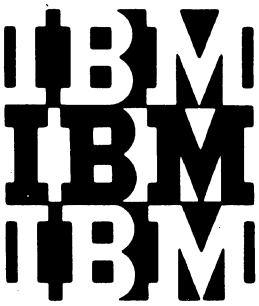
File No. S360-36
Order No. GY28-6773-1



Program Logic

IBM System/360 Operating System:

Time Sharing Option

Command Processor Program Logic Manual

Volume 3

EDIT

Program Number 360S-UT-506

This publication describes the internal logic of the EDIT Command Processor program (program number 360S-UT-506). Included in this manual are discussions of the organization and method of operation of the program, a microfiche directory, tables of data layouts, flowcharts, and a glossary.

The EDIT program processes the EDIT command and subcommands, which are a part of the TSO command language. The EDIT command and its subcommands create and modify data sets. The subcommands are:

BOTTOM	INPUT	RUN
CHANGE	INSERT	SAVE
DELETE	Insert/Replace/Delete	SCAN
DOWN	LIST	TABSET
END	MERGE	TOP
FIND	PROFILE	UP
FORMAT	RENUM	VERIFY
HELP		

This manual is intended for use by persons maintaining the EDIT Command Processor program, or by systems programmers who are altering the program design. It is not intended, nor is it needed for normal operation of the program.

IBM System/360 Operating System: Time Sharing Option, Terminal Monitor Program and Service Routines Program Logic Manual (GY28-6770) contains prerequisite information.

Second Edition (March, 1972)

This edition applies to Release 21 of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest System/360 and System/370 SRL Newsletter, GN20-0360, for the editions that are applicable and current.

This edition, GY28-6773-1, is a major revision of, and obsoletes GY28-6773-0.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratory, Publications Dept., P.O.Box 24, Uithoorn, The Netherlands. Comments become the property of IBM.

Preface

This publication provides IBM customer engineers and other technical personnel with a description of the internal logic and organization of the IBM System/360 Operating System Time Sharing Option EDIT Command Processor.

The publication IBM System/360 Operating System: Time Sharing Option, Terminal Monitor Program and Service Routines Program Logic Manual (GY28-6770) contains prerequisite information. Co-requisite publications are:

IBM System/360 Operating System: Time Sharing Option:

Command Language Reference (GC28-6732) explains the use of TSO commands.

Command Processor Program Logic Manual Volume 4 (GY28-6774) describes the internal logic of the TSO HELP command.

Command Processor Program Logic Manual Volume 6 (GY28-6776) describes the internal logic of the TSO PROFILE and RUN commands.

The following co-requisite publications are only required if the installation has TSO Data Utilities Program Product 5734-UT1:

TSO Data Utilities Copy, Format, List, Merge User's Guide and Reference Manual (SC28-6765) (Program Product publication) explains the use of the TSO FORMAT and MERGE commands.

TSO Data Utilities Copy, Format, List, Merge Program Logic Manual (LY28-6766) (Licensed Program Product publication) describes the internal logic of the TSO FORMAT and MERGE commands.

Specific additional information is available in:

IBM System/360 Operating System:

Supervisor Services and Macro Instructions (GC26-6646) describes linkage conventions for subtasks.

System Programmer's Guide (GC28-6550) contains a description of the ABEND work area.

This manual comprises the following seven sections:

Section 1: Introduction - a discussion of the purpose and function of the EDIT Command Processor.

Section 2: Method of Operation - a functionally oriented description of the way in which the EDIT Command Processor handles the EDIT command and subcommands and error conditions. Method of operation diagrams are on foldout pages at the back of the manual.

Section 3: Program Organization - a description of specific program components (modules and Csects) and flowcharts.

Section 4: Directory - tabular aids for the reader to find sections of code in the microfiche and lists of assembly modules and control sections.

Section 5: Data Areas - detailed descriptions of the various tables, work areas, and parameter lists used by the EDIT Command Processor.

Section 6: Diagnostic Aids - a compilation of information designed to facilitate diagnosis of EDIT program errors.

Section 7: Appendix - formats of the EDIT command and subcommands.

Glossary: A compilation of terms unique to TSO and to the EDIT Command Processor.

More detailed information about the EDIT Command Processor may be found in the associated microfiche listing.

Contents

SUMMARY OF AMENDMENTS (RELEASE 21)
GY28-6773-1 11

SUMMARY OF AMENDMENTS (RELEASE 20.1)
GY28-6773-0) 11

SECTION 1: INTRODUCTION 13
The TSO Environment 13
Starting TSO 14
Logging On 15
Processing TSO Commands 16
TSO Service Routines 17
The EDIT Command Processor 18

SECTION 2: METHOD OF OPERATION 21
Input Processed by the EDIT Program . 21
The Current Line Pointer 25
Output Created by EDIT Program 26
Termination or Suspension of EDIT Program Operation 27
Suspension of EDIT Program Operation 27
Normal Termination of EDIT Program Operation 27
Error Termination of EDIT Program Operation 28
Syntax Checking 29
Use of IPLI and BASIC Syntax Checkers 29
Initialization 33
Processing EDIT Operands 33
Prompting User for Data Set Type . . 48
Obtaining Processor-Dependent Information 48
Termination Processing 48
Controller 49
Controller Processing 49
Error and Attention Handling 53
Abnormal End Exit Routine 53
Attention Exit Routine 53
EDIT Service Routines 54
Invoking TSO Commands (IKJEBECI) . . 55
Initial Copying (IKJEBECO) 56
Allocating and Freeing Data Sets (IKJEBEDA) 56
Final Copying (IKJEBEFC) 57
Line Editing (IKJEBELE) 57
Translating (IKJEBEMR) 58
Selecting EDIT Messages (IKJEBEMS) . 58
Processor Data Table Searching (IKJEBEPS) 60
BASIC Renumbering (IKJEBERN) 61
String Searching (IKJEBESE) 61
BOTTOM Subcommand Processing 62
BOTTOM Processing 62
CHANGE Subcommand Processing 64
Interpreting Operand Combinations . 64
Processing CHANGE Operands 65
DELETE Subcommand Processing 69
Processing DELETE Operands 69
Updating the Current Line Pointer . 70
Invoking the Syntax Checker 70
DOWN Subcommand Processing 72
DOWN Processing 72
END Subcommand Processing 74
FIND Subcommand Processing 76
FIND Processing 76
FORMAT Subcommand Processing 80
Processing FORMAT Operands 80
HELP Subcommand Processing 83
INPUT Subcommand Processing 85
INPUT Processing 86
INSERT Subcommand Processing 93
Processing INSERT Operands 93
Line Insert/Replace/Delete Processing . 96
Processing Line Insert/Replace/Delete Operands . . . 96
Updating the Current Line Pointer . 98
LIST Subcommand Processing 98
Processing LIST Operands 98
MERGE Subcommand Processing 102
Processing MERGE Operands 102
PROFILE Subcommand Processing 106
RENUM Subcommand Processing 109
Processing RENUM Operands 109
RUN Subcommand Processing 112
RUN Processing for Particular Data Set Types 112
GOFORT Data Set Type 112
SAVE Subcommand Processing 116
Processing the SAVE Subcommand 116
SCAN Subcommand Processing 119
Processing SCAN Operands 119
TABSET Subcommand Processing 125
Processing TABSET Operands 125
TOP Subcommand Processing 128
TOP Processing 128
UP Subcommand Processing 130
UP Processing 130
VERIFY Subcommand Processing 132
Processing VERIFY Operands 132
EDIT Access Method 135
Structure 135
Initialization and Final Processing .142
Interface 142
Read, Write, and Delete Operations . .142
Access Method Service Routines 143

SECTION 3: PROGRAM ORGANIZATION 145
Module and Csect Operation Tables . . .154
IKJEBEAA 154
IKJEBEAD (Csect of IKJEBEAA) 155
IKJEBEAE (Csect of IKJEBEMA) 156
IKJEBEAS (Csect of IKJEBEAA) 157
IKJEBEAT (Csect of IKJEBEMA) 158
IKJEBEBO 159
IKJEBECG 160
IKJEBECH 161
IKJEBECI 162
IKJEBECN 164
IKJEBECO 165
IKJEBEDA 166
IKJEBEDE 167
IKJEBEDL (Csect of IKJEBEAA) 168

IKJEBEDO 169
IKJEBEDR (Csect of IKJEBEAA) 170
IKJEBEDS (Csect of IKJEBEAA) 171
IKJEBEDU (Csect of IKJEBEAA) 172
IKJEBEEN 173
IKJEBEEX 174
IKJEBEFC 175
IKJEBEFI 176
IKJEBEFO 177
IKJEBEHE 178
IKJEBEIA 179
IKJEBEIM 180
IKJEBEIN 181
IKJEBEIP 182
IKJEBEIS 184
IKJEBELE 185
IKJEBELI 186
IKJEBELO (Csect of IKJEBEAA) 187
IKJEBELT 188
IKJEBEMA 189
IKJEBEMA (CSECT OF IKJEBEMA) 190
IKJEBEME 192
IKJEBEMR 193
IKJEBEMS 194
IKJEBEMV (Csect of IKJEBEAA) 195
IKJEBEPD (Csect of IKJEBEPS) 196
IKJEBEPS 197
IKJEBEPS (Csect of IKJEBEPS) 198
IKJEBERB (Csect of IKJEBEAA) 199
IKJEBERE 200
IKJEBERN 202
IKJEBERR (Csect of IKJEBEAA) 203
IKJEBERU 204
IKJEBESA 205
IKJEBESA (Csect of IKJEBESA) 206
IKJEBESC 207
IKJEBESE 208
IKJEBESN 209
IKJEBETA 210
IKJEBETO 211
IKJEBEUI 212
IKJEBEUP 213
IKJEBEUT 214
IKJEBEVE 215
IKJEBEWA (Csect of IKJEBEAA) 216
IKJEBEWB (Csect of IKJEBEAA) 217
IKJEBEWR (Csect of IKJEBEAA) 218
IKJEBMA1 (Csect of IKJEBEMA) 219
IKJEBMA2 220
IKJEBMA8 221
IKJEBMA9 222
IKJEBSA9 223
Flowcharts 225

SECTION 4: DIRECTORY 353

SECTION 5: DATA AREAS 357
EDIT Communication Area (IKJEBECA) . . .357
Processor Data Table (IKJEBEPD) 371
IBM- and User-Supplied Tables of
Subcommands 376
EDIT Access Method Work Area (UTILWORK) 377
EDIT Access Method Control Blocks . . .379
Data Blocks 379
Directory Blocks and Lower-level
Directory Blocks 379
Buffers 380
Command Buffer (CBUF) 380
Command Processor Parameter List (CPPL) 381
Command Scan Parameter List (CSPL) . . .381
Command Scan Output Area (CSOA) 382
DAIR Parameter Block (DAPB08) 383
DAIR Parameter Block (DAPB18) 386
GETLINE Parameter Block (GTPB) 388
PUTGET Parameter Block (PGPB) 389
PARSE Parameter List (PPL) 390
PUTLINE Parameter Block (PTPB) 391
Syntax Checker Control Blocks 392
Buffer 392
Command Interface (DELETE Subcommand) 392
Command Interface (RUN Subcommand) . .392
Syntax Checker Communication Area . .393
Option Word 394

SECTION 6: DIAGNOSTIC AIDS 397
Message Cross-Reference Table 397
Using the TEST Command to Diagnose
Errors in the EDIT Program 400
EDIT Messages 401
Module Cross-Reference Table 405
CSECT Cross-Reference Table (Module
IKJEBEAA) 407
Error and Execptional Conditions Tables 408
Register Usage Table 453

SECTION 7: APPENDIX 457
Formats of the EDIT Command and
Subcommands 458
EDIT Command 458
BOTTOM Subcommand 459
CHANGE Subcommand 459
DELETE Subcommand 459
DOWN Subcommand 459
END Subcommand 459
FIND Subcommand 459
FORMAT Subcommand 460
HELP Subcommand 460
INPUT Subcommand 460
INSERT Subcommand 460
Insert/Replace/Delete Subcommand . . .460
LIST Subcommand 460
MERGE Subcommand 461
PROFILE Subcommand 461
RENUM Subcommand 461
RUN Subcommand 461
SAVE Subcommand 462
SCAN Subcommand 462
TABSET Subcommand 462
TOP Subcommand 462
UP Subcommand 462
VERIFY Subcommand 462

GLOSSARY 463

INDEX 467

Chart AA. IKJEBEAD 225
Chart AB. IKJEBEAE 226
Chart AC. IKJEBEAS 227
Chart AD. IKJEBEAT 228
Chart AE. IKJEBEBO 229
Chart AF. IKJEBECG 230
Chart AG. IKJEBECG 231
Chart AH. IKJEBECG 232
Chart AI. IKJEBECG 233
Chart AJ. IKJEBECG 234
Chart AK. IKJEBECG 235
Chart AL. IKJEBECH 236
Chart AM. IKJEBECH 237
Chart AN. IKJEBECH 238
Chart AO. IKJEBECI 239
Chart AP. IKJEBECI 240
Chart AQ. IKJEBECN 241
Chart AR. IKJEBECN 242
Chart AS. IKJEBECN 243
Chart AT. IKJEBECN 244
Chart AU. IKJEBECO 245
Chart AV. IKJEBECO 246
Chart AW. IKJEBECO 247
Chart AX. IKJEBEDA 248
Chart AY. IKJEBEDE 249
Chart AZ. IKJEBEDE 250
Chart BA. IKJEBEDE 251
Chart BB. IKJEBEDL 252
Chart BC. IKJEBEDL 253
Chart BD. IKJEBEDL 254
Chart BE. IKJEBEDO 255
Chart BF. IKJEBEDO 256
Chart BG. IKJEBEDR 257
Chart BH. IKJEBEDS 258
Chart BI. IKJEBEDS 259
Chart BJ. IKJEBEDS 260
Chart BK. IKJEBEDU 261
Chart BL. IKJEBEDU 262
Chart BM. IKJEBEDU 263
Chart BN. IKJEBEDU 264
Chart BO. IKJEBEDU 265
Chart BP. IKJEBEEN 266
Chart BQ. IKJEBEEX 267
Chart BR. IKJEBEFC 268
Chart BS. IKJEBEFC 269
Chart BT. IKJEBEFI 270
Chart BU. IKJEBEFO 271
Chart BV. IKJEBEFO 272
Chart BW. IKJEBEHE 273
Chart BX. IKJEBEIM 274
Chart BY. IKJEBEIM 275
Chart BZ. IKJEBEIM 276
Chart CA. IKJEBEIM 277
Chart CB. IKJEBEIM 278
Chart CC. IKJEBEIN 279
Chart CD. IKJEBEIN 280
Chart CE. IKJEBEIN 281
Chart CF. IKJEBEIN 282
Chart CG. IKJEBEIN 283
Chart CH. IKJEBEIN 284
Chart CI. IKJEBEIN 285
Chart CJ. IKJEBEIN 286
Chart CK. IKJEBEIP 287
Chart CL. IKJEBEIP 288
Chart CM. IKJEBEIS 289
Chart CN. IKJEBEIS 290
Chart CO. IKJEBELE 291
Chart CP. IKJEBELI 292
Chart CQ. IKJEBELI 293
Chart CR. IKJEBELO 294
Chart CS. IKJEBELO 295
Chart CT. IKJEBELT 296
Chart CU. IKJEBELT 297
Chart CV. IKJEBELT 298
Chart CW. IKJEBEMA 299
Chart CX. IKJEBEMA 300
Chart CY. IKJEBEMA 301
Chart CZ. IKJEBEME 302
Chart DA. IKJEBEME 303
Chart DB. IKJEBEMR 304
Chart DC. IKJEBEMS 305
Chart DD. IKJEBEMV 306
Chart DH. IKJEBEPS 307
Chart DI. IKJEBERB 308
Chart DJ. IKJEBERE 309
Chart DK. IKJEBERE 310
Chart DL. IKJEBERE 311
Chart DM. IKJEBERE 312
Chart DN. IKJEBERN 313
Chart DO. IKJEBERR 314
Chart DP. IKJEBERU 315
Chart DQ. IKJEBERU 316
Chart DR. IKJEBESA 317
Chart DS. IKJEBESA 318
Chart DT. IKJEBESA 319
Chart DU. IKJEBESA 320
Chart DV. IKJEBESA 321
Chart DW. IKJEBESA 322
Chart DX. IKJEBESA 323
Chart DY. IKJEBESA 324
Chart DZ. IKJEBESA 325
Chart EA. IKJEBESA 326
Chart EB. IKJEBESA 327
Chart EC. IKJEBESC 328
Chart ED. IKJEBESC 329
Chart EE. IKJEBESC 330
Chart EF. IKJEBESC 331
Chart EG. IKJEBESC 332
Chart EH. IKJEBESE 333
Chart EI. IKJEBESN 334
Chart EJ. IKJEBESN 335
Chart EK. IKJEBESN 336
Chart EL. IKJEBESN 337
Chart EM. IKJEBESN 338
Chart EN. IKJEBESN 339
Chart EO. IKJEBETA 340
Chart EP. IKJEBETO 341
Chart EQ. IKJEBEUI 342
Chart ER. IKJEBEUP 343
Chart ES. IKJEBEUP 344
Chart ET. IKJEBEUT 345
Chart EU. IKJEBEVE 346
Chart EV. IKJEBEWA 347
Chart EW. IKJEBEWB 348

Chart EX. IKJEBEWR 349
Chart EY. IKJEBEWR 350
Chart EZ. IKJEBEWR 351
Chart FA. IKJEBEWR 352

Diagrams

Method of Operation
Diagram 1. Initialization 475
Method of Operation
Diagram 2. Controller 477
Method of Operation
Diagram 3. EDIT Program STAE/STAI Processing 479
Method of Operation
Diagram 4. EDIT Attention Handling . .481
Method of Operation
Diagram 5. BOTTOM Subcommand 483
Method of Operation
Diagram 6. CHANGE Subcommand 485
Method of Operation
Diagram 7. DELETE Subcommand 487
Method of Operation
Diagram 8. DOWN Subcommand 489
Method of Operation
Diagram 9. FIND Subcommand 491
Method of Operation
Diagram 10. FORMAT Subcommand 493
Method of Operation
Diagram 11. HELP Subcommand 495
Method of Operation
Diagram 12. INPUT Subcommand 497
Method of Operation
Diagram 13. INSERT Subcommand 499
Method of Operation Diagram 14. Line Insert/Replace/Delete Subcommand 501
Method of Operation
Diagram 15. LIST Subcommand 503
Method of Operation
Diagram 16. MERGE Subcommand 505
Method of Operation
Diagram 17. PROFILE Subcommand 507
Method of Operation
Diagram 18. RENUM Subcommand 509
Method of Operation
Diagram 19. RUN Subcommand 511
Method of Operation
Diagram 20. SAVE Subcommand 513
Method of Operation
Diagram 21. SCAN Subcommand 515
Method of Operation
Diagram 22. TABSET Subcommand 517
Method of Operation
Diagram 23. TOP Subcommand 519
Method of Operation
Diagram 24. UP Subcommand 521
Method of Operation
Diagram 25. VERIFY Subcommand 523
Method of Operation
Diagram 26. EDIT Access Method Initialization and Final Processing (IKJEBEUI and IKJEBEEX) 525
Method of Operation
Diagram 27. EDIT Access Method Interface (IKJEBEUT) 527
Method of Operation
Diagram 28. EDIT Access Method Write Operation (IKJEBEWR) 529
Method of Operation
Diagram 29. EDIT Access Method Read Operation (IKJEBERR) 531
Method of Operation
Diagram 30. EDIT Access Method Delete Operation (IKJEBEDR) 533
Method of Operation
Diagram 31. EDIT Access Method Record Locate (IKJEBELO) 535
Method of Operation
Diagram 32. EDIT Access Method Record Delete (IKJEBEDL) 537
Method of Operation
Diagram 33. EDIT Access Method TTR Assignment (IKJEBEAD) 539
Method of Operation
Diagram 34. EDIT Access Method Directory Update (IKJEBEDU) 541
Method of Operation
Diagram 35. EDIT Access Method Write Block (IKJEBEWB) 543
Method of Operation
Diagram 36. EDIT Access Method Buffer Assignment (IKJEBEAS) 545
Method of Operation
Diagram 37. EDIT Access Method Directory Search (IKJEBEDS) 547
Method of Operation
Diagram 38. EDIT Access Method Read Block (IKJEBERB) 549
Method of Operation
Diagram 39. EDIT Access Method Wait (IKJEBEWA) 551

Tables

Table 1. EDIT Subcommand Functions (Part 1 of 2) 23
Table 2. Position of Current Line Pointer After Subcommand Operation . . . 25
Table 3. Syntax Checker Usage 31
Table 4. Membername Processing 35
Table 5. Summary of Initialization Operations (Part 1 of 7) 41
Table 6. Summary of BOTTOM Operations 63
Table 7. Summary of CHANGE Operations (Part 1 of 2) 67
Table 8. Summary of DELETE Operations 71
Table 9. Summary of DOWN Operations . 73
Table 10. Summary of END Operations . . 75
Table 11. Summary of FIND Operations (Part 1 of 2) 78
Table 12. Summary of FORMAT Operations 82
Table 13. Summary of HELP Operations . 84
Table 14. Summary of INPUT Operations (Part 1 of 5) 88
Table 15. Summary of INSERT Operations 95
Table 16. Summary of Line Insert/Replace/Delete Operations 97
Table 17. Summary of LIST Operations (Part 1 of 2) 100
Table 18. Summary of MERGE Operations (Part 1 of 2) 104
Table 19. Summary of PROFILE Operations (Part 1 of 2) 107
Table 20. Summary of RENUM Operations .111
Table 21. Summary of RUN Operations (Part 1 of 2) 114
Table 22. Summary of SAVE Operations .118
Table 23. Summary of SCAN Operations (Part 1 of 4) 121
Table 24. Summary of TABSET Operations 127
Table 25. Summary of TOP Operations . .129
Table 26. Summary of UP Operations . .131
Table 27. Summary of VERIFY Operations 133
Table 28. Initialization Routine Program Organization 145
Table 29. Mainline Routines (resident) 146
Table 30. Service Routines (non-resident) 147
Table 31. Message Text Loads 147
Table 32. Subcommand Processors Program Organization (Part 1 of 2) . . .148
Table 33. Field Values for Data Set Types (1 of 4) 372
Table 34. Field Values for Data Set Types Not Used With EDIT (1 of 4) . . .374
Table 35. Explanation of CADSCODE Field Values 375
Table 36. Explanation of CADSATTR and CADSATR2 Field Values 375
Table 37. Explanation of CARECFMD Field Values 375
Table 38. IKJEBEIN Error and Exceptional Conditions (Part 1 of 5) . .408
Table 39. IKJEBEMA Error and Exceptional Conditions 413
Table 40. IKJEBEAE Error and Exceptional Conditions 414
Table 41. IKJEBEAT Error and Exceptional Conditions 414
Table 42. IKJEBECI Error and Exceptional Conditions (Part 1 of 3) . .414
Table 43. IKJEBECO Error and Exceptional Conditions 416
Table 44. IKJEBEDA Error and Exceptional Conditions 417
Table 45. IKJEBEFC Error and Exceptional Conditions 418
Table 46. IKJEBEMR Error and Exceptional Conditions 418
Table 47. IKJEBERN Error and Exceptional Conditions 419
Table 48. IKJEBESE Error and Exceptional Conditions 419
Table 49. IKJEBEBO Error and Exceptional Conditions 419
Table 50. IKJEBECH, IKJEBECG, IKJEBECN Error and Exceptional Conditions (Part 1 of 2) 420
Table 51. IKJEBEDE Error and Exceptional Conditions (Part 1 of 2) . .422
Table 52. IKJEBEDO Error and Exceptional Conditions 423
Table 53. IKJEBEEN Error and Exceptional Conditions 424
Table 54. IKJEBEFI Error and Exceptional Conditions 425
Table 55. IKJEBEFO Error and Exceptional Conditions (Part 1 of 2) . .426
Table 56. IKJEBEIP and IKJEBEIM Error and Exceptional Conditions (Part 1 of 2) 427
Table 57. IKJEBEIS Error and Exceptional Conditions 429
Table 58. IKJEBELI Error and Exceptional Conditions 430
Table 59. IKJEBELT Error and Exceptional Conditions (Part 1 of 2) . .431
Table 60. IKJEBEME Error and Exceptional Conditions (Part 1 of 2) . .432
Table 61. IKJEBERE Error and Exceptional Conditions 434
Table 62. IKJEBERU Error and Exceptional Conditions (Part 1 of 2) . .435
Table 63. IKJEBESA Error and Exceptional Conditions (Part 1 of 5) . .437
Table 64. IKJEBESC, IKJEBESN Error and Exceptional Conditions (Part 1 of 3) 442
Table 65. IKJEBETA Error and Exceptional Conditions 445
Table 66. IKJEBETO Error and Exceptional Conditions 445
Table 67. IKJEBEUP Error and Exceptional Conditions 446
Table 68. IKJEBEVE Error and Exceptional Conditions 446

Table 69. IKJEBEUI Error and Exceptional Conditions 447
Table 70. IKJEBEEX Error and Exceptional Conditions 447
Table 71. IKJEBEUT Error and Exceptional Conditions 448
Table 72. IKJEBEWR Error and Exceptional Conditions 448
Table 73. IKJEBERR Error and Exceptional Conditions 448
Table 74. IKJEBEDR Error and Exceptional Conditions 449
Table 75. IKJEBELO Error and Exceptional Conditions 449
Table 76. IKJEBEDL Error and Exceptional Conditions 450
Table 77. IKJEBEAD Error and Exceptional Conditions 450
Table 78. IKJEBEDU Error and Exceptional Conditions 450
Table 79. IKJEBEWB Error and Exceptional Conditions 451
Table 80. IKJEBEDS Error and Exceptional Conditions 451
Table 81. IKJEBERB Error and Exceptional Conditions 452
Table 82. IKJEBEWA Error and Exceptional Conditions 452

Figures

Figure 1. Relationship of TSO Commands to TSO Processing 14
Figure 2. Main Storage after TSO has Been Started 15
Figure 3. Section of Main Storage after a TSO User has been Logged On . . 16
Figure 4. Section of Main Storage After a TSO User has Entered an EDIT subcommand 20
Figure 5. Symbols Used in Method of Operation Diagrams 21
Figure 6. QSAM and EDIT Access Method Record Formats 22
Figure 7. Formats of Records Passed to Syntax Checkers 29
Figure 8. Interface Between EDIT Program and Syntax Checkers 30
Figure 9. Expansion of Syntax Checker Interface 32
Figure 10. IKJEBECI Output Parameter List 55
Figure 11. IKJEBELE Input Parameter List 57
Figure 12. IKJEBEMS Input Parameters . 58
Figure 13. Format of Insertion Lists Passed to IKJEBEMS 59
Figure 14. Message Module Format and Message Selection 59
Figure 15. IKJEBEMS Output Parameter List 60
Figure 16. IKJEBESE Input Parameter List 61
Figure 17. EDIT Access Method Data Blocks 136
Figure 18. EDIT Access Method Buffers 137
Figure 19. EDIT Access Method Directory Blocks 139
Figure 20. Block Splitting Technique .141
Figure 21. EDIT Initialization Program Organization 150
Figure 22. EDIT Main Line Program Organization 151
Figure 23. EDIT Access Method Program Organization 152
Figure 24. IKJEBEAA Program Organization 153

SUMMARY OF AMENDMENTS
(RELEASE 21)
GY28-6773-1

The changes in this revision include a clarification of minor technical inaccuracies and editorial changes.

SUMMARY OF AMENDMENTS
(RELEASE 20.1)
GY28-6773-0)

TERMINATION PROCEDURES
Clearing of input queue and deleting of input stack for termination of EDIT in a command procedure.

CTLX KEYWORD FOR PROFILE SUBCOMMAND
Keyword added.

FORT DATA SET TYPE KEYWORD
Keyword deleted.

EDIT MESSAGES
Messages added and changed.

Section 1: Introduction

The EDIT Command Processor program is a part of the Time Sharing Option (TSO) of the MVT configuration of the control program. The EDIT Command Processor program (hereinafter called the EDIT program) performs the functions of the EDIT command and subcommands. Formats of the EDIT command and subcommands and acceptable operands are discussed in Section 7.

The TSO Environment

To better understand the environment in which the EDIT Command Processor operates, we should review the operations required before the EDIT program can be used. There are three primary operations that must be performed:

1. The console operator must issue a START TS command to prepare the entire system for the operation of TSO.

2. The terminal user must issue a LOGON command to notify the system that he is going to use TSO, and to define some of the system resources that he will need.

3. The terminal user must issue the EDIT command to initiate EDIT processing.

Figure 1 is a general diagram showing the routines that perform these operations.

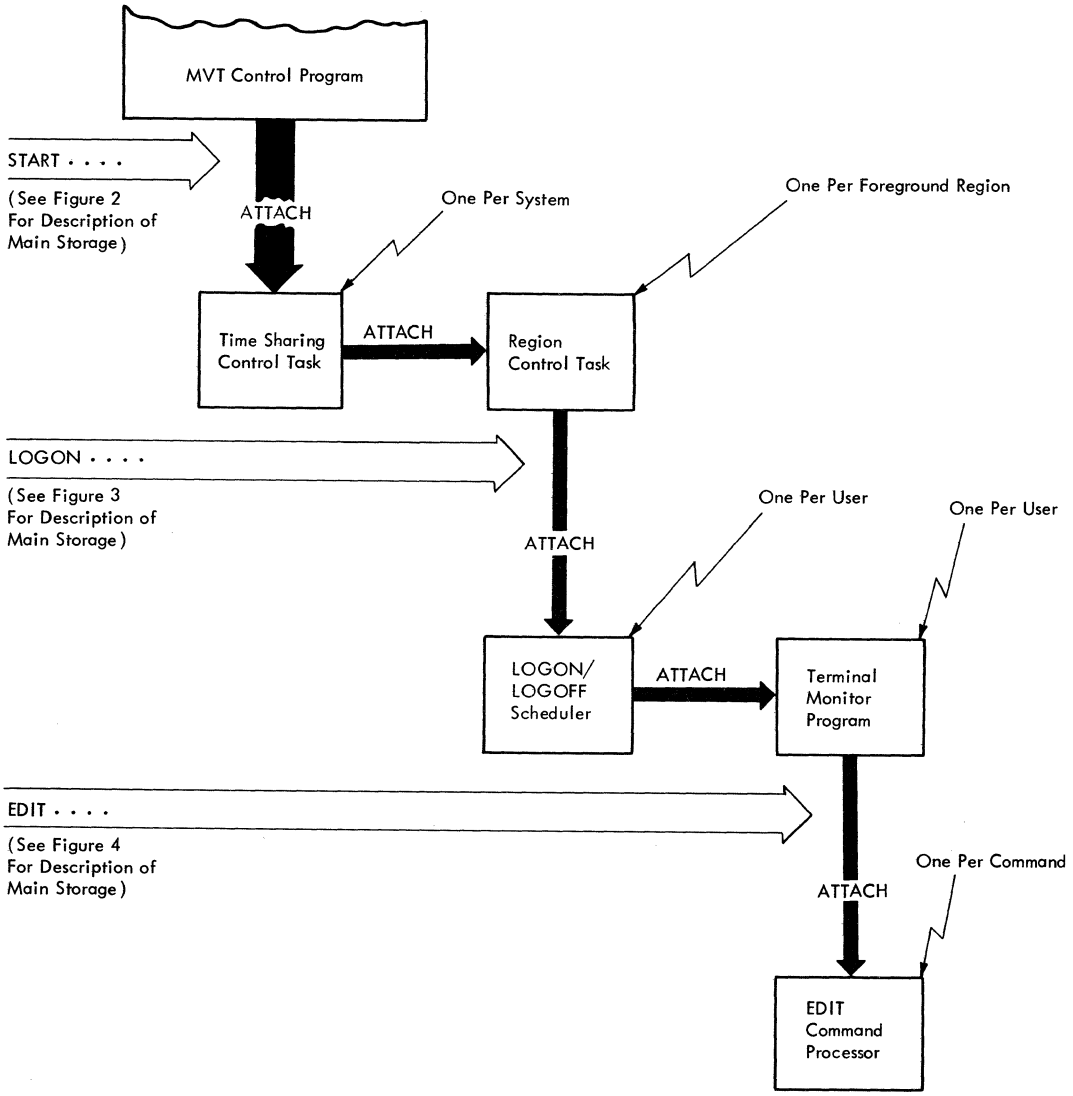


Figure 1. Relationship of TSO Commands to TSO Processing

STARTING TSO

The console operator enables the computing system to perform TSO processing by issuing the START TS command. The MVT control program, in processing this command attaches the Time Sharing Control (TSC) Task. The TSC initialization routine obtains a TSC region and in it, builds the control blocks and buffers that TSO will need. The TSC task attaches a Region Control Task (RCT) for each foreground region that was specified in a cataloged procedure named in the START TS command, and the regions for each RCT are assigned.

The routines of each RCT control the TSO jobs that operate from its region. This control consists primarily of the swapping of jobs in and out of the region when several jobs are competing for space in that region. The RCT routines operate from the TSC region, not from the region that they control. After the RCTs are attached, the system is ready to perform TSO processing. Main storage, in general, is as shown in Figure 2.

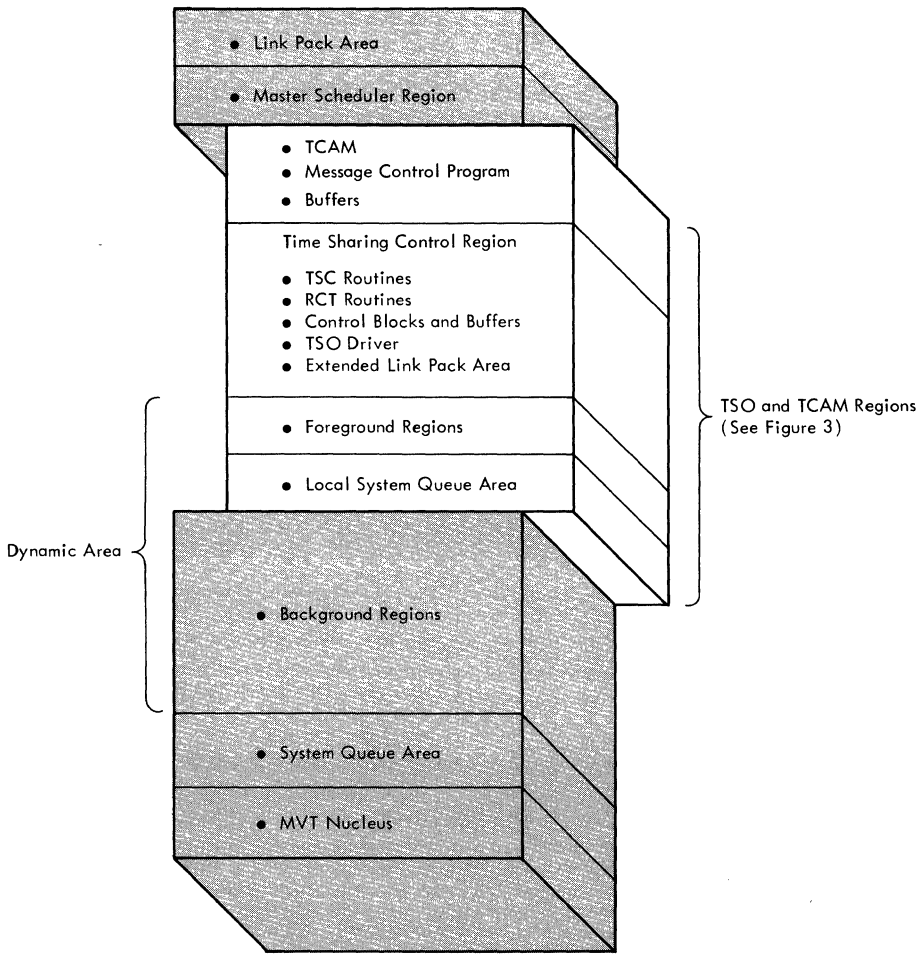


Figure 2. Main Storage after TSO has Been Started

LOGGING ON

After TSO has been "started", a terminal user obtains access to its facilities by logging his terminal onto the system via the LOGON command. This is the start of his terminal session in which he may use the TSO command language to perform his data processing from that terminal. (The session ends when a LOGOFF or another LOGON command is issued from that terminal.)

In processing the LOGON command, the TSC routines assign this session to one of the RCTs. The LOGON/LOGOFF scheduler is invoked by and loaded into the region of that RCT. All processing for this session will occur in this region. The LOGON routines validate the user's password, identify his User Attribute Data Set, and read in the user's profile. The LOGON routine causes the Terminal Monitor Program (TMP) to be attached by constructing JCL statements from information in the LOGON commands and previously-defined catalogue procedures. The JCL defines the TMP as a job step and the MVT initiator attaches it. The TMP is brought into the foreground region assigned to this session.

The TMP processes information specified in the JCL statements, issues a READY message to the terminal and looks for input by invoking the PUTGET service routine. Since at this point, no input (other than LOGON) has yet been entered during this session, the PUTGET routine will find nothing and issue a WAIT macro instruction. Figure 3 shows a typical configuration of the TSO portion of main storage after a user has been logged on.

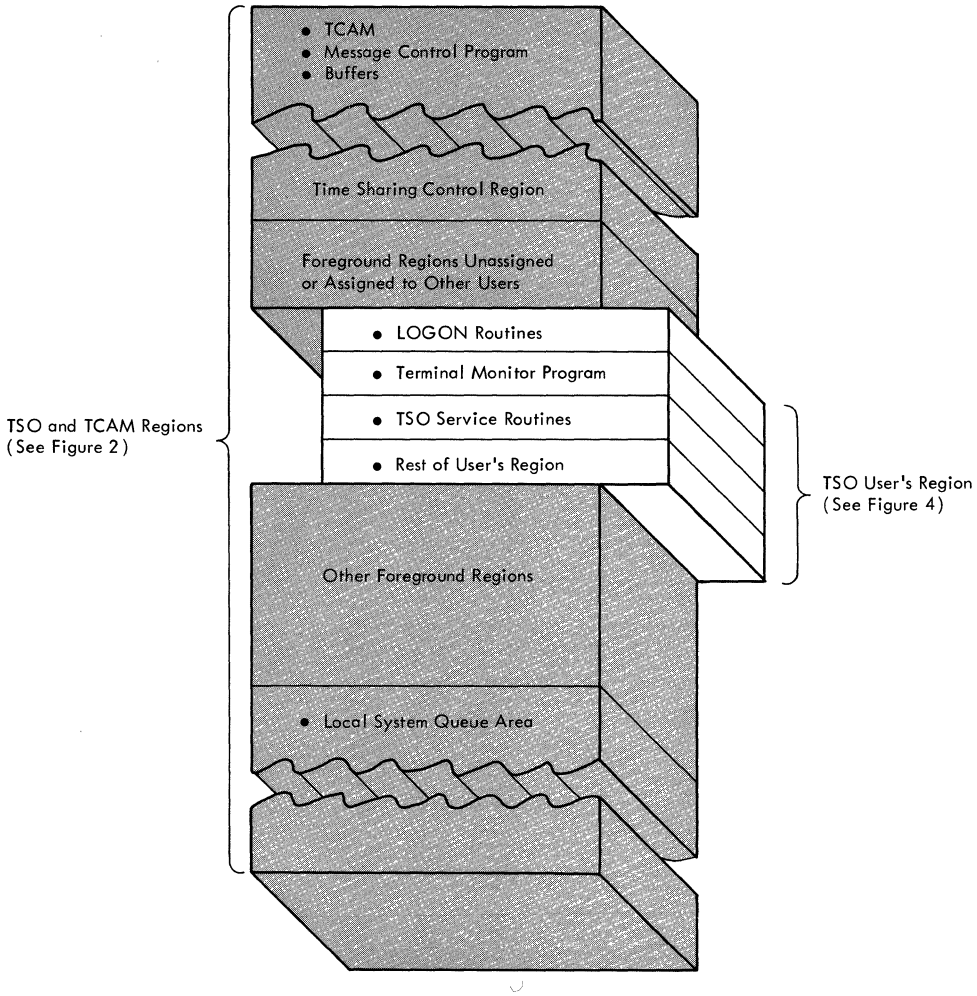


Figure 3. Section of Main Storage after a TSO User has been Logged On

PROCESSING TSO COMMANDS

When a user enters something from his terminal, it is stored in a buffer that is related to his session's TMP. The GETLINE service routine moves this line of input to a TMP buffer and returns control to the TMP.

Terminal Monitor Program

The TMP controls processing during a session; it:

- Requests commands from the terminal when a session is started or when a previously entered command has been processed.

- Has the command checked for validity.
- Invokes the command processor for that command.
- Receives control after a command processor has completed.
- Controls processing when a command processor is terminated because of an error.
- Processes attention interruptions.
- Initiates LOGOFF processing caused when an operator issues a STOP TS command during a session.

When the PUTGET routine returns a line of input, the TMP invokes the Command Scan service routine to see if this input is a syntactically valid TSO command. If it is not a command, the TMP causes a message to be issued to the terminal and waits for more input. If the input is a valid command, the TMP attaches the appropriate command processors.

When a command processor completes or is terminated by an error, CPU control is returned to the TMP. A more detailed description of the operation of the TMP can be found in the publication, IBM System/360 Operating System: Time Sharing Option Terminal Monitor Program and Service Routines Program Logic Manual.

TSO SERVICE ROUTINES

There are seven service routines used during TSO processing. They are normally invoked by the TMP, command processors, and each other. They can be invoked by any processing program. The service routines are:

- GETLINE - obtains a line of input from an area defined as its source of input. Normally this area contains input from the terminal.
- PUTLINE - sends a line of output to the terminal.
- PUTGET - sends a line of output to the terminal, and waits for a line of input as response.
- STACK - places pointers to lines of input into areas (stacks) from which the GETLINE and PUTGET routine obtain data.
- Command Scan - checks the syntax of an input buffer to see if it contains a valid TSO command name.
- PARSE - checks syntax of parameters of TSO commands.
- Dynamic Allocation Interface Routine (DAIR) - provides information to the MVT dynamic allocation routines which allocate, free and concatenate data sets that relate to a session.

The operation of these service routines is also found in the Terminal Monitor Program and Service Routines Program Logic Manual.

The EDIT Command Processor

The EDIT program resides on SYS1.CMDLIB until it is invoked by the user. When the user enters the EDIT command and operands (see Section 7 for external specifications of the EDIT command) a copy of the EDIT program is loaded into the user's region. The Edit data set may be a new data set or an old existing Edit data set; in either case, the name of the data set is that specified on the EDIT command.

The EDIT command enables a TSO user to create data sets and to modify them by adding, replacing, and deleting records within the data sets. A data set can consist of only printable characters in EBCDIC representation. It can contain text or programming language source statements. The user performs work on his data set thru either the Input_mode or the Edit_mode.

In Input mode the user enters successive lines of data. One line of input becomes one record in the data set. Services available in the Input mode include translation of tabulation characters to blanks, interpretation of character and line delete characters, and translation of lower-case characters to upper case. Programming language syntax checkers can be requested to process source statements as they are entered.

In Edit mode the user enters subcommands to point to particular records of the data set, to modify or renumber records, to add and delete records, to control editing of input, or to compile and execute a program. While in the Edit mode, the EDIT program keeps track of the user's position in the data set by means of the current_line_pointer. Subcommands are provided to enable the user to move the current line pointer within his data set. Once the current line pointer is positioned at a particular record, the user can then request a particular function be performed by issuing the appropriate subcommand. Following is a list of the EDIT subcommands and their associated functions:

BOTTOM
: moves the line pointer to the last line of the data set.

CHANGE
: modifies record text.

DELETE
: removes records from the data set.

DOWN
: moves the line pointer toward the end of the data set.

END
: terminates the EDIT command.

FIND
: locates a character string.

FORMAT
: prints out a data set or a portion of a data set in a particular format. Requires IBM COPY, FORMAT, LIST, MERGE Program Product.

HELP
: explains use of EDIT subcommands.

INPUT
: accepts new lines of data from the terminal.

INSERT
inserts records into the data set.

Insert/Replace/Delete
inserts, replaces, or deletes a line of data.

LIST
prints out specific lines of data or the entire data set.

MERGE
merges data sets or parts of data sets into the Edit utility data set. Requires IBM COPY, FORMAT, LIST, MERGE Program Product.

PROFILE
specifies 'delete' characters.

RENUM
numbers or renumbers lines of data.

RUN
compiles, loads, and executes the data set.

SAVE
retains data sets.

SCAN
controls syntax checking.

TABSET
sets the tab positions for editing.

TOP
moves the line pointer to line zero, if line zero exists, otherwise the pointer moves in front of the first line of the data set.

UP
moves the line pointer toward the beginning of the data set.

VERIFY
displays the line referred to by the current line pointer after modification by a subcommand or after movement of the current line pointer.

Main storage after a TSO user has entered an EDIT subcommand is depicted in Figure 4.

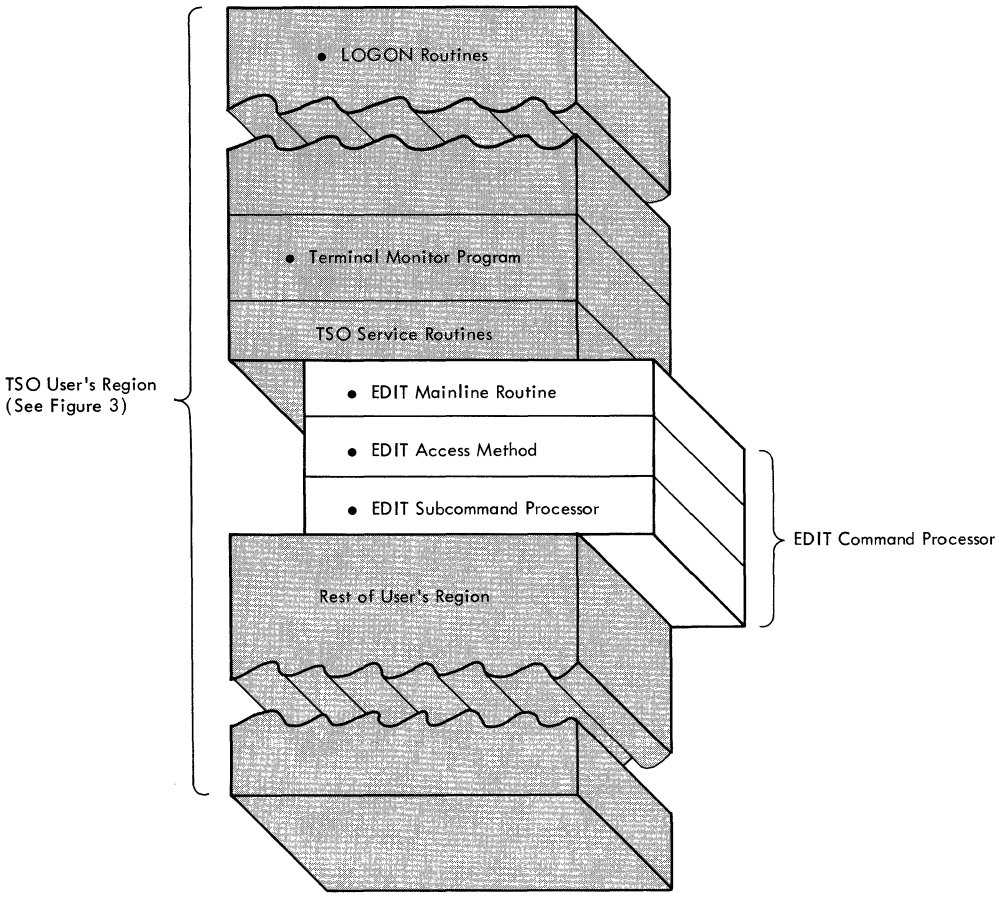


Figure 4. Section of Main Storage After a TSO User has Entered an EDIT subcommand

Section 2: Method of Operation

This section describes the method of operation of the EDIT program; it is divided into six major areas, each of which describes a functional group of routines. These major areas are:

- Initialization Routine
- Controller Routine
- Error and Attention Exit Routines
- EDIT Service Routines
- EDIT Subcommand Processors
- EDIT Access Method Routines

Each area contains functional descriptions, and, where applicable, summary of operation tables, and method of operation diagrams. The method of operation diagrams are on foldout pages at the back of the manual. Figure 5 explains the meaning of the symbols used in these diagrams.

