```

name Identifies the initial macro to be run when editing the data set type that matches this profile. This macro is run before any data is displayed.

NONE

Shows that no macro is to be run at the beginning of each edit session. The editor returns a value of NONE when no initial macro has been specified.

Assignment Statement Syntax

```
ISREDIT (varname) = IMACRO
ISREDIT IMACRO = name
```

varname

The name of a variable to contain the name of the initial macro.

name Same as macro command syntax.

Return Codes

The following return codes can be issued:

0	Normal completion
4	IMACRO set not accepted; profile is locked
12	Invalid name specified
20	Severe error.

Examples

To set the initial macro name to ISCRIPIT:

```
ISREDIT IMACRO ISCRIPIT
```

To set no initial macro:

```
ISREDIT IMACRO NONE
```

To store the name of the initial macro in the variable &IMACNAM:

```
ISREDIT (IMACNAM) = IMACRO
```

INSERT—Prepare Display for Data Insertion

The INSERT macro command appears for one or more blank lines, and allows you to fill them with data.

INSERT

Macro Command Syntax

```
ISREDIT INSERT lptr [numlines]
```

lptr A label or a relative line number that shows which line you want the inserted line or lines to follow.

numlines

The number of lines to appear for data input; these lines are not saved until they contain data. If you do not type a number or if the number you type is 1, only one data input line appears.

Description

Use the INSERT macro command for data input. Inserted lines are initialized with data from the mask line. However, they are not data lines and cannot be referred to by any macro. Inserted lines are deleted if they do not contain data.

You must specify that the line referenced on INSERT should be displayed; otherwise, you will not see the inserted line. Use LOCATE to position a line at the top of the display.

Do not use this command for adding lines with specific data; instead, use the LINE_BEFORE and LINE_AFTER assignment statements.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line number
20	Severe error.

Example

To open a 5-line area for data input after the line with the label .POINT, locate .POINT to position it to the top of the display. Then issue INSERT:

```
ISREDIT LOCATE .POINT  
ISREDIT INSERT .POINT 5
```

LABEL—Set or Query a Line Label

The LABEL assignment statement sets or retrieves the values for the label on the specified line and places the values in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = LABEL lptr  
ISREDIT LABEL lptr = labelname [level]
```

var1 The name of a variable to contain the name of the label.

var2 The name of the variable to contain the nesting level of the label. It must be a 3-character value that is left-padded with zeros.

lptr A line pointer identifying the line for which a label must be set or retrieved. A line pointer can be a label or a relative line number.

Use the LINENUM assignment statement to obtain the current relative line number of a line with a label. See the LOCATE and RESET command descriptions, which use labels to specify line ranges.

labelname

The name of the label. It must begin with a period, followed by 1 to 8 alphabetic characters, the first of which must not be Z. No special characters or numeric characters are allowed. If the label is to be level 0, it must be 5 characters or less. When you want to delete a label, set the label name to blank (' ').

The LINENUM assignment statement can be used to determine whether a label exists. For more information, refer to the description of the LINENUM assignment statement later in this chapter.

level

The highest nesting level at which this label is visible to you or to a macro. Level 0 is the highest level. Labels at this level are visible to you and to all levels of nested macros. Level 1 is not visible to you, but it is visible to all macros, and so on. The level can never exceed the current nesting level. The maximum nesting level is 255. The level number defaults to the current nesting level.

Description

A range of labels is particularly useful for commands that operate on a range of lines, such as those in the following list:

CHANGE	EXCLUDE	LOCATE	SEEK
CREATE	FIND	REPLACE	SORT
DELETE	FLIP	RESET	SUBMIT

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 Label name not returned, specified line has no label
- 8 Label set, but an existing label at the same level was deleted
- 12 Line number specified is beyond the end of data
- 20 Severe error.

Example

To get the line of data at the cursor, look for the next occurrence of the string in the variable &ARG, and then label the line if it is found and currently unlabeled:

```

ISREDIT (NAME) = LINE .ZCSR
ISREDIT FIND &ARG
IF &LASTCC = 0 THEN -
  ISREDIT (LBL,NEST) = LABEL .ZCSR
IF &LBL=&STR() THEN -
  ISREDIT LABEL .ZCSR = .POINT 0

```

LEFT—Scroll Left

The LEFT macro command scrolls data to the left of the current panel position.

LEFT

Macro Command Syntax

ISREDIT LEFT amt

amt The scroll amount, the number of columns (0 - 9999) to scroll, or one of the following operands:

MAX Displays the first page of data to the left.

HALF Displays the next half-panel of data to the left.

PAGE Displays the next full panel of data to the left.

CURSOR

Scrolls until the column on which the cursor is located becomes the first data column on the panel.

DATA Scrolls until the first column on the current panel of data becomes the last column on the next panel.

Description

The editor stops scrolling when it reaches the current BOUNDS setting. For example, if the left bound is position 9 and positions 21 to 92 are displayed, issuing ISREDIT LEFT 20 leaves positions 9 to 80 displayed, not 1 to 72.

To scroll to the left using the panel position when the macro was issued, use USER_STATE assignment statements to save and then restore the panel position operands.

If you define a macro named LEFT, it overrides the LEFT command when used from another macro. LEFT does not change the cursor position and cannot be used in an initial macro. For further information, see the BOUNDS and DISPLAY_COLUMNS descriptions.

Return Codes

The following return codes can be issued:

0	Normal completion
4	No visible lines
8	No data to display
12	Amount not specified
20	Severe error.

Example

To scroll the display to the left by the number of columns specified in variable &COL:

```
ISREDIT LEFT &COL
```

LEVEL—Set or Query the Modification Level Number

The LEVEL macro command allows you to control the modification level that is assigned to a member of an ISPF library.

The LEVEL assignment statement either sets the modification level or retrieves the current modification level and places it in a variable.

See “Version and Modification Level Numbers” on page 31 for more information about level numbers.

Macro Command Syntax

```
ISREDIT LEVEL num
```

num The modification level. It can be any number from 0 to 99.

Assignment Statement Syntax

```
ISREDIT (varname) = LEVEL
ISREDIT LEVEL = num
```

varname

The name of a variable to contain the modification level. The modification level is a 2-digit value that is left-padded with zeros.

num Same as above.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Statistics mode is off; the command is ignored
12	Invalid value specified
20	Severe error.

Examples

To reset the modification level to 1:

```
ISREDIT LEVEL = 1
```

To save the value of the modification level in variable &MODLVL:

```
ISREDIT (MODLVL) = LEVEL
```

LINE—Set or Query a Line from the Data Set

The LINE assignment statement either sets or retrieves the data from the data line specified by a line pointer, and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = LINE lptr
ISREDIT LINE lptr = data
```

varname

Specifies the name of a variable to hold the contents of the specified data line.

lptr Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.

data Specifies that the following forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)

LINE

- Merge format (*string-1 + string-2, operand + string-2, string-1 + operand*)
- Operand:
 - LINE** Data from this line is used.
 - LINE lptr**
Data from the line with the given line pointer (lptr).
 - MASKLINE**
Data from the mask line.
 - TABSLINE**
Data from the tabs line.

Description

The logical data width of the line determines how many characters are retrieved or set. See the description of the DATA_WIDTH command for information on determining the current logical data width.

You must specify the line pointer to set or retrieve a line. To set data on a line, you can use a variety of data formats: (variable), templates, or merging a line with other data. The data on the line is completely overlaid with the data specified on this command.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Data truncated (line shorter than data supplied)
8	Variable not found
12	Invalid line number
16	Variable data truncated
20	Severe error.

Examples

To set comment delimiters in columns 40 and 70, blanking the rest of the line:

```
ISREDIT LINE 1 = < 40 '&STR(/*)' 70 '&STR(*)' >
```

To overlay the first 2 columns of line 2 with //:

```
ISREDIT LINE 2 = LINE + //
```

To merge mask line data with data from variable &VAR:

```
ISREDIT LINE 3 = MASKLINE + (VAR)
```

LINE_AFTER—Add a Line to the Current Data Set

The LINE_AFTER assignment statement adds a line after a specified line in the current data set.

Assignment Statement Syntax

```
ISREDIT LINE_AFTER lptr = [DATALINE] data  
                        [INFOLINE]  
                        [MSGLINE ]  
                        [NOTELINE]
```

lptr Specifies that a line pointer must be used to identify the line after which

the new line is to be inserted. A line pointer of 0 causes the new line to be inserted at the beginning of the current data set. The line pointer can be either a label or a relative line number.

DATALINE

The line inserted is a data line.

INFOLINE

The line inserted is a temporary, non-data line. The line command area shows ===== in high intensity and the data on the line is in high intensity, also. The line can be scrolled left and right and can be as long as the current record length. An information line is protected. Once it has been added to the data, it cannot be referenced.

MSGLINE

The line inserted is a temporary, non-data line. The line command area contains ==MSG> in high intensity and the data on the line is also in high intensity. A message line has a data length of 72 characters, regardless of the data width. Once it has been added to the data, it cannot be referenced.

NOTELINE

The line inserted is a temporary, non-data line. The line command area shows =NOTE= in high intensity and the data on the line is in low intensity. A note line has a data length of 72 characters, regardless of the data width. It cannot be referenced after it is added to the data.

data Specifies that the following data formats can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col,string* >)
- Merge format (*string-1 + string-2, operand + string-2, string-1 + operand*)
- Operand:
 - LINE** Data from the line preceding this line.
 - LINE lptr** Data from the line with the given line pointer (lptr).
 - MASKLINE** Data from the mask line.
 - TABSLINE** Data from the tabs line.

Description

This statement is used for adding lines with specific data. Use the INSERT command for data input.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 Data truncated
- 12 Invalid line number
- 20 Severe error.

LINE_AFTER

Examples

To put a new line that contains the string:

This is the new top line of the data

as the first line of the data set:

```
ISREDIT LINE_AFTER 0 = "This is the new top line of the data"
```

To put the contents of the line labeled .START on a new line following the line labeled .END:

```
ISREDIT LINE_AFTER .END = LINE .START
```

To put the contents of the mask line modified by the variable &DATA after the line whose number is in variable &N:

```
ISREDIT LINE_AFTER &N = MASKLINE + &DATA
```

LINE_BEFORE—Add a Line to the Current Data Set

The LINE_BEFORE assignment statement adds a line before a specified line in the current data set.

Assignment Statement Syntax

```
ISREDIT LINE_BEFORE lptr = [DATALINE] data  
                           [INFOLINE]  
                           [MSGLINE ]  
                           [NOTELINE]
```

lptr Specifies that a line pointer must be used to identify the line before which the new line is to be inserted. A line pointer of 0 is invalid. The line pointer can be either a label or a relative line number.

DATALINE

The line inserted is a data line.

INFOLINE

The line inserted is a temporary, non-data line. The line command area shows ===== in high intensity and the data on the line is in high intensity, also. The line can be scrolled left and right and can be as long as the current record length. An information line is protected. Once it has been added to the data, it cannot be referenced.

MSGLINE

The line inserted is a temporary, non-data line. The line command area contains ==MSG> in high intensity and the data on the line is also in high intensity. A message line has a data length of 72 characters, regardless of the data width. Once it has been added to the data, it cannot be referenced.

NOTELINE

The line inserted is a temporary, non-data line. The line command area shows =NOTE= in high intensity and the data on the line is in low intensity. A note line has a data length of 72 characters, regardless of the data width. It cannot be referenced once it has been added to the data.

data Specifies that the following data formats can be used:

- Simple string
- Delimited string
- Variable

- Template (*< col,string >*)
- Merge format (*string-1 + string-2, operand + string-2, string-1 + operand*)
- Operand (those allowed follow):
 - LINE** Data from the line following this line.
 - LINE lptr**
Data from the line with the given line pointer (lptr).
 - MASKLINE**
Data from the mask line.
 - TABSLINE**
Data from the tabs line.

Description

The LINE_BEFORE statement is used for adding lines with specific data. Use INSERT for data input.

Return Codes

The following return codes can be issued:

- | | |
|----|---------------------|
| 0 | Normal completion |
| 4 | Data truncated |
| 12 | Invalid line number |
| 20 | Severe error. |

Examples

To put the contents of the line labeled .START on a new line preceding the line labeled .END:

```
ISREDIT LINE_BEFORE .END = LINE .START
```

To put the contents of the mask line modified by the variable &DATA before the line whose number is in variable &N:

```
ISREDIT LINE_BEFORE &N = MASKLINE + &DATA
```

LINENUM—Query the Line Number of a Labeled Line

The LINENUM assignment statement retrieves the current relative line number of a specified label, and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = LINENUM label
```

varname

The name of the variable to contain the line number of the line with the specified label. The line number is a 6-digit value that is left-padded with zeros.

label The name of the label for the line whose line number is needed.

Return Codes

The following return codes can be issued:

- | | |
|---|-------------------|
| 0 | Normal completion |
| 4 | Line 0 specified |

LINENUM

8	Label specified, but not found (variable set to 0)
12	Invalid line number
20	Severe error.

Description

Once the line number is retrieved and placed in a variable, it can be used in arithmetic operations. Note that line numbers are relative to the position of the line: first=1, second=2, and so on. Therefore, the value returned by the LINENUM assignment statement is not always be correct if lines are added or deleted before the line number is obtained.

Examples

To determine the number of lines in the data set and set variable &VAR to the last line number:

```
ISREDIT (VAR) = LINENUM .ZLAST
```

That number is 0 if there are no lines.

To set variable &NUM to the line number containing the label .MYLAB:

```
ISREDIT (NUM) = LINENUM .MYLAB
```

LOCATE—Locate a Line

The LOCATE macro command scrolls up or down to a specified line. The line is then displayed as the first line on the panel. There are two forms of LOCATE, specific and generic.

Specific Locate Syntax

The specific form of LOCATE positions a particular line at the top of the panel. You must specify either a line number or a label.

```
ISREDIT LOCATE lptr
```

lptr Specifies that a line pointer must be used for the target. A line pointer can be a label or a relative line number.

If the line pointer is a label, it must be a label that you have previously defined or a editor-defined label, such as .ZFIRST or .ZLAST.

Generic Locate Syntax

The generic LOCATE command positions the panel to the first, last, next, or previous occurrence of a particular kind of line.

```
ISREDIT LOCATE [FIRST] {CHANGE } [lptr-range]
                [LAST ] {COMMAND }
                [NEXT ] {ERROR }
                [PREV ] {EXCLUDED}
                {LABEL }
                {SPECIAL }
```

FIRST Searches from the first line, proceeding forward.

LAST Searches from the last line, proceeding backward.

NEXT Searches from the first line of the page displayed, proceeding forward.

PREV Searches from the first line of the page displayed, proceeding backward.

CHANGE

Searches for a line with a change flag (=CHG>).

COMMAND

Searches for a line with a pending line command.

ERROR

Searches for a line with an error flag (==ERR>).

EXCLUDED

Searches for an excluded line.

LABEL

Searches for a line with a label.

SPECIAL

Searches for any special non-data (temporary) line:

- Bounds line flagged as =BNDS>
- Column identification lines flagged as =COLS>
- Information lines flagged as =====
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged as =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>.

lptr-range

Specifies that two line pointers are required to specify a range of lines in which to search. A line pointer can be a label or a relative line number. Specifying one line pointer is invalid. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

Note: If you try to locate a line using a label that has not been assigned, you will receive a return code of 20. To avoid this, use the LINENUM assignment statement. When using the LINENUM statement, a return code of 8 will be issued if the label does not exist.

```
ISREDIT X = LINENUM .LABEL
```

Return Codes

The following return codes can be issued:

- | | |
|----|--------------------------|
| 0 | Normal completion |
| 4 | Line not located |
| 8 | Empty member or data set |
| 20 | Severe error. |

Examples

To locate the next occurrence of a line with a label:

```
ISREDIT LOCATE NEXT LABEL
```

To locate the first occurrence of a special (non-data) line:

```
ISREDIT LOCATE FIRST SPECIAL
```

To locate the last excluded line:

LOCATE

ISREDIT LOCATE LAST X

To locate the previous line that contains an unprocessed line command:

ISREDIT LOCATE PREV CMD

LRECL—Query the Logical Record Length

The LRECL assignment statement returns the maximum space, in bytes, available for data, COBOL number fields, and sequence number fields.

Assignment Statement Syntax

ISREDIT (varname) = LRECL

varname

The name of a variable to contain the logical record length of the data being edited. The logical record length is a 3-digit value that is left-padded with zeros. If the variable is VDEFINED in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, a length of 3 or 4 is allowed in cases where no data loss occurs.

Description

The value returned by the LRECL assignment statement includes the sequence number field and, for fixed-length records, the COBOL number field, if these number fields are used. For variable-length records, the value returned by LRECL does not include the 4-byte record descriptor word (RDW).

Use the DATA_WIDTH assignment statement to get the maximum space, in bytes, available for data.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid command format
20	Severe error.

Example

To check the logical record length of the data and process the data if the logical record length (LRECL) is 80:

```
ISREDIT (RECLEN) = LRECL
IF &RECLEN = 80 THEN -
...
```

MACRO—Identify an Edit Macro

The MACRO macro command identifies a command as a macro.