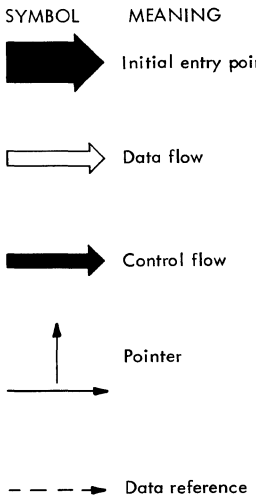


Figure 5. Symbols Used in Method of Operation Diagrams

INPUT PROCESSED BY THE EDIT PROGRAM

The input processed by the EDIT program consists of:

- The EDIT command operands - processed by the initialization routine.
- EDIT subcommands and operands - processed by the controller routine and the subcommand processors.
- Data - processed by the subcommand processors, the service routines, the EDIT Access Method, and Language Processors.

EDIT subcommands are entered during the Edit mode of operation. During Edit mode, the controller routine treats any input as a possible subcommand. Data may be entered during the Input mode of operation. The INPUT subcommand processor handles incoming data. The operating mode of EDIT is switched from Edit to Input when:

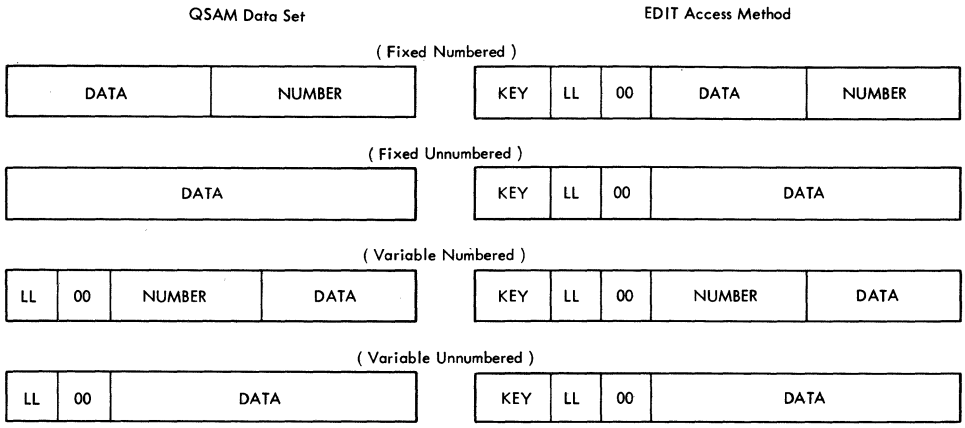
- A null line is entered.
- The INPUT subcommand is entered.
- The INSERT subcommand is entered with no operands.

The operating mode of EDIT is switched from Input to Edit when:

- A null line is entered.
- An attention interrupt is entered.
- A syntactically incorrect line is entered while syntax checking is in effect.
- An exceptional condition occurs during input processing.

The EDIT program assigns a unique key (a binary number) to each input line and treats the key and the data as a record. All accessing of records in the EDIT program is done by record key. If the user has specified that his data set be line-numbered, the EDIT program generates for each new record a line number which is the printable equivalent of the record's key, and if the user specifies unnumbered, the records will contain only a key and data. When a record is displayed, only the data and the line number, if present, will be printed.

The EDIT program converts the format of each record from old QSAM data sets into EDIT Access Method format. Figure 6 shows these two record formats.



Field	Length	Contents
KEY	4	- Key value of record in binary.
LL	2	- Length of record; this includes length field, the DATA field, and NUMBER field, if present. The length value is contained in the first two bytes of this field.
00	2	- Binary zeros.
NUMBER	1-8	- Sequence or line number of record; this is the first eight bytes if the record is variable - length and the last eight bytes if the record is fixed - length. Exceptions: COBOL -- First six bytes. ASM -- Variable length within last eight bytes.
DATA	variable	- User's information; length is dependent on the Edit data set type and user definition of record length.

Figure 6. QSAM and EDIT Access Method Record Formats

Most EDIT subcommands refer to or manipulate a record or a portion of a record as part of their operation. Table 1 lists the EDIT subcommands and indicates which portion of a record they refer to or manipulate during their operation.

Table 1. EDIT Subcommand Functions (Part 1 of 2)

Subcommand	Key	Record	Line number
BOTTOM	Reads last record in data set; sets current line pointer to key of last record.	NA	NA
CHANGE	Reads records by key until record to be changed is located; creates new keys for overflow lines (TEXT NONUM); sets current line pointer to key of last record changed.	Modifies data with in a record; creates new record to contain overflow lines.	NA
DELETE	Uses key to find record or records to be deleted; sets current line pointer to key of record preceding deleted records, or to zero if first record is deleted.	Deletes record.	NA
DOWN	Reads records sequentially by key (toward end of data set) until the specified number has been read; sets current line pointer to key of last record read.	NA	NA
FIND	Reads records until record is found which contains specified character string; sets current line pointer to key of record containing character string.	Searches record for a particular character string.	NA
FORMAT	Reads records by key to locate records selected for formatting.	Writes records into Format data set.	NA
HELP	NA	NA	NA
INPUT	Reads records by key to determine range of keys available for creation of new records; uses keys to create, replace and delete records; sets current line pointer to last record written into data set.	Writes new records into data set; deletes old records.	Creates new line numbers.
INSERT	Uses key to locate line of the data set after which a record is to be written; reads key of record to determine end of available space in data set for new records; sets current line pointer to last record written into data set.	Writes new record into data set.	Creates new line number.

Table 1. EDIT Subcommand Functions (Part 2 of 2)

Subcommand	Key	Record	Line number
Insert/ Replace/ Delete	Uses key to locate line into which a record is to be written, or from which a record is to be deleted; sets current line pointer to key of record written or to key of record preceding record deleted, or to zero if the first record is deleted.	Writes new record into data set; deletes old record.	Creates new line number.
LIST	Uses key to find the records or records to be written to the terminal; sets current line pointer to last record written.	Reads records from the data set; writes them to the terminal.	Writes out line numbers to the terminal.
MERGE	Uses keys to locate records selected for merging; sets current line pointer to last record in data set.	Writes records into Merge data set.	Changes line numbers.
PROFILE	NA	NA	NA
RENUM	Uses key to locate records which are to have new line numbers; changes record keys to conform to new line numbers.	Writes entire data set with renumbered keys to second data set.	Changes or creates line numbers as specified.
RUN	Uses keys to locate all records of the data set.	Writes records into the Run data set.	NA
SAVE	Uses keys to locate all records of the data set.	Writes records into the Save data set.	NA
SCAN	Uses keys to locate specified records of the data set; sets current line pointer to last record scanned.	Reads records into storage.	NA
TABSET	NA	NA	NA
TOP	Reads first record in data set; sets current line pointer to zero.	NA	NA
UP	Reads keys of records (toward beginning of data set) until the specified number has been read; sets current line pointer to key of last record found.		NA
VERIFY	NA	NA	NA

THE CURRENT LINE POINTER

The current line pointer is a device which enables the user and the EDIT program to refer to a particular record in the data set. The current line pointer (a field in the EDIT Communication Area) contains a value which is equal to the value of a particular key. The record associated with this key is the current line, or current record. Many subcommands make use of the current line pointer during their operation. Table 2 lists the EDIT subcommands and indicates which subcommands manipulate the current line pointer.

Table 2. Position of Current Line Pointer After Subcommand Operation

Subcommand	Current Line Pointer Position
BOTTOM	Last line of data set or zero if data set is empty.
CHANGE	Last line changed.
DELETE	Line preceding deleted line, if any, or else, zero.
DOWN	Last line referred to.
END	No change.
FIND	Line containing character sequence, if any, or else, no change.
FORMAT	No change.
HELP	No change.
INPUT	Last line entered.
INSERT	Last line entered.
Line Insert/Replace/Delete	Line inserted or replaced, or line preceding line deleted, if any, or else, zero.
LIST	Last line listed.
MERGE	Last line of data set.
PROFILE	No change.
RENUM	Same relative record.
RUN	No change.
SAVE	No change.
SCAN	Last line scanned, if any.
TABSET	No change.
TOP	Zero.
UP	Last line referred to.
VERIFY	No change.

OUTPUT CREATED BY EDIT PROGRAM

During the operation of EDIT, the user can create a new data set or modify an existing one. The utility data set is a work file used by the EDIT program (specifically, the EDIT Access Method) during EDIT operation. New records entered by the user are inserted into this data set; existing records are modified in this data set. The records in the utility data set are in EDIT Access Method format. The Edit data set, which is the data set the user specified when he entered the EDIT command, is QSAM-formatted. It contains records which have been modified or added during a previous EDIT session. The EDIT program transfers records from the Edit data set into the utility data set (changing their formats during the transferral) when the EDIT session begins. When the user enters the SAVE subcommand, the modified records in the utility data set are transferred into the Edit data set, or another user-specified data set, thus creating an updated Edit data set.

Other data sets created during EDIT operation are:

- The Save data set - created when the user specifies that the results of his EDIT session are to be retained in a data set different from the Edit data set.
- The Run data set - created for execution of the user's data set.
- The Merge data set - created for the MERGE Program Product.
- The Format data set - created for operation of the FORMAT Program Product.
- The Renum data set - created to contain the renumbered records after a RENUM subcommand operation; replaces the utility data set.
- The reverse Polish-notated data set - an in-storage data set created when the data set type is either BASIC or IPLI; created by the BASIC and IPLI language processor.

The EDIT program displays the following types of messages:

- Mode messages - indicate either Input or Edit mode of operation.
- Informational messages (first- and second-level) - inform the user of conditions within the EDIT program; second-level messages, which contain more detailed information, are displayed when the user enters a "?".
- Verify messages - display the current record if the value of the current line pointer has changed or the text of the record has been modified during operation of an EDIT subcommand processor.

Other messages displayed during EDIT program operation are:

- Prompting messages - displayed by the EDIT program or by the PARSE service routine to request more information from the user.
- Syntax checker messages - displayed by the EDIT program or by a syntax checker to inform the user of an error found in a source statement.
- Terminal error messages - displayed by the EDIT abnormal end exit routine or the TMP after an abnormal termination (ABEND) of the EDIT program.

TERMINATION OR SUSPENSION OF EDIT PROGRAM OPERATION

The operation of the EDIT program or of a portion of the program (a subcommand processor, for instance) is suspended or terminated in accordance with what the user has specified or with conditions within the EDIT program.

Termination of an EDIT subcommand processor does not necessarily require the termination of the entire EDIT program. If the termination is normal, the controller regains control and awaits the entry of another subcommand by the user. (See the discussions for each subcommand processor for further information about normal termination of subcommand processors.) Error termination of a subcommand processor does not require termination of the EDIT program if the source of the subcommand was the terminal. The source of input to the EDIT program, however, can be an in-storage command list, or command procedure. A command procedure is a sequence of TSO commands, subcommands, and data. Usually it is stored in a data set that has been created by means of the EDIT command with the CLIST data set descriptive qualifier. If the subcommand which is terminating due to error is from a command procedure, the entire operation of the EDIT program is terminated.

The CAINPROC field of the EDIT Communication Area (IKJEBECA) records the source of current input to the EDIT program. If the setting of CAINPROC is B'0', the terminal is the current source of input; if the setting is B'1', a command list is the current source of input. The EDIT controller updates this field after every return from the PUTGET Service Routine as follows:

- for a return code of 0 from PUTGET, CAINPROC is set to B'0',
- for a return code of 4 from PUTGET, CAINPROC is set to B'1'.

Suspension of EDIT Program Operation

When the user causes an attention interruption by striking the ATTN key, the operation of the EDIT program is suspended and the terminal input queue is cleared (with the TCLEARQ macro instruction). EDIT operation continues according to what the user enters next. See the topic, "Attention Exit Routine" for a more detailed description of attention interruption handling.

Normal Termination of EDIT Program Operation

The EDIT program is terminated normally when the user enters the END subcommand. The operation of the EDIT program is terminated as follows:

- The source of current input to the EDIT program is determined.
- If the current source is the terminal, the user is reminded to save his data set, the terminal input queue is cleared (with the TCLEARQ macro instruction), the input stack is flushed (via the STACK Service Routine), and control is returned to the Terminal Monitor Program with a return code of zero.
- If the current source of input is an in-storage command list, the terminal input queue is cleared, the input stack is flushed, and control is returned to the Terminal Monitor Program with a return code of zero; the user is unable to save his data set.

See the topic, "END Subcommand Processing" for a more detailed description of normal termination.

Error Termination of EDIT Program Operation

When the initialization routine of EDIT terminates due to error, or when a critical command system error or utility I/O error is encountered, the EDIT program is terminated as follows:

- The terminal input queue is cleared (with the TCLEARQ macro instruction).
- The input stack is flushed (via the STACK Service Routine).
- The END subcommand processor is invoked to free resources obtained by the EDIT program.
- Control is returned to the Terminal Monitor program with a return code of 12.

When an EDIT subcommand processor terminates due to error, and the current source of input is an in-storage command procedure, the EDIT program is terminated as described above.

If the current source of input is the terminal, only the input queue is cleared and EDIT processing continues. (Until the first EDIT subcommand is obtained via the PUTGET Service Routine, terminal input is assumed.)

SYNTAX CHECKING

Syntax checking of FORTRAN (E, G, G1, H), IPLI, BASIC, GOFORT, PL/I and PL/I(F) source statements is available while in EDIT program operation. The user can request that each line he enters from the terminal in Input mode be immediately scanned for syntax errors by specifying SCAN on the EDIT command or ON on the SCAN subcommand. Before the record is scanned it is put into the user's data set. If a syntax error is found in a record just entered by the user, an error message is displayed and EDIT switches from Input to Edit mode to enable the user to correct the mistake.

Lines entered by the user which end in a hyphen are not immediately scanned by the syntax checker; they are transferred to the utility data set as entered. If the user's data set type is GOFORT (FREE), the hyphen remains as data. The hyphen is removed for all other data set types.

By specifying the SCAN subcommand with no operands or with line number operands, the user requests that his existing data set be either entirely scanned or partially scanned for syntax errors. Figure 7 describes the formats of records passed to the syntax checkers.

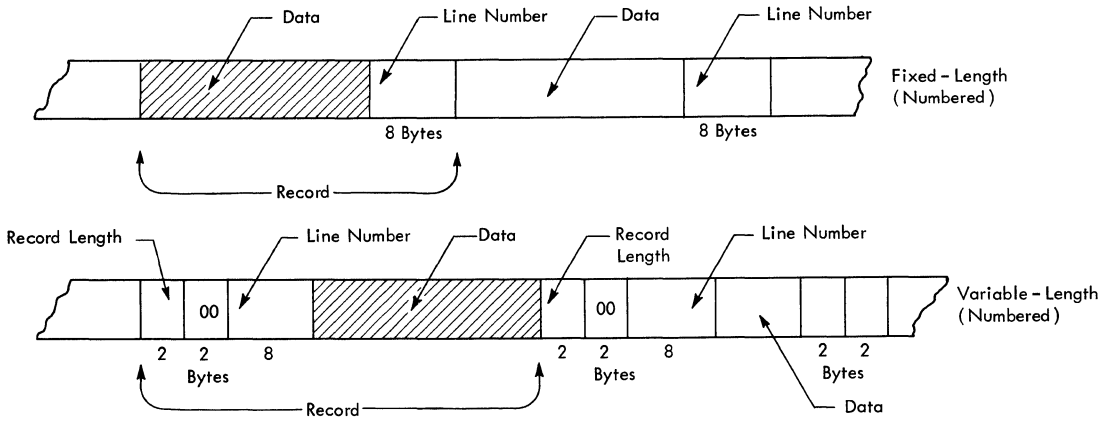


Figure 7. Formats of Records Passed to Syntax Checkers

The syntax checkers operate as separate components of TSO. They perform single-statement syntax checking as opposed to inter-statement checking; that is, a statement is considered valid if it satisfies its category definitions irrespective of preceding statements which may impose restrictions on the statement composition.

Use of IPLI and BASIC Syntax Checkers

The syntax checkers are not only used to scan source statements for proper syntax. If the user's data set type is IPLI or BASIC, the syntax checker or LANGPRCR (the syntax checkers available while in EDIT operation) is also used to delete, to add and to update records in a reverse Polish-notated data set. This data set is required for execution under IPLI or BASIC and is maintained along with the EDIT utility data set as the user performs editing operations. Since the reverse Polish-notated data set must be constantly updated, it is kept in storage; the syntax checker which maintains the data set is loaded into storage regardless of whether syntax checking is requested by the user. The syntax checker is also used when the data set is executed via the RUN subcommand.

To summarize, syntax checkers are invoked by EDIT to scan source statements, to delete, add, and update records from the reverse Polish-notated data set, and to enable program execution of BASIC or IPLI data sets. Statements can be scanned one at a time or several at a time.

The syntax checkers (LANGPRCR) available while in EDIT operation are:

- PLISCAN (for PL/I).
- PLIFSCAN (for PL/IF).
- IPDSNEXC (for FORT E, G, GI, H, and GOFORT).
- IKJNC211 (for BASIC and PL/IF).

The modules of EDIT which invoke syntax checkers are:

- IKJEBESC -- SCAN subcommand
- IKJEBESN -- SCAN subcommand
- IKJEBECG -- CHANGE subcommand
- IKJEBECN -- CHANGE subcommand
- IKJEBEIM -- INPUT subcommand
- IKJEBEIS -- INSERT subcommand
- IKJEBELI -- Insert/Replace/Delete implicit subcommand
- IKJEBERU -- RUN subcommand
- IKJEBEDE -- DELETE subcommand
- IKJEBEMR -- Translation service routine

A standard interface (shown in Figure 8) is provided to enable the EDIT modules to invoke any available syntax checker. Figure 9 shows an expansion of this interface. Detailed information on syntax checker control blocks is given in Section 5: Data Areas.

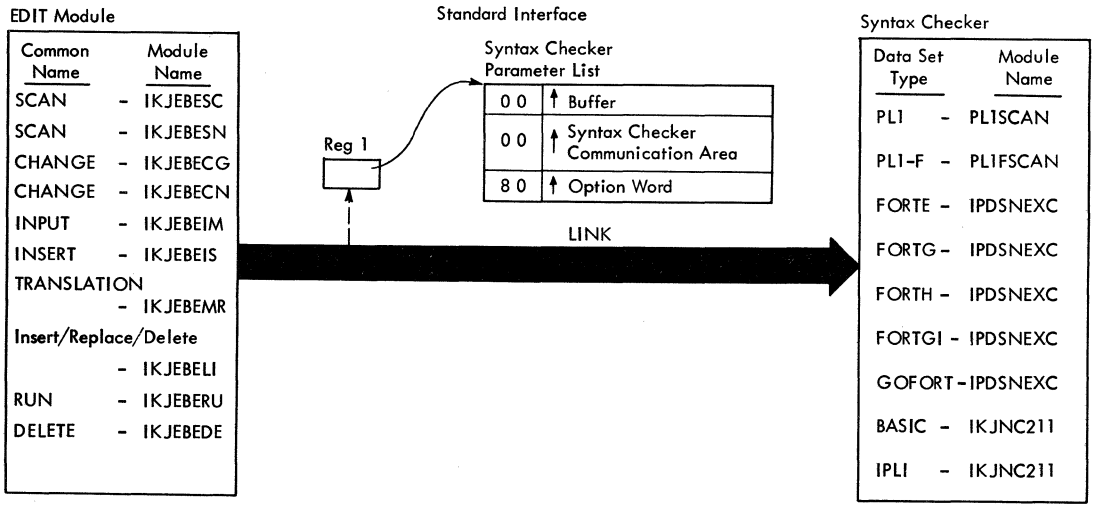


Figure 8. Interface Between EDIT Program and Syntax Checkers

User Exit Routine

If a user exit routine has been supplied at system generation time for a particular user data set type, the user may enter subfield information with this data set type keyword. This subfield information may contain any alphameric data defined as valid, not exceeding 256 characters in length and not containing any blanks, tabulation characters, or commas. This information is passed to the user exit routine to be interpreted and encoded into bytes 0 and 1 of the Option Word in the standard syntax checker interface parameter list.

The user exit routine is passed the address of a three word parameter list in register 1. The contents of this parameter list is as follows:

Word 1 Address of the subfield parameter descriptor element (PDE) returned to the initialization routine by IKJPARS.

Word 2 Address of bytes 0 and 1 of the syntax checker Option Word.

Word 3 Address of command processor parameter list passed to the EDIT Command Processor. This information is used to access the ECT and UPT, if the exit routine wants to use IKJPARS or any of the TMP Service Routines.

(See also Table 63).

Syntax checkers can be invoked by several EDIT modules to perform several different functions. Table 3 summarizes the usage of the syntax checkers.

Table 3. Syntax Checker Usage

EDIT Module	Function Performed
SCAN	Loads syntax checker for initialization.
SCAN	Deletes syntax checker for termination.
SCAN	Passes records from the utility data set for initial translation into reverse Polish notation.
SCAN	Passes records from the utility data set for syntax checking.
SCAN	Passes single record from utility data set for syntax checking.
CHANGE (ITF only)	Passes single record to be scanned and/or translated and added to reverse Polish-notated data set.
INPUT	Passes single record to be scanned and/or translated and added to reverse Polish-notated data set.
INSERT (ITF only)	Passes single record to be scanned and/or translated and added to reverse Polish-notated data set.
Insert/Replace/Delete (ITF only)	Passes single record to be added or deleted from reverse Polish-notated data set.
Translation (ITF only)	After successful renumber or merge operation, deletes old Polish-notated data set and passes records to be translated and inserted into reverse Polish-notated data set.
RUN (ITF only)	Passes pointer to RUN command.
DELETE (ITF only)	Passes control to delete records from reverse Polish-notated data set.

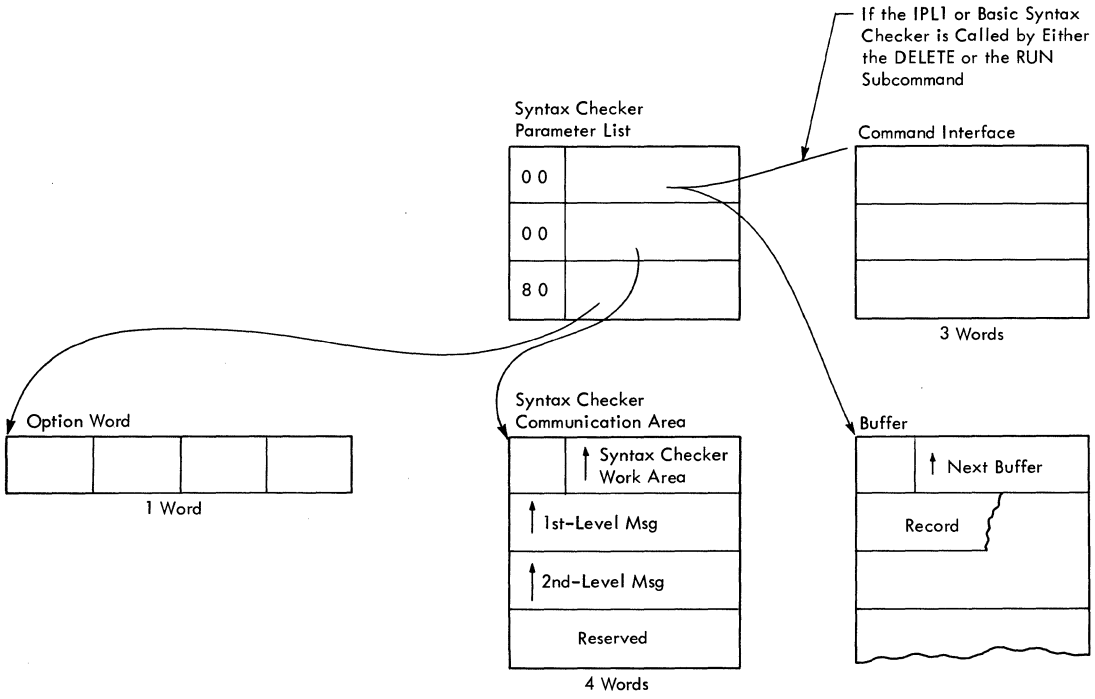


Figure 9. Expansion of Syntax Checker Interface

Initialization

The initialization routine (IKJEBEIN) receives control from the Terminal Monitor Program (IKJEFT02) whenever a valid EDIT command is issued. The initialization routine acquires and initializes work areas used by the EDIT program, processes the operands and keywords specified with the EDIT command, and passes control to the EDIT controller for subsequent subcommand processing.

Prior to validating the command operands and data set attributes specified by the user, the initialization routine:

- Obtains the EDIT Communication Area (IKJEBECA).
- Loads the resident message selection routine (IKJEBEMS).
- Invokes TSO service routine (IKJPARS) to check syntax of EDIT command.
- Performs data set name processing, and updates the EDIT Communication Area.
- Acquires data set type information for inclusion in the EDIT Communication Area. (This information is based on data set type, if entered; if data set type is not entered it is obtained from the descriptive qualifier.)
- Performs NEW/OLD keyword processing, if the data set is specified, or assumed, to be OLD.
- Invokes TSO service routine (IKJDAIR) to dynamically allocate a data set specified by user, if data set is OLD.

After operands have been checked, the initialization routine:

- Initializes the EDIT Access Method (hereafter called the Access Method) by invoking the Access Method initialization routine (IKJEBEUI), if the data set is either new or old and empty, or if a member is to be created for a partitioned data set.
- Loads the Access Method Interface routine (IKJEBEUT) and invokes the initial copy service routine (IKJEBECO) to copy the Edit data set or member into a utility data set, if the data set or member specified by the user contains records.
- Invokes the Controller routine (IKJEBEMA) via the XCTL macro instruction.

  NOTE: When the controller routine is invoked, the CAEDMEM flag (for Partitioned data sets) and the CAEDDISP flag (for sequential data sets) in the EDIT Communication Area (IKJEBECA) will indicate whether Input mode or Edit mode should be established. If the data set specified on the EDIT command is either new or old and empty or if the specified member does not exist, the initialization routine indicates that the Input mode should be entered by setting CAEDMEM or CAEDDISP to 0. If the data set is old and not empty or if the specified member exists, CAEDMEM or CAEDDISP is set to 1, indicating that Edit mode is to be entered.

Processing EDIT Operands

This topic describes the way in which the initialization routine handles particular operands and keywords. Table 5, which follows this topic, summarizes the operations of the initialization routine.

Data Set Name Operand

User Specification	Action of Initialization Routine
User entered fully-qualified data set name and data set type, (EDIT'engbw.parts.data'...ASM)	The fully-qualified data set name and the data set type are indicated in the EDIT Communication Area. (See Obtaining Processor Dependent Information.)
User entered fully-qualified data set name and no data set type (EDIT'engbw.parts.data'.....)	The user is prompted thru IKJPARS to supply the data set type. When this information is provided, the fully-qualified data set name and the data set type are indicated in the EDIT Communication Area.
User entered a partially-qualified data set name and the data set type, (EDIT parts...ASM)	The data set name is fully-qualified thru IKJDFLT and indicated with the data set type in the EDIT Communication Area.
User entered a partially-qualified data set name (rightmost qualifier is not a data set descriptive qualifier) and no data set type, (EDIT engbw.parts.nov....)	The user is prompted thru IKJPARS to supply the data set type. The data set name is fully-qualified thru IKJDFLT. The fully-qualified data set name and the data set type are indicated in the EDIT Communication Area.
User entered a partially-qualified data set name (rightmost qualifier is a data set descriptive qualifier) and no data set type, (EDIT parts.data.....)	The data set name is fully-qualified thru IKJDFLT. The fully-qualified data set name and the data set type are indicated in the EDIT Communication Area.

NEW/OLD Keyword

User Specification	Action of Initialization Routine
User entered NEW on the EDIT command, (EDIT'engbw.parts.data' new...)	EDIT assumes a new data set; IKJEBEIN does not allocate new data sets of the name specified by the user; the allocation of data set 'engbw.parts.data' is deferred until the user enters the SAVE subcommand.
User entered OLD or neither OLD nor NEW on the EDIT command; user specified no member name, (EDIT'engbw.parts.data'...)	Indicates old sequential data set in the Communication Area; invokes IKJDAIR to allocate a data set as OLD,KEEP,KEEP.
User entered OLD or neither OLD nor NEW on the EDIT command; user specified a member name, (EDIT'engbw.parts.data' (memname)old ...)	Processes Membername (See Table 4); indicates old data set in the communication area; invokes IKJDAIR to allocate a data set as SHR,KEEP,KEEP.

Table 4. Membername Processing

User Specification	Action of Initialization Routine
Membername not specified on EDIT Command and data set is partitioned	"TEMPNAME" is used as the member name.
Member specified and member found in data set, or member TEMPNAME defaulted and member TEMPNAME found in data set.	After successful validation of operands and attributes, invokes initial copy routine (IKJEBECO) to create utility work file and checks results as follows: - if an error has occurred (return code from IKJEBECO = 8), returns control to the TMP. - if the member is empty (return code = 4), indicates that the Input mode is to be entered. - if the member is found (return code = 0), indicates that the Edit mode is to be entered.
Membername specified as OLD and member not found in data set. (BLDL return code = 4).	Informs user that member is not OLD and returns control to the TMP.
Membername specified as NEW.	Indicates that the Input mode is to be entered; sets CAEDMEM to 0.
Error has occurred in BLDL process (return code = 8).	Informs user of error and returns control to the TMP.
NOTE: If the user did not specify OLD or NEW and the data set is not successfully allocated by IKJDAIR, the initialization routine prompts the user thru IKJPARS to enter OLD or NEW. If the user enters OLD, the EDIT program terminates after issuing the appropriate error message; if the user enters NEW, allocation is deferred until the user enters SAVE.	

PLI or PLIF Source Margins

User Specification	Action of Initialization Routine
Source margins specified; left margin greater than right margin, or right margin exceeds data length of records.	Inform user that specified margins are invalid and default to 2 and 72.
Source margins specified; left margin less than right margin; right margin less than or equal to data length (LRECL less sequence number length) of records in data set.	Initialize margins with values specified.
NOTE: If CHAR60 is specified for any of the above, a sixty-character character set is indicated by setting the CACHAR60 bit of IKJEBECA to 1. If CHAR48 is specified, CACHAR48 is set to 1. The default is CHAR60.	

SCAN/NOSCAN Keyword

User Specification	Action of Initialization Routine
User entered SCAN; syntax checker is available, and data set type allows scanning.	Sets CASCANON to 1.
User entered SCAN; syntax checker is not available in system.	Informs user that syntax checker is not available in system. Terminates EDIT with return code of 12.
User entered SCAN; syntax checking is not applicable to data set type.	Informs user that SCAN is invalid for data set type; terminates EDIT with return code of 12.

LINE Keyword and Keyword Value

The value entered for the LINE keyword is used as the logical record length (LRECL) for a new data set. This value may be entered by the user only when creating a new data set or editing an old and empty data set. The keyword value or the LRECL from the DS1LRECL field of the DSCB is checked against values obtained from the processor data table (IKJEBEPD) for the particular data set type. If the DS1RECFM field indicates a record format of V or V blocked, and variable format records are not compatible with the data set type, the user is informed and EDIT is terminated.

User Specification	Action of Initialization Routine
Data set is NEW, user did not enter the LINE keyword; the default for RECFM is V.	Initializes CALRECL to default value for LRECL and sets CARECFM to 0.
Data set is NEW, user did not enter the LINE keyword; the default for RECFM is F.	Initializes CALRECL to default value for LRECL and sets CARECFM to 1.
Data set is NEW, user entered LINE keyword; RECFM = F default is required; LINE value is equal to default value.	Initializes CALRECL to value supplied by user and sets CARECFM to 1.
Data set is NEW, user entered LINE keyword; RECFM = F default is not required; LINE value is equal to or less than the maximum value.	Initializes CALRECL to value supplied by user and sets CARECFM to 1.
Data set is NEW, user entered LINE keyword; LINE value is not equal to default value and default value is required.	Informs user that LINE value supplied by him is invalid; initializes CALRECL and CARECFM to proper values according to defaults for the record format. (See Obtaining Processor Dependent Information in this section.)
Data set is NEW, user entered LINE keyword; RECFM = F default is not required; LINE value is greater than maximum value.	Informs user that LINE value supplied by him is invalid; initializes CALRECL and CARECFM to proper values according to defaults for the record format.
Data set is OLD, user did not enter the LINE keyword; RECFM = F default is required; DS1LRECL value is equal to default LRECL.	Initializes CALRECL to value of DS1LRECL and CARECFM to value of DS1RECFM. (Terminates if DS1RECFM other than F or F blocked.)
Data set is OLD, user did not enter the LINE keyword; RECFM = F default is not required; DS1LRECL value is less than or equal to default LRECL.	Initializes CALRECL to value of DS1LRECL and CARECFM to value of DS1RECFM, if record format is F, F blocked, V or V blocked.
Data set is OLD, user entered LINE keyword; RECFM = F default is required and DS1LRECL value is equal to default LRECL, or RECFM = F default is not required and DS1LRECL value is less than or equal to default LRECL.	Initializes CALRECL to value of DS1LRECL and CARECFM according to the value of DS1RECFM, if RECFM is F or F blocked; informs user that LINE keyword is ignored for OLD data sets.
Data set is OLD; RECFM = F default is required; DS1LRECL value is not equal to default LRECL, or RECFM = F default is not required, and DS1LRECL value is greater than maximum LRECL.	Informs user that DS1LRECL is invalid for data set type and terminate EDIT.

BLOCK Keyword and Keyword Value

The value entered for the BLOCK keyword is used as the block size (BLKSIZE) for a new data set. This keyword is invalid for an old data set; if a value is entered for BLOCK and the data set is OLD and not empty, the user is informed that the value will be ignored. Checking of the BLOCK keyword assures compatibility of the values specified for LINE and BLOCK.

User Specification	Action of Initialization Routine
User did not enter the BLOCK keyword.	Uses CABLKS field of the processor data table as follows: for fixed-format data sets, sets CABLKS to the greatest possible integer multiple of LRECL that produces a BLKSIZE less than or equal to the Sysgen maximum, i.e. CABLKS=[CABLKS/CALRECL]X CALRECL (For record format V data default CABLKS is used.)
User entered BLOCK keyword; RECFM = F and BLOCK/LINE is an integer, or RECFM is not F and BLOCK is at least 4 greater than LINE, but less than or equal to Sysgen maximum.	Initializes CABLKS to value supplied by user.
User entered BLOCK keyword; RECFM = F and BLOCK/LINE is not an integer, or RECFM is not F and BLOCK is either less than LINE+4 or greater than Sysgen maximum.	Inform user that BLOCK value supplied by him is invalid; uses Sysgen value as default.

For old data sets, the device-dependent information needed to compute the number of records in the data set is obtained by issuing the DEVTYPE macro instruction. The number of records in the data set is computed according to the following formulae:

T (# of tracks in old data set)=TT(from DS1LSTAR)+1

LB (last block on track)=DS1BLKL-(device overhead for last block)-(device overhead for non-keyed records)

B (other than last block)=[((DS1BLKL*device tolerance factor)/512)]+(device overhead for other than last block)-(device overhead for non-keyed records)

N (# of blocks/track)=1+[(track size - LB)/B]

R (# of records)=T*(DS1BLKL/DS1LRECL)*N for RECFM=F, or
=T*(DS1BLKL/30)*N for RECFM=V.

When computed, the number of records (R) in the OLD data set is saved in CAUTILNO. The Utility Data Set Initialization Routine (IKJEBEUI) will use this value in determining the size of the utility data set to be allocated.

NUM/NONUM Keyword

User Specification	Action of Initialization Routine
NONUM Keyword is specified and data set must be numbered.	Inform user that data set must be numbered and ignore NONUM keyword.
NONUM keyword is specified and data set need not be numbered.	Indicate that data set is not to contain sequence numbers. (CANONUM is set to 1)
NUM keyword specified, data set type is ASM, and sequence numbers do not begin between positions 73 and 80, or NUM operand specified, data set type is ASM, and sequence number length is invalid, i.e., greater than 8, or NUM operand specified, data set type is ASM, and sequence number starting position and length are invalid.	Inform user that sequence number starting position and length are invalid and default to starting position of column 73 and number length of 8.
NUM keyword specified, data set type is ASM, and sequence number starting position and length are valid.	Initialize sequence number starting position and length with values obtained from NUM keyword.

CAPS/ASIS Keyword

User Specification	Action of Initialization Routine
User did not enter CAPS or ASIS keyword.	Sets CACAPS field in the EDIT Communication Area to default for data set type.
User entered CAPS.	Sets CACAPS field to 1.
User entered ASIS.	Checks acceptability of ASIS in data set type; if ASIS is acceptable, sets CACAPS field to 0, if not acceptable, (data set type requires CAPS) informs the user and set CACAPS to 1. (See Obtaining Processor-Dependent Information.)

GOFORT Data Set Type FREE/FIXED Keyword

This keyword specifies the format of input statements.

User Specification	Action of Initialization Routine
User entered FREE.	Ensures that RECFM=V for old data sets and sets CAFREE field in the EDIT Communication Area to 1; if RECFM=F, informs user of error and terminates.
User entered FIXED.	Determines if LRECL = 80 and the RECFM = F; if not and data set is new, informs the user and uses default values; if not, and data set is old, informs the user and terminates; if LRECL=80 and RECFM=F, sets CAFREE to 0.
User did not enter FREE or FIXED.	Defaults to FREE.

User-Generated Data Set Type Keyword

This keyword describes the action taken for user-generated data set type subfields.

User Specification	Action of Initialization Routine
User entered data set type with subfield parameters.	Verify that user exit routine is in the system; if not, inform user and terminate. If in system, invoke exit routine to encode the parameters into the CACHKOPT field.
User entered data set type without subfield parameters.	Same as above.
Note: If no user exit routines were specified at system generation time, this operation is not defined in the initialization routine.	

Table 5. Summary of Initialization Operations (Part 1 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	data set name (fully-qualified)	NEW	None	Terminal Monitor Program	Invokes command scan TSO service routine (IKJSCAN) to determine if EDIT is a valid TSO command; locates EDIT command processor program (IKJEBEIN) in SYS1.CMDLIB; passes control to the EDIT program via ATTACH.	--	--
				IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes the PARSE TSO Service routine (IKJPARS) to syntax check the EDIT command; invokes the Parse TSO service routine (IKJPARS) to prompt the user for the data set type; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in the Edit Communication Area; invokes the Access Method initialization routine (IKJEBEUI) to load the Access Method (IKJEBEAA) and to allocate a utility data set; indicates that the Input mode is to be in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	01	CC-CJ
				IKJEBEPS	Searches the processor data table (IKJEBEPD) for the data set type specified by the user; finds the processor-dependent information for the particular data set type (default blocksize, line number starting column, syntax checker name, etc.) and returns its address to the IKJEBEIN.	--	DH
				IKJEBEUI	Loads the Access Method (IKJEBEAA) into storage; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the utility data set.	26	EQ

(Part 1 of 7)

Table 5. Summary of Initialization Operations (Part 2 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT Continued				IKJEBEMA	Invokes the INPUT subcommand processor (IKJEBEIP) which will write subsequent lines of terminal input via the Access Method into the utility data set; (See INPUT Subcommand Processing).	02	CW-CY
EDIT	data set name (fully-qualified)	OLD (empty sequential data set)-	None	Terminal Monitor Program	Same function as above.	--	--
				IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes the Parse TSO service routine (IKJPARS) to prompt the user for the data set type; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in the Edit Communication Area; invokes IKJDAIR to allocate data set; invokes the Access Method initialization routine (IKJEBEUI) to load the Access Method and to allocate a utility data set; indicates that the Input mode is to be in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	01	CC-CJ
				IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBEMA	Same function as above.	02	CW-CY

(Part 2 of 7)

Table 5. Summary of Initialization Operations (Part 3 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	data set name (partially qualified)	CID	None	Terminal Monitor Program	Same function as above.	--	--
				IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes the IKJPARS routine to syntax check the EDIT command; invokes the Parse TSO service routine (IKJPARS) to prompt the user for the data set type; invokes the TSO default service routine (IKJDFLT) to fully-qualify the data set name; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in the Edit Communication Area; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the Edit data set identified in 'dataset name'; loads IKJEBEUT; invokes the initial copy routine (IKJEBECO) to copy the contents of the Edit data set into the utility data set; deletes IKJEBEUT indicates that the Edit mode is to be in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	01	CC-CJ
				IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBECO	Invokes the Access Method initialization routine (IKJEBEUI) to allocate a utility data set; reads into storage every record from the Edit data set; transforms the format of the records to that used by the Access Method; invokes the IKJEBEUT to write the reformatted records into the utility data set.	--	AU-AW
				IKJEBEMA	Indicates that the Edit mode is in effect by issuing the "Edit" mode message.	02	CW-CY

(Part 3 of 7)

Table 5. Summary of Initialization Operations (Part 4 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	data set name (fully-qualified)	OLD BASIC or IPLI NCSCAN	None None None None	Terminal Monitor Program	Same function as above.	--	--
				IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes the PARSE TSO Service routine (IKJPARS) to syntax check the EDIT command; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in the Edit Communication Area; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the Edit data set identified in 'dataset name'; checks if language processor is available in system; loads IKJEBEUT; invokes the initial copy routine (IKJEBECO) to copy the contents of the Edit data set into the utility data set; deletes IKJEBEUT; indicates that the Edit mode is to be in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	C1	CC-CJ
				IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBECO	Same function as above.	--	AU-AW
				IKJEBEMA	Loads language processor and indicates its address in CAPTCHIC field; invokes IKJEBESC to initialize language processing; indicates that Edit mode is in effect.	C2	CW-CY

(Part 4 of 7)

Table 5. Summary of Initialization Operations (Part 5 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	dataset name (fully-qualified)	OLD	None	Terminal Monitor Program	Same function as above.	--	--
		ASM or CCBOL	None	IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes the PARSE TSO Service routine (IKJPARS) to syntax check the EDIT command; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in Edit Communication Area; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the Edit data set identified in 'dataset name'; determines whether ASIS is acceptable for the data set type by checking the processor-dependent information inserted in the communication area by the processor data table search routine; if ASIS is acceptable, sets CACAPS to zero; loads IKJEBEUT; invokes the initial copy routine (IKJEBECO) to copy the contents of the Edit data set into the utility data set; deletes IKJEBEUT; indicates that the Edit mode is in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	01	CC-CJ
		FORTGI or TEXT or DATA or CLIST or CNTL	None				
		ASIS	None	IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBECO	Same function as above.	--	AU-AW
				IKJEBEMA	Same function as above.	02	CW-CY

(Part 5 of 7)

Table 5. Summary of Initialization Operations (Part 6 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	dataset name (fully-qualified)	OID PLI CHAR48	None integer1 integer2	Terminal Monitor Program	Same function as above.	--	--
				IKJEBEIN	Builds and initializes EDIT communication area (IKJEBECA); invokes PARSE TSO Service routine (IKJPARS) to syntax check the EDIT command; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in Edit Communication Area; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the data set identified in 'data set name'; determines if the values specified for left and right margins are valid, that is, the left margin value is smaller than the right margin value, and the right margin does not exceed the data length, less the sequence number length; sets CACHAR48 in the communication area to indicate that the 48-character character set is to be used; loads IKJEBEUT; invokes the initial copy routine (IKJEBECO) to copy the contents of the Edit data set into the utility data set; deletes IKJEBEUT; indicates that the Edit mode is in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	01	CC-CJ
				IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBECO	Same function as above.	--	AU-AW
				IKJEBEMA	Same function as above.	02	CW-CY

(Part 6 of 7)

Table 5. Summary of Initialization Operations (Part 7 of 7)

Command or Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
EDIT	dataset name (fully-qualified)	OLD GCFORT SCAN NUM	None (FIXED) None	Terminal Monitor Program	Same function as above.	--	--
				IKJEBEIN	Builds and initializes EDIT Communication Area (IKJEBECA); invokes PARSE TSO Service routine (IKJPARS) to syntax check the EDIT command; invokes the processor data table search routine (IKJEBEPS) to obtain data set type information for inclusion in the Edit Communication Area; invokes the TSO data set allocation service routine (IKJDAIR) to allocate the data set identified in "data set name"; determines if syntax checker is available; if syntax checker is available, indicates the fact by setting CASCANON to 1; initializes sequence number starting position and length with default values for NUM keyword; loads IKJEBEUT; invokes the initial copy routine (IKJEBECO) to copy the contents of the Edit data set into the utility data set; deletes IKJEBEUT; indicates that the Edit mode is in effect; passes control to the controller routine (IKJEBEMA) via XCTL.	C1	CC-CJ
				IKJEBEPS	Same function as above.	--	DH
				IKJEBEUI	Same function as above.	26	EQ
				IKJEBECO	Same function as above.	--	AU-AW
				IKJEBEMA	Loads the appropriate syntax checker; invokes the SCAN Subcommand processor to initialize it; indicates that the Edit mode is in effect by issuing the "EDIT" mode message.	C2	CW-CY

Prompting User for Data Set Type

When it is necessary to prompt the user for a data set type, the initialization routine invokes IKJPTGT to prompt the user for a data set type. The response is passed to IKJPARS to be validated. If the response is not a valid data set type, the prompt response buffer is freed and the above processing is repeated until either an attention is issued or a valid data set type is entered. If the response is valid, the initialization routine invokes the processor data table search routine (IKJEBEPS) to obtain the processor-dependent information. The information returned by IKJPARS is saved so that any subfield information associated with the data set type entered may be later processed by the initialization routine.

Obtaining Processor-Dependent Information

The initialization routine invokes the processor data table search routine (IKJEBEPS) via the LOAD and CALL macro instructions. IKJEBEPS searches the processor data table (IKJEBEPD) for an entry corresponding to the data set type, which was passed to IKJEBEPS by the initialization routine. When a matching entry is found, the search routine returns a pointer to the entry back to the initialization routine. The initialization routine, in turn, moves the table entry for the data set type into the CADSTYPE field of the EDIT Communication Area (IKJEBECA) and deletes IKJEBEPS. If the processor-dependent information is unavailable, the initialization routine informs the user prior to deleting IKJEBEPS.

Termination Processing

If an unrecoverable error has occurred during EDIT program initialization, the initialization routine issues the TCLEARQ macro to clear the input queues, invokes the TSO stack service routine (IKJSTCK) to delete all elements but the terminal from the input stack, deletes the service routines, frees the storage used by the EDIT Communication Area (IKJEBECA) and the IKJPARS PDL and returns to the Terminal Monitor Program.

Controller

The controller routine (IKJEBEMA) receives control from the initialization routine (IKJEBEIN) after the operands and keywords specified with the EDIT command have been processed. The controller directs the operation of the EDIT program in accordance with user specifications and requests and with conditions arising within the EDIT program by:

- Invoking the appropriate subcommand processor, when the user enters a subcommand.
- Handling the attention interruption (via the attention exit routine, IKJEBEAT), if the user depresses the ATTN key.
- Invoking the SCAN subcommand processor (IKJEBESC) to call the language processor for initial entry and to send lines, one by one, for translation (if the data set type is BASIC or IPLI), if the user specified the SCAN keyword on the EDIT command.
- Providing for the updating of the reverse Polish-notated data set, if the data set type is either BASIC or IPLI.

  Note: See the topic, "Syntax Checking" for a discussion of the reverse Polish-notated data set.
- Establishing either the Input or the Edit mode of operation, depending upon whether the user's data set is empty or not.
- Handling message verification for all EDIT subcommands except CHANGE.
- Handling error conditions which result in termination of the EDIT program (via the abnormal end exit routine, IKJEBEAE).
- Returning control to the Terminal Monitor Program upon the normal completion of END subcommand processing.

Controller Processing

This topic describes the way in which the controller routine directs the operation of the EDIT program. The operations of the controller routine which relate to normal subcommand processing are summarized in the tables found in the descriptions of the subcommand processors.

Abnormal End and Attention Exits

The controller routine establishes the abnormal end and attention exits by means of the STAE and STAX macro instructions. If the exits cannot be established, the EDIT program is terminated and the controller returns control to the TMP. See the topic, "Error and Attention Handling" for a description of the error and attention exit routines.