Macro Command Syntax

```
ISREDIT MACRO [(var1 [,var2,...])] [PROCESS ]
[NOPROCESS]
```

var1, var2,

The names of the variables that contain parameters, if a macro allows parameters to be specified. Parameters are parsed and placed into the named variables in the order in which they are typed. The last variable contains any remaining parameters. Variables that do not receive a parameter are set to a null string. A parameter is a simple or quoted string, separated by blanks or commas. Quotes can be single (') or double ("), but must be matched at the beginning and end of the string.

PROCESS

Immediately processes all changes and line commands typed at the keyboard.

NOPROCESS

Processes changes and line commands typed at the keyboard when the macro completes processing or a PROCESS statement is found. NOPROCESS must be used if the macro is to use line commands as input to its processing.

See "PROCESS—Process Line Commands" on page 377 for more information.

Description

The MACRO macro command is required in all macros. It must be the first command in a CLIST or REXX macro that is not a CLIST or REXX statement. Similarly, it also must be the first edit command in a program macro.

Return Codes

The following return codes may be returned:

0	Normal completion
8	No parameters are permitted for this processing
12	Syntax Error
20	Severe error.

Examples

To begin a macro, first accepting a member name and optionally a line number range to be placed in the variable &PARM:

```
ISREDIT MACRO (PARM)
ISREDIT COPY AFTER .ZCSR &PARM
```

To begin a macro, checking parameters before processing panel information, testing for missing input, excess input, and non-numeric input:

```
ISREDIT MACRO NOPROCESS (COL,X)
IF &STR(&COL) = &STR() THEN -
  ISREDIT (,COL) = DISPLAY_COLS
ELSE -
  IF &DATATYPE(&COL) = CHAR THEN -
    GOTO MSG
  IF &STR(&X) ^= &STR() THEN -
    GOTO MSG
ISREDIT PROCESS
```

MACRO_LEVEL—Query the Macro Nesting Level

The MACRO_LEVEL assignment statement retrieves the current nesting level of the macro being run, and places the nesting level in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = MACRO_LEVEL
```

varname

The name of a variable to contain the macro nesting level. The nesting level is a 3-digit value that is left-padded with zeros.

Description

The nesting level can be any number between 1 (a macro that you start) and 255. MACRO_LEVEL is used to adjust processing based on whether the macro is started by you or called by another macro. It is required if labels are to be set for the starter of this macro. See "LABEL—Set or Query a Line Label" on page 352 for more information.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid command format
20	Severe error.

Example

To set the label for the caller of the macro at 1 less than the current level:

```
ISREDIT (NESTLEV) = MACRO_LEVEL
ISREDIT LABEL .ZCSR = .XSTR &EVAL(&NESTLEV -1)
```

MASKLINE—Set or Query the Mask Line

The MASKLINE assignment statement sets or retrieves the value of the mask line, which controls the display formatting of your input.

Assignment Statement Syntax

```
ISREDIT (varname) = MASKLINE
ISREDIT MASKLINE = data
```

varname

The name of a variable containing maskline contents.

data Specifies that the following forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< col,string >)
- Merge format (*string-1 + string-2, operand + string-2, string-1 + operand*)
- Operand:

LINE lptr

Data from the line with the given line pointer (lptr).

MASKLINE

Data from the mask line.

TABSLINE

Data from the tabs line.

Description

The MASKLINE assignment statement places the mask line contents in a variable or sets the mask line from a variable. The mask line can contain any characters and serves to initialize inserted lines to the value of the mask line. See the description of templates in “Overlays and Templates” on page 106 for more information on the setting of a mask line.

Be careful not to destroy a DBCS string in the mask line. If shift-out (SO) or shift-in (SI) characters in a mask line are overlaid through the MASKLINE statement, the result is unpredictable.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Data truncated
16	Variable data truncated
20	Severe error.

Examples

To set the mask line to place comment delimiters starting at lines 40 and 70:

```
ISREDIT MASKLINE = <40 '&STR(/*)' 70 '&STR(/*)' >
```

To set the mask line to blanks:

```
ISREDIT MASKLINE = " "
```

MEMBER—Query the Current Member Name

The MEMBER assignment statement retrieves the name of the library member currently being edited, and places it in a variable. If a sequential data set is being edited, the variable is set to blanks.

Assignment Statement Syntax

```
ISREDIT (varname) = MEMBER
```

varname

The name of a variable to contain the name of the library member currently being edited.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid command format

MEMBER

20 Severe error.

Example

To determine if you are editing a library member with a prefix of MIN:

```
ISREDIT (MEMNAME) = MEMBER
IF &SUBSTR(1:3,&MEMNAME ) = MIN THEN -
....
```

MEND—End a Macro in the Batch Environment

The MEND macro command ends a macro that is running in the batch environment. It was required for CLISTs that ran in the batch environment using the MVS/370 operating system. It is not required for MVS/XA*, but can be used.

Macro Command Syntax

```
ISREDIT MEND
```

Description

MEND must be the last processable instruction before the CLIST EXIT statement. If your macro is not running in the batch environment, or if it is not a CLIST, the MEND command is ignored. Refer to *ISPF User's Guide* for more information on running batch jobs.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

If you want to have the proper return code when using the MEND command in a CLIST, the value of the &LASTCC variable must be passed as the exit code. For example:

```
ISREDIT MEND
EXIT CODE(&LASTCC)
```

Example

To update the data and save the changes, first set it up as an initial macro in the batch environment. Then, end that edit session and the macro.

```
ISREDIT MACRO
ISREDIT CHANGE ALL PREFIX BDG BIV
ISREDIT END
ISREDIT MEND
EXIT
```

MODEL—Copy a Model into the Current Data Set

The model name form of the MODEL macro command copies a specified dialog development model before or after a specified line.

The class name form of the MODEL macro command changes the model class that the editor uses to determine the model you want. For more information on edit models, see Chapter 4. Using Edit Models.

Macro Command Model Name Syntax

```
ISREDIT MODEL model-name [qualifier] {AFTER } lptr [NOTES ]
                               {BEFORE}          [NONOTES]
```

model-name

The name of the model to be copied, such as VGET for the VGET service model. This operand can also be one of the options listed on a model selection panel, such as V1 for the VGET service model. However, to use these options with the MODEL macro command, you must already know what they are or else display a model selection panel by using the MODEL primary command. The MODEL macro command does not display model selection panels. Refer to *ISPF Planning and Customizing* for a list of models and model names.

qualifier

The name of a model on a secondary model selection panel, such as TBCREATE for the TBCREATE service model. This operand can also be one of the options listed on a model selection panel, such as G1 for the TBCREATE service model.

For example, a model selection panel allows you to enter T1 to choose table models. It then displays another model selection panel for choosing table models, such as G1 for the TBCREATE service model. Therefore, your MODEL macro command could use either TABLES or T1 as the model-name operand and either TBCREATE or G1 as the qualifier operand. The simplest way would be to use TBCREATE or G1 as the model-name operand and omit the qualifier operand.

To use options with the MODEL macro command, you must already know what they are or else display a model selection panel by using the MODEL primary command. The MODEL macro command does not display model selection panels. Refer to *ISPF ISPF Planning and Customizing* for a list of models and model names.

AFTER

Specifies that the model is to be copied after the line specified by lptr.

BEFORE

Specifies that the model is to be copied before the line specified by lptr.

lptr

A line pointer must be used to specify where the model should be copied. A line pointer can be a label or a relative line number.

NOTES

Explanatory notes appear when a model is copied.

NONOTES

No explanatory notes appear.

Macro Command Class Name Syntax

```
ISREDIT MODEL CLASS class-name
```

CLASS

Specifies that the current model class is to be replaced by class-name. The new class name is used for all models from that point on, until you change the model class again or end the edit session.

MODEL

class-name

Specifies the model class for the current edit session. It must be a name on the Model Classes panel or an allowable abbreviation. The model class coincides with the type of model, such as REXX, COBOL, or FORTRAN.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line pointer (lptr)
20	Severe error.

Example

To copy the VGET model at the end of the current data:

```
ISREDIT MODEL VGET AFTER .ZL
```

MOVE— Move a Data Set or a Data Set Member

The MOVE macro command specifies a member of the partitioned data set being edited to be moved into the data being edited.

Macro Command Syntax

```
ISREDIT MOVE member {AFTER } lptr  
              (member){BEFORE}  
              data set name  
              data.set.name(member)
```

member

A member of the ISPF library or partitioned data set you are editing.

data set name

A partially or fully qualified data set name. If the data set is partitioned you must include a member name in parentheses.

AFTER

Specifies that the member is to be moved after the target specified by lptr.

BEFORE

Specifies that the member is to be moved before the target specified by lptr.

lptr Identifies the target of the move. A line pointer can be a label or a relative line number. If the line pointer is a label, it can be either a label that you define or one of the editor-defined labels, such as .ZF and .ZL.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned a like-named member is copied. If a like-named member does not exist, the name is considered to be a partially qualified data set name.

Description

The member or data set is deleted after the move. For a concatenated sequence of ISPF libraries, the deletion occurs only if the member was in the first library of the concatenation sequence.

See “Copying and Moving Data” on page 50 if you need more information.

Return Codes

The following return codes can be issued:

0	Normal completion
8	End of data before last record read or the specified data set is in use
12	Invalid line pointer (lptr); member not found or BLDL error
16	End of data before first record read
20	Syntax error (invalid name, incomplete range), or I/O error.

Examples

To move the contents of member ABC after the first line in the current data:

```
ISREDIT MOVE ABC AFTER .ZF
```

To move all of data set MOVECOPY.DATA before the line where the cursor is currently positioned:

```
ISREDIT MOVE MOVECOPY.DATA BEFORE .ZCSR
```

NONUMBER—Turn Off Number Mode

The NONUMBER macro command turns off number mode, which controls the numbering of lines in the current data.

Syntax

```
ISREDIT NONUMBER
```

The NONUMBER macro command has no operands.

Description

You can also use the NUMBER OFF macro command to turn off number mode.

When number mode is off, NONUMBER prevents any verification of valid line numbers, generation of sequence numbers, and the renumbering of lines that normally occurs when autonum mode is on.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Example

To turn number mode off by using the NONUMBER command:

```
ISREDIT NONUMBER
```

NOTES—Set or Query Note Mode

The NOTES macro command sets note mode, which controls whether notes are to appear when a dialog development model is inserted into the data.

The NOTES assignment statement either sets note mode, or retrieves the setting of note mode and places it in a variable.

See “MODEL—Copy a Model into the Current Data Set” on page 257 for information about copying dialog development models.

Macro Command Syntax

```
ISREDIT NOTES [ON ]  
              [OFF]
```

ON Displays explanatory notes when a model is copied into the data being edited.

OFF Does not display explanatory notes.

Assignment Statement Syntax

```
ISREDIT (varname) = NOTES  
ISREDIT NOTES = [ON ]  
                [OFF]
```

varname

The name of a variable to contain the value of note mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Return Codes

The following return codes can be issued:

0 Normal completion

20 Severe error.

Examples

To set note mode off:

```
ISREDIT NOTES = OFF
```

To store the value of note mode in variable &NOTEMODE:

```
ISREDIT (NOTEMODE) = NOTES
```

NULLS—Set or Query Nulls Mode

The NULLS macro command sets nulls mode, which determines whether trailing blanks in each data field are written to the panel as blanks or nulls.

The NULLS assignment statement either sets nulls mode or retrieves the setting of nulls mode and places it in a variable.

Macro Command Syntax

```
ISREDIT NULLS [ON STD]
               [ON ALL]
               [OFF  ]
```

ON STD

Specifies that in fields that contain any blank trailing space, the space is to be written as one blank followed by nulls. If the field is entirely empty, it is written as all blanks.

ON ALL

Specifies that all trailing blanks and all-blank fields are written as nulls.

OFF Specifies that trailing blanks in each data field are written as blanks.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = NULLS
ISREDIT NULLS = [ON STD]
                [ON ALL]
                [OFF  ]
```

var1 The name of a variable to contain either ON or OFF.

var2 The name of a variable to contain ALL, STD, or blanks.

ON STD

Same as macro command syntax.

ON ALL

Same as macro command syntax.

OFF Same as macro command syntax.

Description

The term *data field* normally refers to the 72 characters of data on each line. Using hardware tabs, however, you can split each line into multiple fields. See “TABS—Define Tabs” on page 289 for more details.

Blank characters (X'40') and null characters (X'00') both appear as blanks. When you use the I (insert) line command, the data entry area appears as blanks for NULLS ON STD and as nulls for NULLS ON ALL.

Trailing nulls simplify use of the Ins (insert) key on the IBM 3270 keyboard. You can use this key to insert characters on a line if the line contains trailing nulls.

Besides using NULLS, you can create nulls at the end of a line by using the Erase EOF or Del (delete) key. Null characters are never stored in the data; they are always converted to blanks.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

NULLS

Examples

To set nulls mode on with blank trailing space written as one blank followed by nulls and empty fields written as all blanks:

```
ISREDIT NULLS = ON STD
```

To set nulls mode off and thus have trailing blanks in each data field:

```
ISREDIT NULLS = OFF
```

NUMBER—Set or Query Number Mode

The NUMBER macro command sets number mode, which controls the numbering of lines in the current data.

The NUMBER assignment statement either sets number mode, or retrieves the setting of number mode and places it in variables.

Macro Command Syntax

```
ISREDIT NUMBER [ON ] [STD      ] [DISPLAY]
                [OFF] [COBOL   ]
                  [STD COBOL]
                  [NOSTD]
                  [NOCOBOL]
                  [NOSTD NOCOBOL]
```

ON Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. You can also use the RENUM command to turn number mode on and renumber lines.

The editor interprets the STD, COBOL, and DISPLAY operands only when number mode is turned on.

OFF Turns number mode off. You can also use the NONUMBER command to turn number mode off.

STD Numbers the data in the standard sequence field. This is the default for all non-COBOL data set types.

COBOL Numbers the data in the COBOL field. This is the default for all COBOL data set types.

Note: The NUMBER ON COBOL mode is not supported for formatted data sets.

Attention: If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. You can also use CANCEL at any time to end the edit session without saving the data.

STD COBOL

Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. The COBOL numbers

can be out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

NOSTD

Turns standard number mode off.

NOCOBOL

Turns COBOL number mode off.

NOSTD NOCOBOL

Turns both the standard number mode and COBOL number mode off.

DISPLAY

Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. Automatic left or right scrolling is performed, if required, so that the leftmost column of the data window is the first column displayed.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = NUMBER
ISREDIT NUMBER = [ON ] [STD ] [DISPLAY]
                  [OFF] [COBOL ]
                    [STD COBOL]
                    [NOSTD]
                    [NOCOBOL]
                    [NOSTD NOCOBOL]
```

var1 The name of a variable to contain either ON or OFF.

var2 The name of a variable to contain one of the eight combinations in the following list:

NOSTD	NOCOBOL	DISPLAY
STD	NOCOBOL	DISPLAY
NOSTD	COBOL	DISPLAY
STD	COBOL	DISPLAY
NOSTD	NOCOBOL	NODISPL
STD	NOCOBOL	NODISPL
NOSTD	COBOL	NODISPL
STD	COBOL	NODISPL

The value STD, COBOL, or DISPLAY can be placed in *var2*, even when *var1* is set to off. This allows the macro to save and restore number mode. It also allows the macro to set number mode off, while specifying defaults to be used when number mode is changed to on.

ON Same as for macro command syntax.

OFF Same as for macro command syntax.

STD Same as for macro command syntax.

COBOL

Same as for macro command syntax.

NOSTD

Turns standard number mode off.

NUMBER

NOCOBOL

Turns COBOL number mode off.

NOSTD NOCOBOL

Turns both the standard number mode and COBOL number mode off.

STD COBOL

Same as for macro command syntax.

DISPLAY

Same as for macro command syntax.

Description

When number mode is on, NUMBER verifies that all lines have valid numbers in ascending sequence. It renumbers any lines that are either unnumbered or out of sequence, but it does not otherwise change existing numbers.

In number mode, the editor automatically generates sequence numbers in the data for new lines that are created when data is copied or inserted. The editor also automatically renumbers the data when it is saved if autonum mode is in effect.

If the number overlays the shift-in (SI) or shift-out (SO) characters, the double-byte characters are displayed incorrectly and results are unpredictable.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Example

To save the current value of number mode, set number mode off for processing, and then restore the value of number mode:

```
ISREDIT (STAT,VALUE) = NUMBER  
ISREDIT NUMBER OFF  
...  
ISREDIT NUMBER = (STAT VALUE)
```

PACK—Set or Query Pack Mode

The PACK macro command sets pack mode, which controls whether the data is stored in packed format.

The PACK assignment statement either sets pack mode, or retrieves the setting of pack mode and places it in a variable.

The PACK command saves the pack mode setting in the edit profile. See “Packing Data” on page 19 for more information about packing data.

Macro Command Syntax

```
ISREDIT PACK [ON ]  
              [OFF]
```

ON Saves data in packed format.

OFF Saves data in unpacked (standard) format.

If you change pack mode, data is written when an END command is issued.

Assignment Statement Syntax