SCAN Processing

If the user has specified a BASIC or IPLI data set type, the controller loads the appropriate syntax checker into storage. The controller invokes the SCAN subcommand processor (IKJEBESC) to initialize the checker. If the data set type is other than BASIC or IPLI, the controller routine determines if the appropriate syntax checker is to be loaded by checking if CASCANON is set to 1. If it is, the controller loads the syntax checker and invokes the SCAN subcommand processor to initialize the syntax checker. See the topics "Syntax Checking" and "SCAN Subcommand Processing" for more information about syntax checking.

Establishing Mode of EDIT Operation

Depending upon the organization of the data set the controller routine checks the status of either the CAEDMEM or the CAEDDISP field in the EDIT Communication Area (IKJEBECA) to determine if Input or Edit mode should be entered.

The Edit mode is indicated if the data set organization is sequential and CAEDDISP=1, or if the data set organization is partitioned and CAEDMEM=1. The controller prompts the user to enter a subcommand by issuing the Edit mode message thru the PUTGET service rotuine (IKJPTGT).

The Input mode is indicated if the data set organization is sequential and CAEDDISP=0, or if the data set organization is partitioned and CAEDMEM=0. The controller invokes the INPUT subcommand processor (IKJEBEIP). Input received from the terminal is considered to be data; it will be written into the utility data set with no intervention on the part of the controller until the Edit mode is established.

Note: The CAEDMEM field refers to partitioned data set organization, and indicates whether the member exists or is to be created. The CAEDDISP field refers to the disposition of the data set, i.e., the field = 0 for new, or old and empty data sets, or the field = 1 for old (not-empty) data sets.

Obtaining an EDIT Subcommand

After invoking the PUTGET service routine to obtain a subcommand, the controller routine tests the return codes from PUTGET to determine what action to take.

Condition	Action of Controller Routine
Input entered from terminal; PUTGET return code = 0.	Sets CANINPROC bit to zero; validates subcommand. (See "Validating an EDIT subcommand".)
Input is from an in-storage list created by a command procedure; PUTGET return code = 4.	Sets CAINPROC bit to one: validates subcommand. (See "Validating an EDIT subcommand".)
Attention interruption occurred; PUTGET return code = 8.	Process as attention after subcommand termination. See the topic, "Error and Attention Handling".
Input could not be obtained from terminal; PUTGET return code = 12.	Delete second-level messages and invoke PUTGET again.
NOWAIT specified for TPUT; PUTGET return code = 16.	Issues message IKJ52313I and terminates EDIT program.
NOWAIT specified for TGET; PUTGET return code = 20.	Issues message IKJ52313I and terminates EDIT program.
Invalid parameters; PUTGET return code = 24.	Issues message IKJ52313I and terminates EDIT program.
No main storage available; PUTGET return code = 28.	Issues message IKJ52312I and terminates EDIT program.

If the input entered is a null line (that is, a carriage return), the controller invokes the INPUT subcommand processor (IKJEBEIP), thereby establishing the Input mode.

Validating an EDIT Subcommand

The validation of EDIT subcommands is performed in two steps: (1) scanning the subcommand buffer by the TSO service routine, IKJSCAN, and (2) verifying the subcommand by locating it in a subcommand table, by the controller routine. IKJSCAN determines that what was entered is a valid EDIT subcommand candidate or a null line.

Condition	Action Taken by IKJSCAN and Controller
Question mark entered by user.	IKJSCAN sets CSOAQM (in CSOAFLG) to 1; since all question marks should have been handled by PUTGET, a question mark at this point is an error - controller informs user that an invalid subcommand was entered.
Invalid subcommand, that is, what was entered by the user is not a valid EDIT subcommand candidate.	IKJSCAN sets CSOABAD (in CSOAFLG) to 1; controller informs user that an invalid subcommand was entered.
Empty line entered by user. (An "empty" line is a line containing separator characters only).	IKJSCAN sets CSOANOL (in CSOAFLG) to 1; controller processes as a normal return from a subcommand processor. (See the following topic.)
A valid subcommand with or without operands was entered by the user.	IKJSCAN sets CSOAVWP or CSOAVNP (in CSOAFLG) to 1; controller verifies the subcommand and sets CAOPERND (in IKJEBECA) to 1 if operands are present.

The controller verifies that the valid subcommand entered by the user is a valid EDIT subcommand by checking the IBM-supplied table of subcommands and the user-specified table of subcommands. The subcommand tables are defined as (IKJEBMA 8,9) CSECTs of the EDIT controller load module (IKJEBEMA).

Condition	Action Taken by Controller
Subcommand cannot be found in IBM or user subcommand table.	Informs user that subcommand is invalid by issuing message IKJ52366I.
An empty line was entered by user.	Prompts user for new subcommand.
Subcommand found in subcommand table.	Invokes appropriate subcommand processor. (See next topic.)

Invoking an EDIT Subcommand

The controller routine invokes an EDIT subcommand processor to perform the function requested by the user. The controller invokes the subcommand processor via the LINK macro instruction and checks the return code when the subcommand processor relinquishes control.

Condition	Action Taken by Controller
Successful completion of subcommand processing; subcommand return code = 0.	Obtains the next subcommand from the current source of input.
Return code of 4 returned by INSERT subcommand processor (IKJEBEIS).	Invokes the INPUT subcommand processor (IKJEBEIP) to establish the Input mode.
Subcommand termination errors (subcommand return code 8) or invalid subcommand.	Issues TCLEARQ macro for input if the current source of input is the terminal; obtains another subcommand. Starts error termination of EDIT program, if current source of input is an in-storage procedure.
Permanent I/O error in utility data set; subcommand return code = 12.	Terminates the EDIT program.
Syntax checker cannot be initialized; subcommand return code = 16.	If syntax checker recovery is not in progress, indicates that recovery should be attempted by setting CASCRC20 (in CACFLAG4) to 1 and invokes the SCAN subcommand processor (IKJEBESC); if syntax checker recovery is in progress and has failed, indicates that syntax checker clean-up is to be performed by setting CASCRC20 to 0 and invokes IKJEBESC.

Handling an Attention Interrupt

Whenever the return code from a subcommand processor indicates that a new subcommand is to be obtained, that is, the subcommand processor has finished its operation, the controller routine determines if an attention interrupt has occurred during subcommand processing. If an attention interrupt has occurred, the attention exit routine (IKJEBEAT) has updated the CAPTIBFR field of the EDIT Communication Area (IKJEBECA) to point to the attention buffer. The controller deletes any queued second-level messages and obtains the subcommand which is in the attention buffer. See the topic, "Error and Attention Handling" for a description of the attention exit routine.

Verify Message Handling

The verify message is a display of the line number and record pointed to by the current line pointer after the value of the current line pointer has been changed. The user, by means of the VERIFY subcommand, indicates that he wants a display of the record pointed to by the current line pointer if the value of the current line pointer is changed; the subcommand processor updates CALNTOVF (in CACFLAG1) to indicate that the value of the current line pointer has changed during operation of the subcommand and stores current line beginning at CATEMPBF+12. The controller routine determines if the user wants a verify message and whether a verify message is warranted. If a verify message is to be displayed, the controller invokes the message selection service routine (IKJEBEMS).

Termination Processing

The controller routine terminates the EDIT program when an unrecoverable error has occurred in the EDIT program by:

- Deleting the Input Stack (via the STACK service routine).
- Clearing the input queues (via the TCLEARQ macro).
- Invoking the END subcommand processor at entry point IKJEBEXT to free system resources used by EDIT.

When the END subcommand processor returns control, the controller frees the storage used by the EDIT Communication Area (IKJEBECA) and returns to the Terminal Monitor Program with return code 12. The communication area is not freed subsequent to an abnormal end (ABEND). This allows proper closing of any open data sets allocated to EDIT.

Error and Attention Handling

The abnormal end exit routine (IKJEBEAE) intercepts Abend conditions occurring at the EDIT program (task) level. The attention exit routine (IKJEBEAT) handles attention interrupts.

Abnormal End Exit Routine

The abnormal end exit routine:

- Stops automatic line prompting.
- Frees PARSE PDL and input buffer storage.
- Issues diagnostic message IKJ52422I.
- Requests scheduling of a retry routine.

The ABEND completion code is placed in the ECT (ECTRTCD) and the Abend-in-progress flag (the high-order but of EDTRCDF in ECT) is set to 1. IKJEBEAE issues the SPAUTOPT macro to stop automatic line prompting. If EDIT has obtained an input buffer and PARSE PDL storage through a GETMAIN macro instruction, the storage is returned to the system. The abnormal end exit routine then invokes the message selection service routine (IKJEBEMS) to display a two-level message containing the Abend completion code which has been converted into printable hexadecimal. After the message has been displayed, the abnormal end exit routine returns to the caller with the address of the STAE retry routine (CAPTRTRY) in register 0 and a return code of 4 in register 15. If the ECTATRM bit (in the ECT) is set to 1, no message is issued and the retry routine is bypassed, as indicated by a return code of zero to the STAE/ABEND interface routine. Method of Operation Diagram 3 (foldout) shows the relationship between the EDIT program and STAE/STAI processing.

Attention Exit Routine

The attention exit routine receives control after the user has caused an attention interrupt and has entered a line of input. Upon receiving control, the attention exit routine issues the STATUS STOP macro instruction to stop any dispatchable subtasks that the EDIT program has attached. It then examines the input line by invoking the IKJSCAN service routine.

Condition	Action Taken by the Attention Exit
User entered a null line.	Re-starts dispatchable subtasks and returns control to the TIOC routine which invoked the attention exit routine.
User entered a "?".	Invokes the PUTLINE service routine (IKJPUTL) to display any queued second-level messages and the Edit mode message; invokes the GETLINE service routine (IKJGETL) to obtain a new line from the terminal.
User entered neither a null line nor a "?".	Treats the input line as a subcommand; issues POST for the attention ECB (CAATTN); invokes the LANGPRCR module via LINK if BASIC or IPLI language processing is in effect; issues the STATUS macro to restart dispatchable subtasks; returns control to the system. (See the topic, "Handling an Attention Interrupt" in the discussion of the "Controller".)

See Method of Operation Diagram 4 (foldout) for a description of the operation of the attention exit routine.

EDIT Service Routines

The EDIT service routines perform certain operations required by various modules of the EDIT program. These service routines and the functions they perform are:

- IKJEBECI (Command Invoker) - This routine invokes the TSO command processors for the FORMAT, HELP, MERGE, PROFILE, and RUN subcommands of EDIT. It also invokes any TSO command processors placed in the input stack by the TSO RUN command.

- IKJEBECO (Initial Copy) - This routine writes the contents of the Edit data set or an intermediate utility data set into the utility data set.

- IKJEBEDA (Data Set Allocation/Free) - This routine generates a dsname and allocates a data set for a requesting routine by invoking the dynamic allocation service routine (IKJDAIR); when requested, IKJEBEDA frees the data set by invoking IKJDAIR.

- IKJEBEFC (Final Copy) - This routine writes the contents of the utility data set into the Edit data set, the Run data set, the Merge data set, the Format data set, or the Save data set.

- IKJEBELE (Line Edit) - This routine converts lower-case characters to upper-case, translates tabulation characters to the required number of blanks and formats data into records.

- IMJEBEMR (Translation) - This routine calls the BASIC or IPLI language processor to delete an old reverse Polish-notated data set and builds a new reverse Polish-notated data set.

- IKJEBEMS (Message Selection) - This routine selects EDIT program messages requested by EDIT modules and writes them out by invoking the PUTLINE service routine (IKJPUTL).

- IKJEBEPS (Processor Data Table Search) - This routine searches the processor data table (IKJEBEPD) to obtain data set type attributes.

- IKJEBERN (BASIC Renumbering) - Renumbers BASIC data sets and changes line numbers which appear within the records.

- IKJEBESE (String Search) - This routine scans records for a particular character string and returns a pointer to found text to the calling module.

Invoking TSO Commands (IKJEBECI)

This routine invokes the following TSO command equivalents of the EDIT subcommands; FORMAT (a Program Product), HELP, MERGE (a Program Product), PROFILE, and RUN. While operating under RUN, the command invoker also invokes the compiler associated with the data set type, and the loader to execute the user's problem program.

The command invoker receives as input a pointer to a parameter list containing a pointer to the EDIT Communication Area and a pointer to a buffer in which the caller has placed the model command. The command invoker builds a parameter list (shown in Figure 10) and passes control to the specified TSO command by issuing the ATTACH macro instruction with the STAI exit option.

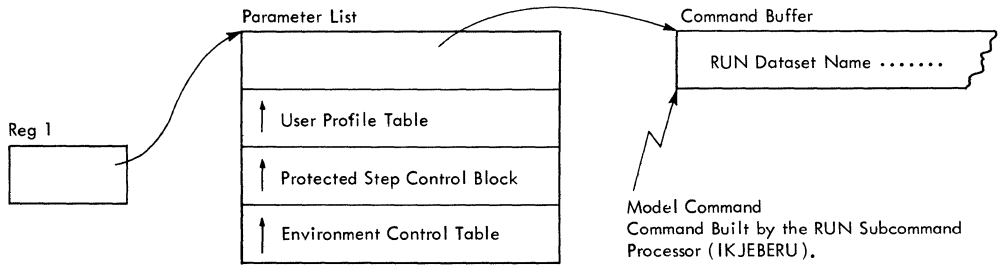


Figure 10. IKJEBECI Output Parameter List

If the subtask ends abnormally, control is passed to the command invoker STAI exit. The command invoker terminates the subtask by issuing the DETACH macro instruction, informs the user that the subcommand has ended abnormally, and returns control with return code of 8 in register 15. The command invoker invokes the TSO data set allocation/free service routine (IKJDAIR) to mark any data sets allocated by the subtask as not in use and issues the DETACH macro instruction to terminate the subtask. The command invoker then returns control with a zero return code, if there is no error.

If the subtask returning normally to the command invoker is the TSO RUN command, the command invoker reads the in-storage commands built by RUN. The command invoker, through a series of ATTACH macro instructions, invokes those programs specified by RUN to compile, load and execute the user's program. If RUN ends normally, the command invoker terminates the subtask by issuing the DETACH macro instruction and invokes IKJDAIR to mark any allocated data sets as not in use. If a subtask completes abnormally, prior to returning control to its caller the command invoker invokes the TSO STACK service routine (IKJSTCK) to delete the pointer to the in-storage command list.

Initial Copying (IKJEBECO)

This routine is used whenever a utility data set must be created from an old, existing data set or member of a partitioned data set which may be accessed by QSAM. The initial copy routine reads the QSAM-formatted data set and transforms the format of the records into the format used by the EDIT Access Method. It then writes the records into the utility data set using either the line number (if the data set is line numbered) or a generated number (if the data set is not line numbered) as the key.

The initial copy routine receives a pointer to the EDIT communication area, which contains the fully-qualified name and the ddname of the data set from which the records are to be copied. The initial copy routine initializes a DCB for the QSAM data set and opens the data set. If the data set is successfully opened, the first read flag (CORDFLAG in the EDIT communication area) is set to 1. The initial copy routine then invokes the Access Method initialization routine (IKJEBEUI) to allocate the utility data set and to initialize the EDIT Access Method. If the Access Method initialization completes normally, the initial copy routine selects the appropriate utility data set record format (fixed-length, numbered or unnumbered, or variable-length, numbered or unnumbered) to be used. The initial copy routine issues a GET macro instruction for each record in the QSAM-formatted data set and changes the record format to that used by the Access Method. After the record's format has been changed, the initial copy routine invokes the Access Method interface routine (IKJEBEUT) to write the record into the utility data set. When all the records from the old QSAM data set have been written into the utility data set, the initial copy routine closes the QSAM data set and returns control to the caller.

Allocating and Freeing Data Sets (IKJEBEDA)

The data set allocating/freeing routine receives control from the FORMAT subcommand processor (IKJEBEFO), the RUN subcommand processor (IKJEBERU), or the MERGE subcommand processor (IKJEBEME) with a pointer to the Communication Area which contains the CASAFLAG field and the name of the subcommand. The CASAFLAG field is set to X'00' if allocation of a new data set is desired, to X'12' for freeing of an allocated data set, to X'10' for marking a DSE not in use, to X'02' for allocation of an old data set. For allocation of a new data set, the allocation routine (IKJEBEDA) builds a DSNAME in the form:
'USERID.SUBCOMMANDNAME.Dyyddd.Ttttttttt.DESCRIPTIVEQUALIFIER'.
USERID is a 1- to 7-byte qualifier obtained from the Protected Step Control Block (PSCB). SUBCOMMANDNAME (the name of the subcommand which invoked IKJEBEDA) is obtained from the Communication Area. Dyyddd (a 6-byte qualifier) is the date obtained when IKJEBEDA issues the TIME macro instruction. Ttttttttt (an 8-byte qualifier) is the time obtained when IKJEBEDA issues the TIME macro instruction. DESCRIPTIVE QUALIFIER is obtained from the Processor Data Table.

After the DSNAME is built, IKJEBEDA inserts it into the DADSN8 field, which is pointed to by an entry in the DAIR parameter block, and invokes the TSO data set allocation service routine (IKJDAIR). IKJDAIR allocates the data set with a disposition of NEW, DELETE, DELETE, or NEW, CATLG, DELETE for the MERGE subcommand. (The first term of the disposition refers to the status of the data set. The second term refers to the disposition of the data set after its use. The third term refers to the disposition of the data set if the job step abnormally terminates.) If the allocation is successful, IKJEBEDA saves the DDNAME, DSNAME and length in the Communication Area, sets a return code of 0 in register 15, and returns control to the caller. If the allocation is unsuccessful, IKJEBEDA issues a message to the terminal via the message selection routine (IKJEBEMS) and returns control to the caller with a return code of 8 in register 15.

IKJEBEDA also frees an existing data set for the RUN, FORMAT, or MERGE subcommand processor by invoking IKJDAIR; IKJDAIR frees the data set with a disposition of DELETE. If IKJDAIR is successful, IKJEBEDA returns a code of 0 and control returns to the caller; if IKJDAIR is unsuccessful, IKJEBEDA issues a message and returns control to the caller with a return code of 8.

"SECTION 5: DATA AREAS" contains the format of the parameters passed to IKJDAIR.

Final Copying (IKJEBEFC)

This routine is used whenever the data being edited is to be used or processed outside of the EDIT program. The data being edited is contained on the utility data set in a record format which is unique to EDIT. In order for the data to be usable outside of EDIT, its record format is changed to QSAM format. The re-formatted records are then written into a QSAM- format data set which has been allocated by the caller of this routine.

The final copy routine determines the name of the data set into which records from the utility data set are to be written, initializes a DCB for the data set, and opens the data set for QSAM output. The final copy routine invokes the Access Method interface routine (IKJEBEUT) to read the records one at a time from the utility data set into storage. After each record is read into storage, its format is changed from that used in the utility data set to QSAM format. After it is re-formatted, the record is written into the QSAM data set by the final copy routine. When all the records of the utility data set have been written, the final copy routine closes the QSAM data set and returns control to the caller.

Line Editing (IKJEBELE)

The line edit routine receives a pointer to the two-word parameter list (shown in Figure 11) as input.

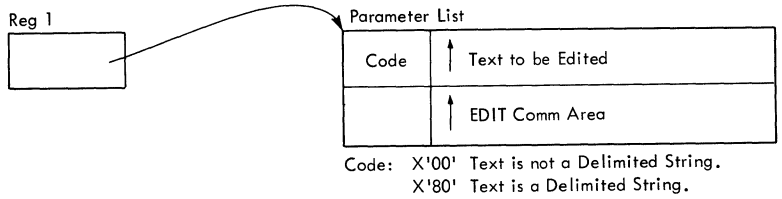


Figure 11. IKJEBELE Input Parameter List

If CAPS were specified on the EDIT command (CACAPS in the Communication area is set to one), the line edit routine converts the text to upper case characters. If the text is not a delimited string and if tabulation settings are in effect (CATABS in the Communication Area is set to X'FF') the line edit routine inserts the required number of blanks and formats the text line accordingly. If text overflow occurs, a pointer to the superfluous text is placed in the first word of the parameter list which is returned to the caller.

If the text is a delimited string or if tabulation settings are not in effect (CATABS is set to X'00'), the line edit routine changes each tabulation character in the text line to a single blank. If the text is not a delimited string, the data is formatted into a record for the EDIT Access Method.

When the text has been edited the line edit routine returns control to the caller with either a 0 or a 4 in register 15. Both return codes indicate a successful line editing operation. A return code of 4 also indicates that an overflow has occurred and that word 1 of the parameter list points to the overflow text.

Translating (IKJEBEMR)

After a successful renumber or merge operation, the reverse Polish-notated data set (an in-storage equivalent of the utility data set which is needed for BASIC or IPLI) must be updated to reflect the changes in the utility data set. IKJEBEMR accomplishes this updating (or translating) by invoking the syntax checker for BASIC and IPLI to delete the old reverse Polish-notated data set and to build a new one.

Upon receiving control from either the RENUM subcommand processor (IKJEBERE) or the MERGE subcommand processor (IKJEBEME), the translating routine invokes the BASIC and IPLI syntax checker to delete the old reverse Polish-notated data set. When the syntax checker returns control, the translating routine invokes the Access Method interface routine (IKJEBEUT) to read each record of the utility data set. The syntax checker is then called to build a new reverse Polish-notated data set using the records from the utility data set as input. After the syntax checker has built the new in-storage data set, it returns control to the translating routine. The translating routine returns control to its caller. See the topic: Syntax Checking for a discussion of the parameters passed by the translating routine to the BASIC and IPLI syntax checker.

Selecting EDIT Messages (IKJEBEMS)

This routine selects an EDIT message (a first-level message, second-level message or current line display) requested by an EDIT module and invokes the TSO PUTLINE service routine (IKJPUTL) to write the message to the terminal. The input to the message selection routine is shown in Figure 12.

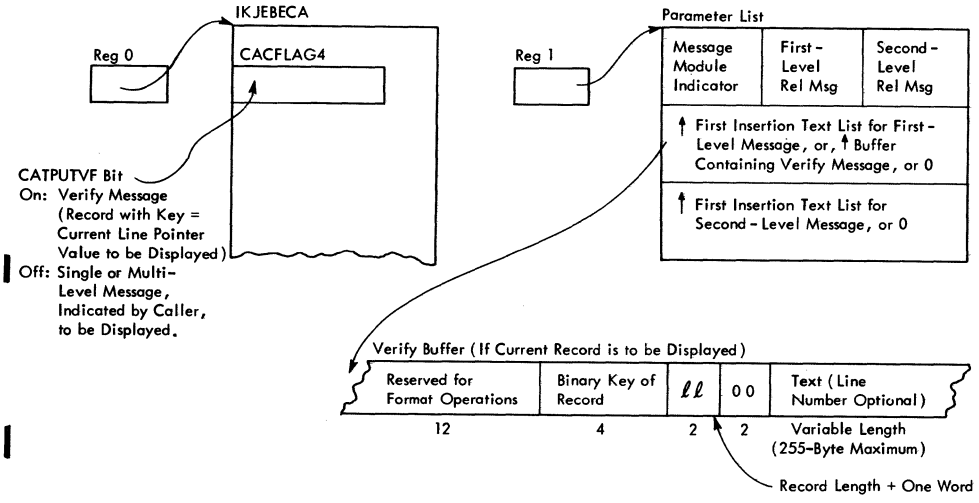


Figure 12. IKJEBEMS Input Parameters

EDIT messages are contained in the message modules IKJEBEM1 thru IKJEBEM7. The messages are arranged in message number sequence with second-level messages immediately following the associated first-level message. Text to be inserted into an EDIT message is generated by the module requesting message output. The text is passed to the message selection routine in an insertion list. (See Figure 13.)

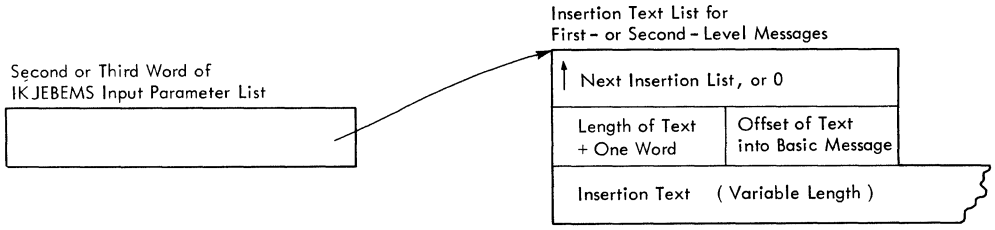


Figure 13. Format of Insertion Lists Passed to IKJEBEMS

The relationship of insertion text to the basic message is in the form: IKJ52303I DATA SET xxxxxx NOT ALLOCATED, REQUIRED VOLUME NOT MOUNTED where: xxxxxx is the insertion text to be supplied by the requesting module, and DATA SET ... NOT MOUNTED is the basic message.

First-and second-level messages are arranged in relative message number sequence in message modules. Figure 14 shows the format of a message module and the way in which a particular message is requested by the message selection input parameter list.

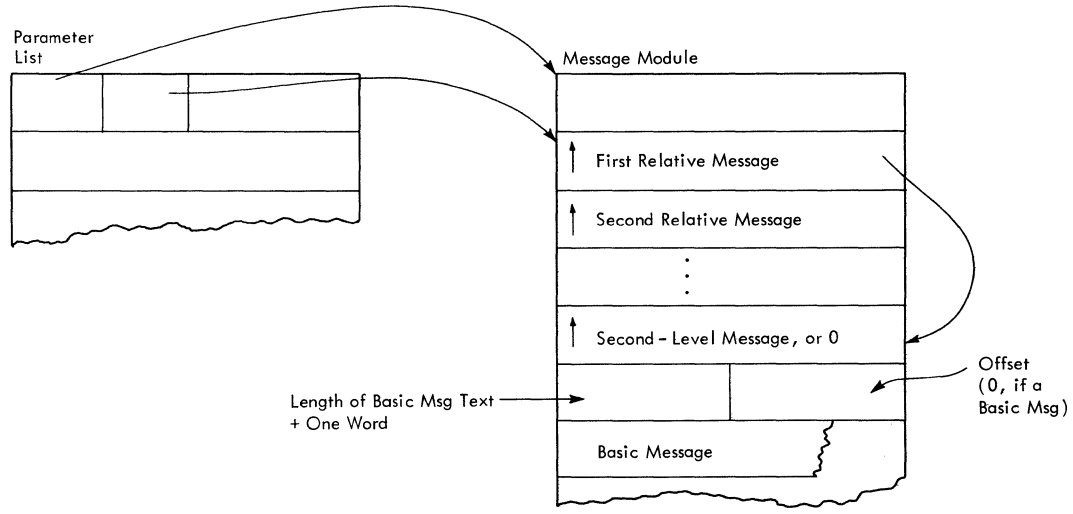


Figure 14. Message Module Format and Message Selection

When the message selection routine receives control, it determines if a verify message or contents of the record pointed to by the current line pointer is to be displayed. If a verify message is to be displayed, the CATPUTVF bit in CACFLAG4 in the EDIT Communication Area has a value of 1 and the second word of the parameter list points to a verify buffer containing the current record and line number, if the data set is line-numbered.

The message selection routine formats the line and invokes the PUTLINE service routine to write the verify message to the terminal. (Figure 14 is a description of the message selection routine output parameter list.)

If a verify message is not to be displayed (CATPUTVF has a value of 0), the input parameter list indicates which message is to be displayed and points to insertion text, if there is any. The message selection routine loads the message module containing the requested message into storage, finds the message, and formats the output parameter list. When the output parameter list has been formatted, the message selection routine invokes the PUTLINE service routine to write the requested message to the terminal. (See Figure 15 for a description of the message selection output parameter list.)

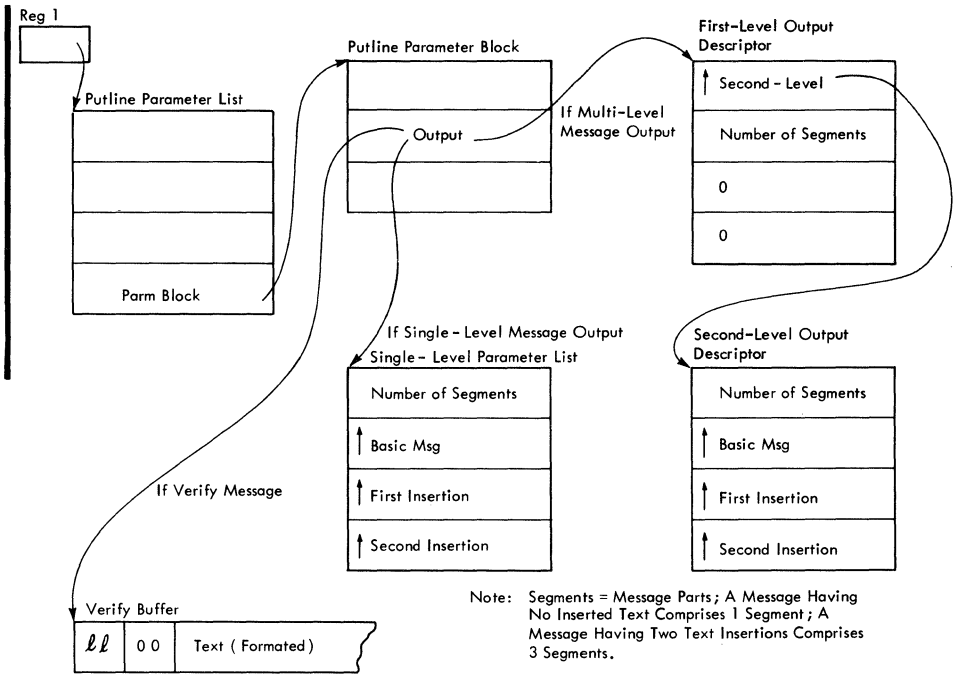


Figure 15. IKJEBEMS Output Parameter List

When the PUTLINE service routine completes its operation, the message selection routine returns control to the caller. SECTION 5: Data Areas contains the format of the PUTLINE Parameter List and the PUTLINE Parameter Block.

Processor Data Table Searching (IKJEBEPS)

The processor data table search routine receives the address of a data set type keyword or the complemented address of a data set name qualifier as input in register 1. It compares the input with the first or second 8-byte field, respectively, in each table entry of the processor data table (IKJEBEPD). The search routine returns control and the address of the table entry, if it is found, to the caller. SECTION 5: DATA AREAS contains the format and the associated data set type values of the processor data table.

BASIC Renumbering (IKJEBERN)

This routine renumbers statements within a BASIC data set. When this routine receives control, Register 1 points to a two-word parameter list, that has the following format:

Word 1 - pointer to the EDIT Communication Area (IKJEBECA).
Word 2 - pointer to a three-word parameter list, that has the following format:

Word 1 - old line number at which to begin renumbering.
Word 2 - new line number.
Word 3 - increment.

Starting at the given old line number, IKJEBERN constructs a table of old and corresponding new line numbers for the entire data set. Then, starting at the line number specified, IKJEBERN renumbers each record and changes all references to line numbers within the records to the corresponding new numbers.

String Searching (IKJEBESE)

The string search routine receives a pointer to the five-word parameter list (shown in Figure 16) as input.

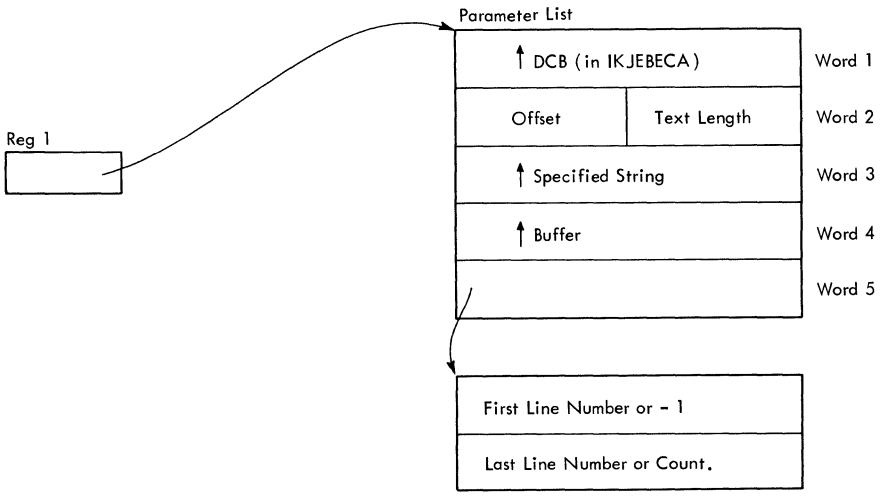


Figure 16. IKJEBESE Input Parameter List

If the data set type is not TEXT, the string search routine invokes the Access Method interface routine (IKJEBEUT) to read each record of the utility data set beginning with the record after the one pointed to by the current line pointer; the first record to be searched has been placed in the buffer by the calling routine. The string search routine then begins its search at the offset into the first record read (a number of characters from the beginning of the record) specified. If the string is not found, the next record is searched, and each succeeding record, until the end of the data set on the end of line range is reached. If the end of the data set is reached and the string has not been found, the string search routine informs the caller via a return code. If the string is found, the record which contains it is placed left-aligned in the buffer (pointed to by word 4 of the parameter list), and a return code of 0 is set.

If the data set type is TEXT, the string search routine, in addition to searching a single record at a time, also searches for the specified character string across two records. If the string is found across two records, a return code of 4 is set.

The string search routine places the second record (containing the final portion of the string into the buffer at the midpoint. When the specified string has been found, or when the data set has been exhausted of records, the string search routine returns control to its caller.

BOTTOM Subcommand Processing

The BOTTOM subcommand positions the current line pointer to the end of the utility data set.

The BOTTOM subcommand processor:

- Reads the last record in the utility data set.
- Sets the current line pointer to the value of the key of the last record.
- Returns control to the controller routine.

BOTTOM Processing

This topic describes the way in which the BOTTOM subcommand processor operates. Table 6, which follows this topic, summarizes the operation of the BOTTOM subcommand processor. Upon receipt of the BOTTOM subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the BOTTOM subcommand processor (IKJEBEBO). IKJEBEBO calls the interface routine for the EDIT Access Method (IKJEBEUT) to read the last record in the data set. (Reading of the last record of the data set is indicated by X'05' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the last record of the data set. If the data set is empty, indicated by a return code of 4 from IKJEBEUT, message number IKJ52501I is selected by the message selection routine (IKJEBEMS) and put out by IKJPUTI. IKJEBEBO sets the current line pointer to zero. If the data set is not empty (IKJEBEUT return code = 0), IKJEBEBO sets the current line pointer to the key of the final record, turns the "line to be verified switch" on, and returns control to IKJEBEMA.

Table 6. Summary of BOTTOM Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
BOTTOM	None	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEBO).	02	CW-CY
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the final record; passes key of final record to IKJEBEBO.	27	ET
				IKJEBEBO	Invokes the Access Method to read the last record of the data set; sets the current line pointer to the final record in the data set.	05	AE

CHANGE Subcommand Processing

The CHANGE subcommand modifies a sequence of characters (a character-string) in a record or a range of records in the utility data set. IKJEBECH is the first load module of three load modules which process the CHANGE subcommand. IKJEBECH establishes program addressability and then calls IKJEBEUT to read the first record in the EDIT data set that has been specified by the user. If no character string has been specified as an operand of the CHANGE subcommand, IKJEBECH will issue appropriate messages until the user enters valid operands. If operands are specified (CAOPERND IN IKJEBECA is set to one) IKJEBECH calls IKJPARS to validate the operands. Upon return from IKJPARS, IKJEBECH analyzes the operands to determine where the change operation is to start, at a specified line number or at the current line pointer. The line count, i.e., number of records to be changed, is interpreted. Depending upon what operands the user specifies, the CHANGE subcommand processor replaces the first reference to the old string of data with a new string in

- One record, if a line number or * were specified as an operand.
- A range of records, if two line numbers or * line count were specified as operands.

Additionally, the CHANGE subcommand processor provides for subsequent data insertion at a specified offset in a record if the user has specified a character count or one string of data as an operand.

Interpreting Operand Combinations

The CHANGE subcommand processor interprets the combinations of operands the user can enter as follows: If * is specified with no numerical operands, IKJEBECH defaults the line count value to 1. If * is specified with one numerical operand, IKJEBECH assumes the operand is a character count and converts it to binary. If * is specified with more than one numerical operand, IKJEBECH converts both to binary and assumes the first numerical operand is a line count and the second is a character count. If only a single numerical operand is specified, IKJEBECH assumes it is a character count and defaults the line count value to 1. If a beginning line number has been specified, IKJEBECH converts it to binary. If two numerical operands are specified, IKJEBECH assumes the first operand is a line number and the second operand is a character count. If two numerical operands and a third operand are specified, IKJEBECH assumes the first operand to be a first line number and the second operand to be a second line number and executes a character count validity checking routine. IKJEBECH assumes the third operand is a character count and converts it to binary.

After the CHANGE subcommand processor (IKJEBECH) has finished analyzing the operands, it calls IKJEBEUT to read the first specified record. If the specified record is not found but a range of records has been specified, IKJEBECH calls IKJEBEUT again to read the next record within the range. Processing continues as outlined in the section, Processing CHANGE Operands.

After the string has been changed, the CHANGE subcommand processor:

- Sets the current line pointer to the value of the key of the last record changed.
- Returns control to the controller routine.

Processing CHANGE Operands

This topic describes the ways in which the CHANGE subcommand processor operates when various operands are specified. Table 7, which follows this topic, summarizes the operations of the CHANGE subcommand processor.

String Data Specified

If string data is specified IKJEBECH invokes another load module of the CHANGE subcommand processor. IKJEBECG first calls the line editing routine IKJEBELE to translate tabulation characters to blanks and the specified string to upper case, if necessary. If only 'string 1' has been specified, another of the three load modules (IKJEBECN) that process the CHANGE subcommand is invoked. If both string 1 and string 2 have been specified and "string 1" is null, the user has requested an insertion operation and the offset at which the change is to occur is set to position 1. The IKJEBECG manipulates the string data as follows:

- If the offset at which the change is to occur is not position one, the data line up to, but not including, the specified offset is moved into a temporary buffer to be saved.
- If 'string 2' is not null, it is the replacement or insertion data and is moved into the temporary buffer next.
- If there is data remaining after the change has been entered, it is moved into the temporary buffer.

IKJEBECG calculates the new length of the data and then moves the new line back into the main buffer. IKJEBECG then calls IKJEBELE to line edit the data into a record. When IKJEBELE returns the record to IKJEBECG, IKJEBEUT is called to write the record into the data set. If line overflow occurs and the data set type is not NONUM TEXT (CANONUM bit is set to 0 or CADSCODE field is not set to CATEXT), IKJEBECG truncates the line and notifies the user. If line overflow occurs and the data set type is NOMUM TEXT (CANONUM bit is set to 1 and CADSCODE field is set to CATEXT). IKJEBECG calls IKJEBELE to line edit the overflow into a record. When the record is returned, IKJEBECG invokes IKJEBEUT to write the record into the EDIT data set with an overflow key. If the data set type is BASIC or IPLI (CADSCODE field is set either to CABASIC or to CAIPLI) and the processor is in the system (CAPCHK field is not set to zero), IKJEBECG updates the reverse Polish data set. The ITF entry code for input or replacement (CASXNCD2 field is set to B'10000') for updating the data set. If the verification is specified (CAVRFYSW bit is set to 1), the changed line and its overflow lines, if any, are passed to IKJEBEMS (the message output routine). IKJEBECG continues processing until all specified records have been changed. It then sets the current line pointer to the last line changed or its last overflow line if one exists. IKJEBECG returns control to IKJEBEMA.

Character Count Specified

If a character count operation has been specified, IKJEBECN calls IKJPUTL to print out each record until the character count is reached or up to the string specified. After each record is printed, IKJEBECN calls IKJGETL to obtain the user's changes for the line from the terminal. IKJEBECN forms a new revised line by joining the IKJPUTL and IKJGETL buffers. It invokes IKJEBELE to format the revised line as a record. If line overflow occurs and the data set type is not NOMUM TEXT (CANONUM bit is set to 0 or CASCODE field is not set to CATEXT), IKJEBECN truncates the line and notifies the user. If line overflow occurs and the data set type is NONUM TEXT (CANONUM bit is set to 1 and CASCODE field is set to CATEXT), IKJEBECN calls IKJEBELE to line edit the overflow into a record. When the record is returned, IKJEBECN invokes IKJEBEUT to write the record into the EDIT data set with an overflow key. If the data set type is BASIC or IPLI (CASCODE field is set either to CABASIC or CAIPLI) and the processor is in the system (CAPTCHK field is not set to zero), IKJEBECN updates the reverse Polish data set. The ITF entry code for input or replacement (CASYNCD2 field is set to B'10000') for updating the data set. IKJEBECN continues processing until all specified records have been changed. It then sets the current line pointer to the last line changed or its last overflow line if one exists. IKJEBECN returns control to IKJEBEMA.

Table 7. Summary of CHANGE Operations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Description	MO ID	Flow Chart ID
CHANGE	linnum1 linnum2 string1 string2	ALL	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEEECH).	02	CW-CY
				IKJEBECH	Invokes the Access Method to read records specified as operands. Issues an XCTL macro to call IKJEBECG, the load module which processes string data.	06	AL-AN
				IKJEEECG	Calls IKJEBELE to line edit string 1 and string 2. Calls IKJEBESE to search for string 1 in line number range specified. forms the new line. Calls IKJEBELE to line edit the data and any overflow into record format. Invokes IKJEEEUT to write the record into the data set.	C6	AF-AK
CHANGE	linnum1 count2 (character count)	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEEECH).	02	CW-CY
				IKJEBECH	Invokes the Access Method to read the record specified as an operand. Issues an XCTL macro to call IKJEEECN, the load module which processes character count.	06	AL-AN
				IKJEBECN	Calls IKJPUTL to print out the specified record at the terminal until the specified character count is reached. Invokes IKJGETL to obtain the user's data changes from the terminal. Forms a new line and calls IKJEEELE to edit it. Invokes IKJEEEUT to write the record into the data set.	06	AW-AT

(Part 1 of 2)

Table 7. Summary of CHANGE Operations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
CHANGE	* (current line pointer) string1	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor, (IKJEEECH).	02	CW-CY
				IKJEBECH	Invokes the Access Method to read record specified as operand (record pointed to by current line pointer). Issues an XCTL macro to call IKJEBECG, the load module which processes string data.	06	AL-AN
				IKJEBECG	Invokes IKJEBELE to tabulate characters and translate them to upper case if necessary. Calls IKJEBESE to search for string 1. Issues an XCTL macro to call IKJEBECN, the load module which processes character count.	06	AF-AK
				IKJEBECN	Same function as above.	06	AQ-AT

DELETE Subcommand Processing

The DELETE subcommand removes one or more records from the utility data set. Upon receipt of the DELETE subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the DELETE subcommand processor (IKJEBEDE). IKJEBEDE determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them. Depending upon what the user has specified, the DELETE subcommand processor:

- Deletes the current record only, if "*" or no operands were specified.
- Deletes a range of records, starting with the current record, if count were specified.
- Deletes a range of records starting and ending with two specified line numbers, if a line number range were specified.

After the specified records have been removed, the DELETE subcommand processor:

- Sets the current line pointer to the value of the key (line number) of the record previous to the deleted record(s).
- Sets CALNTOVF to 1 and reads current line into location CATEMPBF+12.
- Invokes the syntax checker to update the reverse Polish-notated data set, if the data set type is either BASIC or IPLI.

  Note: The topic Syntax Checking describes the use of the reverse Polish-notated data set.

- Returns control to the controller routine.

Processing DELETE Operands

This topic describes the way in which the DELETE subcommand processor operates when particular operands are specified. Table 8, which follows this topic, summarizes the operations of the DELETE subcommand processor.

"*" or No Operands Specified

If no operands are present or if * were entered in place of the subcommand or operand, IKJEBEDE calls the Access Method Interface routine (IKJEBEUT) to delete the current line. (Deleting of the current line is indicated by X'10' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the delete operation routine (IKJEBEDR) which uses the record locate routine (IKJEBELO) to find the current line and the record delete routine (IKJEBEDL) to delete it. After the current line is deleted IKJEBEDE calls IKJEBEUT again, this time to read the previous record. (Reading of the previous record is indicated by X'01' in the option code byte of the parameter list passed to IKJEBEUT.)

"Count" Specified

If count were specified as the operand (for example, if DELETE * 10 was entered), IKJEBEDE calls IKJEBEUT to delete the current line. After the current line is deleted IKJEBEDE calls IKJEBEUT again, this time to read the next line. If the required number of lines has not been deleted, this line is deleted and the next line in the data set is read. After the required number of lines has been deleted or the end of the data set is reached, IKJEBEDE calls IKJEBEUT again to read the previous line.

Line Number Range Specified

If a range of lines were specified (for example, if DELETE 20 40 was entered), IKJEBEDE calls IKJEBEUT to delete the first specified line, that is line 20. (Deleting of a particular line is indicated by X'10' in the option code byte and a record key value in the second word of the parameter list passed to IKJEBEUT.) After the specified line is deleted, IKJEBEDE calls IKJEBEUT again, this time to read the next line. The next line is then deleted. This operation, that is deleting lines one by one, is repeated until the required number of lines are deleted. IKJEBEUT is then called to read the previous line. The required number of lines have been read when:

1. the last line of a specified range of lines has been deleted, or
2. a line with a key value higher than the last line of the specified range has been read, or
3. the last line of the data set has been deleted.

Updating the Current Line Pointer

After the required number of lines have been deleted and the previous line has been located, IKJEBEDE sets the current line pointer (CACURNUM in IKJEBECA) to the previous line. If the search for the previous line is unsuccessful, that is, there are no previous lines, the current line pointer is set to zero. IKJEBEDE calls IKJEBEUT to read the first record in the data set. (Reading of the first record of the data set is indicated by X'04' in the option code byte of the parameter list passed to IKJEBEUT.) If no record is found, message number IKJ52501I is issued through IKJEBEMS. If a record is found, message IKJ52505I is issued.

Invoking the Syntax Checker

IKJEBEDE determines if the data set is BASIC/IPLI by examining the CADSCODE field (in IKJEBECA). If the data set type is BASIC/IPLI and the appropriate syntax checker is available, IKJEBEDE invokes the syntax checker to delete lines from the reverse Polish data set.

Table 8. Summary of DELETE Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
DELETE	linnum1 linnum2	None	None	IKJEPEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEDE).	C2	CW-CY
				IKJEPEDE	Invokes the Access Method to find and delete records indicated by the line number values specified as cperands; sets the current line pointer to the key of the record previous to linnum1 value.	C7	AY-BA
DELETE	*count	None	None	IKJEPEMA	Same function as above.	C2	CW-CY
				IKJEBEDE	Invokes the Access Method to find and delete the current record and the number of records following it (indicated by count); sets the current line pointer to the key of the record previous to the current record.	07	AY-BA
				IKJEPEUT	Acts as interface to IKJEBEAA which finds and deletes specified records; passes key of record previous to current record.	27	ET

DOWN Subcommand Processing

The DOWN subcommand moves the current line pointer toward the end of the data set. The number of lines the line pointer will move is governed by the value in the operand of the DOWN subcommand; if no value is specified, the current line pointer will be moved one line toward the end of the data set. Upon receipt of the DOWN subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the DOWN subcommand processor (IKJEBEDO). IKJEBEDO determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them.

Depending upon what the user has specified, the DOWN subcommand processor:

- Reads records toward the end of the utility data set until the number of records specified by "count" have been read, if an operand were specified.
- Reads the record following the record pointed to by the current line pointer, if no operand were specified.

After the specified number of records have been read, the DOWN subcommand processor:

- Sets the current line pointer to the value of the record last read.
- Turns the line to be verified switch on.
- Returns control to the controller routine.

DOWN Processing

This topic describes the way in which the DOWN subcommand processor operates. Table 9, which follows this topic, summarizes the operation of the DOWN subcommand processor. If no operands are present, or after the operands have been validated, IKJEBEDO calls the interface routine for the EDIT Access Method (IKJEBEUT) to read the last record of the data set. (Reading of the last record of the data set is indicated by X'05' in the option code byte of the parameter list passed to IKJEBEUT). IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the last record of the data set.