```
ISREDIT (varname) = PACK
ISREDIT PACK = [ON ]
                [OFF]
```

varname

The name of a variable to contain the setting of pack mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Example

To set pack mode off:

```
ISREDIT PACK OFF
```

PASTE—Move or Copy Lines from Clipboard

The PASTE macro command moves or copies lines from a clipboard into an edit session.

Syntax

```
ISREDIT PASTE [AFTER] lptr [clipboardname]
              [BEFORE] [KEEP]
```

clipboardname

The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

BEFORE

The destination of the data that is being transferred from the clipboard. BEFORE copies the data *before* the specified label (lptr).

AFTER

The destination of the data that is being transferred from the clipboard. AFTER copies the data *after* the specified label (lptr).

KEEP Records are copied and not removed from the clipboard. If you omit this keyword, the records are removed from the clipboard.

PASTE

Description

PASTE copies or moves lines from a specified clipboard to the current edit session. If lines in the clipboard are longer than the lines in the edit session, they are truncated.

The portion of the line that is saved in the clipboard is only the data portion of the line. Line numbers are *not* saved. If the data was CUT from a data set that had sequence numbers and is PASTEd into an edit session without sequence numbers, or if it was CUT from a data set without sequence numbers and PASTEd into a session with sequence numbers, some shifting of data is likely to occur.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Parameter error. Clipboard is empty or does not exist.
20	Severe error.

Examples

To paste data from the default clipboard to the line after the last line in the edit session:

```
ISREDIT PASTE AFTER .ZLAST
```

To paste data from the default clipboard to the line after the first line in the edit session, without clearing the contents of the clipboard:

```
ISREDIT PASTE AFTER .ZFIRST KEEP
```

PRESERVE

The PRESERVE macro command enables or disables the saving of trailing blanks in the editor. This enables you to override the setting for the field on the edit entry panel called **Preserve VB record length**.

Macro Command Syntax

```
ISREDIT PRESERVE [ON ]  
                [OFF]
```

ON The editor saves all trailing blanks in the record.

OFF Turns truncation on. ISPF removes trailing blanks when saving variable length files. If a line is empty ISPF saves 1 blank. If the line is eight characters long, ISPF pads the record to 9 characters by appending a blank to the end.

Assignment Statement Syntax

```
ISREDIT (varname) = PRESERVE  
ISREDIT PRESERVE = [ON | OFF]
```

varname

The name of a variable to contain the setting of PRESERVE mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

Description

PRESERVE ON causes the editor to save trailing blanks for variable length files. The number of blanks saved for a particular record is determined by one of the following:

- the original record length of the record when it was read in to the editor
- the number of blanks required to pad the record length specified by the SAVE_LENGTH edit macro command
- the length of the record that was saved on disk during a previous SAVE request in the same edit session.

PRESERVE OFF causes the editor to truncate trailing blanks. If a line is empty ISPF saves 1 blank. If the line is eight characters long, ISPF pads the record to 9 characters by appending a blank to the end.

Use of the PRESERVE command does not prevent the editor from working on data past the specified record length. The length set and returned by the PRESERVE command is only used when the data is written and does not affect the operation of other edit functions.

Return Codes

The following return codes can be issued:

0	Normal completion
6	Record format is not variable.
16	Error setting variable.
20	Severe error.

Examples

To save the value of the PRESERVE mode in variable &TRMODE:

```
ISREDIT (TRMODE) = PRESERVE
```

To enable the editor to remove trailing blanks when the data is saved:

```
ISREDIT PRESERVE OFF
```

PROCESS—Process Line Commands

The PROCESS macro command allows the macro to control when line commands or data changes typed at the keyboard are processed.

Macro Command Syntax

```
ISREDIT PROCESS [DEST] [RANGE cmd1 [cmd2]]
```

DEST Specifies that the macro can capture an A (after) or a B (before) line command that you enter. The .ZDEST label is set to the line preceding the insertion point. If A or B is not entered, .ZDEST points to the last line in the data.

RANGE

Must be followed by the names of one or two line commands, either of which you can enter. Use the RANGE_CMD assignment statement to

PROCESS

return the value of the line command entered. This allows the macro to define and then capture a line command that you enter. It can also modify its processing based on which of the two commands was entered.

cmd1 and cmd2

Specifies one or two line command names, which can be 1 to 6 characters; however, if the name is 6 characters long it cannot be used as a block format command (to specify multiple lines) by doubling the last character. The name can contain any alphabetic or special character except blank, hyphen (-), or apostrophe ('). It cannot contain any numeric characters.

The .ZFRANGE label is set to the first line identified by the line command that you have entered, and .ZLRANGE is set to the last line. They can refer to the same line. If the expected RANGE line command was not entered, .ZFRANGE points to the first line in the data and .ZLRANGE points to the last line in the data.

Description

If a line is retrieved before the PROCESS macro command is called, changes made to this line will not be seen. The DEST and RANGE operands allow the macro to identify the line commands that you can enter as additional input to the macro.

This command cannot be specified without first coding the MACRO command with a NOPROCESS operand.

For more information about using the PROCESS command, see "Using the PROCESS Command and Operand" on page 116.

Return Codes

The following return codes can be issued:

- 0 Normal completion.
- 4 Range expected by macro, but you did not specify it; defaults set.
- 8 Destination expected by macro, but you did not specify it; defaults set.
- 12 Both range and destination expected by macro, but you did not specify them; defaults set.
- 16 You entered incomplete or conflicting line commands.
- 20 Severe error.

Note: ISPF does not consider a return code of 12 from the PROCESS edit macro command an error and does not terminate a macro that receives a return code of 12 from the PROCESS edit macro.

Examples

To set up the macro to process the line commands * and # (defined by the macro writer):

```
ISREDIT MACRO NOPROCESS
ISPEXEC CONTROL ERRORS RETURN
ISREDIT PROCESS RANGE * #
IF &LASTCC >= 16 THEN EXIT CODE(&LASTCC)
ISREDIT (CMD) = RANGE_CMD
```

```

ISREDIT (FIRST) = LINENUM .ZFRANGE
ISREDIT (LAST) = LINENUM .ZLRANGE
IF &STR(&CMD) = &STR(*) THEN -
...

```

To place data depending on the location of the A (after) or B (before) line command:

```

ISREDIT MACRO NOPROCESS
ISREDIT PROCESS DEST
ISREDIT LINE_AFTER .ZDEST = "&DATA"

```

To allow processing of the A and B destination line commands and the specification of a range by using the * line command (defined by the macro writer):

```

ISREDIT MACRO NOPROCESS
ISREDIT PROCESS DEST RANGE *

```

See “Using the PROCESS Command and Operand” on page 116.

PROFILE—Set or Query the Current Profile

The control form of the PROFILE macro command appears your current edit profile, defines a new edit profile, or switches to a different edit profile.

The lock form of the PROFILE macro command locks or unlocks the current edit profile.

The PROFILE assignment statement retrieves the name and lock status of the current edit profile and stores those values in variables.

Macro Command Profile Control Syntax

```
ISREDIT PROFILE [name] [number]
```

name The profile name. It can consist of up to 8 alphanumeric characters, the first of which must be alphabetic. The edit profile table is searched for an existing entry with the same name. That profile is then read and used. If one is not found, a new entry is created in the profile table.

If you omit this operand, the current edit profile is used.

number

The number of lines, from 0 through 8, of profile data to be displayed. When you type 0 as the number, no profile data is displayed. When you omit the number operand, the profile modes appear; the =MASK> and =TABS> lines are displayed if they contain data, followed by the =COLS> line.

The =BNDS> line does not appear if it contains the default boundary positions. It does appear when the bounds are set to something other than the default, and no ‘number’ parameter is entered into the PROFILE command.

For more information about displaying and defining a profile, see “Displaying or Defining an Edit Profile” on page 21.

Macro Command Profile Lock Syntax

```
ISREDIT PROFILE {LOCK | UNLOCK}
```

PROFILE

LOCK Specifies that the current values in the profile are saved in the edit profile table and are not modified until the profile is unlocked. The current copy of the profile can be changed, either because of commands you enter that modify profile values (BOUNDS and NUMBER, for example) or because of differences in the data from the current profile settings. However, unless you unlock the edit profile, the saved values replace the changes when you end the edit session.

Caps, number, stats, and pack mode are automatically changed to fit the data. These changes occur when the data is first read or when data is copied into the data set. Message lines (==MSG>) are inserted in the data set to show you which changes occurred.

Note: To force caps, number, stats, or pack mode to a particular setting, use an initial macro. Be aware, however, that if you set number mode on, data may be overlaid.

UNLOCK

Specifies that the editor saves changes to profile values.

See “Locking an Edit Profile” on page 23 for more information about locking and unlocking the profile.

Macro Command Profile Reset Syntax

ISREDIT PROFILE RESET

RESET

Specifies that the ZEDFAULT profile is to be removed and the site-wide configuration for new edit profiles is to be used.

See “Locking an Edit Profile” on page 23 for more information about locking and unlocking the profile.

Assignment Statement Syntax

ISREDIT (var1,var2) = PROFILE

var1 The name of a variable to contain the name of the current edit profile.

var2 The name of a variable to contain the profile status, LOCK or UNLOCK.

Description

Profile names cannot be set by an assignment statement. Instead, use PROFILE to change a profile name, thereby changing the current edit profile and the edit profile values.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Example

To check the lock status of the profile and perform processing if the profile is locked:

```
ISREDIT (,STATUS) = PROFILE
IF &STATUS = LOCK THEN -
...
```

RANGE_CMD—Query a Command That You Entered

The RANGE_CMD assignment statement identifies the name of a line command entered from the keyboard and processed by a macro.

Assignment Statement Syntax

```
ISREDIT (varname) = RANGE_CMD
```

varname

The name of a variable to contain the line command that you entered.

Description

The macro must first issue a PROCESS command to identify all line commands to be processed by this macro. A particular line command within a range can be found by using the RANGE_CMD. For instance, if the following PROCESS command is issued by a macro:

```
PROCESS RANGE Q $
```

The RANGE_CMD statement returns either a Q or a \$. If a range such as Q5 is entered, only Q is returned.

Return Codes

The following return codes can be issued:

```
0      Normal completion
4      Line command not set
8      Line command setting not acceptable
20     Severe error.
```

Example

To determine which line command (* or #) you entered and to process the line command (defined by the macro writer):

```
ISREDIT MACRO NOPROCESS
ISREDIT PROCESS RANGE * #
ISREDIT (CMD) = RANGE_CMD
IF &STR(&CMD) = &STR(*) THEN -
...
ELSE IF &STR(&CMD) = &STR(#) THEN -
.....
```

RCHANGE—Repeat a Change

The RCHANGE command repeats the change requested by the most recent CHANGE command.

Macro Command Syntax

```
ISREDIT RCHANGE
```

RCHANGE

Description

You can use this command to repeatedly change other occurrences of the search string. After a *string* NOT FOUND message appears, the next RCHANGE issued starts at the first line of the current range for a forward search (FIRST or NEXT specified) or the last line of the current range for a backward search (LAST or PREV specified).

Return Codes

The following return codes can be issued:

- | | |
|----|---|
| 0 | Normal completion |
| 4 | String not found |
| 8 | Change error (<i>string-2</i> longer than <i>string-1</i> and substitution was not performed on at least one change) |
| 12 | Syntax error |
| 20 | Severe error. |

Example

To perform a single-line change and then repeat the change from the top if the string was not found:

```
ISREDIT CHANGE C'. the' C'. The' 1 8
IF &LASTCC = 4 THEN-
  ISREDIT RCHANGE
```

RECFM—Query the Record Format

The RECFM assignment statement retrieves the record format of the data set being edited, and places the value in a variable.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = RECFM
```

var1 The name of a variable to contain the type of record format of the data being edited, either F or V:

F Fixed-length records.

V Variable-length records.

var2 The name of a variable to contain the remaining record format information of the data being edited, in the combination of M, A, S, BM, BA, BS, BSM, or BSA:

B Blocked records.

S Standard or spanned records.

M Machine print control character records.

A ASA print control character records.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Example

To place the type of record format in variable RECFM1 and then use either the logical data width (for a fixed data set) or the right display column (for a variable data set):

```
ISREDIT (RECFM1) = RECFM
IF &RECFM1 = F THEN -
  ISREDIT (WIDTH) = DATA_WIDTH
ELSE -
  ISREDIT (,WIDTH) = DISPLAY_COLS
```

To place the remaining record format information in variable RECFM2:

```
ISREDIT (,RECFM2) = RECFM
```

To place the type of record format information in variable RECFM1, and the remaining record format information in variable RECFM2:

```
ISREDIT (RECFM1,RECFM2) = RECFM
```

RECOVERY—Set or Query Recovery Mode

The RECOVERY macro command sets edit recovery mode, which allows you to recover data after a system failure or power outage.

The RECOVERY assignment statement either sets edit recovery mode, or retrieves the edit recovery mode setting and places it in a variable.

Macro Command Syntax

```
ISREDIT RECOVERY [ON [SUSP]]
                  [OFF [WARN]]
                  [OFF NOWARN]
```

ON The system creates and updates a recovery data set for each change thereafter.

OFF The system does not create and update a recovery set.

WARN

This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN

This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP This operand, when specified with the ON operand has no function. It allows existing macros which save and restore the recovery state to continue working. When SUSP is specified by itself, it functions like the ON operand.

RECOVERY

See “Edit Recovery” on page 47 for more information about edit recovery.

Assignment Statement Syntax

```
ISREDIT (var1, var2) = RECOVERY
ISREDIT RECOVERY = [ON [SUSP]]
                   [OFF [WARN]]
                   [OFF NOWARN]
```

var1 The name of a variable to contain the setting of recovery mode, either ON or OFF.

var2 The name of a variable that contains the warning setting, either WARN, NOWARN (when RECOVERY is OFF), or blank or SUSP (when RECOVERY is ON).

ON The system creates and updates a recovery data set for each change thereafter.

OFF The system does not create and update a recovery set.

WARN

This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN

This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP This value indicates that recovery is ON, but that it is suspended due to a previous error.

Return Codes

The following return codes can be issued:

0 Normal completion
20 Severe error.

Examples

To save the value of recovery mode in variable &RECOV:

```
ISREDIT (RECOV) = RECOVERY
```

To set recovery mode OFF:

```
ISREDIT RECOVERY = OFF
```

RENUM—Renumber Data Set Lines

The RENUM macro command immediately turns on number mode and rennumbers all lines, starting with number 100 and incrementing by 100. For any members exceeding 10 000 lines, the increment would be less than 100.

Macro Command Syntax

```
ISREDIT RENUM [ON ] [STD      ] [DISPLAY]
                  [COBOL    ]
                  [STD COBOL]
```

ON Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. It also turns number mode on and renumbers lines.

The STD, COBOL, and DISPLAY operands are interpreted only when number mode is turned on.

STD Numbers the data in the standard sequence field. This is the default for all non-COBOL data set types.

COBOL Numbers the data in the COBOL field. This is the default for all COBOL data set types.

STD COBOL Numbers the data in both fields.

If both STD and COBOL numbers are being generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

DISPLAY Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. The editor automatically scrolls left or right, if required, so that the leftmost column of the data window is the first column displayed.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Examples

To renumber all data lines with standard numbering:

```
ISREDIT RENUM
```

ON and STD are the default operands.

To renumber all data lines with standard and COBOL numbering:

```
ISREDIT RENUM STD COBOL
```

To renumber all data lines with COBOL numbering, bringing the sequence numbers within the data window:

```
ISREDIT RENUM COBOL DISPLAY
```

To turn sequence numbers off:

```
ISREDIT RENUM OFF
```

REPLACE—Replace a Data Set Member

The REPLACE macro command adds or replaces data in a member of the partitioned data set that you are editing, in a member of another partitioned data set, or in a sequential data set.

Macro Command Syntax

```
ISREDIT REPLACE member lptr-range
ISREDIT REPLACE (member) lptr-range
ISREDIT REPLACE dataset lptr-range
ISREDIT REPLACE dataset(member) lptr-range
```

member

The name of the member to be replaced in the partitioned data set currently being edited. If the member does not exist, the editor creates it. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

dataset

The name of a sequential data set that is to be replaced. The data set name can be fully or partially qualified.

dataset(member)

The name of a different partitioned data set and member name to be replaced in the partitioned data set. The data set name can be fully or partially qualified.

lptr-range

Two line pointers that are required to specify the range of lines in the current member that replace data in the other member. A line pointer can be a label or a relative line number. Specifying one line pointer is incorrect.

Return Codes

The following return codes can be issued:

0	Normal completion
8	Member in use
12	Invalid line pointer
20	Syntax error (invalid name, incomplete line pointer value), or I/O error.

Example

To replace member MEM1 with the first 10 lines of the current data:

```
ISREDIT REPLACE MEM1 1 10
```

RESET—Reset the Data Display

The RESET macro command can restore line numbers in the line command area when those line numbers have been replaced by labels, pending line commands, error flags, and change flags. However, to reset any pending line commands, you must have specified the NOPROCESS operand in the MACRO command. RESET can also delete special lines from the display, redisplay excluded lines, and temporarily disable the highlighting of FIND strings.