If the data set is empty, indicated by a return code of 4 from IKJEBEUT, message number IKJ52501I is selected by the message selection routine (IKJEBEMS) and put out by IKJPUTL. IKJEBEDO sets the current line pointer to zero and returns control to IKJEBEMA. If the data set is not empty (IKJEBEUT return code 0), IKJEBEDO calls IKJEBEUT again, this time to locate the record following the current record. (Reading of the record following the current record is indicated by X'02' in the option code byte of the parameter list passed to IKJEBEUT.)

IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the next record. If the record cannot be found, message number IKJ52500I is selected by the message selection routine (IKJEBEMS) and put out by IKJPUTI. IKJEBEDO sets the current line pointer to the last record referred to, turns the line to be verified switch on, and returns control to IKJEBEMA. If the record is found, IKJEBEDO determines if the current line pointer has been moved down the requested number of lines. If the count operand value has not been satisfied, IKJEBEUT is called to read the next record in the data set. If the count has been satisfied, IKJEBEDO sets the current line pointer to the last record referred to. If this last record is the last record in the data set, message number IKJ52500I is issued through IKJEBEMS.

Table 9. Summary of DOWN Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
DOWN	count	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEDO).	02	CW-CY
				IKJEBEDO	Invokes the Access Method to determine if the data set is empty; sets the current line pointer to the key of the record which was the last specified by the count value.	08	BE,BF
				IKJEBEUT	Acts as the interface to IKJEBEAA which finds the specified records; passes key of record last found to IKJEBEDO.	27	ET

END Subcommand Processing

The END subcommand terminates the processing of the EDIT program. If the utility data set has been modified, and the user has not entered SAVE prior to entering END, the END subcommand processor issues message IKJ52555I and prompts the user to enter SAVE or END. If any thing other than SAVE or END is entered, the END subcommand processor returns control to the controller (IKJEBEMA) with a return code of zero. If SAVE is entered, the SAVE subcommand processor (IKJEBESA) is invoked. If END is entered, or after successful completion of SAVE, the END subcommand processor:

- Invokes the Access Method termination routine (IKJEBEEX) to delete the Access Method (IKJEBEAA) and to free the utility data set.
- Invokes the SCAN subcommand processor (IKJEBESC) to delete the syntax checker if it is in storage; (if the CASCANSW field in the EDIT Communication Area is set to 1, there is a syntax checker to be deleted).
- Deletes the message selection (IKJEBEMS) and the line edit (IKJEBELE) service routines.
- Cancels the abnormal end exit routine (IKJEBEAE) via the STAE macro.
- Cancels the attention exit routine (IKJEBEAT) via the STAX macro.
- Returns control to the EDIT controller routine (IKJEBEMA).

The controller routine, upon receiving control from the END subcommand processor, returns to the Terminal Monitor Program by issuing SVC 3. Table 10, which follows, summarizes the operations of the END subcommand processor.

Table 10. Summary of END Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
END	None	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEEN).	C2	CW-CY
				IKJEBEEN	Informs the user if his data set has not been saved; if the user's data set has been saved, invokes the Access Method termination routine (IKJEBEEX) to terminate the Access Method; deletes the Edit service routines and frees the resources used by the Edit command processor; returns control to the controller routine (IKJEBEMA).	--	BP
				IKJEBEMA	Returns control to the Terminal Monitor Program via SVC 3.	C2	CW-CY

FIND Subcommand Processing

The FIND subcommand locates a particular character string in the utility data set. Upon receipt of the FIND subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the FIND subcommand processor (IKJEBEFI). Depending upon what the user has specified, the FIND subcommand processor:

- Locates a character string in the utility data set and returns a pointer to it, if the character string is specified.
- Refers to a character string in the CAFIBFR field of the EDIT Communication Area, locates the same string in the utility data set and returns a pointer to it, if a character string is not specified for a second use of FIND. (The character string in the CAFIBFR field was inserted during the previous use of FIND.)
- Prompts the user (thru IKJPARS) for a character string, if a character string is not specified for the first use of FIND.
- Locates a character string at an offset from the beginning of a record and returns a pointer to it, if "count" and a character string were specified.

Prior to searching for the specified string, the FIND subcommand processor determines if this is the first use of the subcommand for this EDIT session. (Note: The fields mentioned below are in the EDIT Communication area.)

- If this is the first use of FIND (CAFINDIS=0) and no string was entered (CAOPERND=0) the FIND subcommand invokes IKJPARS to prompt the user for the character string.
- If this is the first use of FIND and a string was entered (CAOPERND=1), the FIND subcommand invokes IKJPARS to scan and validate the operand, and the line edit routine (IKJEBELE) to translate the character string.
- If this is not the first use of FIND (CAFINDIS=1) and no string was entered, the FIND subcommand refers to the CAFIBFR field where the string used during the previous FIND operation is stored.

After the specified string is found and saved, the FIND subcommand processor returns control to the controller routine.

FIND Processing

This topic describes the way in which the FIND subcommand processor operates after it determines if the user has previously specified FIND and a string and enters FIND with no operands. Table 11, which follows this topic, summarizes the operations of the FIND subcommand processor. IKJEBEFI calls the Access Method interface routine (IKJEBEUT) to read the record following the current record. (Reading of the record after the current record is indicated by X'02' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the current record. If no offsets were specified by the user IKJEBEFI invokes the string search routine (IKJEBESE) to scan the records for the specified character string. After IKJEBESE has scanned the current record, if the data set type is not TEXT, IKJEBESE continues scanning the records one by one. If the data set type is TEXT, IKJEBESE calls IKJEBEUT to read the next line. (Reading of the next line is indicated by X'02' in the option code byte of the parameter

list passed to IKJEBEUT.) After the next line has been read IKJEBESE scans both lines (current line and following line) for the specified text. If the text is found across the two lines IKJEBESE returns to IKJEBEFI with a special indication (a return code of 04). If the string has not been found, IKJEBESE calls IKJEBEUT to read the next line. After this line is read IKJEBESE scans it. IKJEBESE returns to IKJEBEFI when the rest of the data set has been completely searched in this manner (IKJEBEUT returns an indication that no more lines exist) or when the specified character string has been found. When IKJEBEFI receives control back from IKJEBESE, it notifies the user if the specified character string was not found by issuing message IKJ52506I via the message selection routine (IKJEBEMS). If the text was found, IKJEBEFI updates the current line pointer to the line number of the specified text. In all cases CAFINDIS (in IKJEBECA) is set to 1. If offset is specified, IKJEBEFI scans each record, starting at the current record, at the specified offset only; no search across boundaries is done for TEXT data sets.

Table 11. Summary of FIND Operations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
FIND	string	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEFI).	02	CW-CY
				IKJEBEFI	Invokes the Access Method to read the current record; invokes IKJEBESE to find the string; IKJEBEFI notifies the user when the string is found by setting the current line pointer to the value of the key of record which contains the string.	09	BT
				IKJEBELE	Translates lower case to uppercase, if necessary, and substitutes single blanks for tab characters.	--	CO
				IKJEBEUT	Acts as interface to IKJEBEAA which reads the current record and each succeeding record until the specified string is found or the data set has been completely searched.	27	ET
				IKJEBESE	Invokes the Access Method to read each record in the data set, starting at the record following the current line, scans each record read by the Access Method for the specified address of string; passes string and key of record containing string to IKJEBEFI.	--	EH
FIND	string count	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBEFI	Invokes the Access Method to read each record in the data set, starting at the current line, until the specified string is found; searches for the string at a specified offset (count) within each record; IKJEBEFI notifies the user when the string is found by setting the current line pointer to the key of the record which contains the string.	09	BT
				IKJEBELE	Same function as above.	--	CO
				IKJEBEUT	Same function as above.	27	ET

(Part 1 of 2)

Table 11. Summary of FIND Operations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
FIND	None	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEPFFI	If first use of FIND, prompts for operands; if not first use, determines string to be found and count value from CAFIPFR; proceeds as above, starting at record following the current record.	09	BT
				IKJEPFUT	Same function as above.	27	ET
				IKJEBESE	Same function as above.	--	EH

FORMAT Subcommand Processing

The FORMAT subcommand lists the utility data set, or any part of the data set, in a user-defined format. Upon receipt of the FORMAT subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the FORMAT subcommand processor (IKJEBEFO). IKJEBEFO determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If operands are present, IKJEBEFO calls IKJPARS to scan them. The FORMAT subcommand processor invokes the FORMAT command (a Program Product) through the command invoker (IKJEBECI). The FORMAT command processor interprets the format control words in the text entered by the user and performs the required formatting operation. See the Program Product publication, IBM System/360 Operating System Time Sharing Option: TSO Data Utilities: Copy, Format, List, Merge User's Guide and Reference Manual for a description of the use and function of the format control words. Depending upon what the user has specified, the FORMAT subcommand processor invokes the TSO FORMAT command processor to:

- Format a range of records, if an "*" or a line number were specified.
- Format the entire data set, if no operands were specified.

When the TSO FORMAT command processor has completed processing, it returns control to IKJEBECI, which then returns control to the FORMAT subcommand processor. The FORMAT subcommand processor:

- Frees the Format data set.
- Returns control to the controller routine.

Processing FORMAT Operands

This topic describes the way in which the FORMAT subcommand processor operates when operands are specified. (The Program Product publication, TSO Data Utilities: Copy, Format, List, Merge User's Guide and Reference Manual describes the particulars of the FORMAT subcommand operands; the licensed Program Product publication, TSO Data Utilities: Copy, Format, List, Merge Program Logic Manual describes the internal logic of the TSO FORMAT command processor.) Table 12, which follows this topic, summarizes the operations of the FORMAT subcommand processor.

Line Number Specified

After the operands have been validated, IKJEBEFO checks if the first operand is *. If the first operand is a line number and not *, IKJEBEFO builds a model FORMAT command (including the range of data set lines specified as operands of the FORMAT subcommand) and calls the data set allocation routine (IKJEBEDA). IKJEBEDA builds a DSNAME and invokes IKJDAIR to allocate a data set with a disposition of NEW/DELETE/DELETE. When the data set has been allocated, IKJEBEDA returns to IKJEBEFO. IKJEBEFO calls the final copy routine (IKJEBEFC) to write the contents of the utility data set into the allocated data set. After the copy operation is completed, IKJEBEFO calls IKJEBEDA via LINK to mark the DSE as not in use and calls the command invoker (IKJEBECI) via LINK. The command invoker, in turn, passes control to the TSO FORMAT command via ATTACH. The FORMAT command formats the lines specified in the command model. Upon return from IKJEBECI, IKJEBEDA is invoked to free the data set.

"*" Specified; Unnumbered Data Set

If the first operand is * and the data set is not line numbered, IKJEBEFO calls IKJEBEUT to read the first record of the data set and each succeeding line until the current line (*) is read. By assigning a value to each line, IKJEBEFO is able to give the current line a relative line number. IKJEBEFO builds the FORMAT command model, inserting the relative line values. IKJEBEFO then calls IKJEBEDA to allocate a data set, IKJEBEFC to copy the utility data set into the new data set, IKJEBEDA to mark the DSE as not in use, and IKJEBECI to invoke the TSO FORMAT command. The FORMAT command formats the lines specified in the command model. Upon return from IKJEBECI, the data set is freed.

"*" Specified; Numbered Data Set

If the first operand is * and the data set is line numbered, IKJEBEFO calls IKJEBEUT to read the current line and each succeeding line until the count (FORMAT * 10) has been satisfied. The number of the last line read by IKJEBEUT is inserted by IKJEBEFO into the command model. IKJEBEFO then calls IKJEBEDA to allocate a data set, IKJEBEFC to copy the utility data set into the new data set, IKJEBEDA to mark the DSE as not in use, and IKJEBECI to invoke the TSO FORMAT command. The FORMAT command formats the lines specified in the command model. Upon return from IKJEBECI, the data set is freed.

No Operand Specified

If no operands are present, IKJEBEFO builds the FORMAT command model, specifying that the entire data set is to be formatted. IKJEBEFO then calls IKJEBEDA to allocate a data set, IKJEBEFC to copy the utility data set into the new data set, IKJEBEDA to mark the DSE as not in use, and IKJEBECI to invoke the TSO FORMAT command. The FORMAT command formats the lines specified (the entire data set) in the command model. Upon return from IKJEBECI, the data set is freed.

Table 12. Summary of FORMAT Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
FORMAT	(Refer to the Program Product publication, TSO Data Utilities: COPY, FORMAT, LIST, MERGE) for particulars.			IKJEBEMA	Receives subcommand from command buffer, invokes subcommand processor (IKJEBEFO).	02	CW-CY
				IKJEBEFO	Creates relative line numbers for unnumbered data sets; invokes the EDIT data set allocation/free service routine (IKJEBEDA) to obtain a Format data set; invokes the final copy EDIT service routine (IKJEBEFC) to copy the contents of the utility data set into the Format data set; invokes IKJEBEDA to mark the DSE as not in use; builds a model FORMAT command and invokes the command invoker (IKJEBECI) to pass control to the TSO FORMAT command processor (a program product); invokes IKJEBEDA to free the Format data set.	10	BU,BV
				IKJEBEDA	Builds a DSNAME and invokes the TSO data set allocation service routine (IKJDAIR) to allocate and to free the Format data set and to mark the DSE as not in use.	--	AX
				IKJEBEFC	Invokes the Access Method to find and to read into storage every record in the data set; transforms the records into QSAM format; invokes the Access Method to write the records into the Format data set.	--	BR,BS
				IKJEBECI	Passes control to the TSO FORMAT command processor via Attach with model command as input; returns control to IKJEBEFO when the FORMAT command processor has completed processing.	--	AO,AP
				TSO FORMAT Command	Formats records in the Format data set, as specified in the model command.	--	--

HELP Subcommand Processing

The HELP subcommand provides explanations of the use of the EDIT subcommands. Upon receipt of the HELP subcommand, the controller routine (IKJEBEMA) invokes the HELP/PROFILE subcommand processor (IKJEBEHE). IKJEBEHE sets the third and fourth bytes of the subcommand buffer (the offset field) to zero so that IKJSCAN can be invoked to scan the buffer. IKJEBEHE builds a two-word parameter list which contains pointers to the EDIT Communication Area (IKJEBECA) and to the subcommand buffer. IKJEBEHE passes control (via XCTL) to the command invoker (IKJEBECI). The command invoker, in turn, passes control to the TSO HELP command via ATTACH. (See the publication IBM System/360 Operating System Time Sharing Option Command Processor Program Logic Manual Volume IV for a description of the internal logic of the HELP command processor.) When the system HELP command has completed processing, IKJEBECI receives control and returns to IKJEBEMA. Table 13, which follows, summarizes the operations of the HELP subcommand processor.

Table 13. Summary of HELP Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
HELP	EDIT Sub-command	FUNCTION or SYNTAX or OPERANDS (list)	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEHE).	02	CW-CY
				IKJEBEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO HELP command.	11	BW
				IKJEBECI	Attaches the HELP command.	--	AO,AP
				TSO HELP Command	Describes the function, syntax, or operands applicable to the specified subcommand.	--	--
HELP	None	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBEHE	Same function as above.	11	BW
				IKJEBECI	Same function as above.	--	AO,AP
				TSO HELP Command	Displays a list of all the EDIT subcommands.	--	--
HELP	EDIT sub-command	ALL	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBEHE	Same function as above.	11	BW
				IKJEBECI	Same function as above.	--	AO,AP
				TSO HELP Command	Describes the function, syntax, and operands applicable to the specified subcommand.	--	--

INPUT Subcommand Processing

The INPUT subcommand establishes the Input mode for the EDIT program; the INPUT subcommand processor writes subsequent terminal input into the utility data set. The controller routine (IKJEBEMA) invokes the INPUT subcommand processor (IKJEBEIP and IKJEBEIM) under the following conditions:

1. The user entered the INPUT subcommand.
2. The user entered the INSERT subcommand with no operands.
3. The user entered a null line while in Edit mode.
4. The Input mode was initially entered because the data set was new, or old and empty.

Depending upon what the user has specified, the INPUT subcommand processor:

- Writes records into the utility data set beginning at the first line, if the data set is new, or old and empty.
- Writes records into the utility data set beginning at the line specified, if the INPUT subcommand and an operand were entered.
- Writes records into the utility data set beginning at the line following the one pointed to by the current line pointer, if the INSERT subcommand with no operands were specified.
- Writes records into the utility data set beginning at the line following where the previous Input mode operation ceased, if a null line were entered while in Edit mode.
- Writes records into the utility data set beginning at the line following the last existing record in the data set, if the INPUT subcommand and no operands were specified.

After the INPUT subcommand processor determines which line is to receive the input record, it checks to see if there is room for the new record.

- If there is no room for the new record and the data set is line-numbered, IKJEBEIM issues message IKJ52400I via the message selection routine (IKJEBEMS) and returns control to IKJEBEMA.
- If there is no room for the new record and the data set is not line-numbered, IKJEBEIM rewrites all displaced lines with their original keys incremented by 1.

Prior to inserting the new record, the INPUT subcommand processor determines whether the appropriate syntax checker is available, if the input is to be scanned for syntax errors (the CASCANSW field in the Edit Communication Area is set to 1).

- If the data set is PLI, BASIC, IPLI, FORTRAN (E,G,H,GI) or GOFORT, IKJEBEIM invokes the appropriate syntax checker which scans the input records.
- If a syntax checker detects an error in an input record, the record is inserted into the data set. IKJEBEIM notifies the user by issuing an error message, and returns control to the controller routine. (The controller will then invoke the Edit mode, thereby allowing the user to correct the syntax error by altering the statement just entered.)

After all input records have been inserted, the INPUT subcommand processor:

- Sets the current line pointer to the value of the key of the last record written into the utility data set.
- Returns control to the controller routine.

INPUT Processing

This topic describes the way in which the INPUT subcommand processor operates after it is invoked by the controller routine. Table 14, which follows this topic, summarizes the operations of the INPUT subcommand processor.

INPUT Subcommand and Operands Specified

If the INPUT subcommand were entered, IKJEBEIP determines if operands are present by checking the status of CAOPRND (in IKJEBECA). If operands are present, IKJEBEIP calls IKJPARS to scan them. If no operands are present the defaults are used. After the operands have been validated, IKJEBEIP invokes the second load of Input (IKJEBEIM) via the XCTL macro instruction. IKJEBEIM invokes the access method interface routine (IKJEBEUT) to write each input record into the data set beginning at the line specified. (Writing of a record is indicated by X'20' in the option code byte and a pointer to the line in the data in the second word of the parameter bit passed to IKJEBEUT.) IKJEBEUT branches to the write operation routine (IKJEBEWR) which uses the record locate routine (IKJEBELO) to find the specified line and the move routine (IKJEBEMV) to insert the new record into the found line. If R were specified as an operand for the INPUT subcommand, IKJEBEWR also uses the record delete routine (IKJEBEDL) to delete any records found in lines specified for new records.

INPUT Subcommand and No Operands Specified

If the INPUT subcommand were entered and no operands were specified, IKJEBEIP invokes IKJEBEUT to obtain the last line in the data set. (Reading of the last line in the data set is indicated by X'05' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the read operation routine (IKJEBERR) which uses IKJEBELO to locate the line. After the last line of the data set has been found, IKJEBEIP invokes the second load of INPUT (IKJEBEIM) via the XCTL macro instruction. IKJEBEIM calls IKJEBEUT to write the input records into the data set beginning at the line following the last line. If the data set is line-numbered, the line number assigned to the first new record will be that of the last record plus the increment specified in the INPUT subcommand on the default value.

INSERT Subcommand and No Operands Specified

If the INSERT subcommand with no operands were entered, IKJEBEIP invokes the second load of INPUT (IKJEBEIM) via the XCTL macro instruction. IKJEBEIM invokes IKJEBEUT to insert the new record into the line following the current line. If the data set is line-numbered, the new record will be given a line number 1 greater than the current line.

Null Line Entered in Edit Mode

If a null line was entered while in Edit mode, IKJEBEMA invokes the Input mode. Records will be entered into the data set beginning at the line following where the previous Input mode operation ceased. IKJEBEIP determines the last line number generated by the previous Input mode operation by checking CAIMLLNO (in IKJEBECA), and invokes the second load of INPUT (IKJEBEIM) via the XCTL macro instruction. IKJEBEIM invokes the access method to write the input records into the data set. If Input mode were not previously used, the first new record will be inserted at the last line of the data set plus the last used increment.

Empty Data Set (Data Set is New, or Old and Empty)

If the data set is empty, IKJEBEMA establishes the Input mode by calling IKJEBEIP, which invokes the second load of INPUT (IKJEBEIM) via the XCTL macro instruction. IKJEBEIM invokes the access method to insert the new records. The first record inserted will have a line number of 10; each succeeding line number will be incremented by 10.

Table 14. Summary of INPUT Operations (Part 1 of 5)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
INPUT	linnum incre	R	None	IKJEBEMA	Receives subcommand from buffer, invokes the subcommand processor (IKJEBEIP); treats subsequent terminal input as data.	02	CW-CY
				IKJEBEIP	Processes operands to determine starting line for data insertion; specifies that new records will have line numbers incremented by value in "incre"; invokes Access Method to locate the line in the data set with a line number value of "linnum"; invokes second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes Access Method to replace any record existing in line; invokes the Access Method to write into the line the subsequent terminal input; returns to IKJEBEMA.	12	BX-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which is invoked to replace delete and add records to the utility data set.	27	ET

(Part 1 of 5)

Table 14. Summary of INPUT Cperations (Part 2 of 5)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functicnal Descripticn	MO ID	Flow Chart ID
INPUT	*	I PRCMPT	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBEIP	Processes operands to determine starting line for data insertion; specifies that new records will have line numbers incremented by the last increment used and that the terminal will display a prompting character before each input line; invokes the Access Method to locate the current line; invokes the second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes the Access Method to insert the subsequent data into the line, if it is empty, or after the line, if it is not empty; returns to IKJEBEMA.	12	BX-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which reads the current record to locate the position in the data set at which new records are to be added; writes new records into the data set.	27	ET
INPUT	None	None	None	IKJEBEMA	Save function as above.	02	CW-CY
				IKJEBEIP	Determines that the last record in the data set is to be located; invokes the Access Method to locate the last record in the data set; invokes the second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes the Access Method to write into the data set the subsequent data from the terminal; returns to IKJEBEMA.	12	BY-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which reads the last record in the data set; writes new records into the data set.	27	ET

(Part 2 of 5)

Table 14. Summary of INPUT Operations (Part 3 of 5)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
Input not entered; data set is NEW or OLD and empty.	None	None	None	IKJEBEMA	Receives indication from initialization (IKJEBEIN) that data set is NEW or OLD and empty; sets the INPUT mode by invoking the INPUT subcommand processor (IKJEBEIP).	02	CW-CY
				IKJEBEIP	Determines that the first line in the empty data set is to be located, specifies 10 as the first line number and 10 as the increment for each succeeding line number; invokes the second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes the Access Method to move subsequent data from the terminal into the first line of the data set; returns to IKJEBEMA.	12	BX-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which writes new records into the utility data set.	27	ET

(Part 3 of 5)

Table 14. Summary of INPUT Operations (Part 4 of 5)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
INPUT not entered; a null line is entered while in Edit mode	None	None	None	IKJEBEMA	Sets the Input mode by invoking the INPUT subcommand processor (IKJEBEIP).	C2	CW-CY
				IKJEBEIP	Determines that the last record is to be located, if INPUT has not been entered before in this EDIT session; determines that the record following the last record entered during the previous Input mode operation is to be located, if Input has been entered before; specifies that the line increment be the default value, or the value last used if Input mode were used previously in this EDIT session; invokes the Access Method to locate the specified record; invokes the second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes the Access Method to write the subsequent data into the data set; returns to IKJEBEMA.	12	BX-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which reads and writes records into the utility data set.	27	ET

(Part 4 of 5)

Table 14. Summary of INPUT Operations (Part 5 of 5)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functicnal Descripticn	MO ID	Flow Chart ID
INPUT not entered; INSERT with no operands entered.	None	None	None	IKJEBEMA	Receives subcommand from INSERT subcommand processor (IKJEBEIS) invokes the INPUT subcommand processor (IKJEBEIP); treats subsequent terminal input as data.	02	CW-CY
				IKJEBEIP	Determines that the current record is to be located; specifies that the new record will have a line number of 1 greater than the current record; invokes the Access Method to locate the current record; invokes the second load of INPUT (IKJEBEIM).	12	CK,CL
				IKJEBEIM	Invokes the Access Method to write a new record immediately after the current record; returns to IKJEBEMA.	12	BX-CB
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the current line and moves subsequent data into a new line following the current link.	27	ET

INSERT Subcommand Processing

The INSERT subcommand writes one or more records into the utility data set immediately following the record pointed to by the current line pointer. Upon receipt of the INSERT subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the INSERT subcommand processor (IKJEBEIS). Depending upon what the user has specified, the INSERT subcommand processor:

- Returns control to the controller routine, which invokes the INPUT subcommand, if no operands were specified.
- Writes data into the record following the current record, if "text" were specified.

If the data set is line-numbered, the INSERT subcommand processor either:

- Creates a new record using the next key value, or
- Terminates if the next key value already exists.

If the data set is unnumbered, the INSERT subcommand processor:

- If necessary, "pushes down" record keys following the current record to make room for the record to be inserted.
- Creates a new record with a key of one greater than the current record.

After the text has been inserted, the INSERT subcommand processor:

- Sets the current line pointer to the value of the key of the last record inserted.
- Returns control to the controller routine.

Processing INSERT Operands

This topic describes the way in which the INSERT subcommand processor operates when operands are specified or not specified. Table 15, which follows this topic, summarizes the operations of the INSERT subcommand processor.

No Operands Specified

If no operands are present, IKJEBEIS returns control to IKJEBEMA with a return code of 4. IKJEBEMA then calls the INPUT subcommand processor (IKJEBEIP and IKJEBEIM) to write the lines of input into the data set. Writing into the data set begins at the line following the current line. IKJEBEIM returns control to IKJEBEMA. (See the topic "INPUT Subcommand Processing" for a description of subsequent processing.)

Operands Specified

If an operand is specified (INSERT text) IKJEBEIS calls the Access Method interface routine (IKJEBEUT) to read the next line. (Reading of the next line is indicated by X'02' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the read operation routine. (IKJEBERR) which uses the record locate (IKJEBELO) to find the next line in the data set. After the next line has been located, IKJEBEIS determines if it is empty by checking the return code from IKJEBEUT. (A return code of 4 indicates an empty line.) If the next line in the data set is empty IKJEBEIS calls IKJEBEUT again, this time to write the input record (text) into the data set. Writing of the record is indicated by X'20' in the option code byte and a pointer to the line in the data set (the line following the current line) in the second word of the parameter list passed to IKJEBEUT. IKJEBEUT branches to the write operation routine (IKJEBEWR) which uses the move routine (IKJEBEMV) to insert the new record into the data set. IKJEBEIS sets the current line pointer to the record inserted. If the next line in the data set is not empty and the data set is not line-numbered, IKJEBEIS calls IKJEBEUT to write the input record into the data set. IKJEBEUT branches to IKJEBEWR which uses IKJEBEMV to insert the new record into the data set. Prior to writing of the new record, records with higher keys are displaced to make room for the record to be inserted.

Table 15. Summary of INSERT Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
INSERT	None	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes INSERT subcommand processor (IKJEBEIS).	02	CW-CY
				IKJEBEIS	Returns to IKJEBEMA. (See INPUT subcommand processing).	13	CM,CN
INSERT	data	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes INSERT subcommand processor (IKJEBEIS).	02	CW-CY
				IKJEBEIS	Invokes the Access Method to locate the line following the one pointed to by the current line pointer and to move the data into it.	13	CM,CN
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the line following the current line and moves the input data (indicated by the operand 'data') into the found line. IKJEBEAA which locates the current line and moves subsequent data into a new line following the current line.	27	ET

Line Insert/Replace/Delete Processing

The Line Insert/Replace/Delete function is an implicit subcommand function which inserts, replaces, and deletes particular records in the utility data set. (Note: This subcommand is invoked when the required operand is entered. There is no subcommand name or designation to be entered by the user.) Upon receipt of a number or * as the first character of input while in the Edit mode, the controller routine (IKJEBEMA) invokes the Line Insert/Replace/Delete processor (IKJEBELI). Depending upon what the user has specified, the Insert/Replace/Delete subcommand processor:

- Deletes a single record, if an "*" or a line number were entered.
- Inserts a single record, if a line number, followed by text, is specified, and if the existing record on the utility data set is empty.
- Replaces an existing record with a new record, if a line number, followed by text (the new record), is specified, and if the existing record on the utility data set is not empty.

After the specified record has been deleted, or after the specified line has received the new record, the Line Insert/Replace/Delete subcommand processor:

- Sets the current line pointer.
- Returns control to the controller routine.

Processing Line Insert/Replace/Delete Operands

This topic describes the way in which the Line Insert/Replace/Delete subcommand processor operates when particular operands are specified. Table 16, which follows this topic, summarizes the operations of the Line Insert/Replace/Delete subcommand processor.

"*" or Line Number Specified

If only a required operand is entered, the specified line is deleted. IKJEBELI calls the Access Method interface routine (IKJEBEUT) to delete the record. (Deleting of a record is indicated by X'10' in the option code byte and a pointer to the particular record in the second word of the parameter list passed to IKJEBEUT). IKJEBEUT branches to the delete operation routine (IKJEBEDR) which uses the record locate routine (IKJEBELO) to find the record and the record delete routine (IKJEBEDL) to delete it.

Line Number and Text Specified

If a required operand is followed by text (e.g. * 'data' or 10 'data'), IKJEBELI calls the line edit routine (IKJEBELE) which converts the text to be inserted (data) to uppercase, if CAPS were specified in the EDIT command or defaulted and converts tabs to the specified number of blanks. IKJEBELI calls IKJEBEUT to write the text into the specified line. (Writing is indicated by X'20' in the option code byte and a pointer to the line in the data set in the second word of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the write operation routine (IKJEBEWR) which uses the record locate routine (IKJEBELO) to find the particular line in the data set. If a record exists in the line which IKJEBELO has found, the record delete routine (IKJEBEDL) is called to delete it. After the record is deleted, or if no record exists in the found line, IKJEBEWR branches to the move routine (IKJEBEMV) to insert the new record into the data set. (If the data set type is either IPLI or BASIC, the new record is also passed to the syntax checker which updates the reverse Polish-notated data set.)

Table 16. Summary of Line Insert/Replace/Delete Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
None	linnum	None	None	IKJEBEMA	Receives implicit subcommand buffer; invokes the Insert/Replace/Delete function.	02	CW-CY
				IKJEBELI	Invokes the Access Method to find the record indicated by linnum and to delete it.	14	CP,CQ
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the record with a key equal to linnum and deletes the record.	27	ET
None	*	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBELI	Invokes the Access Method to find the current record and to delete it.	14	CP,CQ
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the current record and deletes it.	27	ET
None	linnum string	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBELI	Invokes the Access Method to find the record indicated by linnum and to replace it with 'string'.	14	CP,CQ
				IKJEBEUT	Acts as interface to IKJEBEAA which creates a record if no key corresponds to 'linnum', or replaces a record having a corresponding key.	27	ET

Updating the Current Line Pointer

If no text were specified, after the record is deleted, the Line Insert/Replace/Delete subcommand processor sets the current line pointer to the value of the key of the record preceding the deleted record. If a line number and text were specified, after the new record is inserted or after the old record is replaced, the Line Insert/Replace/Delete subcommand processor sets the current line pointer to the key of the new record.

LIST Subcommand Processing

The LIST subcommand prints out specific records from the utility data set. Upon receipt of the LIST subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the LIST subcommand processor (IKJEBELT). IKJEBELT determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them. Depending upon what the user has specified, the LIST subcommand processor:

- Displays the entire utility data set, if no operands were specified.
- Displays a range of records starting and ending with two specified line numbers, if a line number range were specified.
- Displays a single record, if a single line number, or "*" were specified.

The records displayed will contain line numbers if the data set is line numbered and the user has not specified the SNUM keyword. The LIST subcommand processor:

- Displays unnumbered records with no line numbers.
- Displays numbered records with line numbers, if SNUM were not specified.
- Displays numbered records with no line numbers, if SNUM were specified.

The LIST subcommand processor returns control to the controller routine.

Processing LIST Operands

This topic describes the way in which the LIST subcommand processor operates when particular operands are specified. Table 17, which follows this topic, summarizes the operations of the LIST subcommand processor.

No Operands Specified

If no operands are present (for example, if LIST was entered), IKJEBELT calls the Access Method interface routine (IKJEBEUT) to read the first record of the data set. (Reading of the first record is indicated by X'04' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the first record. IKJEBELT calls IKJEBEUT again, this time to read the next record in the data set. (Reading of the next record is indicated by X'02' in the option code byte of the parameter list passed to IKJEBEUT.) IKJEBELT calls IKJEBEUT after each succeeding record is read until the end of the data set is reached.

Line_Number_Range_Specified

If a range of lines were specified as operands (e.g. LIST 10 20 or LIST * 20 was entered), IKJEBELT calls IKJEBEUT to read the record represented by the first operand. After this record is read IKJEBELT continues to call IKJEBEUT to read records until the record represented by the second operand is read.

"*"_or_Single_Line_Number_Specified

If a single line were specified as an operand (e.g. LIST 10 or LIST * was entered), IKJEBELT calls IKJEBEUT to read the particular record.

Unnumbered_Data_Set

If the data set is not line-numbered, IKJEBELT calls IKJPUTL to write out the record.

Line-numbered_Data_Set,_SNUM_not_Specified

If the data set is line numbered and SNUM was not specified, IKJEBELT formats the output lines by affixing line numbers to their respective records (separated by one blank from the data) with up to three leading zeros truncated. After the output lines have been formatted, IKJEBELT calls IKJPUTL to write out the records.

Line-numbered_Data_Set,_SNUM_Specified

If the data set is line-numbered and SNUM was specified, IKJEBELT formats the output lines by blanking out the line numbers for fixed-length records or by left-justifying the text for variable-length records and for fixed-length records with line numbers starting in the first byte (COBOL). After the output lines have been formatted, IKJEBELT calls IKJPUTL to write out the records.

Table 17. Summary of IIST Cperations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Operaticn	Functional Description	MO ID	Flow Chart ID
LIST	linnum1 linnum2	NUM	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBELT).	02	CW-CY
				IKJEBELT	Invokes the Access Method to find the record with a key equal to linnum1 and to read it and each succeeding record until the record with a key greater than or equal to linnum2 has been found and read; obtain records from IKJEBUT and invokes PUTLINE to display them one at a time.	15	CT-CV
				IKJEBEUT	Acts as interface to IKJEBAA which locates the records with keys from linnum1 to linnum2.	27	ET
LIST	* count	SNUM	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBELT	Invokes the Access Method to find the current record, or the first record if * has a zero value, and to read it and each succeeding record until 'count' records have been read; displays records via PUTLINE.	15	CT-CV
				IKJEBEUT	Acts as interface to IKJEBAA which locates the record with the current key and the number of records (indicated by 'count') which follow.	27	ET
LIST	linnum	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBELT	Invokes the Access Method to find the record with a key equal to linnum and to read it; displays the record via PUTLINE.	15	CT-CV
				IKJEBEUT	Acts as interface to IKJEBAA which locates the record with a key equal to linnum.	27	ET

(Part 1 of 2)

Table 17. Summary of LIST Operations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Operation	Functional Description	MO ID	Flow Chart ID
LIST	None	NUM	None	IKJEPEMA	Same function as above.	02	CW-CY
				IKJEBELT	Invokes the Access Method to read all the records in the data set; displays the records via PUTLINE.	15	CT-CV
				IKJEBEUT	Acts as interface to IKJEPEAA which locates the first record of the data set and each record thereafter until the end of the data set is reached.	27	ET

MERGE Subcommand Processing

The MERGE subcommand copies all or part of a data set into a specified area within the data set being edited. Upon receipt of the MERGE subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the MERGE subcommand processor (IKJEBEME). IKJEBEME determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If no operands are present IKJEBEME issues an error message. If operands are present, IKJPARS is called to validate them. IKJEBEME invokes IKJEBEDA to allocate a QSAM data set for use by the Final Copy routine (IKJEBEFC). When the data set has been allocated, IKJEBEDA returns to IKJEBEME. IKJEBEME calls IKJEBEFC to copy the utility data set into the QSAM data set. When copy processing has been completed, IKJEBEFC returns to IKJEBEME. IKJEBEME again calls IKJEBEDA, this time to mark the DSE (Data Set Entry) for the QSAM data set as 'not-in-use.' IKJEBEME builds a command buffer in which it sets up a model MERGE command to be used by TSO MERGE command processor. IKJEBEME calls the command invoker, (IKJEBECI), which in turn invokes the TSO MERGE command processor.

When the TSO MERGE command processor has completed processing, it returns control to IKJEBECI which returns control to IKJEBEME. IKJEBEME again invokes IKJEBEDA to allocate the QSAM data set which now contains the output from the system MERGE command processor. When the data set has been allocated, IKJEBEDA returns control to IKJEBEME. IKJEBEME calls the copy routine, IKJEBECO to copy the new QSAM data set to a new utility data set. When copy processing has completed, IKJEBECO returns control to IKJEBEME. IKJEBEME saves the current line pointer which now points to the end of the data set. IKJEBEME invokes IKJEBEDA a final time to free the QSAM data set. IKJEBEDA returns control to IKJEBEME. If the data set is IPLI or BASIC, IKJEBEME calls IKJEBEMR for re-translation of the reverse Polish-notated data set. IKJEBEME calls IKJEBEEX in every case to delete the old utility data set. When IKJEBEME receives control at the completion of MERGE command processing, it returns to the controller routine.

The MERGE subcommand processor invokes the MERGE command (a Program Product). The MERGE command processor combines the specified data sets or portions of data sets. See the Program Product publication, IBM System/360 Operating System Time Sharing Option: TSO Data Utilities: Copy, Format, List, Merge User's Guide and Reference Manual for a description of the use and function of the MERGE control words.

Processing MERGE Operands

This topic describes the way in which the MERGE subcommand processor operates when particular operands are specified. (The Program Product publication TSO Data Utilities: Copy, Format, List, Merge User's Guide and Reference Manual describes the particulars of the MERGE subcommand operands; the licensed Program Product publication, TSO Data Utilities: Copy, Format, List, Merge Program Logic Manual describes the internal logic of the TSO MERGE command processor.) Table 18, which follows this topic, summarizes the operations of the MERGE subcommand processor.

"*" Only Specified

If the dsname is specified as '*', IKJEBEME uses the QSAM data set name obtained from IKJEBEDA as the first operand of the model MERGE command. The second dsname operand of the model command is the name of the QSAM data set.

DSNAME Specified

If a dsname is specified, IKJEBEME uses it as the first operand of the model MERGE command. The second dsname operand of the model command is the name of the QSAM data set.

Linenumber 1 and Linenumber 2 Specified

If linenum 1 and linenum 2 are specified by the user as operands of the subcommand, IKJEBEME moves them into the model MERGE command buffer as operands of the MERGE TSO command.

Linenumber 3 and RENUM Keyword Specified

If the user specified data set is line-numbered and the merge point is '*', IKJEBEME places the current line pointer in the linenum 3 operand position of the model MERGE command. It then moves the keyword operand RENUM into the model command.

Linenumber 3 and NONUM Keyword Specified

If the user specified data set is not line-numbered and the merge point is '*', IKJEBEME calls IKJEBEUT to read the data set backwards from the current line pointer to the beginning to obtain a relative record number. If the user did not specify line number 3 IKJEBEME defaults the value to '*'. It then moves the keyword operand NONUM into the model command.

Table 18. Summary of MERGE Operations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
MERGE	(Refer to the Program Product publication, TSO Data Utilities: COPY, FORMAT, LIST, MERGE for particulars.)			IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEME).	02	CW-CY
				IKJEEEME	Invokes IKJEEEDA to allocate a QSAM data set to be used by the Final Copy routine (IKJEEEFC). Calls IKJEEEFC to copy the utility data set into the QSAM data set. Builds a command buffer in which it sets up a model MERGE command to be used by TSO MERGE command processor. For BASIC or IPLI, invokes IKJEBEMR to update the reverse Polish-notated data set; invokes IKJEBEEX to unallocate the old utility data set.	16	CZ-DA
				IKJEEECI	Passes control to the MERGE command processor via ATTACH with model command as input; returns control to IKJEBEME when the MERGE command processor has completed processing.	--	AO,AP

(Part 2 of 2)

Table 18. Summary of MERGE Operations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
				IKJEBEDA	Allocates the QSAM data set which now contains output from the MERGE command processor.	--	AX
				IKJEBECO	Copies the new QSAM data set to a new utility data set.	--	AU-AW
				IKJEEEFC	Copies old utility data set into an intermediate QSAM data set.	--	BR,BS
				IKJEBEEX	Unallocates old utility data set after merge operation.	26	BQ
				IKJEEEMR	Calls ITF language processor to delete old in-storage data set and to create a new data set.	--	DB
				IKJEECI	Builds a parameter list and invokes the Command to attach the TSO PROFILE command.	11	BW
				IKJEEECI	Attaches the PROFILE command.	--	AC,AP

PROFILE Subcommand Processing

The PROFILE subcommand redefines the set of options which control the flow of information to and from the terminal. Once defined, the options specified become part of the user profile, entered in the User Profile Table. The user profile remains in effect until it is redefined by the user. When User Profile Table entries are redefined, any options not specifically defined by operands on the PROFILE subcommand remain unchanged.

Upon receipt of the PROFILE subcommand, the controller routine (IKJEBEMA) invokes the HELP/PROFILE subcommand processor (IKJEBEHE). IKJEBEHE builds a two-word parameter list which contains pointers to the EDIT Communication Area (IKJEBECA) and to the subcommand buffer. IKJEBEHE insures that the third and fourth bytes of the subcommand buffer contain zeros. The contents of the buffer begin with a header word, which is a fullword, of the form "LL00", where LL is the entire length of the buffer, including this header word. Zeros in the third and fourth bytes are necessary to enable IKJSCAN to scan the buffer. IKJEBEHE passes control (via XCTL) to the command invoker routine (IKJEBECI). The command invoker routine invokes the TSO PROFILE command processor (via ATTACH). (See the publication IBM System/360 Operating System: Time Sharing Option Command Processor Program Logic Manual Volume VI for a description of the internal logic of the PROFILE command processor.) When the PROFILE command has completed processing, IKJEBECI receives control and returns to IKJEBEMA. Table 19 summarizes the operations of the PROFILE subcommand processor.

Table 19. Summary of PROFILE Operations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
PROFILE	None	NOLINE CHAR or CHAR	None (char) (BS)	IKJEBEMA	Receives subcommand from command buffer; invokes HELP/PROFILE subcommand processor (IKJEBEHE).	02	CW-CY
				IKJEBEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO PROFILE command.	11	BW
				IKJEBECI	Attaches the PROFILE command.	--	AO,AP
				TSO PROFILE Command	Nullifies existing line delete characters; specifies a keyboard character ('char') or the keyboard backspace ('BS char') as the character delete indicator.	--	--
PROFILE	None (char)			IKJEEEMA	Same function as above.	02	CW-CY
				IKJEEEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO PROFILE command.	11	BW
				IKJEEECI	Attaches the PROFILE command.	--	AO,AP
		NCCHAR LINE or LINE or LINE	None (char) (ATTN) (CTLX	TSO PROFILE command	Specifies a keyboard character ('char'), the keyboard attention ('ATTN') or the X and CTLX keys on a teletype terminal as the line indicator; nullifies existing character-delete indicators.	--	--
PROFILE	None	NOPROMPT	PR	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEEEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO PROFILE command.	11	BW
				IKJEEECI	Attaches the PROFILE command.	--	AO,AP
				TSO PROFILE Command	Specifies that the user is not to be prompted for required information.	--	--

(Part 1 of 2)

Table 19. Summary of PROFILE Cperations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
PROFILE	None	INTERCOM	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEEEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO PROFILE command.	11	BW
				IKJEEECI	Attaches the PROFILE command.	--	AO,AP
				TSO PROFILE Command	Specifies that the user will accept messages from cther terminal users.	--	--
PROFILE	None	PAUSE	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEEEHE	Builds a parameter list and invokes the Command Invoker to attach the TSO PROFILE command.	11	BW
				IKJEEECI	Attaches the PROFILE command.	--	AO,AP
				TSO PROFILE Command	Specifies that the user will receive any available second-level messages issued during execution of a command procedures.	--	--

(Part 2 of 2)

RENUM Subcommand Processing

The RENUMBER subcommand assigns line numbers to each record of an unnumbered data set or renumbers each record of a numbered data set. Upon receipt of the renumber subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the RENUMBER subcommand processor (IKJEBERE). IKJEBERE determines if operands are present by checking the status of the operand switch. If operands are present (CAOPERND in IKJEBECA is set to 1) IKJEBERE calls IKJPARS to validate the operands. Depending upon what the user has specified, the RENUM subcommand processor:

- Renumbers an entire utility data set from the first record if no operands are specified or if an "old line number" is specified which is equal to the first record.
- Renumbers a range of records from a specified record to the end of the data set if an "old line number" is specified.

The RENUM subcommand processor assigns line number values and an increment value:

- Based on the last increment used by INPUT or RENUM, if no operands are specified. (An increment of 10 is used if this is the first use of either subcommand.)
- Based on values specified as "new line number" operand and "increment" operand.

After the records have been renumbered, the RENUM subcommand processor updates the following fields in IKJEBECA:

- CANONUM bit set to 0 if the data set was previously unnumbered.
- CACURNUM, the current line pointer, set to the same relative record that it pointed to before RENUM processing.
- CAIMLLNO, the last input line number.
- CAIMLINC, the increment used to renumber the data set.
- CASTNUM, the starting line number.
- CADSMODS bit in CACFLAG2, set to 1 to indicate that the data has been modified.
- CAIMPT bit in CACFLAG3, set to 1 to indicate that Input mode is to prompt with line numbers.

The RENUM subcommand processor (IKJEBERE) then returns control to the controller routine.

Processing RENUM Operands

This topic describes the way in which the RENUM subcommand processor operates when various operands are specified. Table 20, which follows this topic, summarizes the operations of the RENUM subcommand processor.

No Operands Specified

If no operands are specified IKJEBERE calls IKJEBEUI to allocate a new output utility data set. If the data set type is BASIC, IKJEBERE calls IKJEBERN and IKJEBEMR to renumber the utility and reverse Polish-notated data sets respectively. If the data set type is IPLI, IKJEBERE renumbers the utility data set and invokes IKJEBEMR to update the run time data set. IKJEBERE invokes the Access Method (IKJEBEAA) via IKJEBEUT, the interface routine, to read the first specified record. If the data set is not already numbered IKJEBERE examines the record and if necessary, formats it. IKJEBERE calls IKJEBEUT to write each updated record into the utility data set. Processing continues as if operands had been specified. See Continued Processing below.

"Old Line Number" Specified

If an old line number has been entered as an operand, it must be validated. IKJEBEUT reads the record specified by old line number. If the new line number is greater than or equal to the old line number (i.e., the record key) processing passes to a copying loop in IKJEBERE. However, if the new line number is less than the old line number IKJEBERE calls IKJEBEUT to read the record preceding the one just read. If the record read is greater than or equal to the new line number, a message is issued and RENUM terminates. Since the data set is already numbered IKJEBERE updates the record keys by adding the user specified increment to the first new key and numbers succeeding lines by the increment. IKJEBERE calls IKJEBEUT to write each updated record into the new utility data set and continues processing.

Continued Processing

- If the record format is fixed and non-blanks are in the area needed for the line number (bit CACRECFM in CACFLAC2 is set to 1), IKJEBERE updates the key and indicates that a truncation message must be issued (ISSTRUNK bit in REMYFLAC is set to 1).

- If the record format is variable (bit CACRECFM in CACFLAC2 is set to 0), IKJEBERE shifts the data to the right to make room for the line number to be entered. If this causes data overflow of non-blank characters and the data set is not "TEXT" type the ISSTRUNK bit in REMYFLAC is set to 1 indicating truncation. If the data set is 'TEXT' type, IKJEBERE breaks the record, builds a new record using the overflow and sets the ISSTRUNK bit to 1.