Macro Command Syntax

```
ISREDIT RESET [CHANGE ] [lptr-range]
              [COMMAND ]
              [ERROR ]
              [EXCLUDED]
              [FIND ]
              [LABEL ]
              [SPECIAL ]
```

You can type the operands in any order. If you do not specify any operands, RESET processes all operands except LABEL.

CHANGE

Removes ==CHG> flags from the line command area.

COMMAND

Removes any pending line commands from the line command area.

ERROR

Removes ==ERR> flags from the line command area.

EXCLUDED

Redisplays any excluded line.

FIND Turns off highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command. However, SEEK and EXCLUDE do not return the highlighting of FIND strings in this manner.

RESET with no operands has the same effect on highlighted FIND strings as RESET FIND.

LABEL

Removes labels from the line command area.

SPECIAL

Deletes any temporary line from the panel:

- Bounds line flagged as =BNDS>
- Column identification lines flagged with =COLS>
- Information lines flagged with =====
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged with =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>.

lptr-range

Specifies that two line pointers are required to specify a range of lines to be reset. A line pointer can be a label or a relative line number. Specifying one line pointer is incorrect. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

Description

RESET scans every line of data for conditions to be reset. If you want to delete a small number of special lines, you can get faster response time if you use the D (delete) line command.

Return Codes

The following return codes can be issued:

RESET

- 0 Normal completion
- 20 Severe error.

Examples

To remove all change flags from the current data:

```
ISREDIT RESET CHANGE
```

To remove all error flags from the current data:

```
ISREDIT RESET ERROR
```

To redisplay all excluded lines between the .START and .STOP labels:

```
ISREDIT RESET EXCLUDED .START .STOP
```

To remove all labels from the current data between and including the .START and .STOP labels:

```
ISREDIT RESET LABEL .START .STOP
```

To remove all special lines from the current data between lines 100 and 200:

```
ISREDIT RESET SPECIAL 100 200
```

RFIND—Repeat Find

The RFIND macro command locates the search string defined by the most recent SEEK, FIND, or CHANGE command, or excludes a line containing the search string defined by the previous EXCLUDE command.

The RFIND command can be used repeatedly to find other occurrences of the search string. After a *string* NOT FOUND message appears, the next RFIND issued starts at the first line of the current range for a forward search (FIRST or NEXT specified), or the last line of the current range for a backward search (LAST or PREV specified).

Macro Command Syntax

```
ISREDIT RFIND
```

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 String not found
- 12 Syntax error
- 20 Severe error (string not defined).

Example

To find a character string, process it, and then repeat the operation for the rest of the data:

```
ISREDIT FIND FIRST C'. the'  
SET RETCODE = &LASTCC;  
DO WHILE &RETCODE = 0
```

...

```

ISREDIT RFIND
SET RETCODE = &LASTCC;
END

```

RIGHT—Scroll Right

The RIGHT macro command scrolls data to the right of the current panel position.

Macro Command Syntax

```
ISREDIT RIGHT amt
```

amt The scroll amount, the number of columns (0 - 9999) to scroll, or one of the following operands:

MAX Displays the last panel of data to the right.

HALF Displays the next half-panel of data to the right.

PAGE Displays the next full panel of data to the right.

CURSOR

Scrolls until the column on which the cursor is located becomes the first data column on the panel.

DATA Scrolls until the last column on the current panel of data becomes the first column on the next panel of data.

Description

The editor stops scrolling when it reaches the current BOUNDS setting. For example, if the right bound is position 100, and positions 9 to 80 are displayed, issuing ISREDIT RIGHT 100 leaves positions 29 to 100 being displayed, not positions 109 to 180.

To scroll to the right using the panel position when the macro was issued, use USER_STATE assignment statements to save and then restore the panel position operands.

If you define a macro named RIGHT, it overrides RIGHT when used from another macro, but has no effect for you. RIGHT does not change the cursor position and cannot be used in an initial macro. See “BOUNDS—Set or Query the Edit Boundaries” on page 311 and “DISPLAY_COLS—Query Display Columns” on page 335 for further information.

Return Codes

The following return codes can be issued:

0	Normal completion
4	No visible lines
8	No data to display
12	Amount not specified
20	Severe error.

RIGHT

Example

To scroll the display to the right by the number of columns specified in variable &RCOL:

```
ISREDIT RIGHT &RCOL
```

RMACRO—Set or Query the Recovery Macro

The RMACRO macro command sets the name of the recovery macro.

The RMACRO assignment statement sets or retrieves the name of the recovery macro set in this edit session.

See “Recovery Macros” on page 118 for more information.

Macro Command Syntax

```
ISREDIT RMACRO {name | NONE}
```

name The name of the recovery macro to be run. The name can be preceded by an exclamation point (!) to show that it is a program macro.

NONE

The name to prevent a recovery macro from being run; conversely, a value of NONE is returned when no recovery macro has been specified.

Assignment Statement Syntax

```
ISREDIT (varname) = RMACRO  
ISREDIT RMACRO = {name | NONE}
```

varname

The name of a variable to contain the name of the recovery macro.

name Same as macro command syntax.

NONE

Same as macro command syntax.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid name specified
20	Severe error.

Example

To set the RMACRO name from the variable &RMAC:

```
ISREDIT RMACRO = &RMAC
```

SAVE—Save the Current Data

The SAVE macro command stores the current data on disk. Generally, you do not need to use SAVE if recovery mode is on. See the DATA_CHANGED, AUTOSAVE, CANCEL, and END commands for more information about saving data.

Macro Command Syntax

```
ISREDIT SAVE
```

Description

The SAVE command writes the data to the same data set from which it was retrieved unless you specified a concatenated sequence of partitioned data sets on the Edit - Entry panel. In that case, the data is saved in the first library in the concatenation sequence, regardless of which library it came from. For a sequential data set, the complete data set is rewritten. For a partitioned data set, the member is rewritten with the same member name. If stats mode is on, the library statistics for the member are automatically updated.

If both number mode and autonum mode are on, the data is automatically renumbered before it is saved.

Return Codes

The following return codes can be issued:

0	Normal completion
4	New member saved
12	Data not saved; not enough PDS space or directory space
20	Severe error.

Example

To check autosave mode and, if it is set to OFF, ensure that changes are saved:

```
ISREDIT (VAR) = AUTOSAVE
IF &VAR = OFF THEN -
  ISREDIT SAVE
```

SAVE_LENGTH—Set or Query Length for Variable Length Data

The SAVE_LENGTH macro command sets or queries the length to be used to save each record in a variable length file. It does not enable you to truncate the non-blank portion of a record, but it does enable you to extend a record. When records are written to disk, they are padded on the end with blanks as needed.

SAVE_LENGTH is a macro command only. It cannot be used as an edit primary command.

Assignment Statement Syntax

```
ISREDIT (varname) = SAVE_LENGTH .lptr
ISREDIT SAVE_LENGTH .lptr = value
```

Description

You can use the SAVE_LENGTH macro command to set or query the minimum length that is used to store an individual record in a variable length data set.

When setting a length, the length is automatically adjusted to include the non-blank portion of the line.

SAVE_LENGTH

When retrieving the length, the number returned reflects the line length that is used to save the line if the save is done immediately. The length is the maximum of either: the length of the nonblank portion of the line *and* the length set by a previous SAVE_LENGTH request, **OR** the length of the nonblank portion of the line *and* the original line length.

You can use the SAVE_LENGTH command in edit macros to define line commands to prompt the user for final record lengths or to check the record length. You might also use it to substitute a visible character for trailing blanks to make editing easier.

Use of the SAVE_LENGTH command does not prevent the editor from working on data past the specified record length. The length set and returned by the SAVE_LENGTH command is only used when the data is written and does not affect the operation of any other edit functions.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 Value supplied on set call was out of range. If the supplied length was too great, it is adjusted to equal the maximum record length. Otherwise, the length was adjusted to the length of the nonblank data portion of the record.
- 6 Record format is not variable. Any value on an assignment request is ignored.
- 16 Error setting variable.
- 20 Severe error.

Examples

To save the number of characters that are saved for the last line in the file when PRESERVE OFF is active:

```
ISREDIT (NCHARS) = SAVE_LENGTH .ZLAST
```

To set the minimum line length for the last line in the file and to set PRESERVE ON active:

```
ISREDIT SAVE_LENGTH .ZLAST = 74
```

Another edit macro sample using the SAVE_LENGTH command can be found in the ISRSETLN member of the ISPF EXEC library.

SCAN—Set Command Scan Mode

The SCAN macro command sets scan mode, which controls the automatic replacement of variables in command lines passed to the editor.

The SCAN assignment statement either sets the value of scan mode (for variable substitution), or retrieves the value of scan mode and places it in a variable.

Macro Command Syntax

```
ISREDIT SCAN [ON ]  
              [OFF]
```

- ON** Specifies that the editor automatically replaces variables in command lines.
- OFF** Specifies that the editor does not automatically replace variables.
- If mode is omitted, the default is ON. Scan mode is initialized to ON when a macro is started.

Assignment Statement Syntax

```
ISREDIT (varname) = SCAN
ISREDIT SCAN = [ON ]
                [OFF]
```

varname

The name of a variable to contain the setting of scan mode, either ON or OFF.

- ON** Same as macro command syntax.
- OFF** Same as macro command syntax.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 20 Severe error.

Example

To set a line whose number is in variable &LNUM to:

```
&SYSDATE is a CLIST built-in function
```

set scan mode off and issue the LINE command with &&SYSDATE as the CLIST function name. The CLIST processor strips off the first &, but, because scan mode is off, the editor does not remove the second &;

```
ISREDIT SCAN OFF
ISREDIT LINE &LNUM = "&&SYSDATE is a CLIST built-in function"
ISREDIT SCAN ON
```

Because the ISPEXEC call interface for REXX EXECs allows you to specify parameters as symbolic variables, a single scan always takes place before the syntax check of a statement. Therefore, the rule of using two ampersands (&) before variable names to avoid substitution of variable names also applies to REXX EXECs.

SEEK—Seek a Data String, Positioning the Cursor

The SEEK macro command finds one or more occurrences of a search string without changing the exclude status of the line.

Macro Command Syntax

```
ISREDIT SEEK string [label-range] [NEXT ] [CHARS ] [X ] [col-1 [col-2]]
                                [ALL ] [PREFIX] [NX]
                                [FIRST] [SUFFIX]
                                [LAST ] [WORD ]
                                [PREV ]
```

SEEK

string The search string you want to find. The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

label-range

Two labels that identify the range of lines SEEK is to search. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

When using a macro that uses NEXT or PREV with a label-range, be careful concerning cursor placement. If the cursor is currently placed below the label-range, and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

If the cursor is currently placed above the label-range, and the PREV occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label-range.

NEXT Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string. NEXT is the default.

ALL Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS

Locates string anywhere the characters match. CHARS is the default.

PREFIX

Locates string at the beginning of a word.

SUFFIX

Locates string at the end of a word.

WORD

Locates string when it is delimited on both sides by blanks or other non-alphanumeric characters.

X Scans only lines that are excluded from the display.

NX Scans only lines that are not excluded from the display.

col-1 and col-2

Numbers that identify the columns SEEK is to search.

Description

Use the FIND macro command instead of SEEK if you want to locate a string and change the exclude status of the line that contains that string at the same time.

You can use SEEK to find a search string, change it with CHANGE, and then exclude it from the display with EXCLUDE.

To find the next occurrence of the letters ELSE without specifying any other qualifications, include the following line in an edit macro:

```
ISREDIT SEEK ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In either an excluded or a nonexcluded line
- Anywhere within the current boundaries.

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

```
ISREDIT SEEK C'ELSE'
```

This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see “Finding, Seeking, Changing, and Excluding Data” on page 53.

Return Codes

The following return codes can be issued:

0	Normal completion
4	String not found
12	Syntax error
20	Severe error.

Examples

The following example finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
ISREDIT SEEK ELSE .E .S LAST SUFFIX X
```

The following example finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines that are labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be stand-alone characters (not part of any other word); they must be found in a nonexcluded line; and they must exist within columns 1 and 5:

```
ISREDIT SEEK ELSE .E .S PREV WORD NX 1 5
```

SEEK_COUNTS—Query Seek Counts

The SEEK_COUNTS assignment statement retrieves the values set by the most recently entered SEEK command and places them in variables.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = SEEK_COUNTS
```

SEEK_COUNTS

- var1** The name of a variable to contain the number of strings found. It must be an 8-character value that is left-padded with zeros.
- var2** The name of a variable to contain the number of lines on which strings were found. It must be an 8-character value that is left-padded with zeros.

Return Codes

The following return codes can be issued:

- 0** Normal completion
- 20** Severe error.

Example

To seek all lines with a blank in column 1 and store the number of such lines in variable &BLNKS:

```
ISREDIT SEEK ALL " " 1
ISREDIT (BLNKS) = SEEK_COUNTS
```

SESSION

The SESSION assignment statement identifies the type of session in which the macro is running, Edit, View, or EDIF. It also identifies if SCLM is active or not.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = SESSION
```

- var1** This variable contains either EDIF, EDIT, or VIEW to identify the type of session.
- var2** This variable contains SCLM if SCLM is active, or four asterisks (****) if SCLM is not active.

Return Codes

The following return codes can be issued:

- 0** Normal completion
- 20** Severe error.

SETUNDO—Set UNDO Mode

The SETUNDO macro command allows the UNDO function to be turned on or off and retrieves the current UNDO status.

Macro Command Syntax

```
ISREDIT SETUNDO [STORAGE]
                  [RECOVER]
                  [ON]
                  [OFF]
```

STORAGE

Enables edit changes to be saved in storage.

RECOVER

Enables edit changes to be saved through the recovery file only. If edit recovery is off, SETUNDO RECOVER turns recovery on.

- ON** Enables edit changes to be saved in storage.
- OFF** Disables the saving of edit changes in storage. If edit recovery is available, the undo command uses the edit recovery file.

Assignment Statement Syntax

```
ISREDIT (varname) = SETUNDO
ISREDIT SETUNDO = [STORAGE]
                  [RECOVER]
                  [ON]
                  [OFF]
```

varname

The name of a variable containing the setting of the UNDO mode, either OFF or RECOVER or STORAGE.

STORAGE

Enables edit changes to be saved in storage.

RECOVER

Enables edit changes to be saved through the recovery file only. If edit recovery is off, SETUNDO RECOVER turns recovery on.

- ON** Enables edit changes to be saved in storage.
- OFF** Disables the saving of edit changes in storage. If edit recovery is available, the undo command uses the edit recovery file.

Description

The SETUNDO macro command enables undo processing. It does not perform the undo function itself. Valid operands are STORAGE, RECOVER, ON, or OFF. If an operand is not supplied, STORAGE is the default.

If SETUNDO is set on by a macro and was not on already, the UNDO function is enabled for all interactions started from the point SETUNDO was turned on.

Note: Changes are saved on the undo chain after:

- SETUNDO STORAGE is specified in a macro, and it was previously OFF or REC, or
- SETUNDO REC is specified in a macro, and it was previously OFF.

It is possible to undo back to a particular point in a macro. This is helpful in debugging edit macros.

Notes:

1. If SETUNDO is disabled through the configuration table, the SETUNDO macro command is accepted and returns a zero return code. It does not turn recovery on.
2. The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *ISPF Planning and Customizing*

Return Codes

The following return codes can be issued:

- 0** Successful completion. SETUNDO was turned on or off, or status remains unchanged because UNDO was already on or off.

SETUNDO

- 20 Severe error. Probably a parameter error (something other than STG, REC, or OFF was specified).

Examples

To disable the saving of edit changes in storage:

```
ISREDIT SETUNDO OFF
```

To enable the saving of edit changes in storage:

```
ISREDIT SETUNDO = STORAGE
```

To store the value of SETUNDO in the variable &SET:

```
ISREDIT (SET) = SETUNDO
```

SHIFT (—Shift Columns Left

The SHIFT (macro command moves characters on a line to the left without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting Data” on page 51 for more information.

Macro Command Syntax

```
ISREDIT SHIFT ( lptr [n]  
                [2]
```

- lptr** Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.
- n** Specifies the number of columns to shift. If this operand is omitted, the default is 2 columns.

Description

The SHIFT (command is limited to shifting columns of data on a single line. If you want to shift columns of data on several lines, each line of data columns must be moved individually.

Return Codes

The following return codes can be issued:

- 0 Normal completion
12 Invalid line number
20 Severe error.

Examples

To shift columns of data 10 columns to the left on the line that contains the cursor:

```
ISREDIT SHIFT ( .ZCSR 10
```

To shift columns of data 2 columns to the left on the line with the label .LAB:

```
ISREDIT SHIFT ( .LAB
```

SHIFT)—Shift Columns Right

The SHIFT) macro command moves characters on a line to the right without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting Data” on page 51 for more information.

Macro Command Syntax

```
ISREDIT SHIFT ) lptr [n]
                    [2]
```

- lptr** Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.
- n** Specifies the number of columns to shift. If this operand is omitted, the default is 2 columns.

Description

The SHIFT) command is limited to shifting columns of data on a single line. If you want to shift columns of data on several lines, each line of data columns must be moved individually.

Return Codes

The following return codes can be issued:

- 0** Normal completion
- 12** Invalid line number
- 20** Severe error.

Examples

To shift columns of data 4 columns to the right on the line that contains the cursor:

```
ISREDIT SHIFT ) .ZCSR 4
```

To shift columns of data 2 columns to the right on the line with the label .LAB:

```
ISREDIT SHIFT ) .LAB
```

SHIFT <—Shift Data Left

The SHIFT < macro command moves the body of a program statement to the left without shifting the label or comments. This command prevents loss of non-blank characters by stopping before shifting non-blank characters past the bound. See “Shifting Data” on page 51 for more information.

Macro Command Syntax

```
ISREDIT SHIFT < lptr [n]
                    [2]
```

- lptr** Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.
- n** Specifies the number of columns to shift. If this operand is omitted, the default is 2 columns.

SHIFT <

Description

The SHIFT < command is limited to shifting data on a single line. To shift data on several lines, you must shift data on each line individually.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line number
20	Severe error.

Examples

To shift data 4 columns to the left on the line that contains the cursor:

```
ISREDIT SHIFT < .ZCSR 4
```

To shift data 2 columns to the left on the line with the label .LAB:

```
ISREDIT SHIFT < .LAB
```

SHIFT >—Shift Data Right

The SHIFT > macro command moves the body of a program statement to the right without shifting the label or comments. This command prevents loss of non-blank characters by stopping before shifting non-blank characters past the bound. See “Shifting Data” on page 51 for more information.

Macro Command Syntax

```
ISREDIT SHIFT > lptr [n]  
[2]
```

lptr Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.

n Specifies the number of columns to shift. If this operand is omitted, the default is 2 columns.

Description

The SHIFT > command is limited to shifting data on a single line. To shift data on several lines, you must shift data on each line individually.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line number
20	Severe error.

Examples

To shift data 4 columns to the right on the line that contains the cursor:

```
ISREDIT SHIFT > .ZCSR 4
```

To shift data 2 columns to the right on the line with the label .LAB:
 ISREDIT SHIFT > .LAB

SORT—Sort Data

The SORT macro command puts data in a specified order.

Macro Command Syntax

```
ISREDIT SORT [label-range] [X ] [sort-field1 ... sort-field5]
                [NX]
```

label-range

Specifies that two labels are required to specify a range of lines for the sort operation; specifying one label is incorrect. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

X Specifies that only excluded lines are to be sorted.

NX Specifies that only nonexcluded lines are to be sorted.

sort-field1 ... sort-field5

Specifies the fields to be used in sorting data. You can specify up to five sort fields as follows:

```
[A] [start-col [end-col]]
[D]
```

where:

A Specifies ascending order. It can either precede or follow the column specification. A is the default.

D Specifies descending order. It can either precede or follow the column specification.

start-col

Defines the starting column of the field that is to be compared. It must be within the current boundaries.

end-col

Defines the ending column of the field that is to be compared. It must be within the current boundaries.

If you specify several fields, you must specify both the starting and ending columns of each field. The fields cannot overlap. If you specify A or D for one field, you must specify it for all fields.

Description

The SORT command operates in two different modes, based on the hexadecimal mode status. If hexadecimal mode is on, the data is ordered according to its hexadecimal representation. If hexadecimal mode is off, data is sorted in the collating sequence defined for the national language being used.

Sorting Data Without Operands

For a SORT command with no operands, the editor compares the data within the current boundaries character by character, and then orders it line by line in the proper collating sequence. It ignores data outside the current boundaries during both operations. This means that only the data inside the current boundaries is

SORT

changed. Labels, excluded lines, line numbers, and change, error, and special line flags are considered associated with the data, and therefore points to the same data fields after the sort as they did before the sort.

For example, if you issue a CHANGE ALL command that changes the first, third, and sixth lines in a data set, these lines are flagged with the change flag, ==CHG>. If you then issue a SORT command that results in the former lines 1, 3 and 6 becoming the first, second and third lines of the sorted file, the changed line flags would now exist on the first, second and third lines of the sorted data set.

It is important to properly set the boundaries before issuing the SORT command. SORT is a powerful tool for editing data that may be formatted in multiple columns. You can set the boundaries, for example, to the first half of a record and sort one column of data. Then you can set the boundaries to the last half of the record and sort a second column of data.

Limiting the SORT Command

You can specify up to five sort fields by labelling starting and ending columns. You can identify each field as having data sorted in ascending or descending order.

Optionally, you can limit sorting to a range of lines by specifying the labels of the first and last lines of the range. You can also limit sorting to either excluded or nonexcluded lines.

If you have labels or line ranges that are between the labels or line ranges specified with the SORT command, you can keep SORT from rearranging them by:

- Excluding them before you enter the SORT command
- Using the NX operand to sort only lines that are not excluded.

See the definition of the NX operand and “EXCLUDE—Exclude Lines from the Display” on page 242 for more information.

Sorting DBCS Data

When sorting data that contains DBCS character strings, you must ensure that no DBCS string crosses the boundaries. Also, all records must have the same format at the boundaries, although the format of the left and right boundaries can differ.

If a boundary divides a DBCS character, or if all records do not have the same format at the boundaries, the result is unpredictable.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Lines were already in sort order
8	No records to sort
16	Not enough storage to perform sort
20	Severe error.

Examples

To sort the data in descending order, using the sort key in columns 15 through 20:

```
ISREDIT SORT D 15 20
```

To sort all excluded lines in ascending order:

```
ISREDIT SORT X A
```

STATS—Set or Query Stats Mode

The STATS macro command sets stats mode, which creates and maintains statistics for a member of a partitioned data set.

The STATS assignment statement either sets stats mode, or retrieves the setting of stats mode and places it in a variable.

Macro Command Syntax

```
ISREDIT STATS [ON ]
              [OFF]
```

ON Creates or updates library statistics when the data is saved.

OFF Does not create or update library statistics.

Assignment Statement Syntax

```
ISREDIT (varname) = STATS
ISREDIT STATS = [ON ]
                [OFF]
```

varname

The name of a variable to contain the setting of stats mode, either ON or OFF.

ON Same as macro command syntax.

OFF Same as macro command syntax.

See “Statistics for PDS Members” on page 30 for more information.

Return Codes

The following return codes can be issued:

```
0      Normal completion
20     Severe error.
```

Examples

To put the value of stats mode in variable &LIBSTAT:

```
ISREDIT (LIBSTAT) = STATS
```

To set stats mode on:

```
ISREDIT STATS = ON
```

To set stats mode off:

```
ISREDIT STATS OFF
```

To reset stats mode from the mode saved in variable &LIBSTAT:

```
ISREDIT STATS = &LIBSTAT
```

SUBMIT—Submit Data for Batch Processing

The SUBMIT macro command submits the member or data set you are editing (or the part of the member or data set defined by the range of line pointers) to be processed as a batch job.

Macro Command Syntax

```
ISREDIT SUBMIT [lptr-range]
```

lptr-range

Specifies that two line pointers are required to specify a range of lines. A line pointer can be a label or a relative line number. Specifying one line pointer is incorrect. The defaults are the editor-defined .ZFIRST and .ZLAST labels.

Description

The editor does not supply a job statement when you enter the SUBMIT command. You can supply job statements as part of the data being submitted. When you supply a job statement, only the job name is logged to the ISPF log data set to ensure the protection of sensitive data.

PDF uses TSO SUBMIT to submit the job.

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error (submit failed).

Examples

To submit the first 20 lines of the data as a batch job:

```
ISREDIT SUBMIT 1 20
```

To submit all of the data as a batch job:

```
ISREDIT SUBMIT
```

TABS—Set or Query Tabs Mode

The TABS macro command:

- Turns tabs mode on and off
- Defines the logical tab character
- Controls the insertion of attribute bytes at hardware tab positions defined with the TABS line command.

The TABS assignment statement does everything the macro command can do. It can also retrieve the setting of tabs mode and place it in a variable.

Use PROFILE to check the setting of tabs mode and the logical tab character. See “Using Tabs” on page 71 if you need more information about using tabs.

Macro Command Syntax

```
ISREDIT TABS [ON ] [STD]
                [ALL]
                [tab-character]
                [OFF]
```

- ON** Turns tabs mode on, which means that logical tabs can be used to break up strings of data. This is the default operand. If no other operands are included, all hardware tab positions (asterisks) that contain a blank or null character are activated because STD is also a default operand. The TABS ON STD message appears in the profile display.
- OFF** Turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab positions, causing the Tab Forward and Tab Backward keys to ignore hardware tabs defined on the =TABS> line. Blanked-out characters occupying these positions reappear. The TABS OFF message appears in the profile display.
- STD** Activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. STD is the default operand. You can use the Tab Forward and Tab Backward keys to move the cursor one space to the right of the attribute bytes. The TABS ON STD message appears in the profile display.
- ALL** Causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over. The Tab Forward and Tab Backward keys can be used to move the cursor one space to the right of these attribute bytes. The TABS ON ALL message appears in the profile display.

tab-character

Defines a single character that is not a number, letter, or command delimiter as the logical tab character. This character is used with hardware tab definitions. The TABS ON tab-character message appears in the profile display.

You can enclose the character in quotes (' or "), although this is not necessary unless you want to use one of the following as the tab character:

```
= ' " < , ( +
```

The ampersand (&), left bracket ([), and right bracket (]) should not be used as tab characters at all.

The tab-character operand causes the data string that follows the logical tab character to align itself one space to the right of the first available hardware tab position when you press Enter. No attribute bytes are inserted.

If no hardware tabs are defined, the editor aligns the data vertically. If software tabs are defined, the first data string is aligned under the first software tab position and the remaining data strings are aligned at the left boundary. If neither software nor hardware tabs are defined, the editor aligns all the data strings at the left boundary.

With the tab-character operand, the Tab Forward and Tab Backward keys ignore hardware tab positions when the tab-character operand is used because no attribute bytes are inserted.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = TABS
ISREDIT TABS = [ON ] [STD]
                    [ALL]
                    [tab-character]
                    [OFF]
```

- var1** The name of a variable to contain the setting of tabs mode, either ON or OFF.
- var2** The name of a variable to contain the tab character and either ALL or STD. This variable may be blank.
- ON** Same as macro command syntax.
- OFF** Same as macro command syntax.
- STD** Same as macro command syntax.
- ALL** Same as macro command syntax.
- tab-character**
Same as macro command syntax.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 20 Severe error.

Examples

To set the tab character to \ and set the tabs mode ON:

```
ISREDIT TABS ON \
```

To set the value of tabs mode from variable &TABVAL:

```
ISREDIT TABS = (TABVAL)
```

TABSLINE—Set or Query Tabs Line

The TABSLINE assignment statement either sets the tabs line, or retrieves the tabs line and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = TABSLINE
ISREDIT TABSLINE = data
```

varname

Specifies the name of a variable to hold the contents of the current tabs line.

data Specifies the data used to set the tabs line. The only valid tab characters for this data are blanks, asterisks (*), hyphens (-), and underscores (_). The following forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< col,string >)

- Merge format (*string-1 + string-2, operand + string-2, string-1 + operand*)
- Operand:
 - LINE lptr**
Data from the line with the given line pointer (lptr).
 - MASKLINE**
Data from the mask line.
 - TABSLINE**
Data from the tabs line.

Return Codes

The following return codes can be issued:

- 0 Normal completion
- 4 Data truncated
- 8 Invalid data detected and ignored
- 20 Severe error (invalid input).

Examples

To store the value of the tabs line in variable &OLDTABS:

```
ISREDIT (OLDTABS) = TABSLINE
```

To set the tabs line to "* ___ * *":

```
ISREDIT TABSLINE = "* ___ * *"
```

To clear the tabs line:

```
ISREDIT TABSLINE = " "
```

To set tabs in columns 1 and 35:

```
ISREDIT TABSLINE = <1,*,35,*>
```

To add a tab in column 36:

```
ISREDIT TABSLINE = TABSLINE + <36,*>
```

TENTER—Set Up Panel for Text Entry

The TENTER macro command provides one very long line wrapped around onto many rows of the panel to allow power typing for text entry. The editor does the formatting for you.

The TENTER command is different from the INSERT command in that the INSERT command inserts a specified number of separate, blank lines and the mask, if any, just as you typed it. With the TENTER command, however, mask line characters are applied only to the new lines created when the text is flowed outside the boundaries. Any mask line characters within the bounds are ignored.

Macro Command Syntax

```
ISREDIT TENTER lptr [numlines]
```

lptr Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.

numlines

Specifies the number of lines displayed for text entry; these lines are not

TENTER

saved unless they contain data. If you do not type a number, the remainder of the panel appears for text entry.

Description

It is important to make sure that the line referenced by the line pointer on TENTER appears; otherwise, the text area will not be visible to you. Use LOCATE to find and display the line for you.

Before you enter text entry mode, consider the following:

- If you are going to be typing text in paragraph form, such as for a memo or letter, make sure caps mode is off. Otherwise, when you press Enter, your text will change to all caps.
- You may want to turn off number mode to prevent sequence numbers from writing over any of your text.
- Make sure the bounds setting is where you want it so that the text flows correctly when you end text entry mode.
- Once you enter text entry mode, no macros can be run.

To enter text entry mode:

1. Include the following command in an edit macro:

```
ISREDIT TENTER lptr numlines
```

where *lptr* is a label or relative line number and *numlines* is the number of blank lines that you want to insert. If the number that you type is greater than the number of rows remaining on the panel, the vertical bar that indicates where you will run out of room does not appear and the keyboard does not lock at the last character position on the panel. When you run the edit macro (see step 2), you can scroll down to bring the additional blank text entry space into view.

2. Run the edit macro. The editor inserts a single continuous blank area for the specified number of rows or to the bottom of the panel.

To begin a new paragraph:

1. Use the return (Enter), cursor movement, or Tab keys to advance the cursor enough spaces to leave one blank row on the panel.

If there are insufficient blank spaces on the panel, the keyboard locks when you try to type beyond the last character position. A vertical bar (|) appears above the cursor at the locked position.

To generate more blank spaces:

1. Press the Reset key to unlock the keyboard.
2. Press Enter.

To end text entry mode:

1. Press Enter. The data is flowed together into a paragraph and any embedded blanks are preserved. The left and right sides of the paragraph are determined by the current bounds.

See “Word Processing” on page 68 and “Entering Text (Power Typing)” on page 71 for more information.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line number
20	Severe error.

Example

To find the last line in the data and set up the display for text entry following the last line:

```
ISREDIT LOCATE .ZL
ISREDIT TENTER .ZL
```

TFLOW—Text Flow a Paragraph

The TFLOW macro command restructures paragraphs. This is sometimes necessary after deletions, insertions, splitting, and so forth. See “Word Processing” on page 68 and “Formatting Paragraphs” on page 68 for more information.

Macro Command Syntax

```
ISREDIT TFLOW lptr [col]
```

lptr Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.

col Specifies the column to which the text should be flowed. If the column number is omitted, it defaults to the right boundary. This is different from the TF (text flow) line command, which defaults to the panel width when default boundaries are in effect.

If a number greater than the right boundary is specified, the right boundary is used.

Return Codes

The following return codes can be issued:

0	Normal completion
12	Invalid line number
20	Severe error.

Example

To limit the flow of text, starting at label .PP, to the displayed columns:

```
ISREDIT (,RCOL) = DISPLAY_COLS
ISREDIT TFLOW .PP &RCOL
```

TSPLIT—Text Split a Line

The TSPLIT macro command moves part or all of a line of text to the following line. This makes it easier for you to add new material to existing text.

Macro Command Syntax

```
ISREDIT TSPLIT [lptr col]
```

TSPLIT

- lptr** Specifies that a line pointer is used to identify the line where the split is to occur. A line pointer can be a label or a relative line number.
- col** Specifies the column at which the text is to be split.

If you omit both operands, the split point is assumed to be the current cursor position.

Description

The TSPLIT macro command is affected by the current setting of the boundaries. For instance, data beyond the right boundary is not moved to the line added by TSPLIT. Data between the split column and the right boundary is moved to a new line. The cursor position is set to the split point.

To rejoin lines, use the TFLOW macro command. See “TFLOW—Text Flow a Paragraph” on page 409 for more information.

For more information about splitting lines and other word processing commands, see “Word Processing” on page 68 and “Splitting Lines” on page 70.

Return Codes

The following return codes can be issued:

- | | |
|----|---------------------|
| 0 | Normal completion |
| 12 | Invalid line number |
| 20 | Severe error. |

Example

To split the line labeled .TOP at column 15:

```
ISREDIT (LINENBR) = LINENUM .TOP  
ISREDIT TSPLIT &LINENBR 15
```

UNNUMBER—Remove Sequence Numbers

The UNNUMBER macro command sets all sequence fields to blanks, turns off number mode, and positions the data so that column 1 is the first column displayed.

Macro Command Syntax

```
ISREDIT UNNUMBER
```

Description

The UNNUMBER command is valid only when number mode is also on. The standard sequence field, the COBOL sequence field, or both, are blanked out.