- IKJEBERE calls IKJEBEEX to close and unallocate the old utility DCB if renumbering is successful.

- IKJEBERE moves the current DCB pointer (CAPTCDCB in IKJEBECA) to the previous DCB pointer field (CAPTPDCB). It then resets the CAPTCDCB field to the address of the new utility DCB and returns to IKJEBEMA.

Table 20. Summary of RENUM Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
RENUM	None	None	None	IKJEEEMA	Receives subcommand from command buffer; invokes subcommand processor.	C2	CW-CY
				IKJEBERE	Invokes the Access Method to find and read each record of the data set; gives the first record a line number of 10 and renumbers the succeeding records, each with an increment of 10 until the data set has been renumbered; after each record is renumbered, invokes the Access Method to write the record into a new utility data set (Renum data set).	18	DJ-DM
				IKJEBEUT	Acts as interface to IKJEEEAA which, beginning at the first record of the data set, reads the records into storage and writes them into the Renum data set.	27	ET
				IKJEEEUI	Acquires the Renum data set.	26	EQ
				IKJEEEEX	Closes and frees data set.	26	BQ
RENUM	New linenum incre old linenum	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEEERE	Invokes the Access Method to find and read the record with a key equal to 'old linenum'; gives this record a new line number equal to 'new linenum'; invokes the Access Method to read the remainder of the data set into storage one by one; renumbers the succeeding records with an increment of 'incre'; invokes the Access Method to write renumbered records into a new utility data set (Renum data set).	18	DJ-DN
				IKJEBEUT	Beginning at the record with a key equal to 'old linenum', reads the records into storage and writes them into the Renum data set.	27	ET
				IKJEBEUI	Same function as above.	26	EQ
				IKJEEEEX	Same function as above.	26	BQ

RUN Subcommand Processing

The RUN subcommand compiles, loads, and executes ASM, BASIC, COBOL, FORTGI, IPLI, PLI, and GOFORT data sets. Upon receipt of the RUN subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the RUN subcommand processor (IKJEBERU). If the data set is not an executable one, i.e., not ASM, BASIC, COBOL, FORTGI, PL/I, PL/I(F), IPLI or GOFORT or any other user-specified types that are non-executable, IKJEBERU returns control to IKJEBEMA. IKJEBERU determines if the data set type is BASIC or IPLI. If the data set type is either BASIC or IPLI, IKJEBERU calls the appropriate syntax checker which runs the data set. For all other data set types, IKJEBERU determines if operands are present by checking the status of the operand switch.(CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them. If no operands are present, or after operands have been validated, depending upon the data set type, the RUN subcommand processor performs the following functions:

- For data set types that can accept in-storage data sets (i.e., the CARUNDS bit is on in IKJEBECA), reads records from the utility data set into storage or allocates a new data set if the utility data set is too large to be contained in a 4K in-storage data set.
- For all other data set types, allocates a new data set.

For the data set types that required an allocated data set, the RUN subcommand processor:

- Invokes the final copy routine (IKJEBEFC) to write a copy of the original data set in QSAM format into the newly allocated data set.
- Invokes the allocating routine (IKJEBEDA) to mark the DSE entry not in use.
- Invokes TSO RUN command processor, for an OBJ-generating data set (i.e., CAOBJGEN bit is on in IKJEBECA). (See the IBM System/360 Operating System Time Sharing Option Command Processor Program Logic Manual Volume VI publication for a description of the internal logic of the RUN command processor).
- Invokes the appropriate prompter whose name is obtained from the CAPRNAME field in (IKJEBECA) or a non-OBJ-generating data set.

  For data set types that can accept in-storage data sets, the appropriate prompter whose name is obtained from the CAPRNAME field in (IKJEBECA) is invoked.

When TSO RUN command processing has been completed, the RUN subcommand processor:

- Returns command to the controller routine.

RUN Processing for Particular Data Set Types

This topic describes the way in which the RUN subcommand processor operates when particular data set types are specified. Table 21, which follows this topic, summarizes the operations of the RUN subcommand processor.

GOFORT Data Set Type

If the data set type accepts in-storage data sets, IKJEBERU attempts to build a copy of the utility data set in storage. A GETMAIN is issued for 4K of buffer storage and records are read using IKJEBEUT. Each record is moved into the buffer (with the record key removed) until all records in the utility data set have been read, or until the buffer storage is exhausted.

If the buffer storage is exhausted before the data set can be built, IKJEBERU invokes IKJEBEDA to allocate the run-time data set, and IKJEBEFC to copy the utility data set into a sequential (run-time) data set.

When the data set is copied into the run-time data set or into storage, IKJEBERU builds a model command and invokes the appropriate prompter via IKJEBECI. The run-time data set is unallocated via IKJEBECA, if data set was allocated.

For any OBJ-generating data set, IKJEBERU invokes IKJEBEDA to allocate an intermediate sequential (run-time) data set and IKJEBEFC to copy the utility data set into the run-time data set. A model RUN command is built and the system RUN command processor is invoked via IKJEBECI. Upon return from IKJEBECI, IKJEBERU invokes IKJEBEDA to unallocate the run-time data set, and the corresponding OBJ data set.

BASIC or IPLI Data Set Type

IKJEBERU invokes the appropriate language processor which is resident in storage while EDIT is being executed.

For any other executable data set type, IKJEBERU invokes IKJEBEDA to allocate an intermediate sequential (run-time) data set and IKJEBEFC to copy the utility data set into the run-time data set. A command is built and its appropriate prompter (whose name is extracted from the CAPRNAME field of IKJEBECA) is invoked via IKJEBECI. Upon return from IKJE BECI, IKJEBERU invokes IKJEBEDA to unallocate the run-time data set.

Note: Execution of source data sets other than IPLI, BASIC, or GOFORT is accomplished through the system Loader. If the 'parameters' operand is specified on the EDIT RUN subcommand, the Loader will pass the specified parameters in accordance with the standard Operating System linkage conventions. See the topic "Program Management Services" in the publication IBM System/360 Operating System: Supervisor Services, GC28-6646 for information about these conventions.

Table 21. Summary of RUN Operations (Part 1 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
RUN	para-meters'		None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBERU).	02	CW-CY
				IKJEBERU	Invokes IKJEBEDA to acquire a Run data set and the Final Copy Routine (IKJEBECF) to read the records from the utility data set into the new Run data set; builds TSO RUN command; invokes the command invoker which attaches the TSO RUN command, for OBJ-generating data sets; otherwise it attaches the approproate prompter.	19	DP,DQ
				IKJEBEDA	Is invoked to: 1. Allocate a QSAM data set. 2. Mark DSE entry not in use. 3. Unallocate the QSAM data set. 4. Unallocate the OBJ data set, if one was generated.	--	AX
				IKJEBEFC	Invokes the Access Method to find and read each record from the utility data set into the Run data set.	--	BR,BS
				IKJEBECI	For an OBJ-generating data set invokes the TSO RUN command; otherwise invokes the appropriate prompter; in both cases passes the command buffer built by IKJEBERU. (RUN subcommand parameters are included). It also invokes the commands in the procedure built by the RUN command, if the data set type was OBJ-generating.	--	AO,AP
				TSO RUN Command	Builds a command procedure required to cause compilation and execution of the RUN data set, for OBJ-generating data set types.	--	OSOS
RUN	None	LMSG SPREC	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBERU	BASIC data set - IKJEBERU invokes LANGPRCR to execute data.	19	DP,DQ
				LANGPRCR	Executes user's data set.	--	--

Table 21. Summary of RUN Operations (Part 2 of 2)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
RUN	None	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBERU).	02	CW-CY
				IKJEBERU	For data set types accepting in-storage data sets invokes IKJEBEUT to read the data set into a 4K in-storage buffer; invokes STACK to put the in-storage data set on the input stack; invokes IKJEBECI to attach the appropriate prompter. For other data set types (excluding BASIC or IPLI), or for data sets over 4K that were acceptable in-storage invokes IKJEBEDA to allocate a run-time data set; invokes IKJEBEFC to copy the utility data set into the run-time data set and invokes IKJEBECI to attach either system RUN or the appropriate prompter.	19	DP,DQ
				IKJEBEUT	Reads records into the 4K in-storage buffer via IKJEBEAA, in the storage of a data set type accepting on in-storage data set	27	ET
				IKJEBEDA	Is invoked to: 1. Allocate a QSAM data set. 2. Mark DSE entry not in use. 3. Unallocate the QSAM data set. 4. Unallocate the OBJ data set, if one was generated.	--	AX
				IKJEBEFC	Copies the utility data set into the sequential run-time data set.	--	BR,BS
				IKJEBECI	For an OBJ-generating data set invokes the TSO RUN command; otherwise invokes the appropriate prompter; in both cases passes the command buffer built by IKJEBERU. (RUN subcommand parameters are included). It also invokes the commands in the procedure built by the RUN command, if the data set type was obj-generating.	--	AO,AP
				TSO RUN Command	Builds a command procedure required to cause compilation and execution of the RUN data set.	--	--

SAVE Subcommand Processing

The SAVE subcommand retains the copy of the Edit data set to which changes have been made. Upon receipt of the SAVE subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the SAVE subcommand processor (IKJEBESA). IKJEBESA determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them.

Depending on whether or not the user has specified a data set name as an operand, the SAVE subcommand processor:

- Saves only the updated version, if one exists, of an Edit data set if the user has not specified an operand.
- Saves the updated version of the Edit data set and retains the original version, if one exists. The updated version is given the data set name specified by the user as operand.
- Returns control to the controller routine.

Processing the SAVE Subcommand

This topic describes the way in which the SAVE subcommand processor operates when operands are specified or not specified. Table 22, which follows this topic, summarizes the operation of the SAVE subcommand processor.

DSNAME Specified

If a data set name is present and not enclosed in quotes, IKJEBESA invokes IKJDFLT to qualify the name. IKJDFLT places the user identification as the left qualifier (if it is not present) and the data set type qualifier as the right qualifier (if it is not present, or if only one index level is specified). If the data set type is present as the right qualifier, IKJEBESA removes it and invokes IKJDFLT to place it back again. If the data set name is enclosed in quotes and has more than one index level, IKJEBESA removes the rightmost qualifier and invokes IKJDFLT to place it back again.

If only a member name is present and a member name was specified on the EDIT command, the Edit data set name is used and is treated as if it were enclosed in quotes. If only a member name is present and no member name was specified on the EDIT command, IKJEBESA invokes IKJDFLT to concatenate the user identification and data set type qualifier. IKJDFLT also determines if the fully-qualified data set name is cataloged.

After the data set name is fully-qualified, IKJEBESA compares it with the fully-qualified Edit data set name. If the names are the same, processing continues as if no operands were present. If the names differ, then the updated Edit data set will be retained with the Save data set name. If the data set name is not cataloged, IKJEBESA calls the data set allocation routine (IKJDAIR) to allocate a new data set with a disposition of NEW, CATLG. If IKJDAIR is unable to allocate a data set with this disposition because a data set already exists with this name, or if the name is cataloged, further processing depends on the data set organization.

If the data set is sequential, IKJEBESA issues a warning message to the user, who may then enter a different data set name to begin processing over again, or a carriage return to continue processing. IKJDAIR is then called to retry allocation of the data set.

If the data set is partitioned and the specified member exists, IKJEBESA issues a warning message to the user, who may then enter a different member name or a carriage return to use the specified member.

When the data set has been allocated, IKJEBESA calls the final copy routine (IKJEBEFC). Processing continues with IKJEBEFC calling IKJEBEUT which reads records from the utility data set, and writes them into the Save data set. Thus the data set with changes is retained with the Save data set name, and the original Edit data set, if any, is undisturbed. IKJEBESA returns control to IKJEBEMA, after calling IKJDAIR to unallocate the data set.

No Operands Specified

If no operands are present, the changed version of the Edit data set that is on the utility data set is to replace the original Edit data set. If the Edit data set is already allocated, IKJEBESA can call the final copy routine (IKJEBEFC) to write the contents of the utility data set into the Edit data set. If the Edit data set is not already allocated (that is, for the first allocation of a NEW data set), the Edit data set name is treated as described under "Dsname Specified." If the IKJDFLT service routine finds that the Edit data set is cataloged when the user specified NEW on the Edit command, the user is warned and prompted to: (1) re-use the data set or (2) specify an alternate data set name. If the Edit data set exists and is partitioned, and if the member name specified exists (NEW was specified on the command), the user is warned and prompted to: (1) to re-use the member or (2) to specify a new member name. Once the Edit data set has been allocated, IKJEBEFC is invoked to copy records from the utility data set into the Edit data set.

Table 22. Summary of SAVE Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
SAVE	dataset name (not the same as the Edit data set name)	None	None	IKJEEEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBESA).	C2	CW-CY
				IKJEEESA	Invokes TSO service routine IKJDFLT to fully qualify data set name; invokes TSO service routine IKJDAIR to allocate a Save data set; invokes the final copy routine (IKJEBEFC) to write the contents of the utility data set into the Save data set.	20	DR-EB
				IKJEBEFC	Invokes the Access Method to find and read each record of the utility data set; writes the records in to the Save data set.	--	BR,BS
SAVE	None or data-set name (same name as Edit data set)	None	None	IKJEEEMA	Same function as above.	C2	CW-CY
				IKJEBESA	Invokes the TSO service routine IKJDFLT to fully qualify data set name, if data set name is specified; determines that data set name specified and Edit data set name are the same; invokes the final copy routine (IKJEEEFC) to write the contents of the utility data set into the Edit data set.	20	DR-EB
				IKJEEEFC	Invokes the Access Method to find and read each record of the utility data set; converts each record to QSAM data set format and writes the records into the Edit data set.	--	BR,BS

SCAN Subcommand Processing

The SCAN subcommand performs syntax checking for statements that will be processed by the PL/I(F), FORTRAN(E), FORTRAN(G) or FORTRAN(H) compiler or by the Code and Go FORTRAN, FORTRAN IV(G1), ITF: BASIC or ITF: PL/I Program Product. Upon receipt of the SCAN subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the first load of the SCAN subcommand processor (IKJEBESC). IKJEBESC determines if the data set type specified by the user is an invalid type for scanning (CASCAN switch in IKJEBECA is set to zero). If so, IKJEBESC invokes the message service routine to issue an error message and returns control to the caller.

Depending upon what the user has specified, the SCAN subcommand processor:

- Loads and initializes the required syntax checker, if the ON keyword were specified.
- Deletes the syntax checker, if the OFF keyword were specified.
- Passes records from the utility data set to the syntax checker in storage, if the line number or "count" operands were specified.
- Passes all the records from the utility data set to the syntax checker in storage, if no operands were specified.

  Note: The BASIC and IPLI syntax checkers are used to update the reverse Polish-notated data set and to scan source statements for syntax errors. The topic Syntax Checking describes the use of the reverse Polish-notated data set.

The SCAN subcommand processor is also invoked by the controller routine when the SCAN keyword of the EDIT command is specified, or when the data set type is BASIC or IPLI.

After the specified records have been syntax checked, the SCAN subcommand processor returns control to the controller routine.

Processing SCAN Operands

This topic describes the way in which the SCAN subcommand processor operates when particular operands are specified. Table 23, which follows this topic, summarizes the operations of the SCAN subcommand processor.

Line Numbers or Count Specified

If either of these operands are present (CAOPERND switch in IKJEBECA is not zero) IKJEBESC calls IKJPARS to check the validity of the operands. IKJEBESC checks the availability of the appropriate syntax checker. If the syntax checker is available in the system (CASYNAME field in IKJEBECA is not equal to 0), IKJEBESC invokes IKJEBESN, the second load module of the SCAN subcommand processor. IKJEBESN loads and initializes the checker if it is not in storage (CAPTCHK field of IKJEBECA=0), and invokes IKJEBEAA, the Access Method to read into storage the range of records specified by the user as operands of the SCAN subcommand. If the data set type is BASIC or IPLI, the syntax checker is already in storage. The module IKJEBEUT acts as an interface between IKJEBESN and IKJEBEAA which locates the record keys corresponding to the user specified keys.

IKJEBEUT informs IKJEBESN if the user specified data set is empty. IKJEBESN calls the appropriate syntax checker for the data set type and informs the user of any syntax errors.

No Operands Specified

If no operands are present (CAOPERND switch in IKJEBECA is zero) IKJEBESC calls IKJEBESN, the second load module of the SCAN subcommand processor. IKJEBESN loads and initializes the syntax checker, if necessary, and invokes the Access Method IKJEBEAA, using the module IKJEBEUT as an interface. IKJEBEAA locates and reads the first record and every record following until the entire data set has been read into storage.

No Operands and ON Keyword Specified

If no operands are specified and the keyword ON is specified each line of input from the user terminal is to be syntax checked as it is entered. IKJEBESC checks the availability of the syntax checker; if the data set type is other than ITF, IKJEBESC loads and initializes the appropriate syntax checker. IKJEBESC turns on the CASCANSW switch in IKJEBECA. From this time on, the INPUT subcommand processor (IKJEBEIP) will pass input records from the terminal to the syntax checker for verification.

No Operands and OFF Keyword Specified

If no operands are specified and the keyword OFF is specified IKJEBESC deletes the syntax checker unless the data set type is BASIC or IPLI. IKJEBESC turns off the CASCANSW switch in IKJEBECA.

Table 23. Summary of SCAN Operations (Part 1 of 4)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
SCAN (NOSCAN specified on EDIT command)	linnum1 linnum2	None	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBESC).	C2	CW-CY
				IKJEBESC	Checks availability of syntax checker; if syntax checker is available, invokes the second load of the SCAN subcommand processor (IKJEBESN) to perform the syntax checking of the specified records.	21	EC-EG
				IKJEBESN	If data set type is not BASIC or IPLI, loads and initializes syntax checker, if it is not in storage (if data set type is BASIC or IPLI, the checker is already in storage); invokes the Access Method to locate and to read into storage the range of records beginning with the record with a key equal to 'linnum1' and ending with the record with a key equal to 'linnum2'; invokes the appropriate syntax checker to perform its function on the records which were read into storage; informs the user of any syntax errors.	21	EI-EN
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the record with a key equal to 'linnum1' and reads it and the succeeding records (ending with record 'linnum2' key value) into storage; informs IKJEBESN if the data set is empty.	27	ET

(Part 1 of 4)

Table 23. Summary of SCAN Cperations (Part 2 of 4)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
SCAN (SCAN specified on EDIT command)	* count	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEPESC	Same function as above.	21	EC-EG
				IKJEBESN	Invokes the Access Method to locate and to read into storage the range of records beginning with the current record and ending when the number of records read into storage is equal to the value specified in 'count'; invokes the appropriate syntax checker (which has already been loaded into storage) to perform its function on the records which were read into storage; informs the user of any syntax errors.	21	EI-EN
				IKJEPEUT	Acts as interface to IKJEBEAA which locates the record with the current key and reads it and the succeeding records into storage until the value specified in 'count' has been satisfied; informs IKJEPESN if the data set is empty.	27	ET

(Part 2 of 4)

Table 23. Summary of SCAN Operations (Part 3 of 4)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
SCAN (NOSCAN specified on EDIT command)	None	None	None	IKJEBEMA	Same function as above.	C2	CW-CY
				IKJEBESC	Same function as above.	21	EC-EG
				IKJEBESN	If data set type is not BASIC or IPLI, loads and initializes syntax checker (if data set type is BASIC or IPLI, checker is already in storage); invokes the Access Method to locate every record in the data set and to read them into storage; invokes the appropriate syntax checker to perform its function on the records which were read into the data sets; informs the user of any syntax errors.	21	EI-EN
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the first record in the data set and reads it and every record thereafter until the entire data set has been read into storage.	27	ET
SCAN (NOSCAN specified on EDIT command)	None	ON	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBESC	Checks availability of syntax checker; if the data set type is other than BASIC or IPLI, and if "SCAN ON" has not been issued previously, loads and initializes the appropriate syntax checker (if the data set type is BASIC or IPLI the syntax checker is already loaded and initialized); input records from the terminal received by EDIT during Input mode from this point on will be passed to the syntax checker; turns on CASCANSW switch in IKJEBECA.	21	EC-EG

(Part 3 of 4)

Table 23. Summary of SCAN Operations (Part 4 of 4)

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
SCAN (SCAN specified on EDIT command)	None	OFF	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEEESC	If the data set type is other than BASIC or IPLI, deletes the syntax checker if it is in storage (the syntax checker is not deleted if the data set type is BASIC or IPLI). Turns off CASCANSW switch.	21	EC-EG
EDIT (SCAN subcommand not specified)	------	SCAN	None	IKJEEEIN	Determines if syntax checker is available; if syntax checker is available, indicates the fact by setting CASCANON to 1 and not setting CASYNAME to 00000000.	01	CC-CJ
				IKJEBEMA	Loads the appropriate syntax checker; invokes the SCAN subcommand processor (IKJEEESC) to initialize it.	02	CW-CY
				IKJEEESC	Initializes the syntax checker which was loaded into storage.	21	EC-EG
END (SCAN subcommand not specified)	None	None	None	IKJEBEEN	Invokes the SCAN subcommand processor (IKJEBESC) to delete the syntax checker, if it is in storage.	--	BP
				IKJEEESC	Deletes the syntax checker in storage.	21	EC-EG
EDIT (SCAN subcommand not specified)	------	NCSCAN BASIC or IPLI	None	IKJEEEIN	If the syntax checker is available, indicates the fact by not setting CASYNAME to 00000000.	01	CC-CJ
				IKJEBEMA	Loads the BASIC or IPLI LANGPRCR; invokes the SCAN subcommand processor (IKJEEESC) to initialize it.	02	CW-CY
				IKJEEESC	Initializes the LANGPRCR which was loaded into storage.	21	EC-EG

TABSET Subcommand Processing

The TABSET subcommand establishes or changes tabulation settings or nullifies any existing tabulation settings. Upon receipt of the TABSET subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the TABSET subcommand processor (IKJEBTA). IKJEBETA determines if operands are present by checking the status of the operand switch (CAOPERND) in IKJEBECA. If operands are present, IKJEBETA calls IKJPARS to validate them.

The TABSET subcommand processor indicates whether tabulation characters are to be translated into blanks and sets new values for tabulation characters (specifies column locations for tabs) by updating a table in the EDIT Communication Area. The Line Edit routine (IKJEBELE) refers to this table when it is invoked to translate tabulation characters into blanks. Depending upon what the user has specified, the TABSET subcommand processor:

- Indicates that tabulation translation is to be performed, if no operands or if the ON keyword were specified.
- Indicates that tabulation translation is not to be performed, if the OFF keyword were specified.
- Sets new values for tabulation characters, if either the ON keyword and a tab list were specified, or if the IMAGE keyword and an input line of tabs were entered.

Note: CATABS, a 12-byte area in IKJEBECA, is used as the tabulation switch and as a table of the tab character values. The first byte is the tabulation switch. The second thru the eleventh bytes contain up to 10 tab character values. The 12th byte contains X'00'.

The TABSET subcommand processor returns control to the controller routine.

Processing TABSET Operands

This topic describes the way in which the TABSET subcommand processor operates when particular operands are specified. Table 24, which follows this topic, summarizes the operations of the TABSET subcommand processor.

No Operands Specified

If no operands are present, IKJEBETA turns on the tabulation switch (CATABS in IKJEBECA) which indicates that translation of the tabulation settings into blanks is to be performed. The tabulation settings will take on the default values specified at Sysgen time on the values specified during a previous TABSET subcommand operation in the current EDIT session.

ON or OFF Keywords Specified

If the ON or OFF keyword were specified, IKJEBETA sets the tabulation switch to perform or not to perform translation of the tabulation settings.

ON Keyword and Integer List Specified

If ON and an integer list were specified, e.g., TAB ON (2 8 72), IKJEBETA stores values in ascending order in the tabulation table (CATABS) and turns the tabulation switch on.

IMAGE Keyword and Input Line of Tabs Specified

TAB IMAGE is followed by a line containing a "t" for each desired tab position to mark the actual column location of the tabulation characters. (A character other than a "t" is considered as a blank in this line.) The tab set character (t) may be entered as either upper or lower case. When the IMAGE keyword is specified using tab set characters IKJEBETA invokes the GETLINE service routine to obtain the line containing the TAB characters, determines the tabulation settings, and stores them in the tabulation table. IKJEBETA updates the CAINPROC field of the EDIT communication area to indicate the input source of the IMAGE line, i.e., terminal input or procedure input.

Table 24. Summary of TABSET Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved In Processing	Functicnal Descripticn	MO ID	Flow Chart ID
TABSET	None	None or ON	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBETA).	02	CW-CY
				IKJEBETA	Specifies that tabulation settings be translated into blanks.	22	EO
TABSET	(integer list)	None or ON	None	IKJEBEMA	Same function as above.	C2	CW-CY
				IKJEBETA	Specifies that tabulation settings specified in the integer list be translated into the proper number of blanks.	22	EO
TABSET	None	IMAGE	None	IKJEBEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBETA).	C2	CW-CY
				IKJEBETA	Treats next input line as series of tabset characters. Obtains the next line from the current source of input; determines the tabulation settings from the cccurances of 't' in the input line; stores these values.	22	EO
TABSET	None	OFF	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBETA	Specifies that tabulation characters be translated into single blanks.	22	EO

TOP Subcommand Processing

The TOP subcommand positions the current line pointer to the beginning of the utility data set; in an unnumbered data set, the beginning of the data set is the position preceding the first record. In a numbered data set, line number zero is the beginning.

The TOP subcommand processor:

- Reads the first record in the utility data set.
- Sets the current line pointer to zero.
- Returns control to the controller routine.

TOP Processing

This topic describes the way in which the TOP subcommand processor operates. Table 25, which follows this topic, summarizes the operation of the TOP subcommand processor. Upon receipt of the TOP subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the TOP subcommand processor (IKJEBETO). IKJEBETO calls the interface routine for the EDIT Access Method (IKJEBUT) to read the first record in the data set. (Reading of the first record of the data set is indicated by X'04' in the option code byte of the parameter list passed to IKJEBEUT.)

IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the first record of the data set. If the data set is empty, indicated by a return code of 4 from IKJEBEUT, message number IKJ52501I is issued through the message selection routine (IKJEBEMS) and put out by IKJPUTL. IKJEBETO sets the current line pointer to zero. If the data set is not empty (IKJEBEUT return code = 0), IKJEBETO sets the current line pointer to zero. If a record exists for line zero, the line to be verified switch (CALNOTOVF in IKJEBECA) is turned on. IKJEBETO returns control to IKJEBEMA. If record zero does not exist, and if the verify switch (CAVRFYSW in IKJEBECA) is turned on, IKJEBETO issues message IKJ52504I with an insertion of "0" or "*" depending upon whether the data set is line-numbered or not.

Table 25. Summary of TOP Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Chart ID
TOP	None	None	None	IKJEBEMA	Recieves subcommand from command buffer; invokes subcommand processor (IKJEBETO).	02	CW-CY
				IKJEBETO	Invokes the Access Method to find the first record in the data set; sets the current line pointer to 0.	23	EP
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the first record of the data set.	27	ET

UP Subcommand Processing

The UP subcommand moves the current line pointer toward the beginning of the utility data set. The number of lines the line pointer will move is governed by the value in the operand of the UP subcommand; if no value is specified, the current line pointer will be moved one line toward the beginning of the data set. Upon receipt of the UP subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the syntax is valid, IKJEBEMA invokes the UP subcommand processor (IKJEBEUP). IKJEBEUP determines if operands are present by checking the status of the operand switch CAOPERND in IKJEBECA). If operands are present, IKJPARS is called to scan them.

Depending upon what the user has specified, the UP subcommand processor:

- Reads records toward the beginning of the utility data set until the number of records specified by "count" have been read, if an operand were specified.
- Reads the record previous to the record pointed to by the current line pointer, if no operand were specified.

After the specified number of records have been read, the UP subcommand processor:

- Sets the current line pointer to the value of the record last read.
- Turns the line to be verified switch on.
- Returns control to the controller routine.

UP Processing

This topic describes the way in which the UP subcommand processor operates. Table 26, which follows this topic, summarizes the operation of the UP subcommand processor. If no operands are present, or after the operands have been validated, IKJEBEUP calls the interface routine for the EDIT Access Method (IKJEBEUT) to read the first record of the data set. (Reading of the first record of the data set is indicated by X'04' in the option code byte of the parameter list passed to IKJEBEUT.)

IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBELO) to find the first record of the data set. If the data set is empty, indicated by a return code of 4 from IKJEBEUT, message number IKJ52501I is issued through the message selection routine (IKJEBEMS) and put out by IKJPUTL. IKJEBEUP sets the current line pointer to zero and returns control to IKJEBEMA. If the data set is not empty (IKJEBEUT return code = 0), IKJEBEUP calls IKJEBEUT again, this time to locate the record previous to the current record. (Reading of the record previous to the current record is indicated by X'01' in the option code byte of the parameter list passed to IKJEBEUT.)

IKJEBEUT branches to the read operation routine (IKJEBERR) which uses the record locate routine (IKJEBEIO) to find the previous record. If the record cannot be found, message number IKJ52505I is issued through the message selection routine (IKJEBEMS). IKJEBEUP sets the current line pointer to the last record referenced, turns the line to be verified switch on, and returns control to IKJEBEMA. If the record is found, IKJEBEUP determines if the current line pointer has been moved up the requested number of lines. If the count operand value has not been satisfied, IKJEBEUT is called to read the next previous record in the data set. If the count has been satisfied, IKJEBEUP sets the current line pointer to the last record referenced. If this last record is the first record in the data set, message number IKJ52505I is issued through IKJEBEMS.

Table 26. Summary of UP Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Functional Description	MO ID	Flow Cart ID
UP	count	None	None	IKJEBEMA	Receives subcommand from subcommand buffer; invokes subcommand processor (IKJEBEUP).	02	CW-CY
				IKJEBEUP	Invokes IKJEBEUT to determine if the data set is empty, and if it is not, to locate the record previous to the current record; continues to invoke IKJEBEUT to find records with lower keys until the number of records indicated by 'count' have been located; sets the current line pointer to the key value of the last record found by the Access Method.	24	ER,ES
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the first record in the data set to determine if the data set is empty and which locates the record previous to the current record and records with still lower keys until the 'count' value has been satisfied; informs IKJEBEUP whether the data set is empty or not and passes it the key of the record last found by the Access Method.	27	ET
UP	None	None	None	IKJEBEMA	Same function as above.	02	CW-CY
				IKJEBEUP	Invokes IKJEBEUT to determine if the data set is empty, and if it is not, to locate the record previous to the current record; sets the current line pointer to the key value of this record.	24	ER,ES
				IKJEBEUT	Acts as interface to IKJEBEAA which locates the first record in the data set to determine if the data set is empty and which locates the record previous to the current record; IKJEBEUT informs if the data set is empty or passes the key of the record found by the Access Method.	27	ET

VERIFY Subcommand Processing

The VERIFY subcommand provides for the display of the record pointed to by the current line pointer after a portion of the record has been altered by the CHANGE subcommand, or after the value of the current line pointer has been changed by the operation of an EDIT subcommand processor. Upon receipt of the VERIFY subcommand, the controller routine (IKJEBEMA) calls the command scan routine (IKJSCAN) to validate the subcommand. If the subcommand is valid, IKJEBEMA invokes the VERIFY subcommand processor (IKJEBEVE). The VERIFY subcommand processor indicates whether the user wants a record display or not by setting a flag in the EDIT Communication Area. This flag is tested by the CHANGE subcommand processor and by the EDIT controller routine, which acts in behalf of the other EDIT subcommand processors. Depending upon what the user has specified, the VERIFY subcommand processor:

- Indicates that a record will be displayed when the text of the current line or the value of the current line pointer is changed, if no operands or the ON keyword were specified.
- Indicates that a record will not be displayed, if the OFF keyword were specified.

The VERIFY subcommand processor returns control to the controller routine.

Processing VERIFY Operands

This topic describes the way in which the VERIFY subcommand processor operates when particular operands are specified. Table 27, which follows this topic, summarizes the operations of the VERIFY subcommand processor.

No Operands Specified

IKJEBEVE determines if operands are present by checking the status of the operand switch (CAOPERND in IKJEBECA). If no operands were specified, IKJEBEVE sets the verify switch (CAVRFYSW) to 1, indicating that during the operation of subsequent subcommands, records will be displayed.

Operands Specified

If an operand were specified, IKJEBEVE calls IKJPARS to scan the operand. If the operand is valid, IKJEBEVE sets the verify switch to 1 (if ON were specified) or to 0 (if OFF were specified).

Table 27. Summary of VERIFY Operations

Subcommand	Operand	Keyword	Keyword Value	Modules Involved in Processing	Function Description	MO ID	Flow Chart ID
VERIFY	None	ON or None	None	IKJEEEMA	Receives subcommand from command buffer; invokes subcommand processor (IKJEBEVE).	C2	CW-CY
				IKJEBEVE	Turns verify switch (CAVRFYSW) on to indicate that every time the text of a line is altered, or every time the current line pointer is changed as a result of a subcommand operation, the line is to be displayed at the terminal.	25	EU
VERIFY	None	OFF	None	IKJEEEMA	Same function as above.	C2	CW-CY
				IKJEBEVE	Turns verify switch off.	25	EU

EDIT Access Method

Inherent in the operation of most subcommands is the reading, writing, and deleting of records in the utility data set. Records can be read into, or written or deleted from the data set sequentially (forward and backward) and directly by record key. The EDIT Access Method is the portion of the EDIT program which performs these functions.

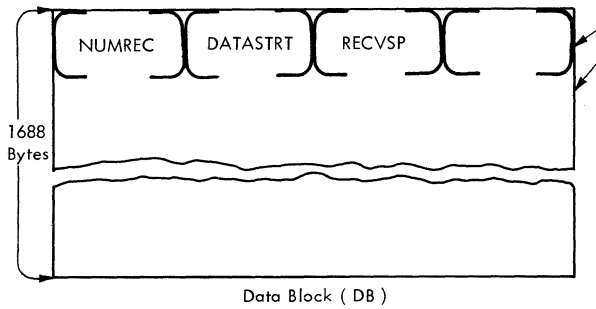
STRUCTURE

The EDIT Access Method comprises four major functional areas:

- Initialization and Final Processing.
- Interface.
- Read, Write, and Delete Operations.
- Access Method Service Routines.

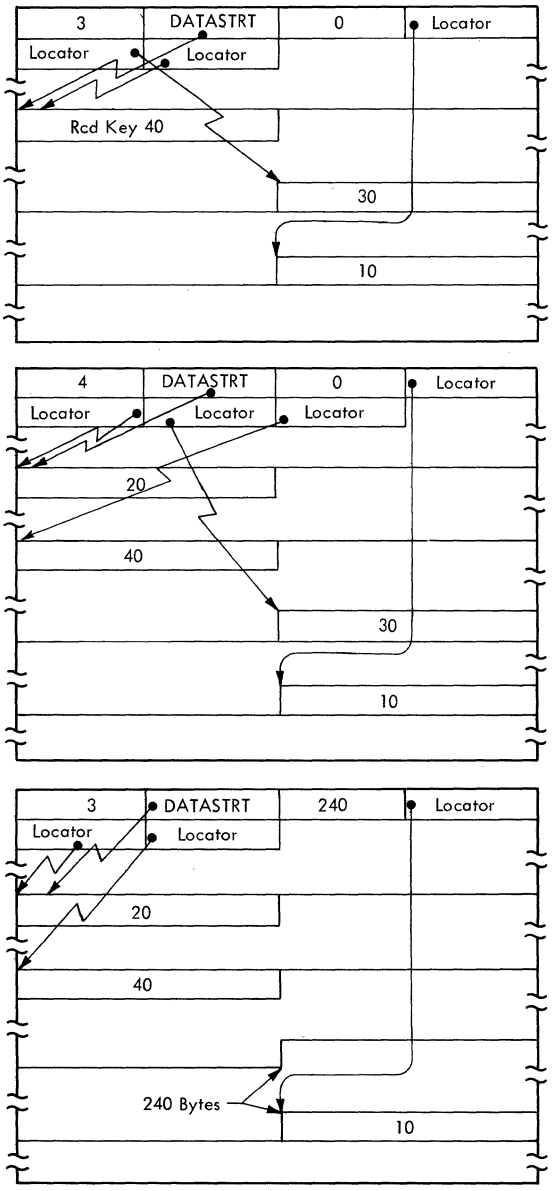
The EDIT Access Method handles variable and fixed-length, numbered and unnumbered records.

These records are inserted into data blocks which reside on a direct-access device. When needed, the data blocks are read into one of three buffers. Figures 17 and 18, respectively, describe the formats of the data blocks and the buffers.



Data Block Field Descriptions

NUMREC	(2 Bytes)-Number of Records in the Data Block
DATASTRT	(2 Bytes)-Pointer to where Next Record Should Go.
RECVSP	(2 Bytes)-Number of Bytes of Recoverable, Unused Space between the Position Pointed to by DATASTRT and the End of the Data Block.
LOCATOR ENTRY	(2 Bytes)-Points to a Record. The Locator Entries are in Ascending Sequence Based on Record Key.



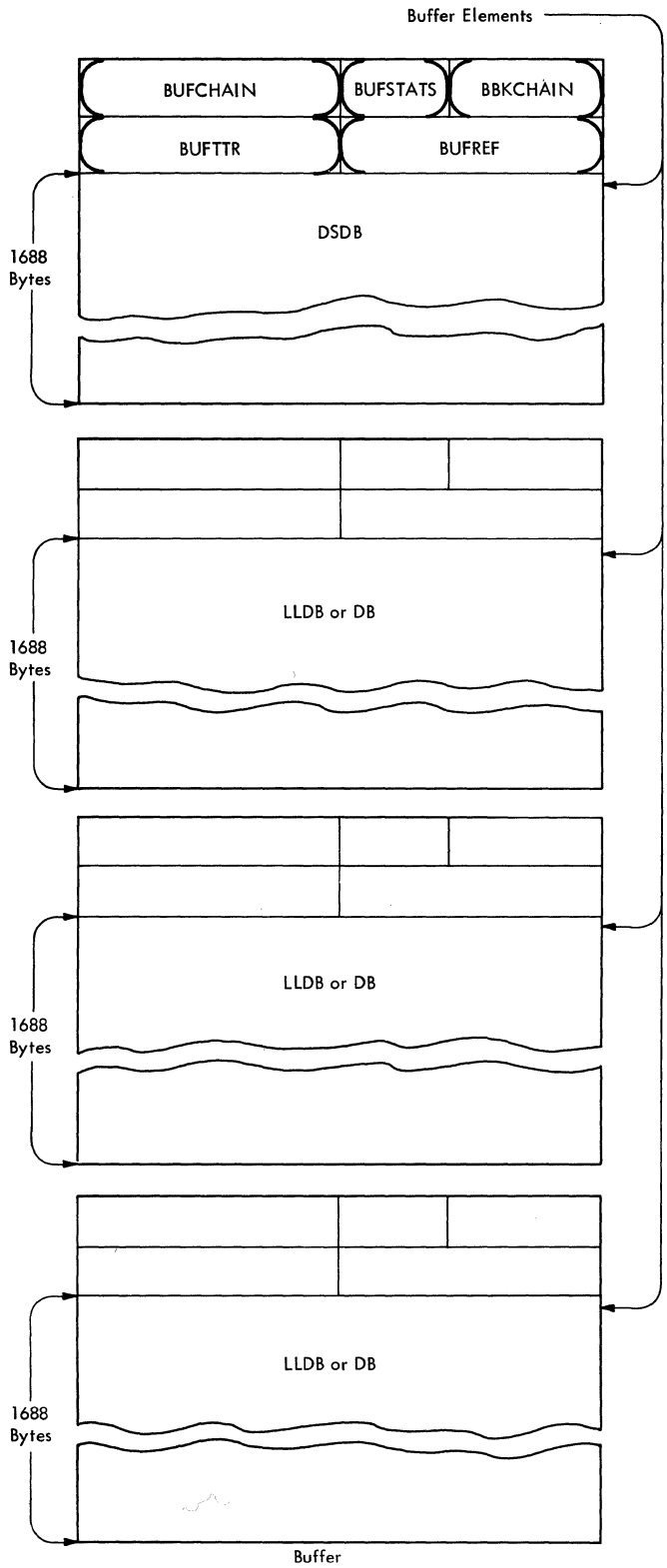
The Following Sequence Illustrates the Changes in Data Block Fields when Records are Added and Deleted.

- Original Condition - Data Block Contains 3 Records and No Recoverable Unused Space. DATASTRT Field, Pointing to Beginning of Record which was Last Entered in the Data Block, Indicates that Space Exists for More Records.

- Record Added - New Record with Key of 20 Added to Data Block. New Locator for Record is Built and Inserted into Proper Position. DATASTRT Field is Updated and Now Points to Beginning of New Record.

- Record Deleted - 240-Byte Record with Key of 30 is Deleted. Locator for the Record is Deleted. Locator for Record with Key of 40 is Moved Down. Value of 240, Representing Recoverable, Unused Space in Area of Data Block Occupied by Records, is Inserted in the RECVSP Field.

Figure 17. EDIT Access Method Data Blocks



Buffer

BUFCHAIN (4 Bytes) - Address of Next Buffer Element in Chain, or Zero if this is the Last Element.

BUFSTATS (1 Byte) - Buffer Status Flags.

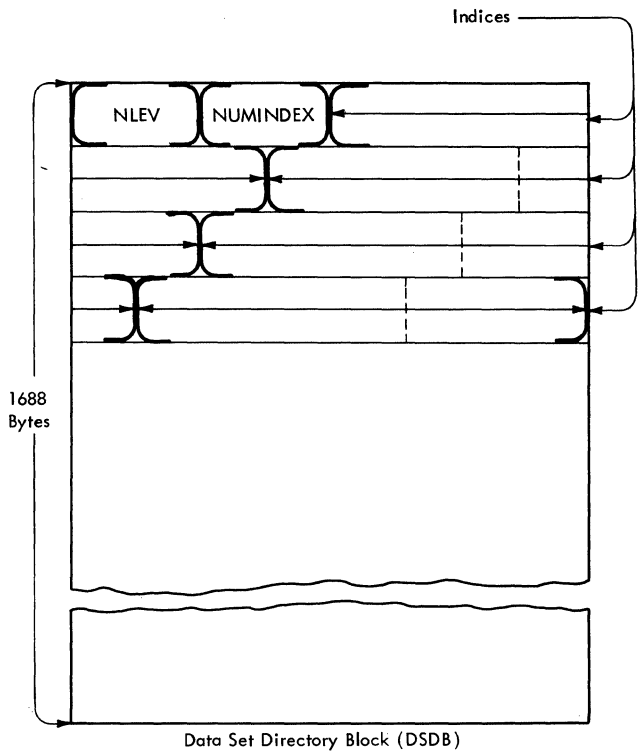
BBKCHAIN (3 Bytes) - Address of Previous Buffer Element in Chain, or Zero if this is the First Element.

BUFTTR (4 Bytes) - TTR of the Physical Block which was Last Read into the Buffer Element, or into which the Buffer Element is to be Written.

BUFREF (4 Bytes) - Address of Index Entry or Record Locator Last Found.

Figure 18. EDIT Access Method Buffers

A directory block provides the mechanism for accessing a particular record. Contained within each directory block are pointers to data blocks or other directory blocks. Associated with each data block is the highest key of the records within the block. When a particular record is requested the Access Method first searches the directory block for the first data block whose highest key is greater than or equal to the key of the desired record. When the data block is identified, the Access Method reads the record (if the data block is in storage) or reads the data block into a buffer (if the block is on the direct-access device) and then reads the record. Figure 19 describes the format of the directory block.



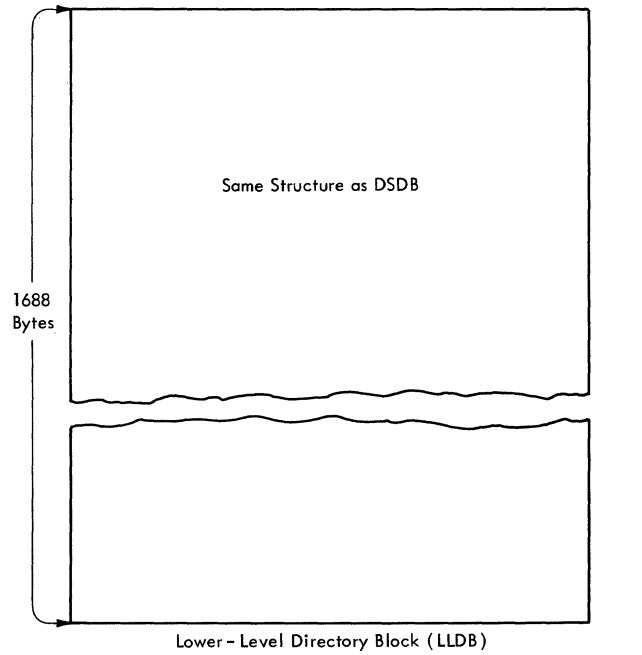
Data Set Directory Block Field Description

NLEV - (2 Bytes) Number of LLDB Levels. Zero Level Always Points to Data Blocks.

NUMINDEX - (2 Bytes) Nbr of Index Entries.

INDEX ENTRY - (7 Bytes) 4-Byte Binary Key Value and 3-Byte TTR. The 3-Byte TTR Points to a Data Block or to a Lower-Level Directory Block. The 4-Byte Key Indicates the Highest Key Value in the Data Block or the Lower-Level Directory Block.

Note: This Block is Always in the First Buffer Position in the Main Storage Assigned to the EDIT Access Method.



Lower-Level Directory Block Field Descriptions

NLEV - (2 Bytes) Level Nbr of this LLDB Zero Level Always Points to Data Blocks.

NUMINDEX - (2 Bytes) Nbr of Index Entries.

INDEX ENTRY - (7 Bytes) 4-Byte Binary Key Value and 3-Byte TTR. The 3-Byte TTR Points to a Data Block or to Another LLDB. The 4-Byte Key Indicates the Highest Key Value in the Data Block or the LLDB.

Figure 19. EDIT Access Method Directory Blocks

If the data set becomes very large, the number of pointers to data blocks may exceed the size of the Directory block. In this case the directory block contents are written into two lower-level directory blocks. This operation is called block splitting. (See Figure 20 for a description of how this operation is accomplished.) The new directory block will now point to two lower-level directory blocks, which in turn point to data blocks. If a lower-level directory block becomes full, a new lower-level directory block is created by writing half the contents of the original into the new. Lower-level directory blocks are also stored on the direct-access device.

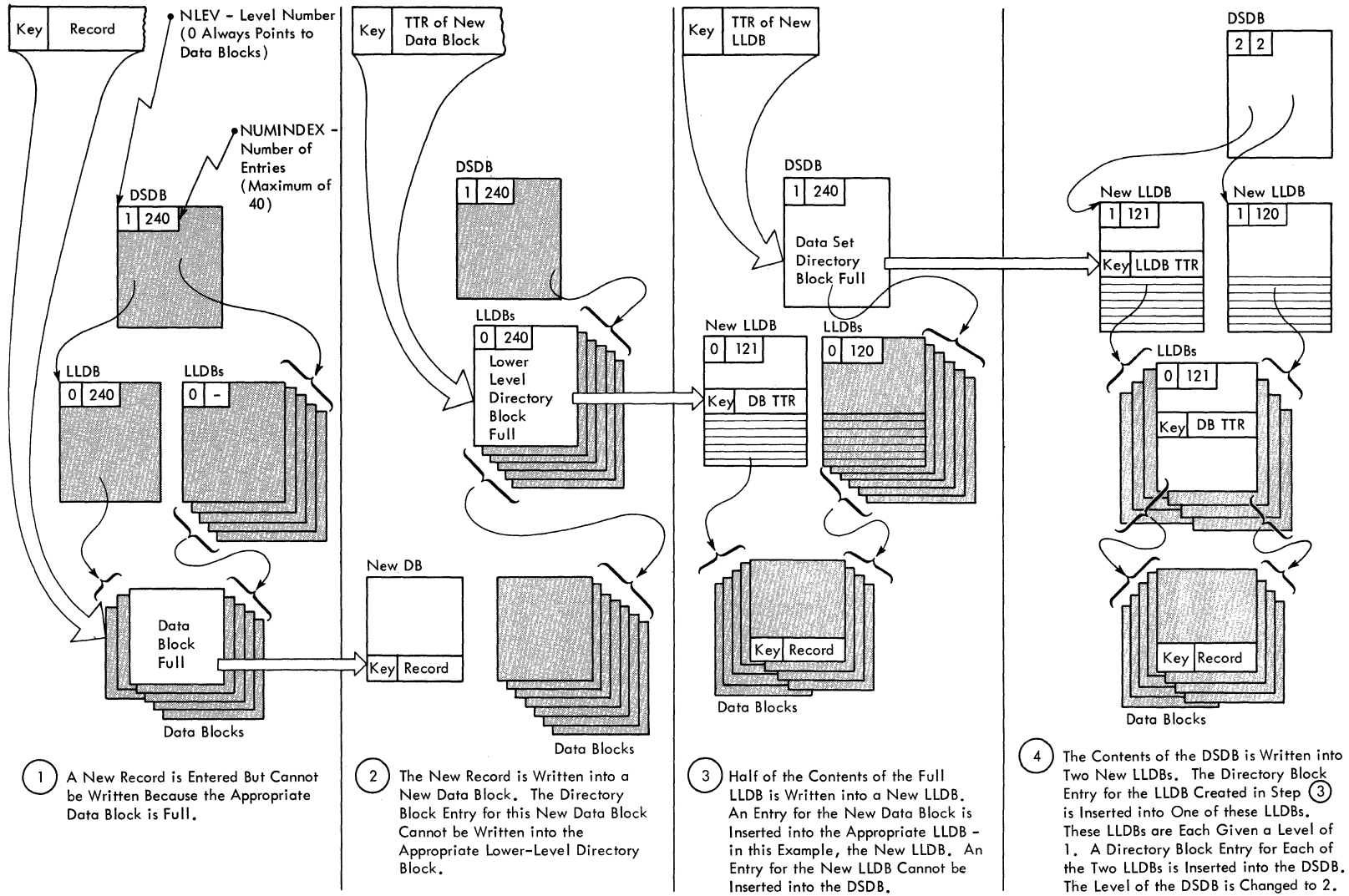


Figure 20. Block Splitting Technique

(1) A New Record is Entered But Cannot be Written Because the Appropriate Data Block is Full.

(2) The New Record is Written into a New Data Block. The Directory Block Entry for this New Data Block Cannot be Written into the Appropriate Lower-Level Directory Block.

(3) Half of the Contents of the Full LLDB is Written into a New LLDB. An Entry for the New Data Block is Inserted into the Appropriate LLDB - in this Example, the New LLDB. An Entry for the New LLDB Cannot be Inserted into the DSDB.

(4) The Contents of the DSDB is Written into Two New LLDBs. The Directory Block Entry for the LLDB Created in Step (3) is Inserted into One of these LLDBs. These LLDBs are Each Given a Level of 1. A Directory Block Entry for Each of the Two LLDBs is Inserted into the DSDB. The Level of the DSDB is Changed to 2.

The data set on direct-access which contains the data blocks (and the lower-level directory blocks, if neccessary) searched by the EDIT Access Method is called the utility data set. Each unit of storage on the device is called a block and is referred to by a TTR.

INITIALIZATION AND FINAL PROCESSING

Initialization of the EDIT Access Method consists of acquiring the resources needed by the Access Method (the utility data set, a work area, buffers) and of loading the Access Method routines into storage. Access Method initialization occurrs during the initialization phase of EDIT, and whenever an additional utility data set is required by an EDIT module.

Final processing involves freeing the resources acquired during initialization. Method of Operation Diagram 26 (foldout) describes the method of operation of the EDIT Access Method initialization (IKJEBEUI) and final processing (IKJEBEEX).

INTERFACE

The interface receives requests for Access Method functions, selects the appropriate operation (read, write, or delete), and returns results (pointers to found records, indications of successful operation, etc.) to the caller. The interface is the only agent of communication between the Access Method and the subcommand processors. Method of Operation Diagram 27 (foldout) describes the method of operation of the interface (IKJEBEUT).

READ, WRITE, AND DELETE OPERATIONS

These operations are selected by the interface to perform work for the caller of the Access Method. There are three kinds of work performed:

- Reading operations - locating a particular record or a number of records in the utility data set; moving the record to a location in storage.
- Writing operations - locating a particular line in the data set; writing a record into the line.
- Deleting operations - locating a particular record in the data set; deleting the record.

The following diagrams describe the method of operation of the read, write, and delete operations:

- Method of Operation Diagram 28. Write operation (IKJEBEWR).
- Method of Operation Diagram 30. Delete operation (IKJEBEDR).
- Method of Operation Diagram 29. Read operation (IKJEBERR).

ACCESS METHOD SERVICE ROUTINES

The Access Method service routines are used by the read, write, and delete operations:

- To handle data records - locating, deleting, moving.
- To handle the directories - searching, updating, moving (of control fields).
- To handle the blocks - assigning, reading, writing.
- To handle the buffers - assigning.
- To handle I/O - waiting.

The following diagrams (foldouts) describe the method of operation of the Access Method service routines:

- Method of Operation Diagram 31. Record locate routine (IKJEBELO).
- Method of Operation Diagram 32. Record delete routine (IKJEBEDL).
- Method of Operation Diagram 33. TTR assignment routine (IKJEBEAD).
- Method of Operation Diagram 34. Directory update routine (IKJEBEDU).
- Method of Operation Diagram 35. Write block routine (IKJEBEWB).
- Method of Operation Diagram 36. Buffer assignment routine (IKJEBEAS).
- Method of Operation Diagram 37. Directory search routine (IKJEBEDS).
- Method of Operation Diagram 38. Read block routine (IKJEBERB).
- Method of Operation Diagram 39. Wait routine (IKJEBEWA).

Section 3: Program Organization

This section describes the relationships among the modules and csects in the EDIT program. Included in this section are:

- Tables which list the various modules and csects in the EDIT program.
- Program organization diagrams which pictorialize these relationships.
- Module and csect operation tables which describe the operation of each module and csect in the EDIT program.
- Flowcharts.

All EDIT program modules are named in the form IKJEBExx, where:

IKJ is the Time Sharing Option identifier
EB indicates a data set utility
E indicates the EDIT data set utility
xx identifies the particular module of the EDIT program.

The following tables list the functions of and the relationships among the modules and csects of the EDIT program.

Table 28. Initialization Routine Program Organization

Load Module	Csect	Called By	Linkage	Function
IKJEBEIN	IKJEBEIN	TMP	ATTACH	Edit Initialization
	IKJEBIN1			Parse PCL
	IKJEBIN2			Parse PCL
	IKJEBIN3			Builds Error Messages
	IKJEBIN4			Validity Check Exit
	IKJEBIN5			Data Set Type Prompting
	IKJEBIN6			Parse PCL
	IKJEBIN7			Process Partially-Qualified Dsname
	IKJEBIN8			Process Operands

Table 29. Mainline Routines (resident)

Load Module	Csect	Called By	Linkage	Function
IKJEBEMA	IKJEBEMA	IKJEBEIN	XCTL	Controller
	IKJEBMA0			Message Text
	IKJEBEAE	Supervisor	CALL	Abnormal End Exit
	IKJEBEAT	Supervisor	CALL	Attention Exit
	IKJEBEUT	All	CALL	Access Method Interface
	IKJEBMA2	IKJEBEAE	CALL	
		IKJEBEAT	CALL	
		IKJEBEMA	CALL	
	IKJEBMA8			IBM Subcommand Table
	IKJEBMA9			User Subcommand Table
IKJEBEMS	IKJEBEMS	IKJEBEIN all	LOAD call	Message Selection
IKJEBELE	IKJEBELE	IKJEBECG	CALL	Line editing
		IKJEBECN		
		IKJEBEIP		
		IKJEBEIS		
		IKJEBELI		
		IKJEBEFI		
IKJEBEUT	IKJEBEUT	IKJEBEIN	LOAD	Utility Interface
IKJEBEAA	IKJEBEAA	IKJEBEUI	LOAD	Edit Access Method
	IKJEBEAD			Add a Block
	IKJEBEAS			Assign a Buffer
	IKJEBEDL			Record Removal
	IKJEBEDR	IKJEBEUT	CALL	Delete Record
	IKJEBEDS			Directory Search
	IKJEBEDU			Directory Update
	IKJEBELO			Locate Record
	IKJEBEMV			Move Data
	IKJEBERB			Read Block
	IKJEBERR	IKJEBEUT	CALL	Read Record
	IKJEBEWA			Wait on I/O
	IKJEBEWB			Write Block
	IKJEBEWR	IKJEBEUT	CALL	Write Record

Table 30. Service Routines (non-resident)

Load Module	Csect	Called By	Linkage	Function
IKJEBEPS	IKJEBEPS IKJEBEPD	IKJEBEIN	LOAD/CALL	Table Search Processor Data Table
IKJEBECO	IKJEBECO	IKJEBEIN IKJEBEME	LINK	Initial Copy
IKJEBEFC	IKJEBEFC	IKJEBEFO IKJEBEME IKJEBERU IKJEBESA	LINK	Final Copy
IKJEBEUI	IKJEBEUI	IKJEBECO IKJEBEIN IKJEBERE	LINK	Utility Initialization
IKJEBEEX	IKJEBEEX	IKJEBECO IKJEBEMA IKJEBERE IKJEBEME IKJEBEEN IKJEBEMR	LINK	Utility Exit
IKJEBEDA	IKJEBEDA IKJEBDAM	IKJEBEFO IKJEBEME IKJEBERU	LINK	Utility Allocation Message Text
IKJEBECI	IKJEBECI	IKJEBEHE IKJEBEFO IKJEBEME IKJEBERU	XCTL LINK LINK LINK	Command Invoker
IKJEBEMR	IKJEBEMR	IKJEBEME IKJEBERE	LINK	Basic Interface

Table 31. Message Text Loads

Load Module	Csect	Called By	Linkage	Function
IKJEBEM1	IKJEBEM1	IKJEBEMS	LOAD	Message Load 1
IKJEBEM2	IKJEBEM2	IKJEBEMS	LOAD	Message Load 2
IKJEBEM3	IKJEBEM3	IKJEBEMS	LOAD	Message Load 3
IKJEBEM4	IKJEBEM4	IKJEBEMS	LOAD	Message Load 4
IKJEBEM5	IKJEBEM5	IKJEBEMS	LOAD	Message Load 5
IKJEBEM6	IKJEBEM6	IKJEBEMS	LOAD	Message Load 6
IKJEBEM7	IKJEBEM7	IKJEBEMS	LOAD	Message Load 7

Table 32. Subcommand Processors Program Organization (Part 1 of 2)

Load Module	Csect	Called By	Linkage	Function
IKJEBEBO	IKJEBEBO	IKJEBEMA	LINK	BOTTOM
IKJEBECG	IKJEBECG	IKJEBECH	XCTL	CHANGE-String Operation
	IKJEBESE	IKJEBECG	CALL	String Search
IKJEBECH	IKJEBECH	IKJEBEMA	LINK	CHANGE-Initialization
	IKJEBCH1			Parse PCL
	IKJEBCH2			Parse PCL
	IKJEBCH3			Parse PCL
	IKJEBCH4			Parse PCL
IKJEBECN	IKJEBECN	IKJEBECH	XCTL	CHANGE-Char Count Operation
		IKJEBECG	XCTL	
IKJEBEDE	IKJEBEDE	IKJEBEMA	LINK	DELETE
	IKJEBED1			Parse PCL
IKJEBEDO	IKJEBEDO	IKJEBEMA	LINK	DOWN
	IKJEBEDL			Parse PCL
IKJEBEFI	IKJEBEFI	IKJEBEMA	LINK	FIND
	IKJEBFI0			Parse PCL
	IKJEBFI1			Parse PCL
	IKJEBESE	IKJEBEFI	CALL	
IKJEBEFO	IKJEBEFO	IKJEBEMA	LINK	FORMAT
	IKJEBEFL			Parse PCL
IKJEBEHE	IKJEBEHE	IKJEBEMA	LINK	HELP
IKJEBEIM	IKJEBEIM	IKJEBEIP	XCTL	INPUT-Input Operation
IKJEBEIP	IKJEBEIP	IKJEBEMA	LINK	INPUT-Initialization
	IKJEBXT1			Validity Check Exit
	IKJEBIP1			Parse PCL
IKJEBEIS	IKJEBEIS	IKJEBEMA	LINK	INSERT
	IKJEBIS1			Parse PCL
IKJEBELI	IKJEBELI	IKJEBEMA	LINK	Line Insert
	IKJEBLI1			Parse PCL
IKJEBELT	IKJEBELT	IKJEBEMA	LINK	LIST
	IKJEBLT1			Parse PCL
IKJEBEME	IKJEBEME	IKJEBEMA	LINK	MERGE
	IKJEBMEM			Message Text
	IKJEBME0			Parse PCL
IKJEBEPR	IKJEBEPR	IKJEBEMA	LINK	PROFILE
	IKJEBEPL			Parse PCL
IKJEBERE	IKJEBERE	IKJEBEMA	LINK	RENUM
	IKJEBRE4			Parse PCL
IKJEBERU	IKJEBERU	IKJEBEMA	LINK	RUN
	IKJEBRU0			Parse PCL

(Part 1 of 2)

Table 32. Subcommand Processors Program Organization (Part 2 of 2)

Load Module	Csect	Called By	Linkage	Function
IKJEBESA	IKJEBESA	IKJEBEMA	LINK	SAVE
	IKJEBSA1			Parse PCL
	IKJEBSA2			Parse PCL
	IKJEBSA8			Message Text
	IKJEBSA9			Issue Messages
IKJEBESC	IKJEBESC	IKJEBEMA	LINK	SCAN (Part 1)
	IKJEBSC1			Parse PCL
IKJEBESN	IKJEBESN	IKJEBESC	XCTL	SCAN (Part 2)
IKJEBETA	IKJEBETA	IKJEBEMA	LINK	TABSET
	IKJEBTA0			Parse PCL
IKJEBETO	IKJEBETO	IKJEBEMA	LINK	TOP
IKJEBEUP	IKJEBEUP	IKJEBEMA	LINK	UP
	IKJEBUP0			Parse PCL
IKJEBEVE	IKJEBEVE	IKJEBEMA	LINK	VERIFY
	IKJEBVEP			Parse PCL

The organization of the EDIT program is shown in the following figures:

- Figure 21. EDIT Initialization Program Organization
- Figure 22. EDIT Main Line Program Organization
- Figure 23. EDIT Access Method Program Organization
- Figure 24. IKJEBEAA Program Organization

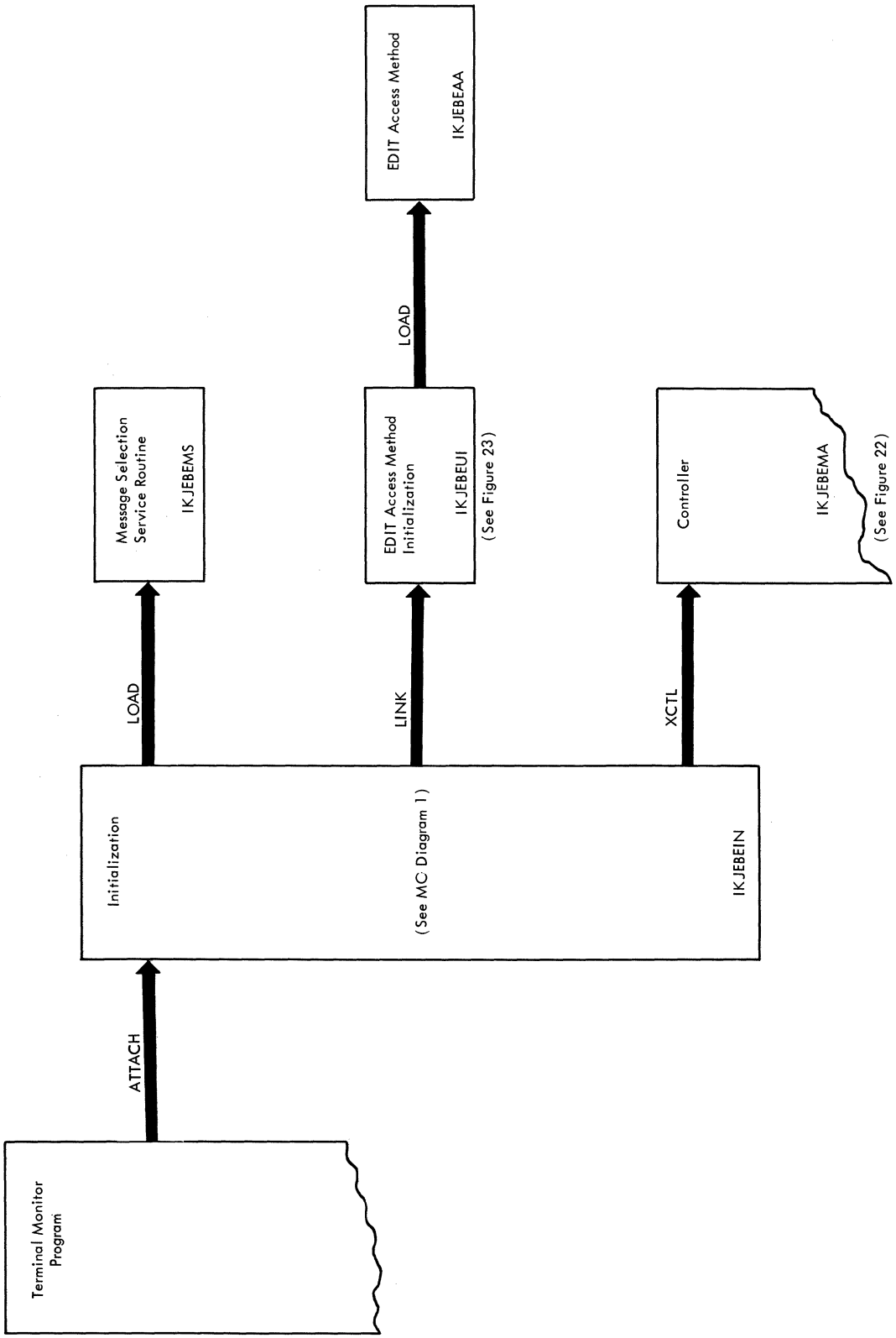


Figure 21. EDIT Initialization Program Organization

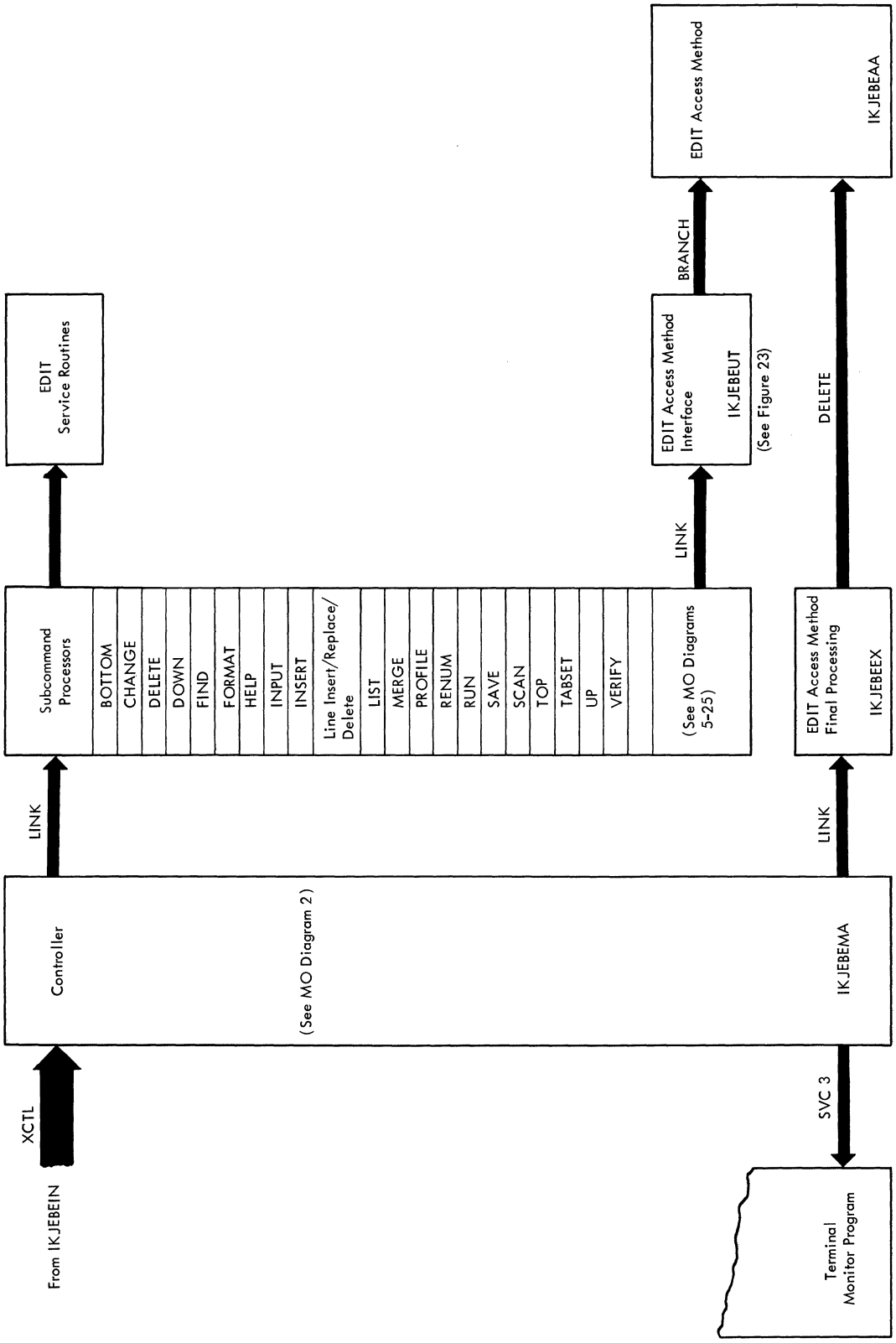


Figure 22. EDIT Main Line Program Organization

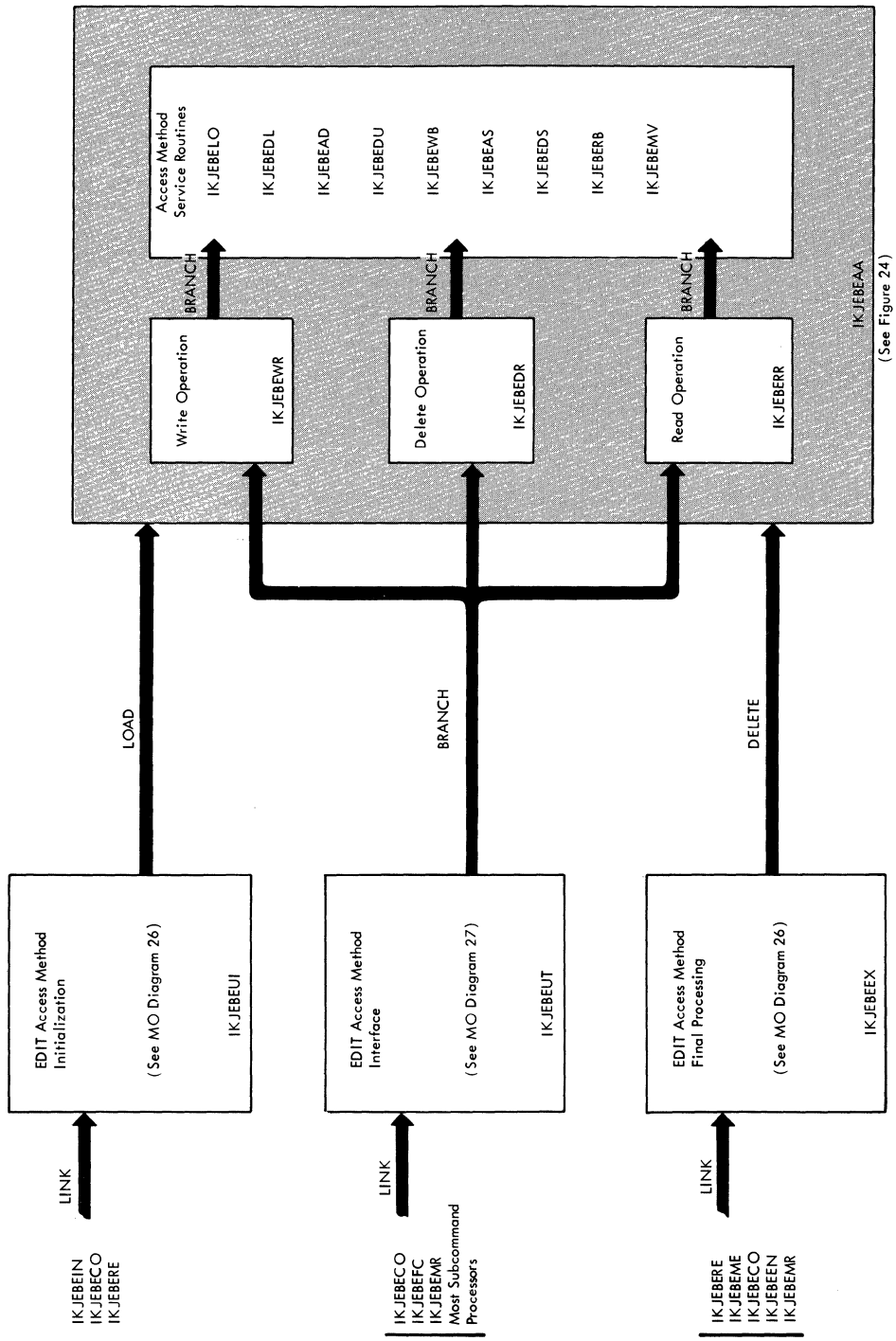


Figure 23. EDIT Access Method Program Organization

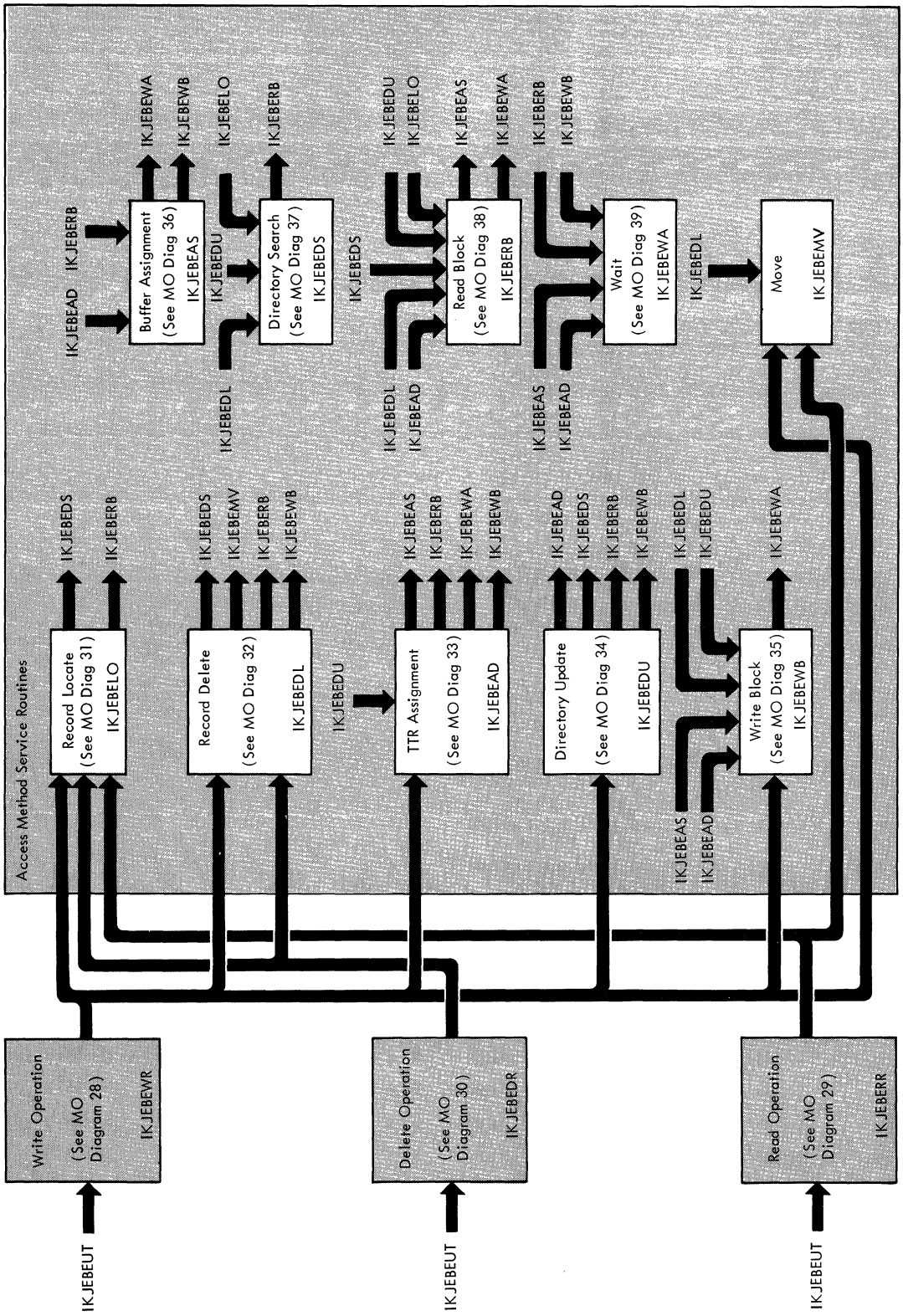


Figure 24. IKJEBEAA Program Organization

Module and Csect Operation Tables

IKJEBEAA

Module Name: IKJEBEAA

Common Name: EDIT Utility Access Method

Entry: From module IKJEBEUT at entry points IKJEBEDR, IKJEBERR, IKJEBEWR.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Load module containing the following access method csects:

IKJEBEAD - TTR assignment routine
IKJEBEAS - Buffer assignment routine
IKJEBEDL - Record delete routine
IKJEBEDR - Delete operation routine
IKJEBEDS - Directory search routine
IKJEBEDU - Directory update routine
IKJEBELO - Record locate routine
IKJEBEMV - Move record routine
IKJEBERB - Read block routine
IKJEBERR - Read operation routine
IKJEBEWA - Wait routine
IKJEBEWB - Write block routine
IKJEBEWR - Write operation routine

See Csect Operation Tables for description of the operation of these csects.

Data Areas Defined by this Module: See Csect Operation Tables.

Data Areas Updated by this Module: See Csect Operation Tables.

Routines or Modules Called: See Csect Operation Tables.

Exits: See Csect Operation Tables.

Register Usage: See Csect Operation Tables.

IKJEBEAD (CSECT OF IKJEBEAA)

Csect_Name: IKJEBEAD

Common_Name: TTR assignment routine

Entry:
Branch from csect IKJEBEDU at entry point IKJEBEAD.
Branch from csect IKJEBEWR at entry point IKJEBEAD.

Attributes:

Operation: Obtains a new physical block from direct-access device.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

DCBEBQX (in UTILWORK) - pointer to first empty block in utility data set
BUFFTTR (in buffer) -

Routines_or_Modules_Called:

IKJEBEAS - Assigns new buffer
IKJEBERB - Reads in empty block
IKJEBEWB - Formats block
IKJEBEWA - Waits on completion of I/O

Exits: Return to caller with following return codes:

0 - successful operation
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEAE (CSECT OF IKJEBEMA)

Csect_Name: IKJEBEAE

Common_Name: Abnormal end exit routine.

Entry: Asynchronously from module IGC0401C at entry point IKJEBEAE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Processes abend situations arising within the Edit Command Processor; attempts recovery from Abend situations, and if successful, allows the user to save his data set.

Data_Areas_Defined_by_this_Module:

CATEMPBF -
CASVAREA - 18 word save area in IKJEBECA.

Data_Areas_Updated_by_this_Module:

CANXTSVA - in IKJEBECA
CASVAREA - in IKJEBECA
CAVRFYSW - in IKJEBECA
CAABEND - in IKJEBECA

ECTRTCD - in ECT
ECTRCDF - in ECT

Routines_or_Modules_Called:

IKJEBMA3 - Input stack clean-up routine
IKJEBEMS - Message selection routine

Exits: Return to caller with following return codes:

0 - complete Abend processing
4 - pass control to retry routine whose address is specified in Register 0 (zero).

Register_Usage:

Entry
- All registers are saved.
- Register 0 contains an entry code.
  - For entry code 0, 4, and 8:
    Register 1 contains the address of the STAE work area.
  - For entry code 12:
    Register 1 contains the user area parameter specified on the STAE macro instruction and Register 2 contains the abend completion code.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEAS (CSECT OF IKJEBEAA)

Csect Name: IKJEBEAS

Common Name: Buffer assignment routine

Entry:
Branch from csect IKJEBEAD at entry point IKJEBEAS.
Branch from csect IKJEBERB at entry point IKJEBEAS.

Attributes:

Operation: Obtains an I/O buffer.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

Routines or Modules Called:

IKJEBEWB - Writes out contents of buffer
IKJEBEWA - Waits on completion of I/O

Exits: Return to caller with following return code:

0 - successful operation
12 - I/O error

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEAT (CSECT OF IKJEBEMA)

Csect Name: IKJEBEAT

Common Name: Attention exit routine.

Entry: Asynchronously from module IKJEAR05 at entry point IKJEBEAT.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Posts the Attention ECB when the user enters a new subcommand after striking the attention key; processes second level messages when the user enters a question mark.

Data Areas Defined by this Module:

CAATNWKA - WORK/SAVE area for IKJEBEAT (in IKJEBECA)

Data Areas Updated by this Module:

CAATNBUF - in IKJEBECA
CAATTN - in IKJEBECA
CAPTIBFR - in IKJEBECA
CAPTGTBF - in IKJEBECA

Routines or Modules Called:

IKJSCAN - SCAN service routine
IKJPTGT - PUTGET service routine
LANGPRCR - language processor or syntax checker
IKJEBMA2 - input stack clean-up routine

Exits: Return to caller.

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the Attention Exit Parameter List.

Exit
- All registers are restored.

IKJEBEBO

Module_Name: IKJEBEBO

Common_Name: BOTTOM subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEBO.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Moves current line pointer to the last line in the data set.

Data_Areas_Defined_by_this_Module:

CASCWKA - formatted for local storage
CATEMPBF - contains line to be verified
CASVAREA - save area in IKJEBECA

Data_Areas_Updated_by_this_Module:

CALNTOVF - is set to 1 to indicate a verify request
CANXTSVA - updated pointer to next available save area
CACURNUM - current line pointer

The_following_CSECTS_are_contained_in_this_load_module:

IKJEBEBO - main line routine

Routines_or_Modules_Called:

IKJEBEUT - Access method interface routine
IKJEBEMS - Message selection routine

Exits: Return to caller with following return codes.

0 - successful operation
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBECG

Module_Name: IKJEBECG

Common_Name: CHANGE subcommand processor (String Operations)

Entry: XCTL from module IKJEBECH to entry point IKJEBECG.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Performs all text replacement operation.

Data_Areas_Defined_by_this_Module:

CASCWOKA - Variables, parameter lists for IKJEBELE and IKJEBESE
CABERPL - Buffers for records to be changed
CATEMPBF - Buffers to build changed record
CASVAREA - Save area

Data_Areas_Updated_by_this_Module:

CANXTSVA - Pointer to next available save area
CACURNUM - Current line pointer
CASYNBER - Pointer to record to be passed to LANGPRCR
CASYNOD2 - Entry code to LANGPRCR
CASYNMS1 - Pointer to first-level LANGPRCR message
CASYNMS2 - Pointer to second-level LANGPRCR message
CATPUTVF - Verify message switch

Routines_or_Modules_Called:

IKJEBEMS - message selection routine
IKJEBEUT - access method interface routine
IKJEBELE - line edit routine
LANGPRCR - Language processor or syntax checker
IKJEBESE - String search routine

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set
XCTL to IKJEBECN

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- Upon Return to IKJEBEMA all registers except 15 are restored.
- Register 15 contains a return code. Upon XCTL to IKJEBECN all registers except register 1 are restored.

IKJEBECH

Module_Name: IKJEBECH

Common_Name: CHANGE subcommand processor (initialization phase).

Entry: LINK from module IKJEBEMA at entry point IKJEBECH.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Checks validity of all CHANGE subcommand operands and reads in first record.

Data_Areas_Defined_by_this_Module:

CASCWKA (in IKJEBECA) - Variables
CABFRPL (in IKJEBECA) - Buffer for record to be changed
CASVAREA (in IKJEBECA) - Save area

Data_Areas_Updated_by_this_Module:
CANXTSVA (in IKJEBECA) - Pointer to next Save area
CASRPLST (in IKJEBECA) - Parameters for PUTLINE, GETLINE and PARSE

The_Following_Csects_Are_Contained_in_This_Load_Module:

IKJEBECH - Main line routine
IKJEBCH1 - Main PCL
IKJEBCH2 - Prompt PCL
IKJEBCH3 - Character count PCL
IKJEBCH4 - Line count PCL

Routines_or_Modules_Called:

IKJPARS - Validates CHANGE subcommand operands
IKJEBEMS - Message output
IKJPUTL - PUTLINE service routine; writes out messages to the terminal
IKJGETL - GETLINE service routine; obtains terminal input
IKJEBEUT - Access method interface routine

Exits: Return to caller with following return codes:

XCTL to IKJEBECG or IKJEBECN
 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers except 13 are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.
- Upon XCTL all registers except 1 are restored.

IKJEBECI

Module_Name: IKJEBECI

Common_Name: Command invoker

Entry:
LINK from module IKJEBERU at entry point IKJEBECI.
XCTL from module IKJEBEHE at entry point IKJEBECI.
LINK from module IKJEBEFO at entry point IKJEBECI.
LINK from module IKJEBEME at entry point IKJEBECI.
Called asynchronously from supervisor at entry points: STAEEXIT, STAIEXIT, STAERTRY.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Attaches the TSO command processors for the following EDIT subcommands: RUN, MERGE, FORMAT, and HELP.

Data_Areas_Defined_by_this_Module:

CASRWKA (in IKJEBECA)
CATEMPBF (in IKJEBECA)
CASVAREA (in IKJEBECA)

Data_Areas_Updated_by_this_Module:

CAATTN (in IKJEBECA)
CAMODMSG (in IKJEBECA)
CANXTSVA (in IKJEBECA)
CAPTIBFR (in IKJEBECA)
CASRPLST (in IKJEBECA)
ECTMSGF (in ECT)
ECTRCDF (in ECT)
ECTRTCD (in ECT)
CACFLAG2
CATMPLST

The_following_Csects_are_contained_in_this_load_module:

IKJEBECI
IKJEBCIM

Routines_or_Modules_Called:

IKJEBEMS - Message output.
IKJPTGT - PUTGET service routine; prompts user and obtains responses. Obtains command placed in input stack by RUN command.
IKJDAIR - Mark DSE entries not in use.
IKJSTCK - Add end-of-data marker to input stack. Delete procedure elements from input stack when error is encountered.
IKJSCAN - Check for valid command name.

Exits: Normal - return to caller with following return codes.

0 - successful operation
4 - unsuccessful MERGE or GOFORT operation
Asynchronous - return to caller from STAIEXIT with return code of 16.
- return to caller from STAEEXIT with following return codes.
0 - no retry
4 - retry
8 - unsuccessful completion

Register Usage:

Normal entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Asynchronous entry
- All registers are saved.

Normal exit
- All registers except 15 are restored.
- Register 15 contains a return code.

Asynchronous exit
- All registers except 15 are restored.
- Register 15 contains a return code.
- If retry is specified, register 0 contains the address of a retry routine.

IKJEBECN

Module_Name: IKJEBECN

Common_Name: CHANGE subcommand processor (character count operations)

Entry:
Via XCTL from module IKJEBECH at entry point IKJEBECN.
Via XCTL from module IKJEBECG at entry point IKJEBECN.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Displays specified characters of record at the terminal; obtains user's modifications to record; writes modifications into data set.

Data_Areas_Defined_by_this_Module:

CASCWKA (in IKJEBECA)
CABFRPL (in IKJEBECA)
CATEMPBF (in IKJEBECA)
CASVAREA

Data_Areas_Updated_by_this_Module:

CASRPLST (in IKJEBECA)
CAAXTSVA (in IKJEBECA)
CACURNUM (in IKJEBECA)
CASYNBFR (in IKJEBECA)
CASYNCD2 (in IKJEBECA)
CASYNMS1 (in IKJEBECA)
CASYNMS2 (in IKJEBECA)

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBECN

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJEBEUT - Access Method interface routine
IKJEPUTL - PUTLINE service routine
IKJEBELE - Line edit service routine
IKJEGETL - GETLINE service routine
LANGPRCR - Syntax checker

Exits: Return to caller with following return codes.

0 - successful operation
8 - unsuccessful completion
12 - I/O error in data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBECO

Module_Name: IKJEBECO

Common_Name: Initial copy routine

Entry:
Via LINK from module IKJEBEIN at entry point IKJEBECO.
Via LINK from module IKJEBEME at entry point IKJEBECO.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Reads records from an old data set; converts record format from old data set format to EDIT Access Method format; writes the newly-formatted records into the utility data set.

Data_Areas_Defined_by_this_Module:

CASRWKA -
CATEMPBF -
CASVAREA -

Data_Areas_Updated_by_this_Module:

CANONUM	CAINCRE	CANXTSVA
CADSMODS	CASTNUM	
CACURNUM	CAPTCDCB	
CADSNOFF	CADSNPTR	
CADSNOF2	CADSNPT2	

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBECO -

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJEBEUI - Access method initialization routine
IKJEBEUT - Access method interface routine
IKJEBEEX - Access method final processing routine

Exits: Return to caller with following return codes.

0 - successful operation
4 - data set empty
8 - unsuccessful operation; error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDA

Module_Name: IKJEBEDA

Common_Name: Data set allocation/free routine

Entry:
from module IKJEBEME at entry point IKJEBEDA.
from module IKJEBERU at entry point IKJEBEDA.
from module IKJEBEFO at entry point IKJEBEDA.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Generates a dsname and invokes IKJDAIR to allocate a data set by that name; frees a data set by invoking IKJDAIR.

Data_Areas_Defined_by_this_Module:

CASRWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CANXTSVA -
CASADDN -
CASADISP -
CASADSN -
CASADSNL -
CASAUNCG -
CASRPLST -

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEDA - main line routine
IKJEBDAM - message control section

Routines_or_Modules_Called:

IKJDAIR - Allocates and frees data sets.
IKJEBEMS - Message output

Exits: Return to caller with following return codes.

0 - successful operation
4 - unsuccessful operation

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDE

Module_Name: IKJEBEDE

Common_Name: DELETE subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEDE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Deletes specified lines from the data set and resets the current line pointer if the delete operation was successful.

Data_Areas_Defined_by_this_Module:

CASCWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CALNTOVF - in IKJEBECA
CANXTSVA - in IKJEBECA
CASYNBFR - in IKJEBECA
CASYNCD2 - in IKJEBECA
CASRPLST - in IKJEBECA

CACURNUM

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEDE - main line routine
IKJEBDE1 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates DELETE subcommand operands
IKJEBEUT - Access method interface routine
IKJEBEMS - Message output
LANGPRCR - Language processor or syntax checker

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDL (CSECT OF IKJEBEAA)

Csect_Name: IKJEBEDL

Common_Name: Record delete routine

Entry:
Branch from csect IKJEBEDR at entry point IKJEBEDL.
Branch from csect IKJEBEWR at entry point IKJEBEDL.

Attributes:

Operation: Deletes record locators from data blocks.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

BUFSTATS (in buffer)
BUFREF (in buffer)
NUMREC (in data block)
DATASTRT (in data block)
RECVSP (in data block)
Locator (in data block)
NUMINDEX (in directory block)

Routines_or_Modules_Called:

IKJEBEDS - Searches directory
IKJEBERB - Reads directory block
IKJEBEMV - Moves record locators
IKJEBEWB - Writes out block

Exits: Return to caller with following return codes:

0 - successful operation
8 - data set empty
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDO

Module Name: IKJEBEDO

Common Name: DOWN subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEDO.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Moves current line pointer toward end of data set.

Data Areas Defined by this Module:

CASCWKA -
CATEMPBF -
CASVAREA -

Data Areas Updated by this Module:

CACURNUM
CALNTOVF
CANXTSVA -
CASRPLST -

The Following Csects are Contained in this Load Module:

IKJEBEDO - main line routine
IKJEBDOO - PCL for IKJPARS

Routines or Modules Called:

IKJPARS - Validates DOWN subcommand operands
IKJEBEUT - Access method interface routine
IKJEBEMS - Message output

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDR (CSECT OF IKJEBEAA)

Csect Name: IKJEBEDR

Common Name: Delete operation

Entry: from module IKJEBEUT at entry point IKJEBEDR.

Attributes:

Operation: Locates and deletes a particular record.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

PRIMEKEY (in UTILWORK) -
SAVEKEY (in UTILWORK) -

Routines or Modules Called:

IKJEBELO - Locates a particular record by key.
IKJEBEDL - Deletes a particular record by key.

Exits: Return to caller with following return codes:

 0 - successful operation
 4 - record could not be found
12 - I/O error

Register Usage:

Entry
- All registers are saved
- Register 0 contains the address of an optional new key.
- Register 1 contains the address of UTILWORK.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDS (CSECT OF IKJEBEAA)

Csect Name: IKJEBEDS

Common Name: Directory search routine

Entry:
Branch from csect IKJEBEDL at entry point IKJEBEDS.
Branch from csect IKJEBEDU at entry point IKJEBEDS.
Branch from csect IKJEBELO at entry point IKJEBEDS.

Attributes:

Operation: Locates a particular data block by key and TTR address.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

BUFREF (in buffer) -

Routines or Modules Called:

IKJEBERB - Reads next directory block level.

Exits: Return to caller with following return codes:

0 - successful operation
4 - data block could not be found
8 - data set is empty
12 - I/O error

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEDU (CSECT OF IKJEBEAA)

Csect Name: IKJEBEDU

Common Name: Directory update routine

Entry: Branch from csect IKJEBEWR at entry point IKJEBEDU.

Attributes:

Operation: Adds new block key/TTR combination to a particular directory block.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

BUFSTATS (in buffer)
BUFREF (in buffer)
DCBNLEV (in UTILWORK)
NUMINDEX (in directory block)
NLEV (in directory block)

Routines or Modules Called:

IKJEBEAD - Assigns a new block.
IKJEBEDS - Searches directory for block by key.
IKJEBERB - Reads into storage the next higher directory block level.
IKJEBEWB - Writes updated block.

Exits: Return to caller with following return codes:

 0 - successful operation
12 - I/O error

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a three-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEEN

Module_Name: IKJEBEEN

Common_Name: End Subcommand processor.

Entry: Link from module IKJEBEMA at entry point IKJEBEEN when an End subcommand is entered. Link from module IKJEBEMA at entrypoint IKJEBEXT when a terminal error condition encountered.

Attributes: Refreshable, non-privileged, read-only, enable.

Operation: IKJEBEEN determines if the user has saved his data set; if not, it invokes IKJEBESA. IKJEBEEN terminates Edit by invoking IKJEBEEN to unallocate the utility data set, invoking IKJEBESC to perform final entry processing for syntax checkers, and cancels the attention and abnormal end exits.

Data_Areas_Defined_by_this_Module:

CAMAWKA
CATEMPBF
CABFRPL
CASVAREA

Data_Areas_Updated_by_this_Module:

CAATTNTS
CAENDSC
CANXTSVA
CAOPERND
CAPTGTBF
CAPTIBFR
CASRPLST

The_Following_Csects_are_contained_in_this_Load_Module:

IKJEBEEN - mainline routine
IKJEBENO - message control section

Routines_are_modules_called:

IKJEBEMS - message selection routine
IKJPTGT - Putget service routine
IKJEBESA - Save subcommand processor
IKJEBEEX - Access method final processing routine
IKJEBESC - Scan subcommand processor.