Return Codes

The following return codes can be issued:

- | | |
|----|--------------------|
| 0 | Normal completion |
| 12 | Number mode not on |

20 Severe error.

Example

To set all sequence fields to blanks, turn number mode off, and position the panel so that column 1 is the first column displayed:

```
ISREDIT UNNUMBER
```

UP—Scroll Up

The UP macro command scrolls data up from the current panel position.

Macro Command Syntax

```
ISREDIT UP amt
```

amt The scroll amount, the number of lines (0 - 9999) to scroll, or one of the following operands:

MAX Displays the first panel of data.

HALF Displays the previous half-panel of data.

PAGE Displays the previous full panel of data.

CURSOR

Scrolls until the line on which the cursor is located becomes the last data line on the panel.

DATA Scrolls until the first data line on the current panel becomes the last data line on the next panel.

Description

To scroll up using the panel position when the macro was issued, use `USER_STATE` assignment statements to save and then restore the panel position operands.

When you issue the UP command, the non-data lines on the panel affect the number of lines scrolled. However, if you define a macro named UP, it only overrides UP when used from another macro. UP does not change the cursor position and cannot be used in an initial macro.

The actual number of lines to appear on the panel is determined by:

- The number of lines excluded from the panel
- The terminal display size and split panel line
- The number of special temporary lines displayed, such as the `==ERR>`, `==CHG>`, `=PROF>`, `=MASK>`, `=BNDS>`, `=TABS>`, `==MSG>`, `=NOTE=`, `=COLS>`, and `=====` lines.

The first line displayed is determined in one of two ways: (1) a LOCATE command can actually set the line to be first on the panel, or (2) the first line to be displayed depends on whether the cursor was explicitly set by a CURSOR assignment statement or implicitly set by a SEEK, FIND, CHANGE, or TSPLIT command. Since the cursor must be on the panel, the line that is first on the panel may be different from the line that was first when you started the macro.

Return Codes

The following return codes can be issued:

0	Normal completion
2	No more data UP
4	No visible lines
8	No data to display
12	Amount not specified
20	Severe error.

Examples

To scroll up to the top of the data set:

```
ISREDIT UP MAX
```

To display the previous half panel of data:

```
ISREDIT UP HALF
```

To display the previous full panel of data:

```
ISREDIT UP PAGE
```

To make the line where the cursor is placed the last one on the display:

```
ISREDIT UP CURSOR
```

To display the previous page less one line:

```
ISREDIT UP DATA
```

USER_STATE—Save or Restore User State

The USER_STATE assignment statement saves or restores the state of edit profile values, FIND, CHANGE, SEEK, and EXCLUDE values, and panel and cursor values.

Assignment Statement Syntax

```
ISREDIT (varname) = USER_STATE
ISREDIT USER_STATE = (varname)
```

varname

The name of a variable to contain your status information.

Note: The information in the variable is saved in an internal format that is subject to change. Dependence on the format can lead to macro errors.

Description

USER_STATE can be used at the beginning of a macro to save conditions, and at the end of a macro to restore the conditions that may have changed during processing. Many of the values saved by USER_STATE can be saved and restored individually. The USER_STATE assignment statement is a simple way of saving many values with a single statement.

The following edit modes and values are saved and restored by USER_STATE:

AUTOLIST	CURSOR	NOTES	RECOVERY
AUTONUM	HEX	NULLS	STATS
AUTOSAVE	IMACRO	NUMBER	TABS
BOUNDS	MASKLINE	PACK	TABSLINE
CAPS	MODEL CLASS	PROFILE	

Return Codes

The following return codes can be issued:

0	Normal completion
20	Severe error.

Examples

To save the user state in variable &STATUS:

```
ISREDIT (STATUS) = USER_STATE
```

To restore the user state from variable &STATUS:

```
ISREDIT USER_STATE = (STATUS)
```

VERSION—Set or Query Version Number

The VERSION macro command allows you to change the version number assigned to a member of an ISPF library.

The VERSION assignment statement either sets the version number, or retrieves the version number and places it in a variable.

For more information about version numbers, see “Version and Modification Level Numbers” on page 31.

Macro Command Syntax

```
ISREDIT VERSION num
```

num The version number. It can be any number from 1 to 99.

Assignment Statement Syntax

```
ISREDIT (varname) = VERSION
ISREDIT VERSION = num
```

varname

The name of a variable to contain the version number. The version number is a 2-digit value that is left-padded with zeros.

num Same as macro command syntax.

Return Codes

The following return codes can be issued:

0	Normal completion
4	Stats mode is off, the command is ignored
12	Invalid value specified (the version must be 1 to 99)
20	Severe error.

VERSION

Examples

To save the version number in variable &VERS:

```
ISREDIT (VERS) = VERSION
```

To set the version number to 1:

```
ISREDIT VERSION 1
```

To set the version number from variable &VERS:

```
ISREDIT VERSION = &VERS
```

VIEW—View from within an Edit Session

The VIEW macro command allows you to view a member of the same partitioned data set during your current edit session.

Macro Command Syntax

```
ISREDIT VIEW member
```

member

A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Your initial edit session is suspended until the view session is complete. Editing sessions can be nested until you run out of storage.

To exit from the view session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

The VIEW service call, ISPEXEC VIEW, is an alternate method of starting view. It offers the option of viewing another data set and specifying an initial macro.

For more information on using the VIEW service, refer to *ISPF Services Guide*

Return Codes

The following return codes can be issued:

0	Normal completion
12	Your error (invalid member name, recovery pending)
20	Severe error.

Examples

To view the member OLDMEM in your current ISPF library:

```
ISREDIT VIEW OLDMEM
```

VOLUME—Query Volume Information

The VOLUME assignment statement retrieves the volume serial number (or serial numbers) and the number of volumes on which the data set resides.

Assignment Statement Syntax

```
ISREDIT (var1,var2) = VOLUME
```

var1 The name of a variable to contain the serial number of the volume on which the data set resides. For a multivolume data set, this will be the serial number of the first volume. The volume serial number is a six character value.

var2 The name of a variable to contain the number of volumes the data set occupies. The number of volumes is a two character value.

Return Codes

The following return codes can be issued:

0 Normal completion

4 The data set is a multivolume data set and the shared pool variable ZEDMVOL is set to contain all the volume serial numbers of the data set. ZEDMVOL has the length of the number of volumes times six.

20 Severe error.

Examples

To retrieve just the volume serial number of the data set:

```
ISREDIT (VOL) = VOLUME
```

To retrieve just the number of volumes the data set occupies:

```
ISREDIT (,NUMVOL) = VOLUME
```

To retrieve both the volume serial number and the number of volumes the data set occupies:

```
ISREDIT (VOL,NUMVOL) = VOLUME
```

XSTATUS—Set or Query Exclude Status of a Line

The XSTATUS assignment statement either sets the exclude status of the specified data line, or retrieves the exclude status of the specified data line and places it in a variable.

Assignment Statement Syntax

```
ISREDIT (varname) = XSTATUS lptr
ISREDIT XSTATUS lptr = X | NX
```

varname

The name of a variable to contain the exclude status, either X or NX.

lptr Specifies that a line pointer must be used. A line pointer can be a label or a relative line number.

X Specifies that the specified line is to be excluded.

NX Specifies that the specified line is to be shown (nonexcluded).

Description

Exclude status determines whether the line is excluded.

XSTATUS

If you want to exclude several lines at one time, the EXCLUDE command should be used. Similarly, to show several lines at one time, use the FIND command.

Return Codes

The following return codes can be set:

- 0 Normal completion
- 8 An attempt to set a line status to NX could not be performed. The line has a pending line command on it. For example, if an excluded line contains an M line command in the line command area, then the MOVE/COPY IS PENDING message is displayed and the lines cannot be shown. The reset command can be used to remove your line commands from the line command area.
- 12 Line number is not an existing line.
- 20 Severe error.

Examples

Use XSTATUS together with SEEK and CHANGE to preserve the exclude status of a line. For example, to store the exclude status of the line whose number is in variable &N in variable &LINEX:

```
ISREDIT (LINEX) = XSTATUS &N
```

To exclude line 1:

```
ISREDIT XSTATUS 1 = X
```

To locate a string and change it, saving and then restoring the exclude status:

```
ISREDIT SEEK &DATA  
IF &LASTCC = 0 THEN -  
  DO  
    ISREDIT (XLINE) = XSTATUS .ZCSR  
    ISREDIT CHANGE &DATA &NEWDATA .ZCSR .ZCSR  
    ISREDIT XSTATUS .ZCSR = (XLINE)  
  END
```

Part 4. Appendixes

Appendix A. Abbreviations for Commands and Other Values

The following list includes the command names and keywords that can be abbreviated, followed by the allowable abbreviations. To improve readability, do not use abbreviations in edit macros. ISPF scans the NUMBER macro as a command. If you want to define NUMBER as a program macro and use the abbreviated form, define the abbreviations as program macros also.

Edit Line Commands

BOUNDS	BOUND	BNDS	BND	BOU
COLS	COL			
LCC	LCLC			
MDD	MDMD			
TABS	TAB			
UCC	UCUC			

Edit Primary Commands

BOUNDS	BOUND	BNDS	BND	BOU		
CANCEL	CAN					
CHANGE	CHA	CHG	C			
CREATE	CRE					
DEFINE	DEF					
DELETE	DEL					
EXCLUDED	EXCLUDE	EXC	EX	X		
FIND	F					
HILITE	HILIGHT	HI				
LEVEL	LEV					
LOCATE	LOC	L				
MODEL	MOD					
NONULLS	NONULL	NONUL				
NONUMBER	NONUMBR	NONUMB	NONUM			
NOTABS	NOTAB					
NOTES	NOTE					
NULLS	NULL	NUL				
NUMBER	NUMB	NUM				
PROFILE	PROF	PRO	PR			
RECOVERY	RECOVER	RECOVRY	RECVRY	RECOV	RECVR	
	REC					
RENUM	REN					
REPLACE	REPL	REP				
RESET	RES					
SETUNDO	SETU					
SUBMIT	SUB					
TABS	TAB					
UNNUMBER	UNNUMB	UNNUM	UNN			
VERSION	VERS	VER				

Parameters

Parameters

AFTER	AFT
BEFORE	BEF

Keywords/Operands

Keywords/Operands

CHANGE	CHG			
CHARS	CHAR			
COMMAND	COM			
CURSOR	CUR			
DISABLED	DISABLE	DISAB		
DISPLAY	DIS	DISP	DISPL	
DOLOGIC	DO			
ERROR	ERR			
IFLOGIC	IF			
LABEL	LABELS	LAB		
PREFIX	PRE			
RECOVER	RECOVERY	REC		
SPECIAL	SPE			
STANDARD	STD			
STORAGE	STG	STORE	STOR	STO
SUFFIX	SUF			
VERTICAL	VERT			

Scroll Amounts

CUR	CSR	C
DATA	D	
HALF	H	
MAX	M	
PAGE	P	

Appendix B. Edit-Related Sample Macros

The following edit macros are shipped with ISPF in the IBM-supplied ISPF samples library.

Sample Macros

These macros can be used in problem resolution.

ISRCUT

An ISPF Edit macro written in REXX that writes lines from a file to the user's PROFILE pool for later inclusion by the ISRPASTE macro.

ISRONLY

An ISPF Edit macro written in REXX that combines the ISPF Edit commands EXCLUDE and FIND such that *only* the lines containing the search string are displayed.

ISRPASTE

An ISPF Edit macro written in REXX that writes lines from the user's PROFILE pool into the current file. This macro is used in conjunction with the ISRCUT macro.

Edit-Related Sample Macros

Index

Special Characters

((column shift left), line command 156
) (column shift right), line command 158
! (exclamation point), for implicit edit macro 116
& prefix for edit commands 17
> (data shift right), line command 162
< (data shift left), line command 159
&LASTCC variable 119
{ } (one operand required) 154, 209, 299, 300
| (OR symbol) 154, 209, 300
.ZCSR 66, 112
.ZDEST 112, 117
.ZFIRST 66, 112
.ZFRANGE 112, 117
.ZLAST 66, 112
.ZLRANGE 112, 117

Numerics

3850 virtual volumes, accessing 8

A

A (after), line command 163, 164
A operand, REXX TRACE statement 123
abbreviations for commands and other values 419
ACCOUNT command 9
add a data set member 386
add data 277
adding
 a line 177, 356
 edit macro command 96
 models 81
alias, assigning 236, 332
alias name, defining with edit macro 115
application-wide macros 30
assignment statement
 AUTOLIST 307
 AUTONUM 308
 AUTOSAVE 309
 BLKSIZE 311
 BOUNDS 311
 CAPS 315
 CHANGE COUNT 319
 CTL_LIBRARY 324
 CURSOR 326
 DATA_CHANGED 329
 DATA_WIDTH 330
 DATAID 331
 DATASET 331
 description 104
 DISPLAY_COLS 335
 DISPLAY_LINES 336
 EXCLUDE_COUNTS 341
 FIND_COUNTS 344
 FLIP 345
 FLOW_COUNTS 345

assignment statement (*continued*)

 HEX 346
 how to use 106
 IMACRO 351
 LABEL 112, 352
 LEVEL 354
 LINE 355
 LINE_AFTER 356
 LINE_BEFORE 358
 LINENUM 359
 LRECL 362
 MACRO_LEVEL 112, 364
 MASKLINE 364
 MEMBER 365
 notation conventions 299
 NOTES 370
 NULLS 370
 NUMBER 372
 PACK 374
 parentheses guidelines 106
 PROFILE 379
 RANGE_CMD 117, 381
 RECFM 382
 RECOVERY 383
 reference section 299
 RMACRO 118, 390
 SCAN 104, 392
 SEEK_COUNTS 395
 STATS 403
 summary 300
 TABS 404
 TABSLINE 406
 USER_STATE 412
 VERSION 413
 XSTATUS 415
attribute bytes, used with tabs 74
AUTOLIST
 assignment statement 308
 macro command 307
 primary command 213
autolist mode
 defined 23
 querying the value 307
 setting the value 213, 307
automatic generation of source listing 213, 307
automatic saving of data 217, 309
AUTONUM
 assignment statement 308
 macro command 308
 primary command 23, 215
autonum mode 23
AUTOSAVE
 assignment statement 309
 macro command 309
 primary command 23, 217
autosave mode, defined 23

B

B (before), line command 50, 166
batch, ending a macro 366

batch processing, submitting data for 288, 404
batch processing, using edit macros in 111
beginning an edit session 4
BLKSIZE, assignment statement 311
block size, retrieving 311
boundaries
 controlling 218, 311
 default 29
 definition line 29
 setting 168
BOUNDS
 assignment statement 311, 312
 line command 168
 macro command 311, 312
 primary command 218
BROWSE
 macro command 313
 primary command 220
built-in command
 disabling 236, 332
 processing 219
built-in labels 66
BUILTIN
 macro command 314
 primary command 219

C

C (copy), line command
 description 170
 used with CREATE command 232
 used with REPLACE command 278
CANCEL
 macro command 315
 primary command 221
canceling edit changes 221, 315
CAPS
 assignment statement 315, 316
 DBCS data 222
 macro command 315, 316
 primary command 24, 221
caps mode
 defined 24
 overview 24
 querying the value 315
 setting the value 221, 315
CHANGE
 macro command
 column-dependent data, defined 57
 DBCS data 57
 description 316, 318
 EBCDIC data 57
 RCHANGE command 273, 381
 saving and restoring values 412
 primary command
 column-dependent data, defined 57
 DBCS data 57

CHANGE (*continued*)
 description 53, 222, 223
 EBCDIC data 57
 qualifying search strings 59
 specifying search strings 54
 repeating 60
 change a data string 222, 316
 change count, retrieving 319
 CHANGE_COUNTS, assignment
 statement 319
 changed lines 27
 changing data 53
 changing models 86
 character string
 changing 222, 223
 finding 243, 342
 how to use 55
 specifying 54
 characters
 converting 221, 315
 converting to lowercase 180
 converting to uppercase 203
 displaying hexadecimal 247, 346
 CLIST CONTROL statements 123
 CLIST edit macro statements 89, 95
 CLIST WRITE statements 122
 COBOL sequence field, defined 33
 COLS, line command 172
 column identification line,
 displaying 172
 column limitations 59
 column positions, referring to 115
 column shifting
 DBCS data 52
 destructive 51
 line command 51
 columns
 identifying 172
 line command 172
 query display 335
 shift left 398
 shift right 399
 command, PROFILE RESET 26
 command, querying 381
 command names, overriding 116
 command procedure statements 96
 command scan mode, setting the
 value 392
 commands, reversing last edit 290
 Compare, edit command 224, 320
 compare command 224, 320
 compare command examples 226
 compare command return codes 321
 compare command syntax 225, 320
 compress data 268, 374
 CONLIST operand, CLIST CONTROL
 statement 123
 CONTROL, ISPEXEC statement 120
 control and display your profile 270, 379
 control edit recovery 273, 383
 control null spaces 265, 370
 control version number 295, 413
 controlled library status, retrieving 324
 controlling the edit boundaries 218, 311
 controlling the edit environment 21
 controlling the search for a data
 string 57

convert characters to lowercase 180
 converting characters 221, 315
 converting note lines to data 186
 COPY
 macro command 322
 primary command
 description 227, 228
 how to use 50
 copy a model into the current data
 set 257, 366
 copying data
 into the current data set 50
 lines of data 170
 macro command 322
 primary command 227
 using edit macro 107
 CREATE
 macro command 323, 324
 primary command
 description 231
 how to use 49
 creating
 a data set member 231, 323
 data 49
 new data 10
 CTL_LIBRARY, assignment
 statement 324
 current member name, querying 365
 CURSOR, assignment statement 326
 positioning cursor on command
 line 327
 cursor position
 querying the value 326
 setting the value 326
 cursor values, saving and restoring 412
 Cut and Save Lines 235, 328
 Cut Macro command 328
 Cut Primary command 235

D
 D (delete) line command 174
 data
 adding 277
 canceling changes 221, 315
 changing 53, 222, 316
 column-dependent, defined 57
 compressing 268, 374
 controlling the string search 57
 converting data 203
 copying 50, 227, 322
 copying lines 170
 creating 49
 creating new 10
 DBCS considerations 57
 deleting 238, 334
 description 223
 EBCDIC considerations 57
 editing existing 11
 excluding 53, 242, 339
 finding 53, 243, 342
 inserting 351
 managing 49
 moving 50
 packing 19
 replacing 49, 277
 retrieving the changed status 329
 retrieving the ID 331

data (*continued*)
 retrieving the width 330
 saving automatically 217, 309
 saving the current 283, 390
 seek a data string 393
 shift left 399
 shift right 400
 shifting 51, 52
 sorting 286, 401
 split a line 409
 submitting for batch processing 288,
 404
 test flow a paragraph 409
 DATA_CHANGED, assignment
 statement 329
 data-changed status, retrieving 329
 data field, defined 371
 data in controlled libraries, editing 19
 data lines, referring to 114
 data modes 24
 data set
 adding a member 386
 copying a model into 257, 366
 creating a member 231, 323
 creating a new 10
 editing a member 239, 338
 editing existing 11
 generating statistics 288, 403
 moving a member 260, 368
 password specification 9
 renumbering lines automatically 274,
 384
 replacing a member 386
 retrieving the current name 331
 security 9
 DATA_WIDTH, assignment
 statement 330
 DATAID, assignment statement 331
 DATASET, assignment statement 331
 DBCS data
 CHANGE command 57
 column shifting 52
 display boundary 10
 hardware tabs 73, 74
 SORT command 287, 402
 TE (text entry) line command 71
 TF (text flow) 69
 TS (text split) line command 70
 debugging edit macros 121
 default operands 154, 209, 300
 DEFINE
 edit macro command 98, 115
 macro command 332
 primary command 236
 define tabs mode 289, 404
 defining
 a name 236, 332
 an alias for a command 115
 an edit profile 21
 defining macros
 implicit 116
 overriding command names 116
 resetting definitions 115
 scope of definitions 115
 using an alias 115
 DELETE
 macro command 334

DELETE (*continued*)
 primary command 238
 deleting
 edit macro labels 113
 labels 66
 lines 174, 238
 models 86
 delimited string 54
 destination, specifying 117
 destructive shift, defined 51
 dialog development models 77
 dialog service errors, debugging 121
 dialog service requests 97
 dialog variable name, defined 105
 direction of the search 58
 disabling a command 115
 disabling a macro or built-in
 command 236, 332
 display and control your profile 270, 379
 display boundary, DBCS data 10
 DISPLAY_COLS, assignment
 statement 335
 display columns 335
 DISPLAY_LINES, assignment
 statement 336
 display model notes 264, 370
 displaying an edit profile 21
 displaying hexadecimal characters 247,
 346
 distributed edit 3
 DOWN, macro command 336
 duplicating lines 190

E

EBCDIC data 57
 edit
 beginning a session 4
 canceling changes 221, 315
 column shifting 51
 command reference section 209
 command summary 16
 considerations 18
 controlling the boundaries 218, 311
 controlling the environment 21
 controlling the recovery 273, 383
 copying data 50
 creating data 49
 data display panel 11
 displaying processed commands 17
 editing data in controlled libraries 19
 ending a session 15
 entry panel 10
 excluding lines 65
 introduction to 3, 14
 line commands 16
 macro command 18, 338
 managing data 49
 models 77
 modes 23
 moving data 50
 number mode 33
 option 2 4
 primary command
 description 239
 example 240
 syntax 239
 primary commands, description 17

edit (*continued*)
 profiles 21
 recursive 239, 338
 replacing data 49
 rules for entering line commands 153
 selecting the editor 4
 sequence number display 33
 sequence number format 32
 sequence numbers 32
 shifting columns 51
 shifting data 51, 52
 splitting text 68
 text entry 68
 text flow 68
 undisplayable characters 14
 undoing edit interactions 74
 word processing 68
 Edit - Entry panel 10
 edit, distributed 3
 edit a member 239, 338
 edit assignment statements
 elements
 keyphrase 105
 overlays 106
 value 104
 how to use 106
 manipulating data 107
 Edit command errors, debugging 121
 edit commands and PF key
 processing 17
 edit compare command 224, 320
 Edit data display panel 11
 edit macro
 alias name 115
 ALLMBRS macro 135
 assignment statements 96, 104
 BOX macro 130
 CLIST macro, differences from
 program macros 98
 column positions, referring to 115
 command procedure statements 96
 command summary 18
 commands 96
 creating 95
 data lines, referring to 114
 defining 115
 definition of 3
 description 89
 dialog service requests 97
 FINDCHGS macro 138
 identifying 362
 IMBED macro 132
 implicit definition using an
 exclamation point 116
 initial macro 29
 introduction to 89
 labels
 description 112
 editor-assigned 112
 passing 114
 referring to 113
 using 112
 levels 112
 line command functions, how to
 perform 108
 MASKDATA macro 141
 messages 111

edit macro (*continued*)
 naming 103
 NOPROCESS operand 116
 parameters 109
 PFCAN macro 129
 PROCESS command and
 operand 116
 program macro
 description 97
 differences from CLIST macros 98
 differences from REXX macros 98
 parameter passing 98
 running 102
 writing 99
 recovery macro 118
 reference section 299
 replacing built-in edit commands 116
 resetting a command to previous
 status 115
 return codes 118
 REXX macro, differences from
 program macros 98
 samples 127
 testing
 CLIST CONTROL statements 123
 CLIST WRITE statements 122
 description 121
 experimenting with edit macro
 commands 124
 return codes 120
 REXX SAY statements 122
 REXX TRACE statements 123
 TEXT macro 127
 TSO commands 97
 using 89
 variable substitution 104
 variables 103
 Edit mode defaults 25
 edit processing of PF keys 17
 edit profile
 autolist mode 213
 autonum mode 215, 308
 autosave mode 217, 309
 boundary settings 168
 caps mode 221
 control and display 270, 379
 defaults 25, 26
 defining 21
 definition of 21
 displaying 21
 initial macro 253, 351
 lock 270, 379
 modifying 23
 naming 21
 note mode 264
 nulls mode 265
 profile name 21
 recovery macro 282
 saving and restoring 412
 specifying 8
 tabs mode 289
 types 21
 Edit Profile Initialization, Site-wide 25
 edit profile name, definition 21
 edit profiles, locking 23
 edit recovery
 Edit Recovery panel 47

- edit recovery (*continued*)
 - turning off 48
 - turning on 47
- edit session, ending 241, 338
- editing existing data 11
- editor, ISPF 4
- editor-assigned labels 66
- eliminating labels 66
- END
 - macro command 338
 - primary command 241
- end a macro 366
- END command 217
- end the edit session 241, 338
- ending an edit session 15
- enter text 197
- error codes for severe errors 119
- error lines 27
- EXCLUDE
 - macro command 339
 - primary command
 - description 53, 242, 243
 - qualifying search strings 59
 - specifying search strings 54
 - repeating 60
- EXCLUDE_COUNTS, assignment statement 341
- exclude counts, querying the value 341
- exclude status of a line, set or query 415
- excluded line limitations 60
- excluded lines, redisplaying 65
- excluding a line 65, 205, 339
- excluding data 53
- explicit shifts, defined 51
- extent of a search 57

F

- F (show first line), line command 175
- FIND
 - macro command
 - description 342, 343
 - RFIND command 282, 388
 - saving and restoring values 412
 - when to use instead of SEEK 394
 - primary command
 - description 53, 243, 244
 - qualifying search strings 59
 - specifying search strings 54
 - repeating 60
- FIND_COUNTS, assignment statement 344
- find counts, querying the value 344
- finding a data string 243
- finding a search string 342
- finding data 53
- finding models 85
- flagged lines
 - changed lines 27
 - error lines 27
 - special lines 27
- FLIP
 - assignment statement 345
 - definition 66
 - macro command 345
 - primary command 245
- FLOW_COUNTS, assignment statement 345

- flow counts, querying the value 345
- Format Name field 10
- formatted edit mode, defined 185
- formatting input 364

G

- generate sequence numbers 266, 372
- generating data set statistics 288, 403
- guidelines for using the editor 18

H

- Hardware Tab field, defined 73
- hardware tabs
 - DBCS data 74
 - defining 73
 - description 71
 - fields, how to use 73
- HEX
 - assignment statement 346
 - macro command 346
 - primary command 24, 247
- hexadecimal characters
 - displaying 247, 346
 - format 24
 - mode 247, 346
 - string 54
- HILITE
 - macro command
 - description 350
 - how to use 347
 - primary command
 - description 253
 - how to use 250
- HILITE function description 34

I

- I (insert) line command 177
- I operand, REXX TRACE statement 123
- identify an edit macro 362
- identify columns 172
- IMACRO
 - assignment statement 351
 - macro command 351
 - primary command 24, 253
- implicit macro definition 116
- implicit shifts, defined 51
- initial macro, specifying 253, 351
- initial macros
 - DEFINE commands used in 115
 - specifying in the EDIT service call 30
 - specifying on the Edit - Entry panel 30
 - starting 29
- Initialization, Site-wide Edit Profile 25
- INSERT, macro command 351
- inserting
 - data 351
 - lines 177
- interactive column numbers 115
- introduction to edit macros 89
- ISPEXEC 97
- ISPF, definition 3
- ISPF list data set 213, 307

- ISPF Workstation Tool Integration dialog 3
- ISRCUT edit macro 421
- ISREDIT service 98
- ISREDIT statements 96, 108
- ISRONLY edit macro 421
- ISRPASTE edit macro 421

K

- keeping an edit command on the command line 17
- keyphrase, defined 105
- kinds of search strings 54

L

- L (show last line), line command 179
- L operand, REXX TRACE statement 123
- LABEL
 - assignment statement
 - description 352, 353
 - overview 112
 - querying the value 352
 - setting the value 352
 - labeled line, querying 359
- labels
 - defined 66
 - deleting 66
 - editor-assigned 66
 - eliminating 66
 - in macro commands 66
 - specifying a range 67
- labels in edit macros
 - deleting 113
 - description 112
 - editor-assigned 112
 - how to use 112
 - levels 112
 - nested macros 113
 - passing 114
 - referring to 113
- languages for edit macros 89, 95
- LC (lowercase), line command 180
- left
 - scroll 353
 - shift columns 398
 - shift data 399
- LEFT
 - macro command 353
- LEVEL
 - assignment statement 354
 - macro command 354
 - primary command 254
- level number, specifying 254, 354
- library status, retrieving 324
- limiting the SORT command 287, 402
- LINE
 - adding 358
 - assignment statement 355
 - querying the number 355
 - querying the value 355
 - setting the value 355
- LINE_AFTER, assignment statement 356
- LINE_BEFORE, assignment statement 358
- Line Command field, resetting 54

- line command functions in edit
 - macros 108
- line command summary 154
- line commands
 - ((column shift left) 156
 -) (column shift right) 158
 - > (data shift right) 162
 - < (data shift left) 159
 - A (after) 163
 - B (before) 166, 167
 - BOUNDS 168
 - C (copy) 170
 - COLS 172
 - D (delete) 174
 - description 153
 - F (show first line) 175
 - I (insert) 177
 - L (show last line) 179
 - LC (lowercase) 180
 - M (move) 182
 - MASK 184
 - MD (make dataline) 186
 - notation conventions 154
 - O (overlay) 188
 - R (repeat) 190
 - rules for entering 153
 - S (show line) 65, 193
 - summary 154
 - TABS 195
 - TE (text entry) 68, 71, 197
 - TF (text flow) 68, 200
 - TS (text split) 68, 202
 - UC (uppercase) 203
 - usage 16
 - X (exclude) 60, 65, 205
- line label
 - querying the value 352
 - setting the value 352
- line number, ordinal 256
- line pointer
 - COPY macro command 322
 - CREATE macro command 323
 - CURSOR assignment statement 326
 - DELETE macro command 334
 - incomplete 324
 - INSERT macro command 352
 - invalid 323, 369
 - LABEL assignment statement 352
 - LINE_AFTER assignment statement 356
 - LINE assignment statement 355
 - LINE_BEFORE assignment statement 358
 - LOCATE macro command 360
 - MASKLINE assignment statement 365
 - MODEL macro command 367
 - MOVE macro command 368
 - referring to labels 114
 - SHIFT (macro command 398
 - SHIFT) macro command 399
 - SHIFT > macro command 400
 - SHIFT < macro command 399
 - SUBMIT macro command 404
 - TABSLINE assignment statement 407
 - TENTER macro command 407
 - TFLOW macro command 409

- line pointer (*continued*)
 - TSPLIT macro command 410
 - XSTATUS assignment statement 415
- line pointer range
 - CREATE macro command 323
 - DELETE macro command 334
 - LOCATE macro command 361
 - REPLACE macro command 386
 - RESET macro command 387
 - SUBMIT macro command 404
- line range 67
- LINENUM, assignment statement 359
- lines
 - adding 177
 - copying 170
 - deleting 174, 334
 - exclude status 415
 - excluded limitations 60
 - excluding 65, 242, 339
 - inserting 177
 - locating 255, 360
 - moving 182
 - numbering automatically 215
 - overlying 188
 - query display 336
 - renumbering automatically 274, 384
 - repeating 190
 - show 193
 - show the first 175
 - showing the last 179
 - specifying ranges 66
 - splitting 70, 409
- literal character string, defined 104
- LMF 6
- LMF lock — errors 6
- LMF lock ignored 6
- LOCATE
 - macro command
 - generic syntax 360
 - specific syntax 360
 - primary command
 - generic syntax 256
 - specific syntax 256
- locate lines 255, 360
- Lock — Never 6
- Lock — No 6
- Lock — Yes 6
- lock your profile 270, 379
- locking an edit profile 23
- logical record length, querying 362
- logical tabs, description 72
- lowercase operands 154, 209, 300
- lptr
 - COPY macro command 322
 - CURSOR assignment statement 326
 - DELETE macro command 334
 - incomplete 324
 - INSERT macro command 352
 - invalid 323, 369
 - LABEL assignment statement 352
 - LINE_AFTER assignment statement 356
 - LINE assignment statement 355
 - LINE_BEFORE assignment statement 358
 - LOCATE macro command 360

- lptr (*continued*)
 - MASKLINE assignment statement 365
 - MODEL macro command 367
 - MOVE macro command 368
 - referring to labels 114
 - SHIFT (macro command 398
 - SHIFT) macro command 399
 - SHIFT > macro command 400
 - SHIFT < macro command 399
 - TABSLINE assignment statement 407
 - TENTER macro command 407
 - TFLOW macro command 409
 - TSPLIT macro command 410
 - XSTATUS assignment statement 415
- lptr-range
 - CREATE macro command 323
 - DELETE macro command 334
 - LOCATE macro command 361
 - REPLACE macro command 386
 - RESET macro command 387
 - SUBMIT macro command 404
- LRECL, assignment statement 362

M

- M (move), line command
 - description 182
 - used with CREATE command 232
 - used with REPLACE command 278
- macro
 - ending in batch 366
 - specifying a recovery 282, 390
 - specifying an initial 253, 351
- MACRO, macro command 362
- Macro Command Profile Reset Syntax 380
- macro commands
 - abbreviations 419
 - assignment statements 104
 - AUTOLIST 307
 - AUTONUM 308
 - AUTOSAVE 309
 - BOUNDS 311
 - BROWSE 313
 - BUILTIN 314
 - CANCEL 315
 - CAPS 315
 - CHANGE 316
 - COPY 322
 - CREATE 323
 - DEFINE 332
 - DELETE 334
 - disabling 236, 332
 - DOWN 336
 - EDIT 338
 - END 338
 - EXCLUDE 339
 - FIND 342
 - FLIP 345
 - HEX 346
 - HILITE 347
 - identifying 236, 332
 - IMACRO 351
 - INSERT 351
 - introduction to 89
 - ISRCUT 421
 - ISRONLY 421

macro commands (*continued*)