Exits: Return to caller with following return codes:

0 - A subcommand other than End or SAVE was entered in response to prompting message.
12 - End processing complete.

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEEX

Module_Name: IKJEBEEX

Common_Name: EDIT Access Method final processing routine.

Entry: Via link from modules IKJEBECO, IKJEBEEN, IKJEBEME, IKJEBEMR and IKJEBERE at entry point IKJEBEEX.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Performs the final processing on the utility data set and deletes the Access Method routines.

Data_Areas_Defined_by_this_Module:

TEMPAREA in UTILWORK
CASCWKA in IKJEBECA

Data_Areas_Updated_or_Freed_by_this_Module:

CASRWKA - in IKJEBECA
Utility Work area (UTILWORK)

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEEX - Main line routine.

Routines_or_Modules_Called:

IKJDAIR - Dynamic Allocation Interface Routine.
IKJEBEMS - Message Selection Routine.

Exits: Return to caller.

Register_Usage:

Entry
- All registers are saved.
- Register 0 contains the address of UTILWORK.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers are restored.

IKJEBEFC

Module_Name: IKJEBEFC

Common_Name: Final copy routine

Entry:
LINK from module IKJEBEFO at entry point IKJEBEFC.
LINK from module IKJEBEME at entry point IKJEBEFC.
LINK from module IKJEBESA at entry point IKJEBEFC.
LINK from module IKJEBERU at entry point IKJEBEFC.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Reads records from the utility data set; converts record format from EDIT Access Method to QSAM; writes the newly formatted records into a QSAM data set.

Data_Areas_Defined_by_this_Module:

CATEMPBF - in IKJEBECA
CASRWKA - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CADSNOFF - CADSNOF2 -
CADSNPTR - CADSNPT2 -
CANXTSVA -

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEFC - Mainline routine

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJEBEUT - Access method interface routine

Exits: Return to caller with following return codes.

0 - successful operation
4 - empty utility data set
8 - QSAM I/O error
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEFI

Module_Name: IKJEBEFI

Common_Name: FIND subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEFI.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Updates the current line pointer if successful.

Data_Areas_Defined_by_this_Module:

CASCWKA - in IKJEBECA
CABFRPL - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CAFIBFR - in IKJEBECA
CAFINDIS - in IKJEBECA
CALNTOVF - in IKJEBECA
CANXTSVA - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEFI - Main line routine
IKJEBFI0 - PCL for IKJPARS, string not required case.
IKJEBFI1 - PCL for IKJPARS, string required case.
IKJEBESE - String search service routine.

Routines_or_Modules_Called:

IKJPARS - Validates FIND subcommand operands
IKJEBEUT - Access method interface routine
IKJEBEMS - Message output
IKJEBELE - Line edit routine

Exits: Return to caller with following return codes.

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEFO

Module Name: IKJEBEFO

Common Name: FORMAT subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEFO.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Builds a model FORMAT command and passes control to module IKJEBECI to invoke the system FORMAT command.

Data Areas Defined by this Module:

CASCWKA in IKJEBECA
CABFRPL in IKJEBECA
CASVAREA in IKJEBECA

Data Areas Updated by this Module:

CASAFNCP - in IKJEBECA
CANXTSVA - in IKJEBECA
CASAFLAG - in IKJEBECA
CASADSN - in IKJEBECA
CASRPLST - in IKJEBECA

The Following Csects are Contained in this Load Module:

IKJEBEFO - Main line routine
IKJEBFOO - PCL for IKJPARS

Routines or Modules Called:

IKJEBEDA - Data set allocation/unallocation routine.
IKJEBEUT - Access method interface routine
IKJEBEMS - Message output
IKJEBEFC - Copies utility data set into QSAM data set format.
IKJEBECI - Invokes the system FORMAT command.
IKJPARS - Validates the FORMAT subcommand operands.

Exits: Return to caller with following return codes:

 0 - successful operation or empty data set
 8 - unsuccessful completion
12 - I/O error in utility data set

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEHE

Module Name: IKJEBEHE

Common Name: HELP/PROFILE subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEHE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Resets the offset field of the subcommand buffer to zero. Passes control to IKJEBECI, which invokes the appropriate system command processor.

Data Areas Defined by this Module:

CASCWKA - in IKJEBECA
CASVAREA - in IKJEBECA

Data Areas Updated by this Module:

CANXTSVA - in IKJEBECA.
Offset field of subcommand buffer as pointed to by CAPTIBFR.

The Following Csects are Contained in this Load Module:

IKJEBEHE - Main line routine

Routines or Modules Called: None

Exits: XCTL to IKJEBECI (Command Invoker) which invokes the system HELP or PROFILE command.

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- Registers 2 through 12 are restored.
- Register 1 contains the address of IKJEBECA.
- Register 15 contains the address of the parameter list for XCTL.

IKJEBEIA

Module_Name: IKJEBEIA

Common_Name: Initialization Message Processing Routine

Entry: Link from module IKJEBEIN at Entry point IKJEBEIA.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Selects the appropriate message based upon an index passed by the caller. Builds all necessary insertion records for the message.

Data_Areas_Defined_by_the_Module: A work area supplied by the calling routine.

CASVAREA

Data_Areas_Updated_by_this_Module:

CADSNLEN
CADSNOFF
CADSNPTR
CADXTSVA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEIA

Routines_or_Modules_Called:

IKJEBEMS - message section routine

Exits: Return to caller with the following return codes.

```
 0 - successful operation
 4 - attention interrupt
 8 - "nowait" specified, function not complete
12 - invalid parameters
16 - conditional getmain failure
```

Register_usage:

Entry
- All registers are saved.
- Register 1 contains the address of a six word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEIM

Module_Name: IKJEBEIM

Common_Name: Input subcommand processor, second load.

Entry: Via XCTL from module IKJEBEIP at entry point IKJEBEIM.

Attributes: Refreshable, non-privileged, read-only, enable.

Operation: Obtains lines of Data from the user, prepares input data for Edit, causes the input data to be placed into the utility data set.

Data_Areas_Defined_by_this_Module:

CASCWKA
CATEMPBF
CABFRPL
CASVAREA

Data_Areas_Updated_by_this_Module:

CACURNUM
CAMIMFLG
CAIMIR
CAIMLINC
CAIMLLNO
CANXTSVA
CAIMSC
CAMODMSG
CASYNBFR
CASYNCD2
CASYNECD
CASYNIS
CASTNUM
CANXTREC
CASRPLST

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEIM

Routines_or_Modules_Called:

IKJEBEUT - access method interface routine
IKJEBELE - line edit service routine
IKJEBEMS - message selection routine
LANGPRCR - language processor or syntax checker

Exits: Return to caller with the following return codes.

 0 - successful operation
12 - I/O error on utility data set
16 - return code 20 received from LANGPRCR

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEIN

Module_Name: IKJEBEIN

Common_Name: Initialization routine

Entry: Via ATTACH from Terminal Monitor Program at entry point IKJEBEIN.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Obtains the Edit Communication area; validates operands entered on the Edit Command; initializes and/or defaults values in the communication area.

Data_Areas_Defined_by_this_Module:

See operation tables for csects of IKJEBEIN.

Data_Areas_Updated_by_this_Module:

See operation tables for csects of IKJEBEIN.

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEIN - Allocates Edit data set
IKJEBIN1 - PCL for Edit command
IKJEBIN2 - PCL for NEW/OLD prompt
IKJEBIN3 - invokes message routine IKJEBEIA
IKJEBIN4 - IKJPARS validity check exit
IKJEBIN5 - prompt routine for data set type
IKJEBIN6 - PCL for Data Set Type prompt
IKJEBIN7 - routine isued to build a fully qualified data set name
IKJEBIN8 - validates command operands

Routines_or_Modules_Called:

See operation tables for csects of IKJEBEIN.

Exits:

See operation tables for csects of IKJEBEIN.

Register_Usage:

See operation tables for csects of IKJEBEIN

IKJEBEIP

Module Name: IKJEBEIP

Common Name: INPUT subcommand processor

Entry: Via LINK from module IKJEBEMA at entry point IKJEBEIP.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Issues the INPUT mode message; determines if entry was from an INPUT subcommand, a null line in Edit mode or from the INSERT subcommand; completes initialization for INPUT processing.

Data Areas Defined by this Module:

CASCWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CABFRPL - in IKJEBECA
CASVAREA - in IKJEBECA

Data Areas Updated by this Module:

CAFREEDL - in IKJEBECA
CAIMCIN - in IKJEBECA
CAIMFLG - in IKJEBECA
CAIMIR - in IKJEBECA
CAIMLINC - in IKJEBECA
CANXTSVA - in IKJEBECA
CAIMLLNO - in IKJEBECA
CAIMPT - in IKJEBECA
CAIMSC - in IKJEBECA
CAIMSFPT - in IKJEBECA
CAINSAVE - in IKJEBECA
CAMODMSG - in IKJEBECA
CANXTREC - in IKJEBECA
CAPTPRSD - in IKJEBECA
CASTNUM - in IKJEBECA
CASRPLST - in IKJEBECA

The Following Csects are Contained in this Load Module:

IKJEBEIP - main line routine
IKJEBXT1 - IKJPARS validity check exit
IKJEBIP1 - PCL for IKJPARS

Routines or Modules Called:

IKJPARS - Validates INPUT subcommand operands
IKJEBEUT - Access method interface routine
IKJEBEMS - Message output syntax checkers

Exits: Return to caller with following return codes:

Normal: XCTL to module IKJEBEIM with a pointer to IKJEBECA in Register 1.

Error: Return to caller with the following return codes.
0 - successful operation
8 - unsuccessful completion
12 - I/O error on utility data set
XCTL to IKJEBEIM

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit (Normal)
- Register 2-12 are restored.
- Register 1 contains the address of IKJEBECA.
- Register 15 contains XCTL parameter list.

Exit (Error)
- All register except 15 are restored.
- Register 15 contains a return code.

IKJEBEIS

Module_Name: IKJEBEIS

Common_Name: INSERT subcommand processor.

Entry: LINK from module IKJEBEMA at entry point IKJEBEIS.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Inserts a line of input following the position of the current line pointer or causes the input mode to be entered.

Data_Areas_Defined_by_this_Module:

CASCWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CABFRPL - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CAFREEDL - in IKJEBECA
CAIMIS - in IKJEBECA
CAPTPRSD - in IKJEBECA
CASYNBFR - in IKJEBECA
CASYNCD2 - in IKJEBECA
CASYNECD - in IKJEBECA
CASYNIS - in IKJEBECA
CASRPLST - in IKJEBECA
CANXTSVA - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEIS - main line routine
IKJEBIS1 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates INSERT subcommand operands
LANGPRCR
IKJEBEMS - Message output
IKJEBEUT - Access method interface routine
IKJEBELE - Edits inserted text for tabs and converts lower to upper case

Exits: Returns to caller with following return codes:

 0 - successful operation
 4 - invoke input to process insert request
 8 - unsuccessful completion
12 - I/O error in utility data set
16 - return code of 20 from LANGPRCR

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBELE

Module_Name: IKJEBELE

Common_Name: Line editing routine.

Entry:
CALL from module IKJEBEIP at entry point IKJEBELE.
CALL from module IKJEBELI at entry point IKJEBELE.
CALL from module IKJEBEIS at entry point IKJEBELE.
CALL from module IKJEBECH at entry point IKJEBELE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Converts lower case characters to upper case; removes tabulation characters and shifts text lines according to the positions of the tabs; formats lines according to the data set attributes.

Data_Areas_Defined_by_this_Module:

CASRWKA (in IKJEBECA) -
CATEMPBF (in IKJEBECA) -

Data_Areas_Updated_by_this_Module:

CANXTSVA (in IKJEBECA) -

Routines_or_Modules_Called: None.

Exits: Return to caller with following return codes.

0 - no line overflow
4 - line overflow
8 - GETMAIN failure

Register_Usage:

Entry
- All registers except 13 are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.
- First word of parameter list contains pointer to overflow text, if any.

IKJEBELI

Module_Name: IKJEBELI

Common_Name: Line-insert/replace/delete subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBLI1.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Inserts a new line in a data set, deletes an existing line, or replaces an existing line. Called whenever a number or * is the first character of an input line while in edit mode.

Data_Areas_Defined_by_this_Module:

CASCWKA in IKJEBECA
CATEMPBF in IKJEBECA
CABFRPL in IKJEBECA
CASVAREA in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CAFREEDL - in IKJEBECA
CALNTOVF - in IKJEBECA
Subcommand buffer
CAPTPRSD - in IKJEBECA
CASYNBFR - in IKJEBECA
CASYNCD2 - in IKJEBECA
CASYNECD - in IKJEBECA
CASRPLST - in IKJEBECA
CANXTSVA - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBELI - main line routine
IKJEBLI1 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJEBEUT - Access method interface routine
IKJEBELE - Edits input line for tabs and upper case characters
IKJPARS - Validates line-insert/delete/replace subcommand operands
IKJEBEMS - Message output
LANGPRCR

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set
16 - Return code of 20 from LANGPRCR

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBELO (CSECT OF IKJEBEAA)

Csect_Name: IKJEBELO

Common_Name: Record locate routine

Entry:
Branch from csect IKJEBEDR at entry point IKJEBELO.
Branch from csect IKJEBERR at entry point IKJEBELO.
Branch from csect IKJEBEWR at entry point IKJEBELO.

Attributes:

Operation: Finds a particular record by key.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

PRIMEKEY (in UTILWORK)
BUFREF (in buffer)

Routines_or_Modules_Called:

IKJEBEDS - Searches directory for data block containing a particular key.
IKJEBERB - Reads data block into storage.

Exits: Return to caller with following return codes:

0 - successful operation
4 - record could not be found
8 - data set is empty
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBELT

Module_Name: IKJEBELT

Common_Name: LIST subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBELT.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Writes out one or more lines from the utility data set.

Data_Areas_Defined_by_this_Module:

CABFRPL in IKJEBECA
CASCWKA in IKJEBECA
CASVAREA in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CANXTSVA - in IKJEBECA
CASRPLST - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBELT - main line routine
IKJEBLT1 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates LIST subcommand operands
IKJEBEUT - Access method interface routine
IKJPUTL - PUTLINE service routine; writes out messages to the terminal
IKJEBEMS - Message output

Exits: Return to caller with following return codes:

0 - successful operation
8 - unsuccessful completion
12 - I/O error

Register_Usage:

Entry
- All registers except 13 are saved.
- Register 1 contains the address of IKJEBECA.
- Register 13 contains the address of a save area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEMA

Module_Name: IKJEBEMA

Common_Name: Edit controller

Entry: Via an XCTL from module IKJEBEIN to entry point IKJEBEMA.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Load module containing the following control sections:

Data_Areas_Defined_by_this_Module: See operation tables for csects of IKJEBEMA.

Data_Areas_Updated_by_this_Module: See Operation table for csects of IKJEBEMA.

The_Following_Csects_are_Contained_in_this_Load_module:

IKJEBEMA - Main line control routine
IKJEBMA1 - Message processing routine
IKJEBMA2 - buffer and stack clean-up routine
IKJEBMA8 - IBM subcommand table
IKJEBMA9 - USER subcommand table
IKJEBEAE - abnormal end exit
IKJEBEAT - attention exit
IKJEBEUT - access method interface routine

See CSECT operation tables for description of the operation of these csects.

Routines_or_Modules_Called: See operation tables for csects of IKJEBEMA.

Exits: See operation tables for csects of IKJEBEMA.

Register_Usage: See operation tables for csects of IKJEBEMA.

IKJEBEMA (CSECT OF IKJEBEMA)

Module Name: IKJEBEMA

Common Name: Edit main line control routine.

Entry: XCTL from module IKJEBEIN at entry point IKJEBEMA.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Obtains EDIT subcommands from the terminal user; selects the mode (edit or input) of EDIT operation; invokes the subcommand processors; terminates the EDIT function when unrecoverable errors occur; performs the END subcommand function.

Data Areas Defined by this Module:

CAMAWKA - in IKJEBECA
CABFRPL - in IKJEBECA
CASVAREA - in IKJEBECA

Data Areas Updated by this Module:
CAATTNIS - in IKJEBECA
CAIMSC - in IKJEBECA
CAINITSC - in IKJEBECA
CALNTOVF - in IKJEBECA
CAMODMSG - in IKJEBECA
CANXTSVA - in IKJEBECA
CAOPERND - in IKJEBECA
CAPICHK - in IKJEBECA
CAPTGTBF - in IKJEBECA
CAPTIBFR - in IKJEBECA
CAPTLE - in IKJEBECA
CAPTRTRY - in IKJEBECA
CAPTSCMD - in IKJEBECA
CAPTUT - in IKJEBECA
CASCMDLN - in IKJEBECA
CASCRC20 - in IKJEBECA
CASTAEPL - in IKJEBECA
CASTAXPL - in IKJEBECA
CASRPLST - in IKJEBECA
CATPUTVF - in IKJEBECA
ECTMSGF in ECT
ECTSCMD in ECT
IKJEBECA - EDIT Communication Area

Routines or Modules Called:

IKJEBEMS - Message output
IKJEBEIP - INPUT subcommand processor
IKJEBELI - Line insert/replace/delete subcommand processor
IKJEBEEX - Access method final processing routine
IKJEBEBO - BOTTOM subcommand processor
IKJSCAN - SCAN service routine
IKJPTGT - PUTGET service routine; prompts user and obtains responses.
IKJEBECH - CHANGE subcommand processor
IKJEBEFO - FORMAT subcommand processor
IKJEBELT - LIST subcommand processor
IKJEBERE - RENUM subcommand processor
IKJEBEDE - DELETE subcommand processor
IKJEBEHE - HELP subcommand processor
IKJEBEME - MERGE subcommand processor
IKJEBERU - RUN subcommand processor
IKJEBEDO - DOWN subcommand processor
IKJEBEIS - INSERT subcommand processor

IKJEBEPR - PROFILE subcommand processor
IKJEBESA - SAVE subcommand processor
IKJEBEFI - FIND subcommand processor
IKJEBESC - SCAN subcommand processor (first load)
IKJEBETA - TABSET subcommand processor
IKJEBETO - TOP subcommand processor
IKJEBEUP - UP subcommand processor
IKJEBEVE - VERIFY subcommand processor
IKJEBEEN - END subcommand processor
IKJEBEXT - END subcommand processor, abnormal termination portion
IKJEBMA1 - Message processing routine
IKJEBMA2 - Input stack clean-up routine.

Exits: Return to the Terminal Monitor Program with following return codes:

 0 - normal completion
12 - abnormal termination

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEME

Module_Name: IKJEBEME

Common_Name: MERGE subcommand processor.

Entry: LINK from module IKJEBEMA at entry point IKJEBEME.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: This subcommand processor will invoke the system MERGE command to either copy all or a part of the Edit data set into itself at some specified or merge all or part of another data set into the one being edited.

Data_Areas_Defined_by_this_Module:

CASCWKA in IKJEBECA
CABFRPL in IKJEBECA
CASVAREA in IKJEBECA

Data_Areas_Updated_by_this_Module:

CAATTNIS - in IKJEBECA
CADSMODS - in IKJEBECA
CAIMLLNO - in IKJEBECA
CANXTSVA - in IKJEBECA
CAPTPDCB - in IKJEBECA
CAPTCDCB - in IKJEBECA
CAPTPRSD - in IKJEBECA
CASAFLAG - in IKJEBECA
CASAINCP - in IKJEBECA
CASAUNCG - in IKJEBECA
CASRPLST - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEME - Mainline routine
IKJEBMEM - Message control section
IKJEBME0 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates MERGE subcommand operands
IKJEBEDA - Data set allocation/unallocation service routine
IKJEBEFC - Copies utility data set into QSAM data format
IKJEBECI - Invokes the system MERGE command.
IKJEBECO - Converts record format from old data set format to EDIT Access Method format.
IKJEBEMR - Merged data set translation routine.

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set
16 - syntax checker not operational

Register_Usage:
Entry
- All registers except 13 are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEMR

Module_Name: IKJEBEMR

Common_Name: Merged data set translation routine.

Entry:
from module IKJEBERE at entry point IKJEBEMR.
from module IKJEBEME at entry point IKJEBEMR.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Re-translates the ITF reverse Polish data set after a successful merge of renumber operation.

Data_Areas_Defined_by_this_Module:

CASRWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CAMODMSG - in IKJEBECA
CANXTSVA - in IKJEBECA
CAPTPDCB - in IKJEBECA
CASYNBFR - in IKJEBECA
CASYNCD2 - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEMR - main line routine
IKJEBMRM - Messages control section

Routines_or_Modules_Called:

IKJEBEUT - Access method interface routine
LANGPRCR - Language processor or syntax checker
IKJEBEEX - access method final processing routine

Exits: Return to caller with following return codes.

0 - successful operation
4 - unsuccessful operation
8 - syntax checker not operational

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEMS

Module_Name: IKJEBEMS

Common_Name: Message selection routine.

Entry: Via a branch from modules IKJEBEBO, IKJEBECG, IKJEBECH, IKJEBECI, IKJEBECN, IKJEBECO, IKJEBEDA, IKJEBEDE, IKJEBEDO, IKJEBEEN, IKJEBEEX, IKJEBEFC, IKJEBEFI, IKJEBEFO, IKJEBEIM, IKJEBEIP, IKJEBEIS, IKJEBELI, IKJEBELT, IKJEBEMA, IKJEBEME, IKJEBEMR, IKJEBEPR, IKJEBERE, IKJEBERM, IKJEBESA, IKJEBESC, IKJEBESN, IKJEBETA, IKJEBETO, IKJEBEUI, IKJEBEUP, and IKJEBEVE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Selects and issues message requested by calling module.

Data_Areas_Defined_by_this_Module:

CAMSWKA - in IKJEBECA
CATEMPBF - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CASRPLST - in IKJEBECA
CANXTSVA - in IKJEBECA
CAPTMSGM - in IKJEBECA
CATPUTVF - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBEMS - Main line routine

Routines_or_Modules_Called:

IKJPUTL - PUTLINE service routine; issues messages.

Exits: Return to caller with following return codes from PUTLINE.

 0 - successful operation
 4 - attention interrupt
 8 - 'NOWAIT' specified, function not complete
12 - invalid parameters
16 - conditional GETMAIN not satisfied.

Register_Usage:

Entry
- All registers are saved.
- Register 0 contains the address of IKJEBECA.
- Register 1 contains the address of a three-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEMV (CSECT OF IKJEBEAA)

Csect_Name: IKJEBEMV

Common_Name: Move routine

Entry:
Branch from csect IKJEBEDL at entry point IKJEBEMV.
Branch from csect IKJEBERR at entry point IKJEBEMV.
Branch from csect IKJEBEWR at entry point IKJEBEMV.

Attributes:

Operation: Moves records.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

Routines_or_Modules_Called: None.

Exits: Return to caller.

Register_Usage:

Entry
- All registers are saved.
- Register 0 contains the length of the data to be moved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers are restored.

IKJEBEPD (CSECT OF IKJEBEPS)

Module Name: IKJEBEPD

Common Name: Processor data table.

Entry: NA

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Contains processor dependent information, is not executable

Data Areas Defined by this Module: NA

Data Areas Updated by this Module: NA

The following Csects are Contained in this Load Module: NA

Routines or Modules Called: None

Exits: NA

Register Usage: NA

IKJEBEPS

Module_Name: IKJEBEPS

Common_Name: Processor data table search routine.

Entry: Via a LOAD and CALL from any module requiring processor dependent information at entry point IKJEBEPS.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Searches IKJEBEPD (processor data table, which is the second csect of the load module) for processor-dependent information relating to a particular data set type.

Data_Areas_Defined_by_this_Module: See operation tables for csects of IKJEBEPS.

IKJEBEPD (second csect of IKJEBEPS) - processor data table.

Data_Areas_Updated_by_this_Module: See operation table for csects of IKJEBEPS.

Routines_or_Modules_Called: See operation table for csects of IKJEBEPS.

IKJEBEPS - Processor Search Routine
IKJEBEPD - Processor Data Table.

Exits: Return to caller with following return codes. See operation tables for csects of IKJEBEPS.

0 - successful operation; entry found in table
4 - entry not found

Register_Usage: See operation tables for csects of IKJEBEPS.

Entry
- All registers except 13 are saved.
- Register 1 contains the address or complimented address of an eight-byte field containing the data set type name for which information from the processor data table is required.
- Register 13 contains the address of a save area.

Exit
- All registers except 0, 1, and 15 are restored.
- Register 0 contains
- Register 1 contains the address of the found table entry.
- Register 15 contains a return code.

IKJEBEPS (CSECT OF IKJEBEPS)

Module_Name: IKJEBEPS

Common_Name: Processor Table Search Routine

Entry: Via LOAD/CALL at entry point IKJEBEPS.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Using the search argument passed this routine, search the Processor Table for an equal compare.

The_Following_Csects_are_Contained_in_this_Load_Module: Does not apply.

Data_Areas_Defined_by_this_Module: None.

Data_Areas_Updated_by_this_Module: None.

Routines_or_Modules_Called: None.

Exits: Return to caller with the following return codes:

0 - Register 1 contains the address of the requested table entry.
4 - Requested table entry not found.

Register_Usage:

Entry
- All registers except 13 are saved.
- Register 1, if negative, contains the complemented address of the search argument for data set name qualifers; if positive, contains the address of the search argument for data set name qualifers.

Exit
- All registers except 1 and 15 are restored.
- Register 1 contains a pointer to the table entry requested for return code 0.
- Register 15 contains a return code.

IKJEBERB (CSECT OF IKJEBEAA)

Csect Name: IKJEBERB

Common Name: Read block routine

Entry:
Branch from csect IKJEBEAD at entry point IKJEBERB.
Branch from csect IKJEBEDL at entry point IKJEBERB.
Branch from csect IKJEBEDS at entry point IKJEBERB.
Branch from csect IKJEBEDU at entry point IKJEBERB.
Branch from csect IKJEBELO at entry point IKJEBERB.

Attributes:

Operation: Reads a particular block into storage.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

BUFTTR (in buffer)

Routines or Modules Called:

IKJEBEAS - Assigns a buffer.
IKJEBEWA - Waits on completion of I/O.
IECPCNVT - Converts TTR to actual track address

Exits: Return to caller with following return codes:

0 - successful operation
12 - I/O error

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBERE

Module_Name: IKJEBERE

Common_Name: RENUMBER subcommand processor.

Entry: LINK from module IKJEBEMA at entry point IKJEBERE.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Renumbers a selected portion or the entire data set; numbers a previously unnumbered data set.

Data_Areas_Defined_by_this_Module:

CABFRPL - in IKJEBECA
CASCWKA - in IKJEBECA
CATEMPBF - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CACURNUM - in IKJEBECA
CADSMODS - in IKJEBECA
CAIMPT - in IKJEBECA
CAIMFLG - in IKJEBECA
CAIMLINC - in IKJEBECA
CAIMLLNO - in IKJEBECA
CAINCRE - in IKJEBECA
CALNTOVF - in IKJEBECA
CANONUM - in IKJEBECA
CANXTSVA - in IKJEBECA
CAPTPDCB - in IKJEBECA
CAPTCDCB - in IKJEBECA
CASTNUM - in IKJEBECA
CASYNCD2 - in IKJEBECA
CASYNLN - in IKJEBECA
CASYNMS1 - in IKJEBECA
CASYNSW - in IKJEBECA
CASRPLST - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBERE - Main line routine
IKJEBRE4 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates RENUM subcommand operands
IKJEBEMS - Message output
IKJEBEUT - Access method interface routine
IKJEBEUI - Access method initialization routine
IKJEBEEX - Access method final processing routine
IKJEBERN - BASIC renumber routine
IKJEBEMR - Data set translate routine

Exits: Return to caller with the following return codes.
 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBERN

Common Name: BASIC Renumber service routine

Entry: LINK from IKJEBERE at entry point IKJEBERN

Attributes: Refreshable, non-privileged, read-only, enabled

Operation: Renumbers a selected part on the entire BASIC data set; updates line number references in statements.

Data Areas Defined by this Module:

CASRWKA
CATEMPBF
CASVAREA

Data Areas Updated by this Module:

CACURNUM
CAIMLINC
CAIMLLNO
CAINCRE

The Following Csects are Contained in this Load Module:

IKJEBERN

Routines or Modules Called:

IKJEBEMS
IKJEBEUT

Exits: Return to caller with following return codes:

0 - successful operation
4 - syntax error in data set (close new data set)
8 - I/O error (close old data set)

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a parameter list.

Exit
- All registers are restored.
- Register 1 contains the address of a parameter list.

IKJEBERR (CSECT OF IKJEBEAA)

Csect_Name: IKJEBERR

Common_Name: Read operation routine

Entry: From module IKJEBEUT at entry point IKJEBERR.

Attributes:

Operation: Reads a particular record into storage.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

PRIMEKEY (in UTILWORK)
SAVEKEY (in UTILWORK)

Routines_or_Modules_Called:

IKJEBELO - Locates a particular record by key.
IKJEBEMV - Moves the located record to a particular place in storage.

Exits: Return to caller with following return codes:

 0 - successful operation
 4 - record could not be found
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a one- to three-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBERU

Module_Name: IKJEBERU

Common_Name: RUN subcommand processor.

Entry: ATTACH from module IKJEBEMA at entry point IKJEBERU.

Attributes: Refreshable, non-privileged, read-only, enabled.

Operation: Builds a command for the system RUN command processor; calls ITF to execute an ITF data set.

Data_Areas_Defined_by_this_Module:

CASCWKA - in IKJEBECA
CABFRPL - in IKJEBECA
CASVAREA - in IKJEBECA

Data_Areas_Updated_by_this_Module:

CAMODMSG - in IKJEBECA
CANXTSVA - in IKJEBECA
CASADISP - in IKJEBECA
CASADSNL - in IKJEBECA
CASYNBFR - in IKJEBECA
CASYNCD2 - in IKJEBECA
CASYNSW - in IKJEBECA
CASYNWA - in IKJEBECA
CASRPIST - in IKJEBECA

The_Following_Csects_are_Contained_in_this_Load_Module:

IKJEBERU - Main line routine
IKJEBRU0 - PCL for IKJPARS

Routines_or_Modules_Called:

IKJPARS - Validates RUN subcommand operands.
IKJEBEDA - Allocates and frees data set.
IKJEBEMS - Message output.
IKJEBEFC - Copies utility data set into QSAM data set format.
IKJEBEUT - Access method interface routine.
IKJEBECI - Command invoker routine
IKJSTCK - Stack routine
LANGPRCR - Language processor or syntax checker.

Exits: Return to caller with following return codes.

 0 - successful operation.
 8 - unsuccessful completion
12 - I/O error on utility data set.

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBESA

Module Name: IKJEBESA

Common Name: SAVE subcommand processor.

Entry: from module IKJEBEMA at entry point IKJEBESA.

Attributes: Reentrant.

Operation: Causes the utility data set to be saved as the permanent data set with the name specified on the SAVE subcommand or with the name specified on the EDIT command.

Data Areas Defined by this Module: See operations tables for csect of IKJEBESA.

Data Areas Updated by this Module:

CASADSN - DSNAME of SAVE data set
CASAMEMB - Membername of Save data set
CASADDN - DDNAME of SAVE data set
CASAPSWD - Password for SAVE data set

The Following Csects are Contained in this Load Module:

IKJEBESA - Main line routine
IKJEBSA1 - PCL for IKJPARS (SAVE subcommand)
IKJEBSA2 - PCL for IKJPARS (prompt for membername)
IKJEBSA8 - Message control section
IKJEBSA9 - Message processing routine

Routines or Modules Called:

IKJDFLT - Fully-qualifies data set name and searches catalog
IKJPARS - Validates SAVE subcommand operands
IKJEBEMS - Message output
IKJDAIR - Allocates SAVE data set
IKJEBEFC - Copies utility data set into QSAM data set format. See operations tables for csect of IKJEBESA.
IKJPTGT - Issues prompt messages and obtains input

Exits: Return to caller with following return codes:

0 - successful operation
8 - unsuccessful completion
12 - I/O error See operations tables for csect of IKJEBESA.

Register Usage:

Entry
- All registers except 13 are saved.
- Register 1 contains the address of the EDIT Communication Area.
- Register 13 contains the address of a save area. See operations tables for csect of IKJEBESA.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code. See operations tables for csect of IKJEBESA.

IKJEBESA (CSECT OF IKJEBESA)

Module Name: IKJEBESA

Common Name: SAVE subcommand processor, main line routine.

Entry: Via LINK from module IKJEBEMA or IKJEBEEN at entry point IKJEBESA.

Attributes: Re-entrant and refreshable.

Operation: Causes the utility data set to be saved as the permanent data set with the name specified on the SAVE subcommand or with the name specified on the Edit command.

Data Areas Defined by this Module:

CABFRPL - (in IKJEBECA)
CASCWKA - (in IKJEBECA)
CATEMPBF - (in IKJEBECA)
CASVAREA - (in IKJEBECA)

Data Areas Updated by this Module:

CAATTNIS-	CADSNOFF-	CAEDDDN-	CANXTSVA-	CASAALOC-
CADSMODS-	CADSNOF2-	CAEDDISP-	CAPRSPDL-	CASADSN-
CADSNPTR-	CADSNRC2-	CAEDMEM-	CAPTIBFR-	CASADISP-
CADSNPT2-	CAEDDSOR-	CAEDMEMB-	CAPTPRSD-	CASADDN-
CADSNLN2-	CAEDALOC-	CAEDUNCG-	CASRPLST-	CASADSNL-
CASADSOR-	CASAFNCP-	CASAMEM-	CASAMEMB-	CASAPSWD-

CASAUNCG-(in IKJEBECA)

Routines or Modules Called:

IKJEBEMS - Message select routine
IKJDFLT - Default service routine
IKJPTGT - Putget service routine
IKJDAIR - Dynamic allocation interface service routine
IKJEBEFC - Final copy routine
IKJPARS - Validates SAVE subcommand operands
IKJEBSA9 - SAVE message processing routine

Exits: Return to caller with the following return codes:

 0 - successful operation
12 - I/O error in utility data set

Register Usage:

Entry:
- All registers are saved.
- Register 1 contains the address of IKJEBECA.

Exit:
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBESC

Module_Name: IKJEBESC

Common_Name: SCAN subcommand processor.

Entry:
Via LINK from module IKJEBEMA at entry point IKJEBESC.
Via LINK from module IKJEBEEN at entry point IKJEBESC.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Loads and deletes syntax checker; performs initial copy processing for syntax checker.

Data_Areas_Defined_by_this_Module:

CABFRPL - (in IKJEBECA)
CASCWKA - (in IKJEBECA)
CASVAREA - (in IKJEBECA)

Data_Areas_Updated_by_this_Module:

CAPROMPT-	CASYNAME-	CASYNPWA-
CAPTCHK-	CASYNBFR-	CASRPLST-
CASCANON-	CASYNMS1-	CANXTSVA-
CASCANSW-	CASYNMS2-	(all in IKJEBECA)
CASCRC20-	CASYNPTO-	

The_following_Csects_are_contained_in_this_load_module: IKJEBESC

Routines_or_Modules_Called:

IKJPARS - Validates SCAN subcommand operands
IKJEBEMS - Message output
IKJEBEUT - Access method interface routine
IPDSNEXC - Syntax checker
IKJNC211 - Syntax checker
PLISCAN - Syntax checker
PLIFSCAN - Syntax checker
IKJEBESN - Second load of SCAN subcommand processor

Exits:
Normal -
XCTL to IKJEBESN (second load of SCAN subcommand processor).
Error -
Return to caller with following return code:
8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.
- Register 13 contains the address of a save area.

Exit
Normal -
- Registers 2-12 are restored.
- Register 1 contains the address of the EDIT Communication Area.
- Register 15 contains the address of the XCTL parameter list.

Error -
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBESE

Module Name: IKJEBESE (Csect name), link-edited with the load modules IKJEBEFI and IKJEBECG.

Common Name: String search routine.

Entry:
via a BALR instruction from module IKJEBEFI at entry point IKJEBESE.
via a BALR instruction from module OKJEBECG at entry point IKJEBESE.

Attributes: Reentrant.

Operation: Searches a given number of lines for a particular string of data.

Data Areas Defined by this Module:

Data Areas Updated by this Module:

Routines or Modules Called:

IKJEBEUT - Access method interface routine

Exits: Return to caller with following return codes.

0 - successful operation; text found in one line
4 - successful operation; text found across lines
8 - text not found
12 - I/O error in utility data set

Register Usage:

Entry
- All registers are saved.
- Register 0 contains the address of the EDIT Communication Area.
- Register 1 contains the address of a parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code (in the first byte) and the offset to the found text (in the last three bytes).

IKJEBESN

Module_Name: IKJEBESN

Common_Name: SCAN subcommand processor.

Entry: Via XCTL from module IKJEBESC at entry point IKJEBESN.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Interfaces with the appropriate syntax checker passing lines of input to be syntax checked; causes diagnostic messages to be issued when syntax errors are diagnosed.

Data_Areas_Defined_by_this_Module:

CABFRPL - (in IKJEBECA)
CASCWKA - (in IKJEBECA)
CATEMPBF - (in IKJEBECA)
CASVAREA - (in IKJEBECA)

Data_Areas_Updated_by_this_Module:

CACURNUM-
CALNTOVF-
CAPTCHK-
CASYNBFR-
CASYNPWA-
CASYNPTO-
CASYNMS1-
CASYNMS2-
CANXTSVA (all in IKJEBECA)

The_following_Csects_are_contained_in_this_load_module:

IKJEBESN - Main line routine

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJEBEUT - Access method interface routine
IPDSNEXC - Syntax checker
IKJNC211 - Syntax checker
PLISCAN - Syntax checker
PLIFSCAN - Syntax checker

Exits: Return to caller with following return codes.

0 - successful operation
8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBETA

Module Name: IKJEBETA

Common Name: TABSET subcommand processor

Entry: Via LINK from module IKJEBEMA at entry point IKJEBETA.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Inserts up to 10 tab settings independent of tab settings specified at terminal; controls whether tabs are to be in effect or not.

Data Areas Defined by this Module:

CASCWKA - (in IKJEBECA)
CASVAREA

Data Areas Updated by this Module:

CANXTSVA - (in IKJEBECA)
CATABS - (in IKJEBECA)
CASRPLST - (in IKJEBECA)

The following Csects are contained in this load module:

IKJEBETA - Main routine
IKJEBETA0 - Parse PCL Csect

Routines or Modules Called:

IKJPARS - Validates TABSET subcommand operands.
IKJGETL - GETLINE service routine; obtains terminal input.
IKJEBEMS - Message output.

Exits: Return to caller with following return codes:

0 - successful operation
8 - unsuccessful completion

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBETO

Module Name: IKJEBETO

Common Name: TOP subcommand processor

Entry: Via LINK from module IKJEBEMA at entry point IKJEBETO.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Causes current line pointer to be moved to before the first line in the data set or to line number zero.

Data Areas Defined by this Module:

CASCWKA - (in IKJEBECA)
CATEMPBF - (in IKJEBECA)
CASVAREA

Data Areas Updated by this Module:

CALNTOVF - (in IKJEBECA)
CACURNUM - (in IKJEBECA)
CANXTSVA - (in IKJEBECA)

The following Csects are contained in this load module:

IKJEBETO - Main routine

Routines or Modules Called:

IKJEBEMS - Message output
IKJEBEUT - Acts as interface to the EDIT Access Method

Exits: Return to caller with following return codes:

 0 - successful operation
12 - I/O error in utility data set

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEUI

Module Name: IKJEBEUI

Common Name: EDIT Access Method initialization routine.

Entry:
Via LINK from module IKJEBEIN at entry point IKJEBEUI.
Via LINK from module IKJEBECO at entry point IKJEBEUI.
Via LINK from module IKJEBERE at entry point IKJEBEUI.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Allocates the resources used by the access method routines; loads the access method routines (IKJEBEAA) into storage.

Data Areas Defined by this Module:

CATEMPBF - Temporary workarea
UTILWORK - EDIT Access Method work area
CASRWKA - Work area
CASVAREA

Data Areas Updated by this Module:

CANXTSVA - (in IKJEBECA)
CAPTCDCB - (in IKJEBECA)
CASRPLST - (in IKJEBECA)
DCBBUFAD - (in UTILWORK)
DCBDDNAM - (in UTILWORK)
DCBSYNAD - (in UTILWORK)
DCBWBKAD - (in UTILWORK)
UTILDCB - (in UTILWORK)

The following Csects are contained in this load module:

IKJEBEUI - Main routine

Routines or Modules Called:

IKJDAIR - Allocates utility data set
IKJEBEMS - Message output

Exits: Return to caller with following return codes:

0 - successful operation
4 - unsuccessful operation; insufficient storage
8 - unsuccessful operation; unable to allocate utility data set
12 - unsuccessful operation; unable to open utility data set

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 0 and 15 are restored.
- Register 0 contains the address of UTILWORK.
- Register 15 contains a return code.

IKJEBEUP

Module_Name: IKJEBEUP

Common_Name: UP subcommand processor.

Entry: Via LINK from module IKJEBEMA at entry point IKJEBEUP.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Causes current line pointer to be moved toward beginning of data set.

Data_Areas_Defined_by_this_Module:

CASCWKA - (in IKJEBECA)
CATEMPBF - (in IKJEBECA)
CASVAREA

Data_Areas_Updated_by_this_Module:

CALNTOVF - (in IKJEBECA)
CACURNUM - (in IKJEBECA)
CANXTSVA - (in IKJEBECA)

The_following_Csects_are_contained_in_this_Load_Module:

IKJEBEUP - Main routine
IKJEBUP0 - Parse PCL

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJPARS - Validates UP subcommand operands
IKJEBEUT - Acts as interface to the EDIT Access Method

Exits: Return to caller with following return codes:

 0 - successful operation
 8 - unsuccessful completion
12 - I/O error in utility data set

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEUT

Module_Name: IKJEBEUT

Common_Name: EDIT Access Method interface routine.

Entry: Via BALR from modules IKJEBEBO, IKJEBECG, IKJEBECH, IKJEBECN, IKJEBECO, IKJEBEDE, IKJEBEDO, IKJEBEFC, IKJEBEFI, IKJEBEFO, IKJEBEIM, IKJEBEIP, IKJEBEIS, IKJEBELI, IKJEBELT, IKJEBEME, IKJEBEMR, IKJEBERE, IKJEBERU, IKJEBESC, IKJEBESN, IKJEBETO, and IKJEBEUP.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Selects proper operation routine (read, write, or delete) based on function desired by caller.

Data_Areas_Defined_by_this_Module:

TEMPAREA - (in UTILWORK)

Data_Areas_Updated_by_this_Module:

CADSMODS - (in IKJEBECA)
CANXTSVA - (in IKJEBECA)
CAUTILNO - (in IKJEBECA)
DABUFNO - (in UTILWORK)
DCBSTATS - (in UTILWORK)
VTABLE - (in UTILWORK)

The_following_Csects_are_contained_in_this_load_module:

IKJEBEUT - Main routine

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJEBERR - Read operation routine
IKJEBEWR - Write operation routine
IKJEBEDR - Delete operation routine

Exits: Return to caller with following return codes:

 0 - successful operation
 4 - unsuccessful read or delete operation; record not found
 8 - I/O error
12 - invalid line number

Register_Usage:

Entry
- All registers are saved.
- Register 0 contains the address of the EDIT Communication Area.
- Register 1 contains the address of a three-word parameter list.

Exit
- All registers except 1 and 15 are restored.
- Register 15 contains a return code.
- Register 1 contains address of record for read operation.

IKJEBEVE

Module_Name: IKJEBEVE

Common_Name: VERIFY subcommand processor

Entry: LINK from module IKJEBEMA at entry point IKJEBEVE.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Sets the verify switch in the EDIT Communication area depending on the subcommand operand entered.

Data_Areas_Defined_by_this_Module:

CASCWKA - (in IKJEBECA)
CASVAREA

Data_Areas_Updated_by_this_Module:

CASRPLST - (in IKJEBECA)
CAVRFYSW - (in IKJEBECA)
CAFREEDL - (in IKJEBECA)
CANXTSVA - (in IKJEBECA)

The_following_Csects_are_contained_in_this_Load_Module:

IKJEBEVE - Mainline routine
IKJEBVEP - PCL for IKJPARS

Routines_or_Modules_Called:

IKJEBEMS - Message output
IKJPARS - Validates VERIFY subcommand operands

Exits: Return to caller with following return code:

0 - successful operation
8 - unsuccessful completion

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of the EDIT Communication Area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEWA (CSECT OF IKJEBEAA)

Csect_Name: IKJEBEWA

Common_Name: Wait routine

Entry:
Branch from csect IKJEBEAD at entry point IKJEBEWA.
Branch from csect IKJEBEAS at entry point IKJEBEWA.
Branch from csect IKJEBERB at entry point IKJEBEWA.
Branch from csect IKJEBEWB at entry point IKJEBEWA.

Attributes:

Operation: Waits on completion of an I/O operation ina particular buffer.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

TEMPAREA (in UTILWORK)
BUFSTATS (in buffer)
BUFTTR (in buffer)
DCBSTATS (in UTILWORK)

Routines_or_Modules_Called:

WAIT
CHECK
NOTE
SYNADAF
SYNADRLS

Exits: Return to caller with following return codes:

0 - successful operation
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEWB (CSECT OF IKJEBEAA)

Csect Name: IKJEBEWB

Common Name: Write block routine

Entry:
Branch from csect IKJEBEAD at entry point IKJEBEWB.
Branch from IKJEBEAS at entry point IKJEBEWB.
Branch from IKJEBEDL at entry point IKJEBEWB.
Branch from IKJEBEDU at entry point IKJEBEWB.
Branch from IKJEBEWR at entry point IKJEBEWB.

Attributes:

Operation: Writes a particular block into the utility data set.

Data Areas Defined by this Csect:

Data Areas Updated by this Csect:

DCBBUFIO (in UTILWORK)
DCBSTATS (in UTILWORK)
BUFSTATS (in buffer)

Routines or Modules Called:

IKJEBEWA - Waits on completion of I/O.
IECPCNVT - Converts TTR to actual track address

Exits: Return to caller with following return codes:

0 - successful operation
12 - I/O error

Register Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a two-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBEWR (CSECT OF IKJEBEAA)

Csect_name: IKJEBEWR

Common_Name: Write operation.

Entry: from module IKJEBEUT at entry point IKJEBEWR.

Attributes:

Operation: Writes a new record into the utility data set.

Data_Areas_Defined_by_this_Csect:

Data_Areas_Updated_by_this_Csect:

BUFSTATS (in buffer)
NUMREC (in buffer)
BUFREF (in buffer)
DATASTRT (in buffer)
DCBRECNO (in UTILWORK)
DCBSTATS (in UTILWORK)

Routines_or_Modules_Called:

IKJEBELO - Locates old record with same key as new record.
IKJEBEDL - Deletes old record.
IKJEBEAD - Obtains a new physical block from direct-access device.
IKJEBEWB - Writes a block into storage.
IKJEBEMV - Moves records in storage.
IKJEBEDU - Updates the directory block.

Exits: Return to caller with following return codes:

 0 - successful operation
12 - I/O error

Register_Usage:

Entry
- All registers are saved.
- Register 1 contains the address of a three-word parameter list.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBMA1 (CSECT OF IKJEBEMA)

Csect Name: IKJEBMA1

Common Name: Message processing routine for IKJEBEMA.

Entry: Via BRANCH from module IKJEBEMA at entry point IKJEBMA1.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Using an index passed by the caller, this routine selects and completes the appropriate insertion and calls the message select routine to put the message to the user.

Data Areas Defined by this Module:

CATEMPBF - in IKJEBECA
CASVAREA

Data Areas Updated by this Module:

CANXTSVA - in IKJEBECA

The following Csects are Contained in this Load Module: NA

Routines or Modules Called:

IKJEBEMS - Message select service routine

Exits: Return to caller with following PUTLINE return codes.

```
 0 - successful operation
 4 - attention interrupt
 8 - 'NOWAIT' specified, function not complete
12 - invalid parameters
16 - conditional GETMAIN not satisfied
```

Register Usage:

Entry
- All registers are saved.
- Register 1 contains a pointer to a five word parameter list of the following format:
  +0 Pointer to IKJEBECA
  +4 Relative first-level message
  +8 Relative second-level message
  +12 Message-dependent information
  +16 Message-dependent information

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBMA2

Csect Name: IKJEBMA2

Common Name: Input stack clean-up routine.