ISRPASTE 421
labels 66
LEFT 353
LEVEL 354
LOCATE 360
MACRO 362
MEND 366
MODEL 366
MOVE 368
NONNUMBER 369
notation conventions 299
NOTES 370
NULLS 370
NUMBER 372
PACK 374
PROCESS 377
PROFILE 379
RCHANGE 273, 381
RECOVERY 383
reference section 299
RENUM 384
REPLACE 386
RESET 386
RFIND 282, 388
RIGHT 389
RMACRO 118, 390
SAVE 390
SCAN 392
SEEK 53, 393
SETUNDO 396
SHIFT (398
SHIFT) 399
SHIFT > 400
SHIFT < 399
SORT 401
STATS 403
SUBMIT 404
summary 300
TABS 404
TENTER 68, 407
TFLOW 68, 409
TSPLIT 68, 409
UNNUMBER 410
UP 411
usage 18
VERSION 413
VIEW 414

Macro Commands

CUT 328
PASTE 375

macro definitions, resetting 115

MACRO_LEVEL, assignment statement 114, 364

macro nesting level

querying 364
retrieving 112

managing data 49

mask, defined 184

MASK, line command 184

mask line, set or query 364

MASKLINE, assignment statement

description 364, 365
overlays 106
using 106

MD (make dataline), line command 186

MEMBER, assignment statement 365

member, editing 239, 338

member name, querying 365

MEND, macro command 366

messages, displayed from edit macros 92, 111

mixed data, used with data strings 98

Mixed Mode field 10

model

adding 81

changing 81, 86

class, defined 77

copying into the current data set 257, 366

deleting 81, 86

edit, defined 77

finding 81, 85

hierarchy 77

kinds 77

locating 85

logical name 77

macro command 366

name, defined 78

primary command 257

qualifier, defined 78

using 79

model notes, displaying 264, 370

model selection panels 79

modes, edit 23, 24

modification flag 256

modification level, description 31

modification level number, specifying 254, 354

modifying an edit profile 23

MOUNT authority 9

MOVE

macro command 368

primary command 50, 260

move a data set member 260, 368

moving a line of data in an edit

macro 108

moving data into the current data set 50

moving lines 182

multiple parameters in an edit

macro 110

N

name, defining 236, 332

naming edit macros 103

nested macros, starting 112

nesting level, querying 364

NOCONLIST operand, CLIST CONTROL statement 123

NOLIST operand, CLIST CONTROL statement 123

non-destructive shifting, defined 52

NONNUMBER

macro command 369

primary command 264

NOPROCESS 116

normal, defined for stats mode 31

NOSYMLIST operand, CLIST CONTROL statement 123

notation conventions

line commands 154

macro commands 299

primary commands 209

note lines, converting to data 186

note mode

description of 24

querying the value 370

setting the value 264, 370

NOTES

assignment statement 370

macro command 370

primary command 24, 264

notes, displaying model 264, 370

null spaces, controlling 265, 370

NULLS

assignment statement 370

macro command 370

primary command 24, 265

nulls mode

description of 24

querying the value 370

setting the value 265, 370

NUMBER

assignment statement 372

macro command 372

primary command

description 24, 266

DISPLAY operand 33

number, specifying the modification

level 254, 354

number mode

defined 24

description 24, 266

initializing 33

setting, edit 32

turning off 264, 369

used with RENUM command 274, 384

numbering lines automatically 215, 308

numbers

controlling version 295, 413

generating sequence 266, 372

modification level 31

remove sequence 293, 410

sequence 31

turning off number mode 264, 369

O

O (overlay), line command 188

O operand, REXX TRACE statement 123

operand notation

lowercase 154, 209, 300

OR symbol (|) 209, 300

stacked 154, 209, 300

underscored defaults 154, 209, 300

ordinal line number 256

overlying lines 188

overlays, guidelines on how to

perform 106

overriding, built-in edit commands 116

P

PACK

assignment statement 374

macro command 374

primary command 24, 268

pack mode 24, 268

packing data, edit 19

panel

excluding lines 205

- panel (*continued*)
 - process the 377
 - resetting the 386
 - set up for text entry 407
- panel data, resetting 280
- panel values, saving and restoring 412
- panels
 - Edit data display 11
 - Edit Entry 7, 241
 - edit profile display 22, 272
 - Edit Recovery 47
 - model selection 79
- parameters in an edit macro 109
- passing labels 114
- passing parameters to an edit macro
 - description 109
 - multiple 110
 - processing an Edit command 98
 - program macros 98
- password protection 9
- Paste Lines 268, 375
- Paste Macro command 375
- Paste Primary command 268
- PDF, defined 3
- PF key processing in edit 17
- PF keys, scroll commands 15
- picture string 54, 55
- power typing, defined 71
- prepare display for data insertion 351
- Preserve command 269
- PRESERVE command 16
- PRESERVE macro 376
- primary commands
 - abbreviations 419
 - AUTOLIST 23, 213
 - AUTONUM 23, 215
 - AUTOSAVE 23, 217
 - BOUNDS 218
 - BROWSE 220
 - BUILTIN 219
 - CANCEL 221
 - CAPS 24, 221
 - CHANGE 53, 222
 - COPY 50, 227
 - CREATE 49, 231
 - DEFINE 236
 - DELETE 238
 - displaying after processing 17
 - EDIT 239
 - END 241
 - EXCLUDE 53, 242
 - FIND 53, 243
 - FLIP 66, 245
 - HEX 24, 247
 - HILITE 250
 - IMACRO 24, 253
 - LEVEL 254
 - LOCATE 255
 - MODEL 257
 - MOVE 50, 260
 - NONUMBER 264
 - notation conventions 209
 - NOTES 24, 264
 - NULLS 24, 265
 - NUMBER 24, 266
 - PACK 24, 268
 - PROFILE 23, 270

- primary commands (*continued*)
 - RECOVERY 24, 273
 - reference section 209
 - RENUM 274
 - REPLACE 49, 277
 - RESET 66, 280
 - RMACRO 282
 - SAVE 283
 - SETUNDO 24, 284
 - SORT 286
 - STATS 24, 288
 - SUBMIT 288
 - summary 209
 - TABS 24, 289
 - UNDO 290
 - UNNUMBER 293
 - usage 17
 - VERSION 295
 - VIEW 296
- Primary Commands
 - CUT 235
 - PASTE 268
- PROCESS, macro command
 - description 378
 - used with RANGE_CMD assignment statement 381
- PROCESS command and operand 116
- processing built-in commands 219, 314
- PROFILE
 - assignment statement 379
 - macro command
 - description 379
 - profile control syntax 379
 - profile lock syntax 379
 - primary command
 - description 23, 271
 - display or define a profile 21
 - profile control syntax 270
 - profile lock syntax 270
- profile, edit
 - autolist mode 213, 366
 - autonum mode 215, 308
 - autosave mode 217, 309
 - boundaries 218
 - boundary settings 168
 - caps mode 221
 - control and display 270, 379
 - defining 21
 - description 21
 - displaying 21
 - initial macro 253, 351
 - lock 270, 379
 - locking 23
 - modifying 23
 - note mode 264
 - nulls mode 265
 - recovery macro 282
 - saving and restoring 412
 - tabs mode 289
 - types 21
- profile defaults 25, 26
- PROFILE RESET command 26
- Profile Reset Syntax 271
- Profile Reset Syntax, Macro Command 380
- program macros
 - defined 97

- program macros (*continued*)
 - differences from CLISTS 98
 - differences from REXX EXECs 98
 - how to write 99
 - implicit definition 116
 - passing parameters 98
 - running 102
- pseudo-lock, defined 325

Q

- qualifying the search string 59
- query
 - a line 355
 - autolist mode 307
 - autonum mode 308
 - autosave mode 309
 - block size 311
 - caps mode 315
 - change count 319
 - command entered 381
 - controlled library status 324
 - current member name 365
 - cursor position 326
 - data-changed status 329
 - data ID 331
 - data set name 331
 - data width 330
 - display columns 335
 - display lines 336
 - edit boundaries 311
 - edit profile 379
 - exclude counts 341
 - exclude status for a line 415
 - find counts 344
 - flow counts 345
 - hexadecimal mode 346
 - initial macro 351
 - line label 352
 - line number 359
 - logical record length 362
 - macro nesting level 364
 - mask line 364
 - modification level number 354
 - note mode 370
 - nulls mode 370
 - number mode 372
 - pack mode 374
 - record format 382
 - recovery mode 383
 - seek counts 395
 - tabs line 406
 - tabs mode 404
 - version number 413
- Query Volume Information 414

R

- R (repeat) line command 190
- R operand, REXX TRACE statement 123
- range
 - specifying 117
 - using labels to specify 67
- RANGE_CMD, assignment statement
 - description 117, 381
 - used with the PROCESS command 381

RC variable 119

RCHANGE, macro command
 description 273, 381, 382
 used to repeat CHANGE
 command 60

RECFM, assignment statement 382

record format, query 382

recovery
 controlling edit 273, 383
 edit 47
 macro 118, 282, 390
 mode 24, 273, 383

RECOVERY
 assignment statement 383
 macro command 383
 primary command 24, 273

recursive editing, defined 239, 338

redisplaying excluded lines 65

referring to column positions 115

referring to data lines 114

reformatting a paragraph 200

relative line number of cursor, setting or
 retrieving 326

relative line numbers 114

remove sequence numbers 293, 410

removing lines 238, 334

RENUM
 macro command 384
 primary command 274

RENUMBER primary command,
 DISPLAY operand 33

renumbering lines automatically 274,
 384

repeating a change 273, 381

repeating a search
 RCHANGE command, Edit 60
 RFINN command, Edit 60

repeating lines 190

REPLACE
 macro command 386
 primary command
 description 277, 278
 how to use 49

replace a data set member 386

replacing
 data 49, 277
 lines 108

RESET
 macro command 386
 primary command 280

RESET command, PROFILE 26

reset the data display 386

reset the data panel 280

resetting macro definitions 115

resetting the Line Command field 54

retrieving the change count 319

retrieving the controlled library
 status 324

retrieving the data-changed status 329

retrieving the data ID 331

retrieving the data set name 331

retrieving the data width 330

return codes
 &LASTCC variable 119
 0 to 20 118
 above 20 119
 ISPF editor 119

return codes (*continued*)
 RC variable 119

reverse last data change 290

REXX edit macro statements 89, 95

REXX SAY statements, using to debug
 edit macros 122

REXX TRACE statements, using to debug
 edit macros 123

RFINN command
 description 282, 388
 used to repeat FINN and EXCLUDE
 commands 60

RIGHT
 macro command 389
 scroll 389

RMACRO
 assignment statement
 description 390
 overview 118
 macro command 390
 primary command
 description 282, 283
 overview 118

S

S (show line), line command
 description 193
 redisplaying excluded lines 65

S operand, REXX TRACE statement 123

sample edit macros 127

SAVE
 macro command 390, 391
 primary command 283

save data automatically 217, 309

SAVE_LENGTH command 391

save the current data 283, 390

saving and restoring
 CHANGE macro command
 values 412
 cursor and panel values 412
 edit profile 412
 FINN macro command values 412

SCAN
 assignment statement 392
 macro command 392

SCAN assignment statement 104

scope of macro definitions 115

scroll
 down 336
 left 353
 right 389
 up 411
 using PF keys 15

search
 controlling 57
 DBCS search string, delimiting 54
 extent 57
 qualifying 59
 starting point and direction 58

search strings
 character 54
 delimited 54
 finding 342
 hexadecimal 54
 picture 54
 simple 54

security, data set 9

SEEK, macro command
 description 53, 393, 394
 when to use instead of FINN 343

seek a data string 393

SEEK_COUNTS, assignment
 statement 395

seek counts, query 395

sequence numbers
 display 33
 format 32
 generating 266, 372
 initializing 33
 setting, edit 32

set
 a line 355
 autolist mode 307
 autonom mode 308
 autosave mode 309
 caps mode 315
 command scan mode 392
 cursor position 326
 edit boundaries 218, 311
 edit profile 379
 exclude status for a line 415
 hexadecimal mode 247, 346
 initial macro 351
 line label 352
 mask 184
 mask line 364
 modification level number 354
 note mode 264, 370
 nulls mode 265, 370
 number mode 372
 pack mode 374
 recovery mode 383
 tabs line 406
 tabs mode 289, 404
 version number 413

set UNDO command 284

setting the edit boundaries 218, 311

SETUNDO
 macro command 396
 primary command 75, 284

SHIFT (, macro command 398

SHIFT), macro command 399

SHIFT >, macro command 400

SHIFT <, macro command 399

shift columns
 left 398
 right 399

shift data
 left 399
 right 400

shifting data
 edit
 columns 51
 explicit 51
 implicit 51
 non-destructive 52

show lines 193

show the first line 175

show the last line 179

SI characters, delimiting a search 54

simple editing 14

simple string 54

Site-wide Edit Profile Initialization 25

- site-wide macro 18
- SO characters, delimiting a search 54
- software tab field, defined 196
- software tabs
 - defining 73
 - description 71
 - fields, how to use 196
- SORT**
 - macro command
 - DBCS data 402
 - description 401
 - limiting 402
 - without operands 401
 - primary command
 - DBCS data 287
 - description 286
 - limiting 287
 - without operands 286
 - sorting data 286, 401
 - source listing, create 213, 307
 - spaces, controlling null 265, 370
 - special lines 27
 - specify a recovery macro 118, 282, 390
 - specifying
 - an initial macro 18, 29, 253, 351
 - the level number 254, 354
 - split screen limitations 59
 - splitting a line of text 202
 - splitting lines 70
 - splitting text 68
 - stacked operands 154, 209, 300
 - standard sequence field, defined 32
 - starting point of a search 58
 - statistics
 - creation and maintenance of 30
 - generating for a data set 288, 403
- STATS**
 - assignment statement 403
 - macro command 403
 - primary command 24, 288
- stats mode 24, 30
- strings, kinds of search
 - character 54
 - delimited 54
 - hexadecimal 54
 - picture 54
 - simple 54
- SUBMIT**
 - macro command 404
 - primary command 288
- submit data for batch processing 288, 404
- SYMLIST operand, CLIST CONTROL statement 123
- Syntax, Macro Command Profile
 - Reset 380
- Syntax, Profile Reset 271

T

- TABS**
 - assignment statement 404
 - controlling and querying 72, 404
 - line command
 - defining hardware tabs 73
 - defining software tabs 73
 - description 195
 - limiting hardware tab columns 73

- TABS** (*continued*)
 - line command (*continued*)
 - using software tab fields 196
 - macro command 404
 - primary command 24, 289
- tabs line
 - querying the value 406
 - setting the value 406
- tabs mode
 - description 24, 72
 - setting the value 289, 404
- TABSLINE, assignment statement 406
- TE (text entry), line command
 - DBCS data, using a DBCS terminal 71
 - description 71, 197
 - example 198
 - syntax 197
- template (overlay)
 - definition 106
 - how to design 106
- TENTER, macro command 407
- text entry
 - in word processing 68
 - line command 197
 - setting up the panel 407
- text flow 68
- text flowing a paragraph 200, 409
- text split a line 409
- TF (text flow), line command
 - DBCS data, using a DBCS terminal 69
 - description 68, 200
- TFLOW, macro command 409
- TS (text split), line command
 - DBCS data 70
 - description 202
- TSO commands in edit macros 97
- TSPLIT, macro command 409
- turn off number mode 264, 369

U

- UC (uppercase), line command 203
- underscored operands 154, 209, 300
- undisplayable characters 14
- UNDO**
 - primary command 290
 - SETUNDO requirement 396
 - with SETUNDO macro 284
- undoing edit interactions
 - description 290, 291
 - how to use 74
 - UNDO primary command 290
- UNDOSIZE 75
- UNNUMBER**
 - macro command 410
 - primary command 293
- UP, macro command 411
- uppercase, converting data to 203
- uppercase commands and operands 154, 209, 299
- USER_STATE, assignment statement 412
- using the ISPF editor 3

V

- value portion of an edit macro statement 104
- variable substitution, controlling 104
- variables in edit macros 103
- verifying parameters 116
- VERSION**
 - assignment statement 413
 - macro command 413
 - primary command 295
- version number
 - controlling 295, 413
 - description 31
- VIEW**
 - macro command 414
 - primary command 296
- VOLUME** assignment statement 414
- Volume Information 414

W

- writing program macros 97, 99

X

- X (exclude), line command
 - using 60, 65
- XSTATUS, assignment statement 415

Z

- ZDEFAULT edit profile 26
- ZEDITCMD variable 110
- ZEDLMSG 111
- ZEDSAVE variable 329
- ZEDSMSG 111
- ZUSERMAC variable 30

Readers' Comments — We'd Like to Hear from You

Interactive
System Productivity Facility (ISPF)
Edit and Edit Macros
OS/390 Version 2 Release 8.0

Publication No. SC28-1312-03

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

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

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape



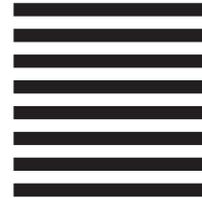
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department CGMD / Bldg 062
P.O. Box 12195
Research Triangle Park, NC
27709-2195



Fold and Tape

Please do not staple

Fold and Tape



File Number: S370/4300-39
Program Number: 5647-A01



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

SC28-1312-03