Entry: Via branch from IKJEBEMA, IKJEBEAE or IKJEBEAT at entry point IKJEBMA2.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: For entry code 0 (zero), free current input buffer, if necessary, for entry code 1. Clear input and output queues and delete the input stack; for entry code 2, delete the input stack.

Data Areas Defined by this Module:

CASVAREA - 18 word save area in IKJEBECA

Data Areas Updated by this Module:

CANXTSVA - (in IKJEBECA)
CAOPERND - (in IKJEBECA)
CAPTGTBF - (in IKJEBECA)
CASRPLST - (in IKJEBECA)

The Following Csects are Contained in this Load Module: NA

Routines or Modules Called:

IKJSTCK - STACK service routine

Exits: Return to caller with following return codes.

0 - successful operation
4 - unsuccessful operation

Register Usage:

Entry
- All registers are saved.
- Register 1 contains a pointer to a work area.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

IKJEBMA8

Csect_Name: IKJEBMA8

Common_Name: IBM-supplied table of subcommands.

Entry: NA

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Table of IBM-supplied subcommands. (See Section 5: Data Areas for Description of Table Format.)

Data_Areas_Defined_by_this_Module: None

Data_Areas_Updated_by_this_Module: None

The_Following_Csects_are_Contained_in_this_Load_Module: NA

Routines_or_Modules_Called: NA

Exits: NA

Register Usage:

Entry
NA

Exit
NA

IKJEBMA9

Csect_Name: IKJEBMA9

Common_Name: User-supplied table of subcommands.

Entry: NA

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Table of user-supplied subcommands. (See Section 5: Data Areas for Description of Table Format.

Data_Areas_Defined_by_this_Module: None

Data_Areas_Updated_by_this_Module: None

The_Following_Csects_are_Contained_in_this_Load_Module: NA

Routines_or_Modules_Called: NA

Exits: NA

Register Usage:

Entry
NA

Exit
NA

IKJEBSA9

Csect Name: IKJEBSA9

Common Name: SAVE message processing routine.

Entry: Via BRANCH from module IKJEBESA at entry point IKJEBSA9.

Attributes: Refreshable, read-only, non-privileged, enabled.

Operation: Prepares message insertions and selects error message to be put to the user by the message select routine.

Data Areas Defined by this Module:

CABFRPL - in IKJEBECA
CASVAREA

Data Areas Updated by this Module:

CADSNPT2

The Following Csects are Contained in this Load Module: NA

Routines or Modules Called:

IKJEBEMS - Message selection routine

Exits: Return to caller with following Putline return codes.

0
4
8
12

Register Usage:

Entry
- All registers are saved.
- Register 1 contains a relative error number.

Exit
- All registers except 15 are restored.
- Register 15 contains a return code.

Flowcharts

Chart AA. IKJEBEAD

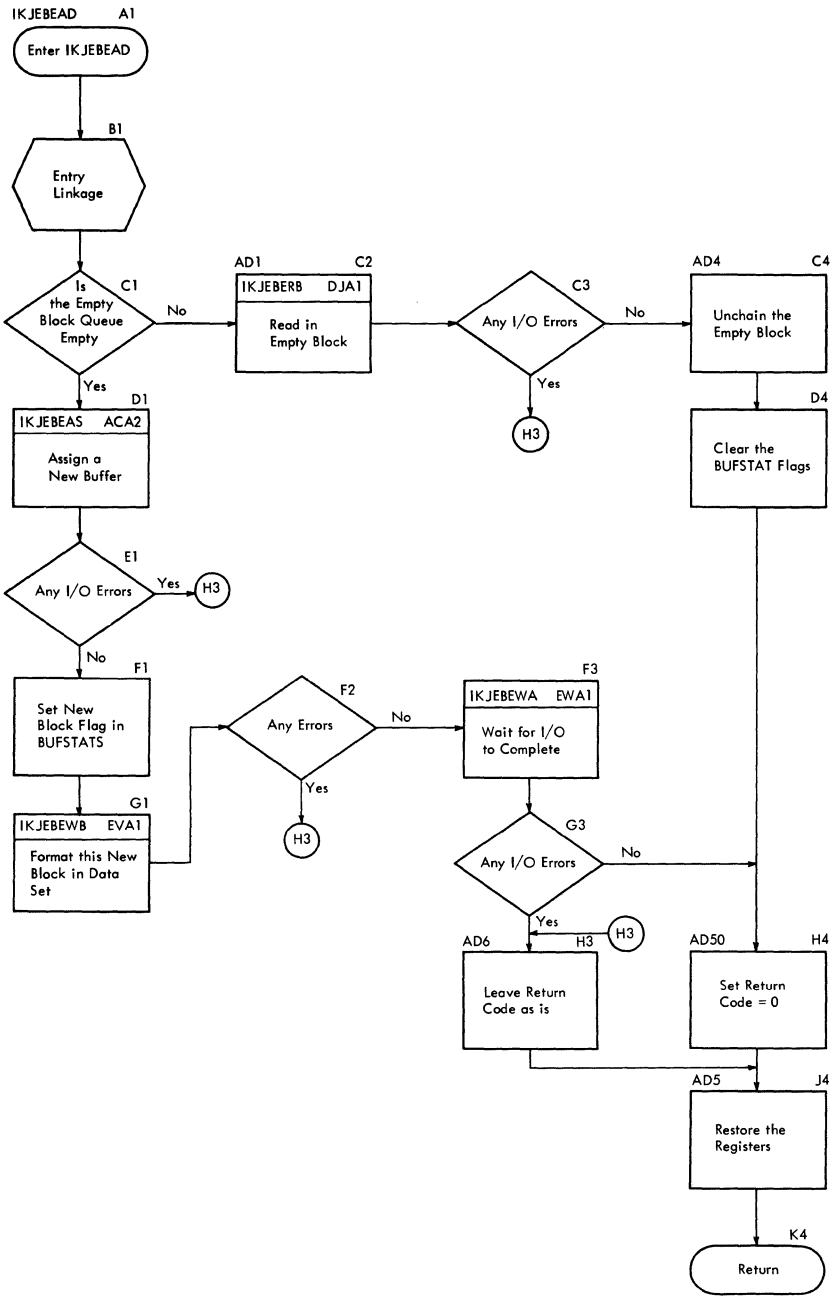


Chart AB. IKJEBEAE

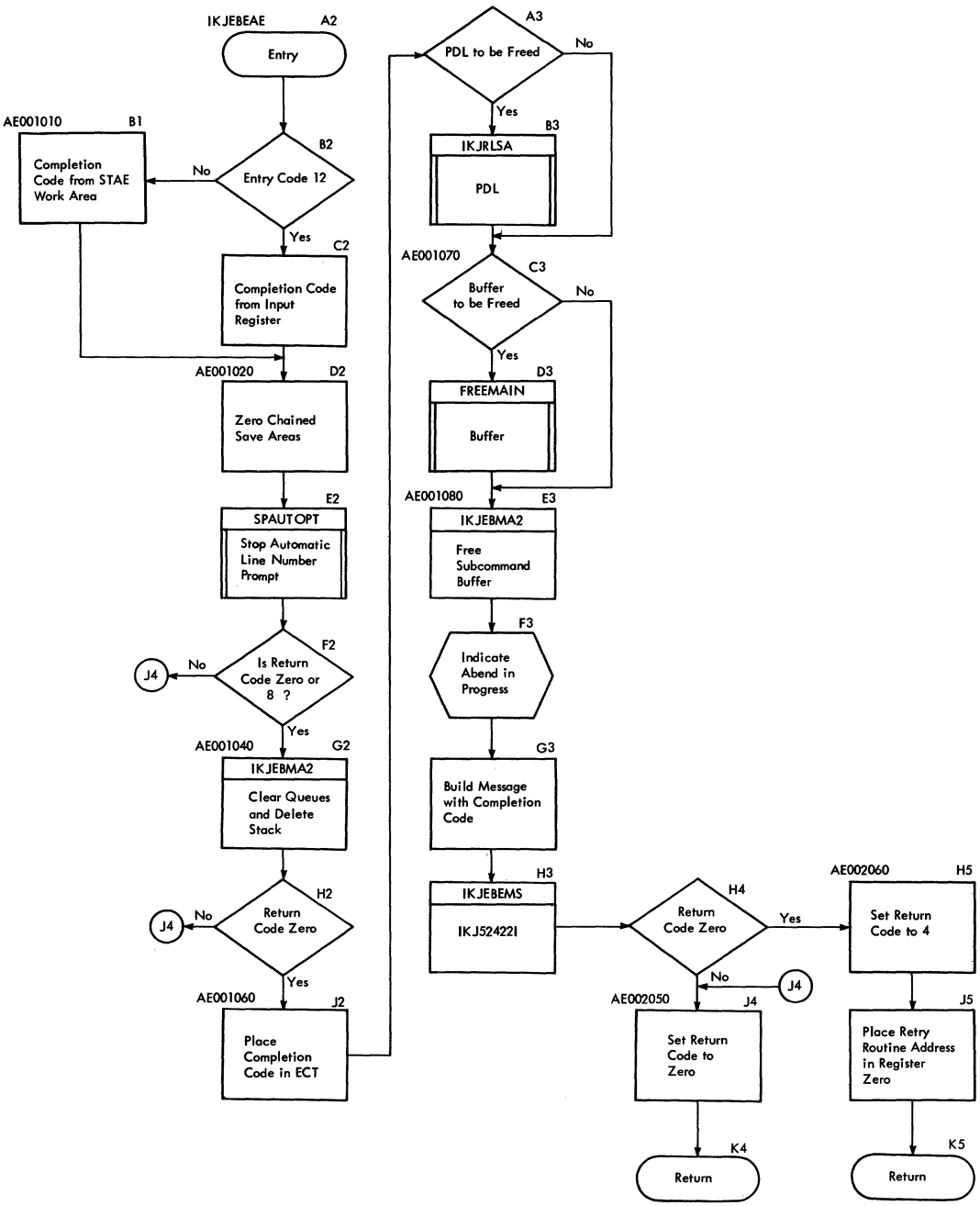


Chart AC. IKJEBEAS

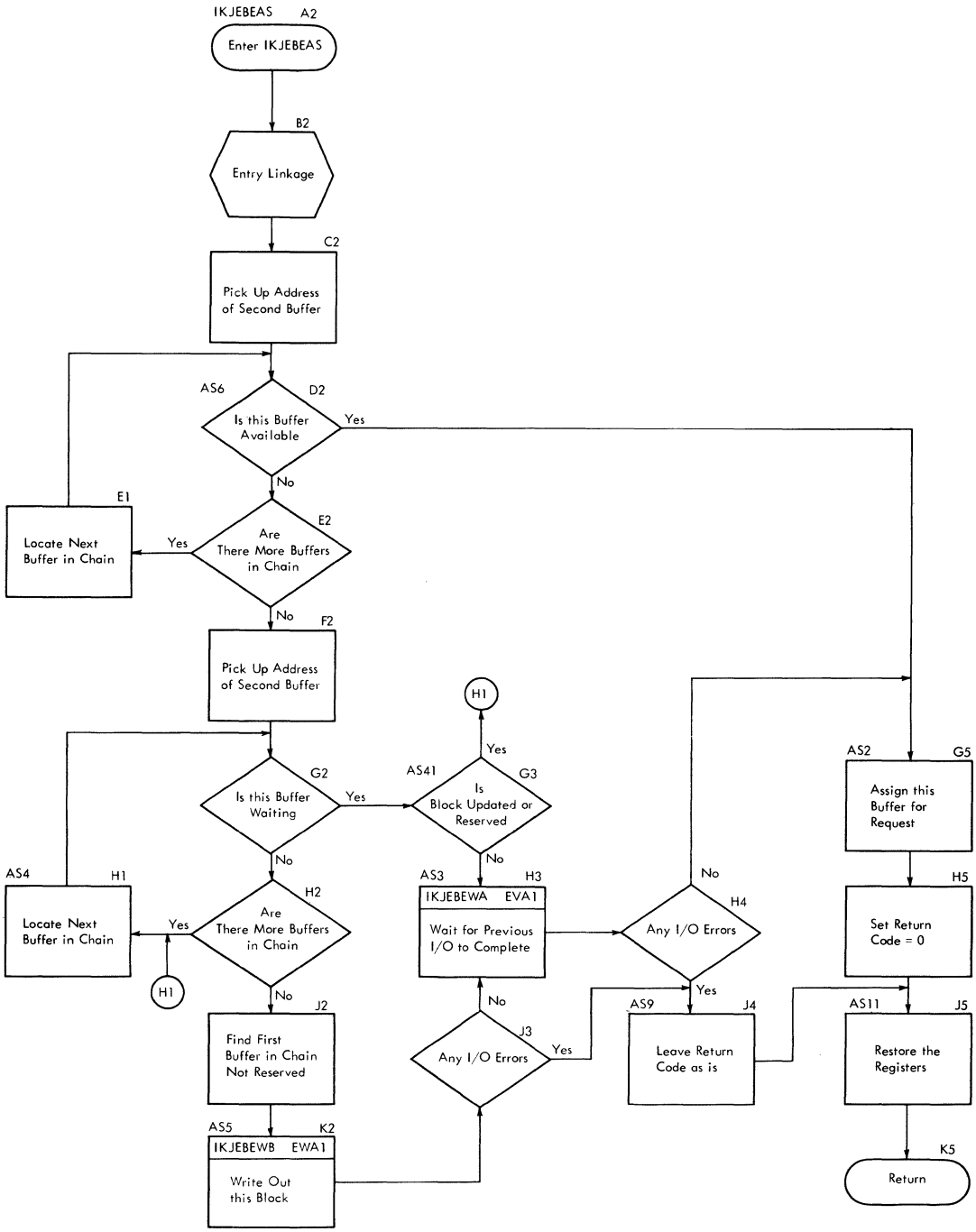


Chart AD. IKJEBEAT

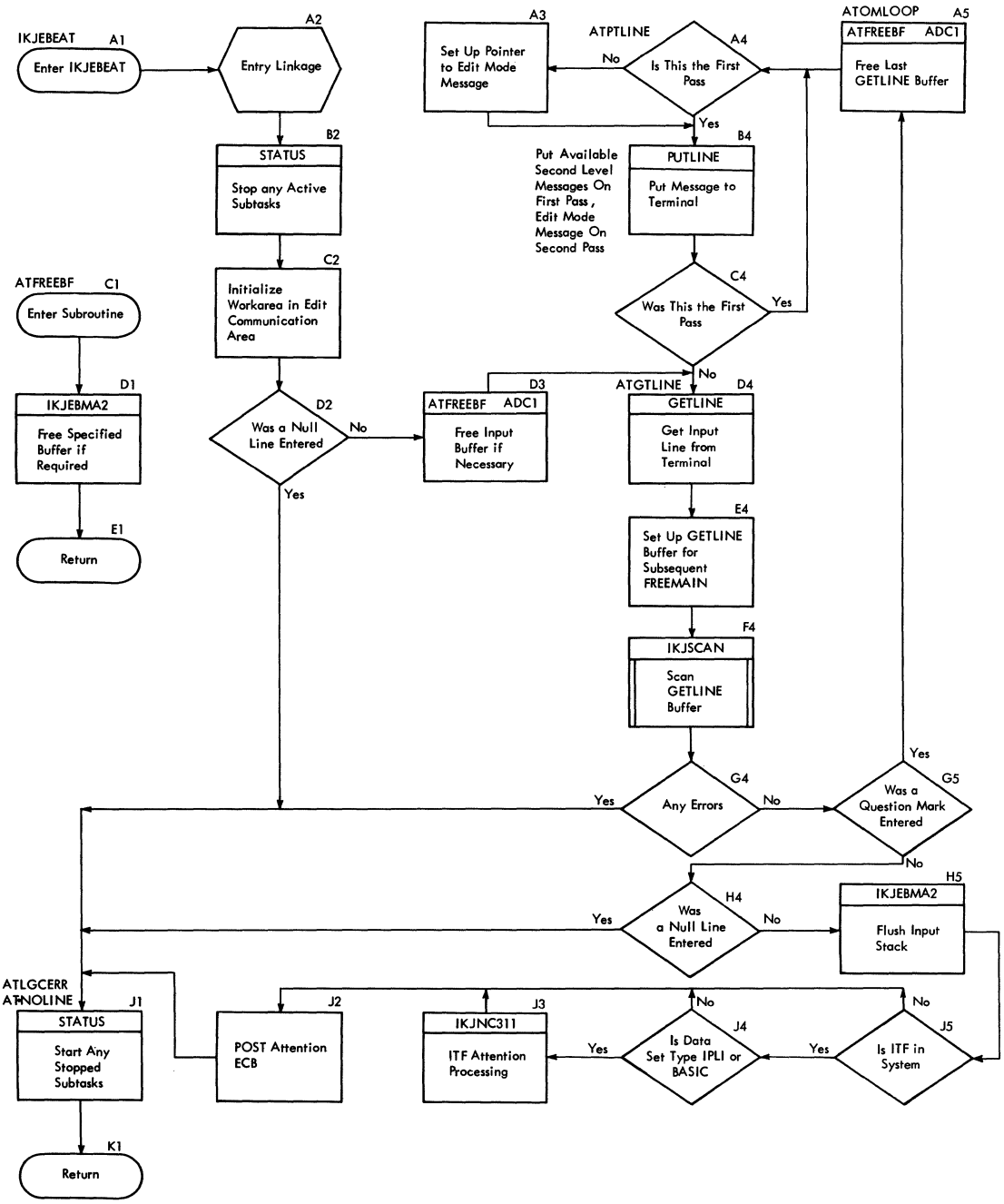


Chart AE. IKJEBEBO

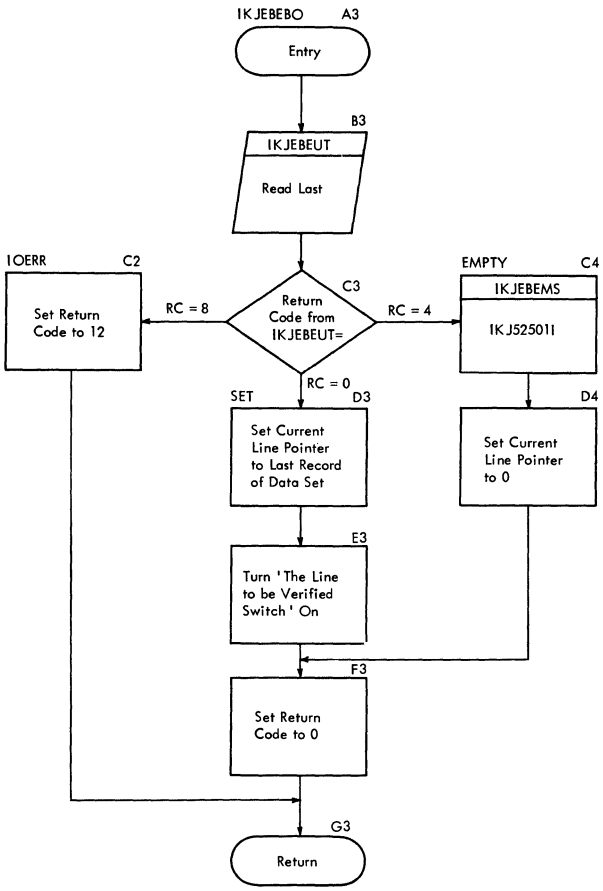


Chart AF. IKJEBECG

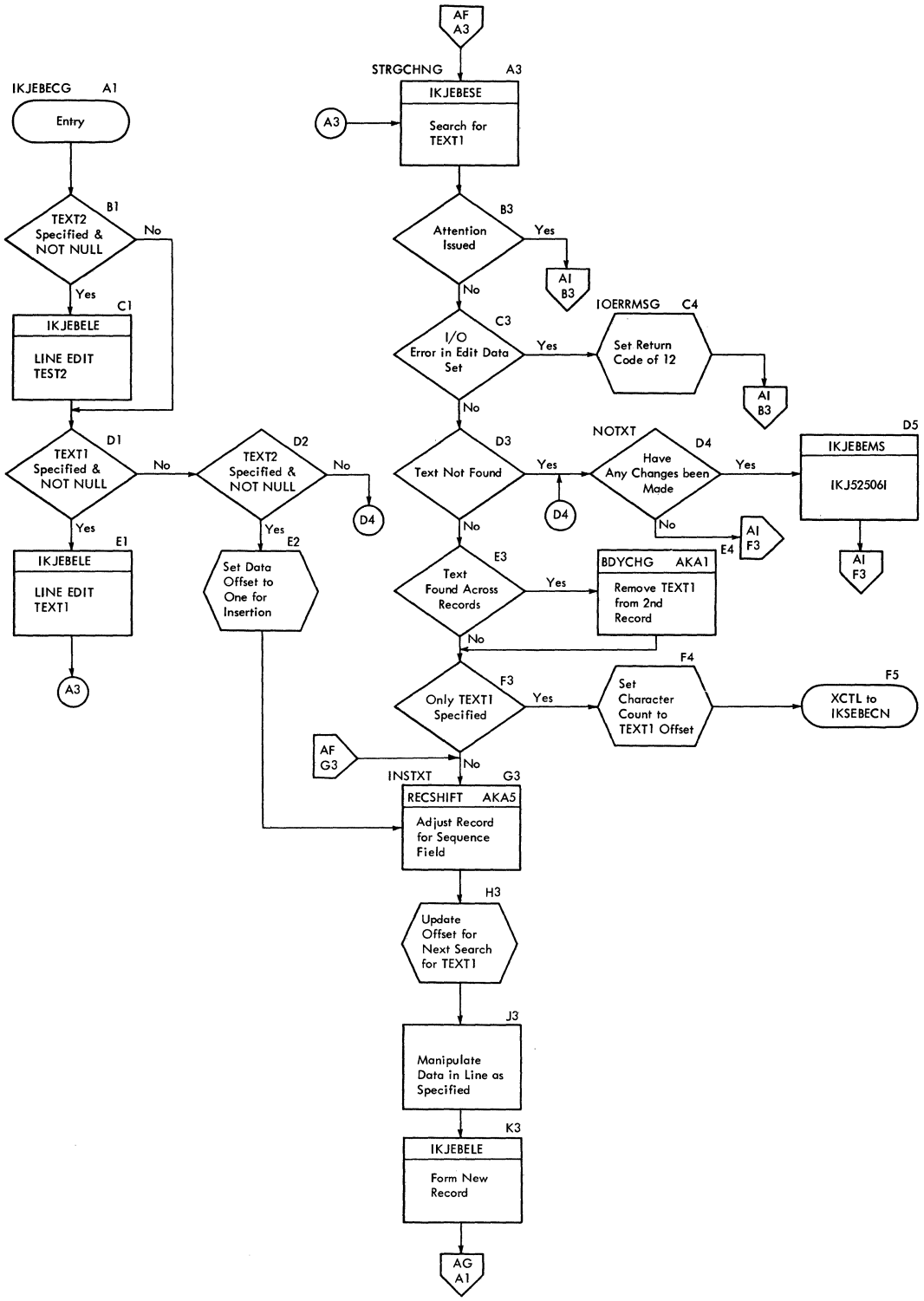


Chart AG. IKJEBECG

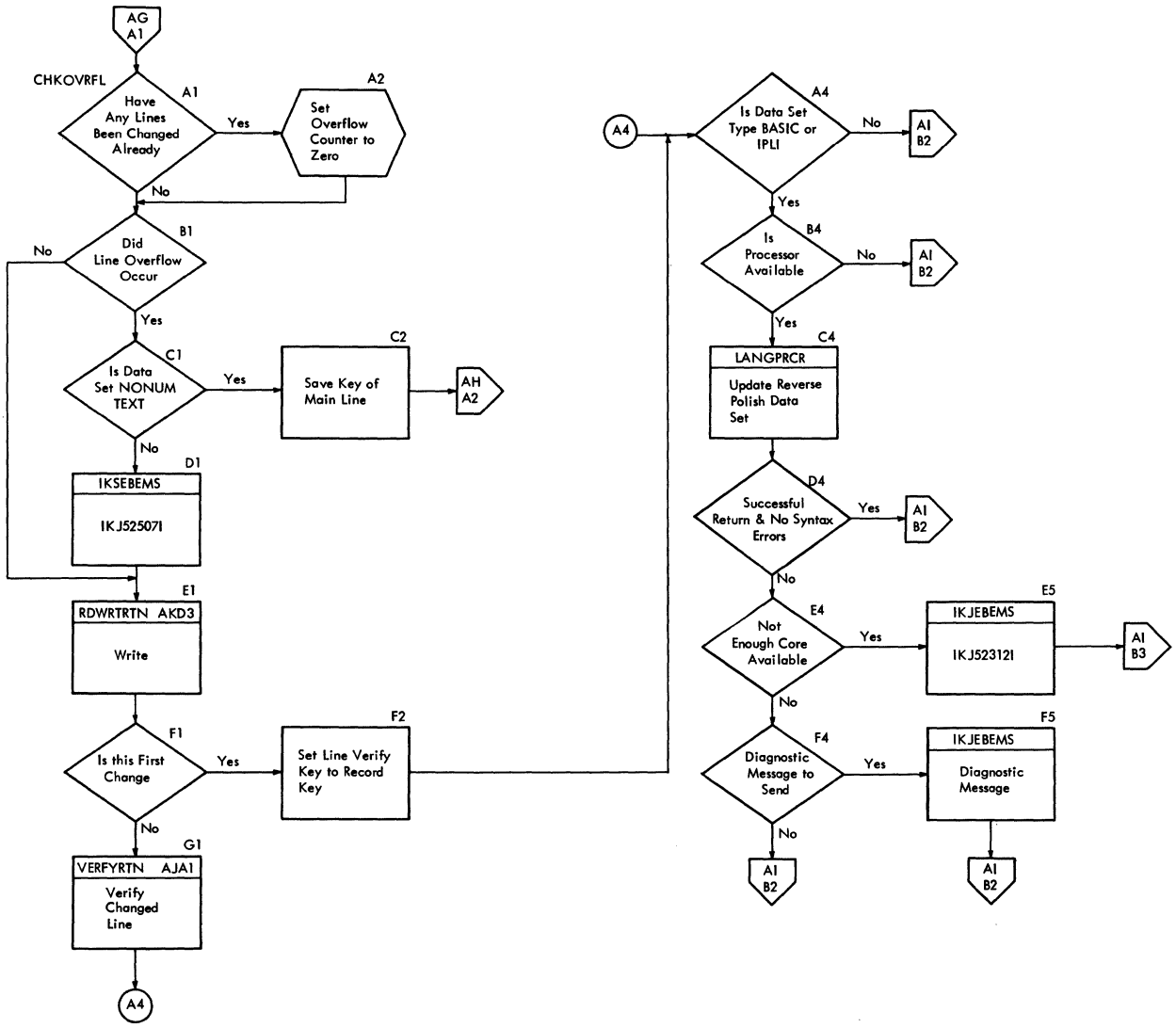


Chart AH. IKJEBECG

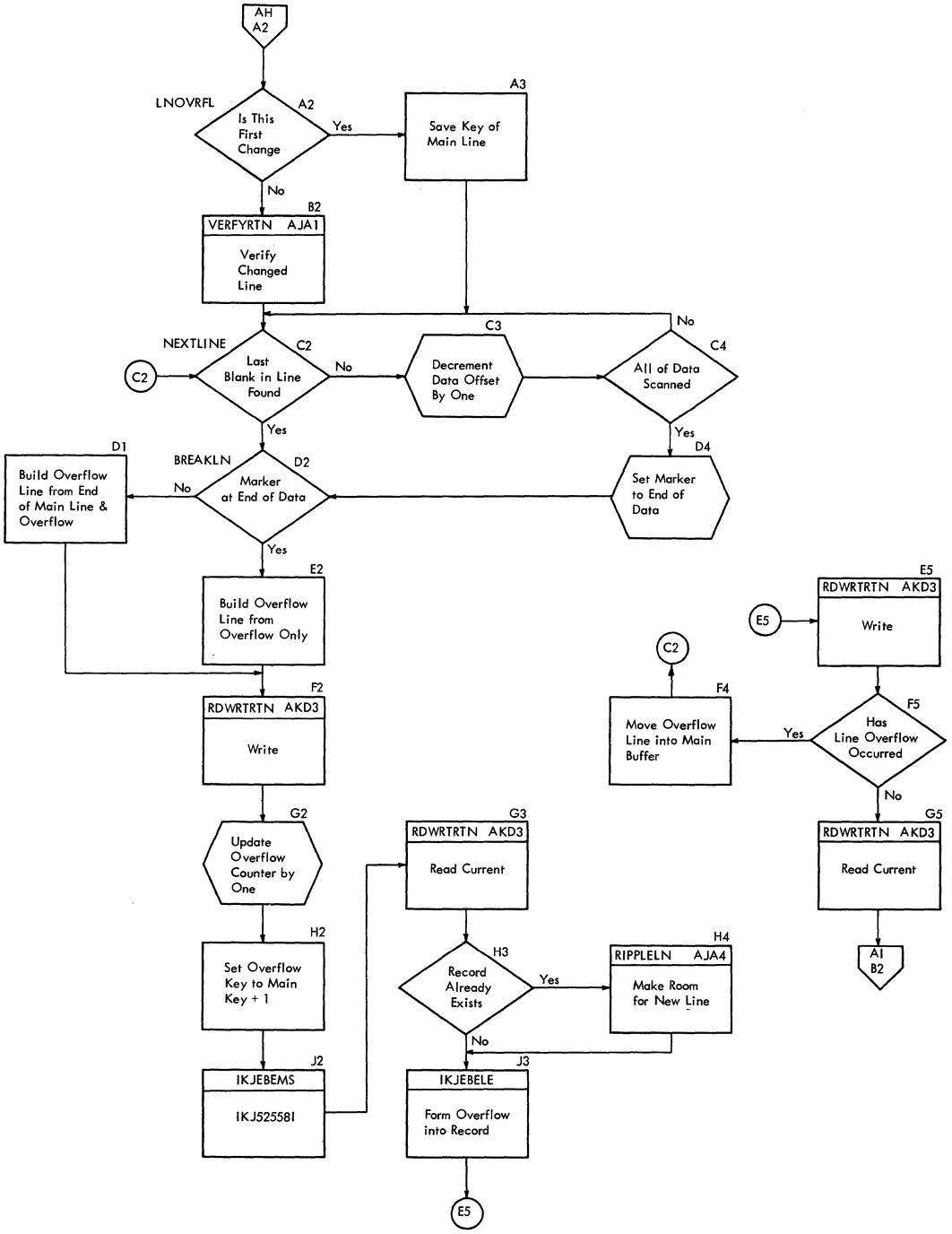


Chart AI. IKJEBECG

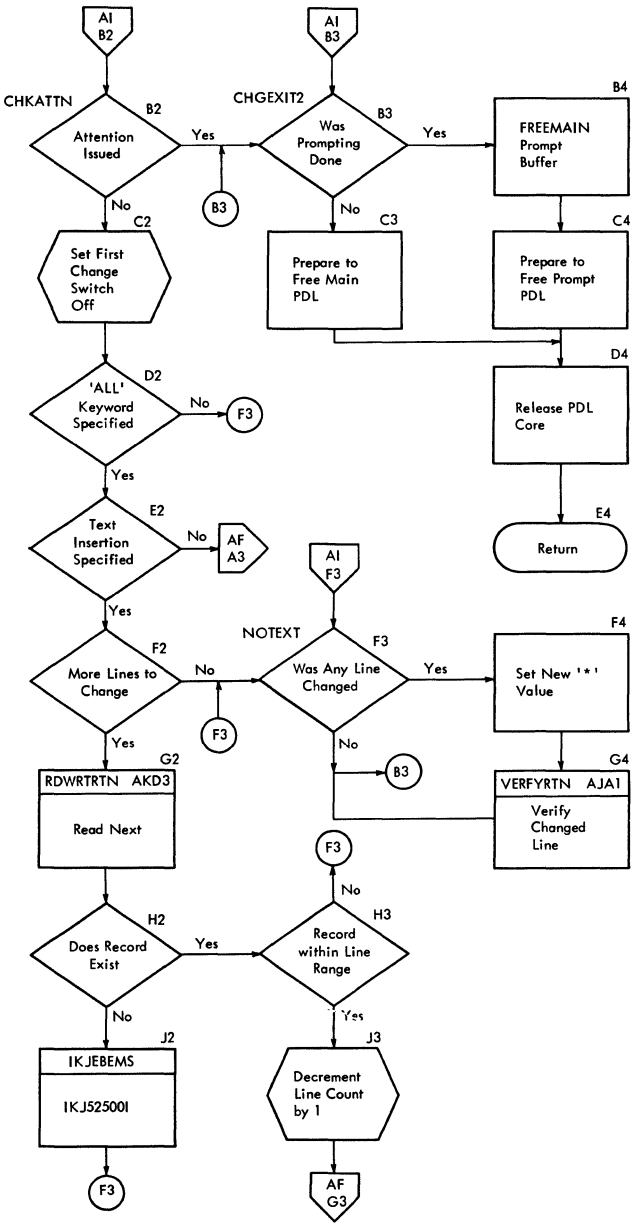


Chart AJ. IKJEBECG

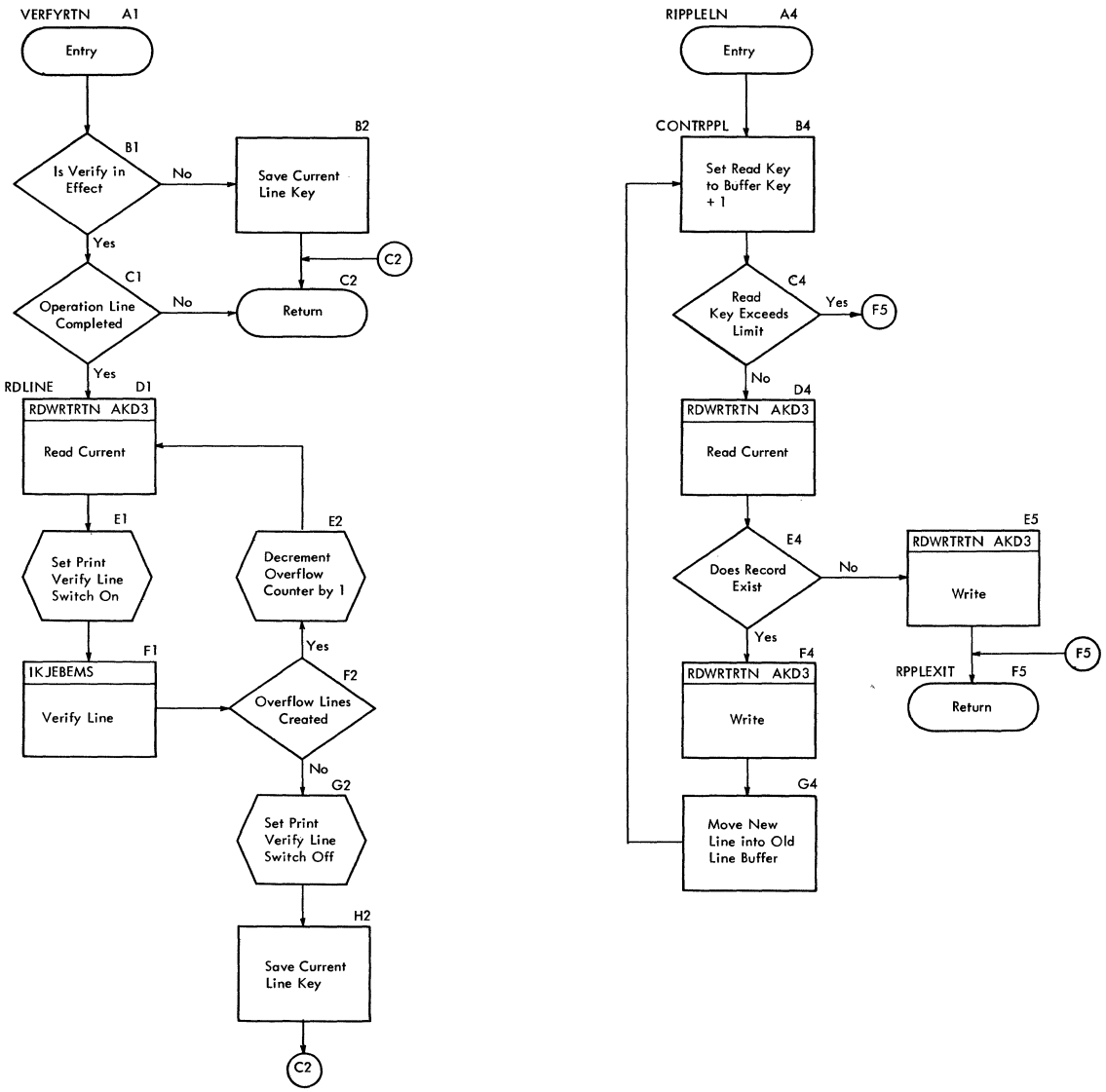


Chart AK. IKJEBECG

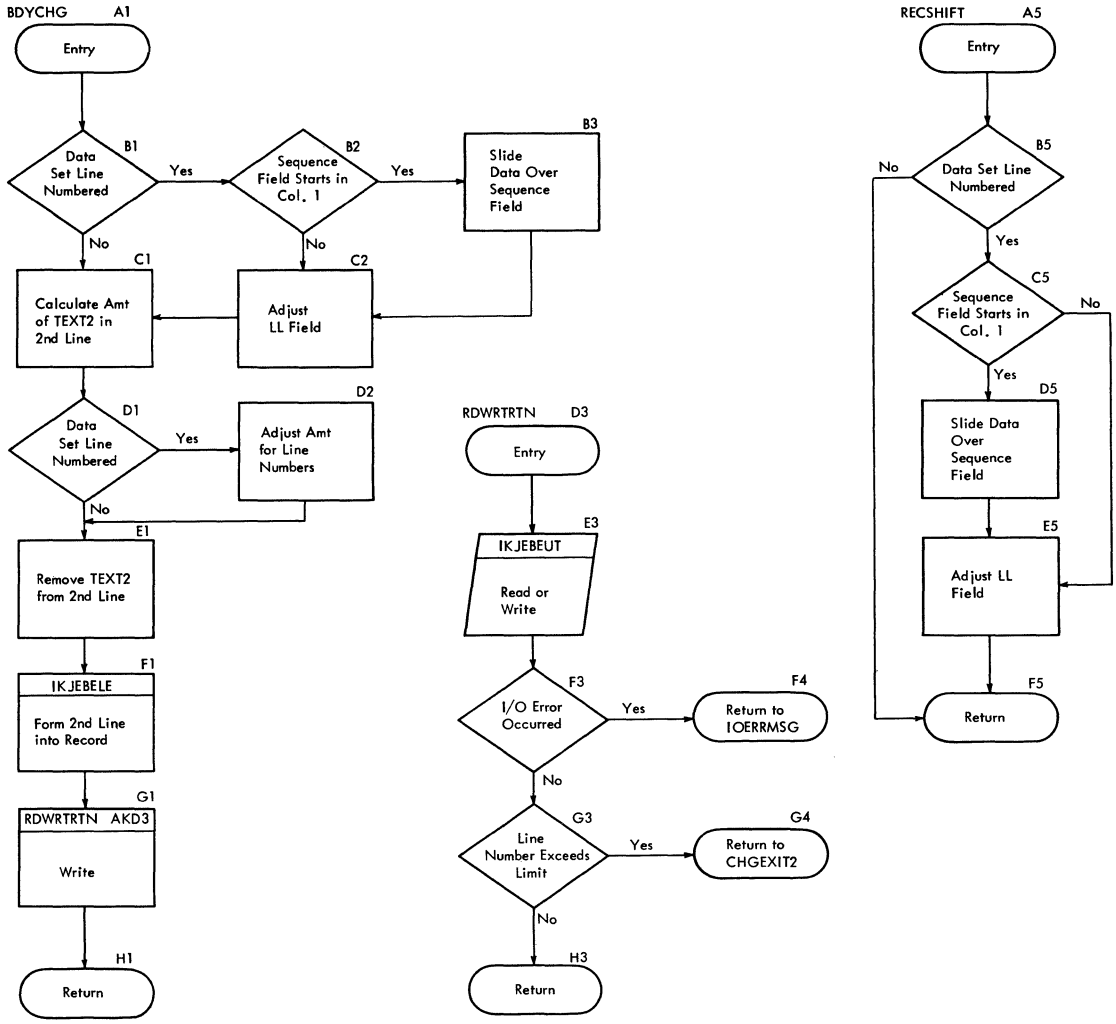


Chart AL. IKJEBECH

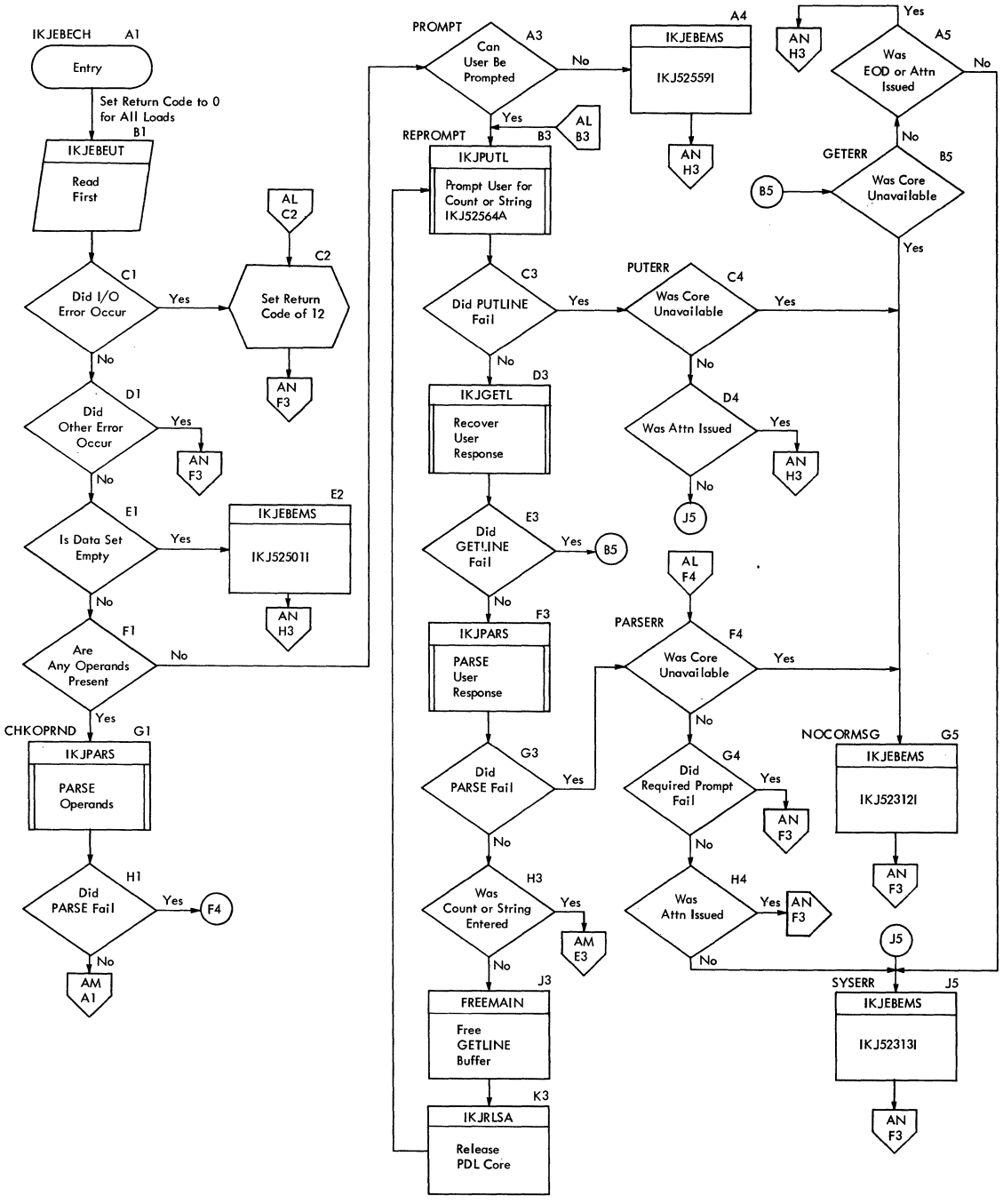


Chart AM. IKJEBECH

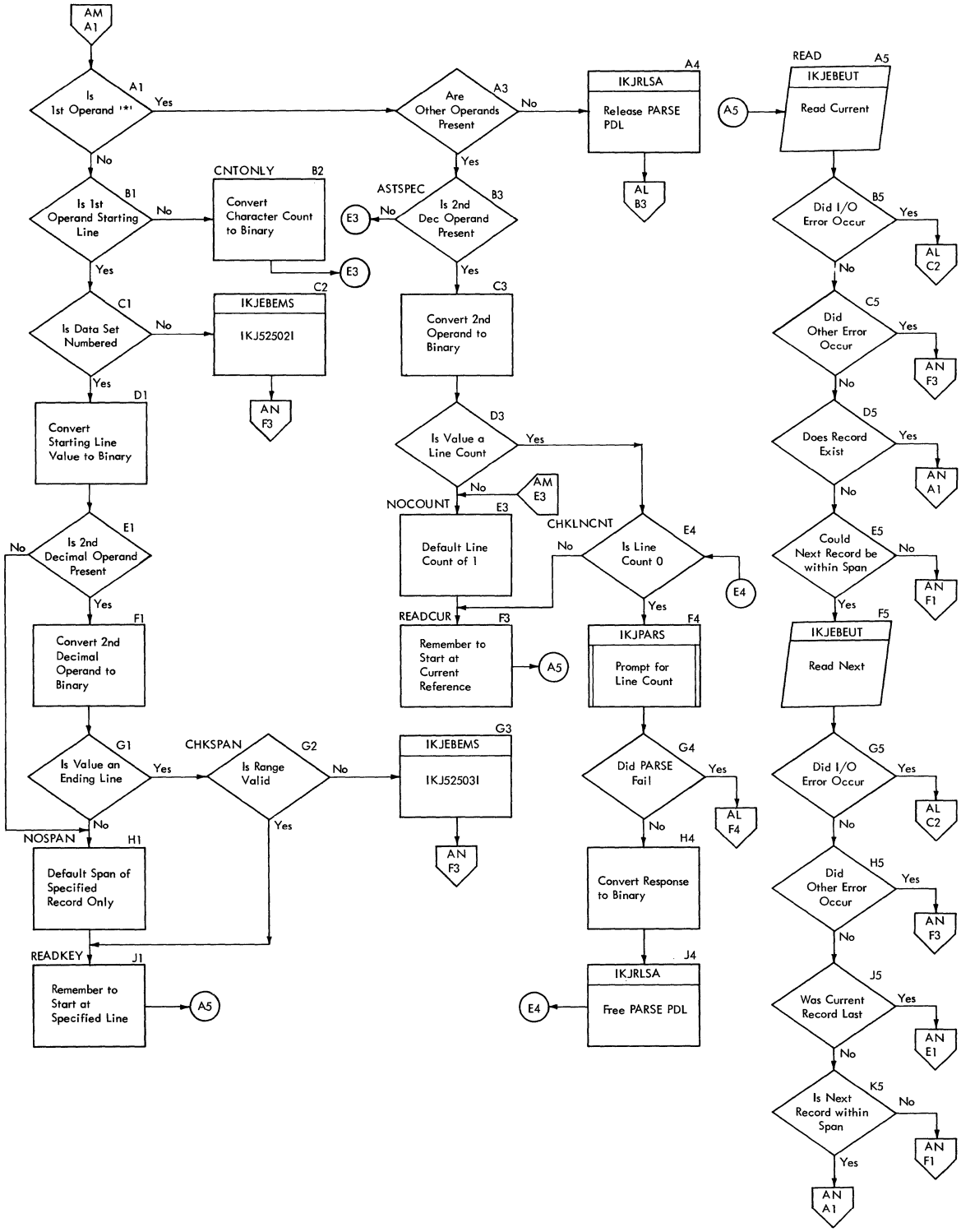


Chart AN. IKJEBECH

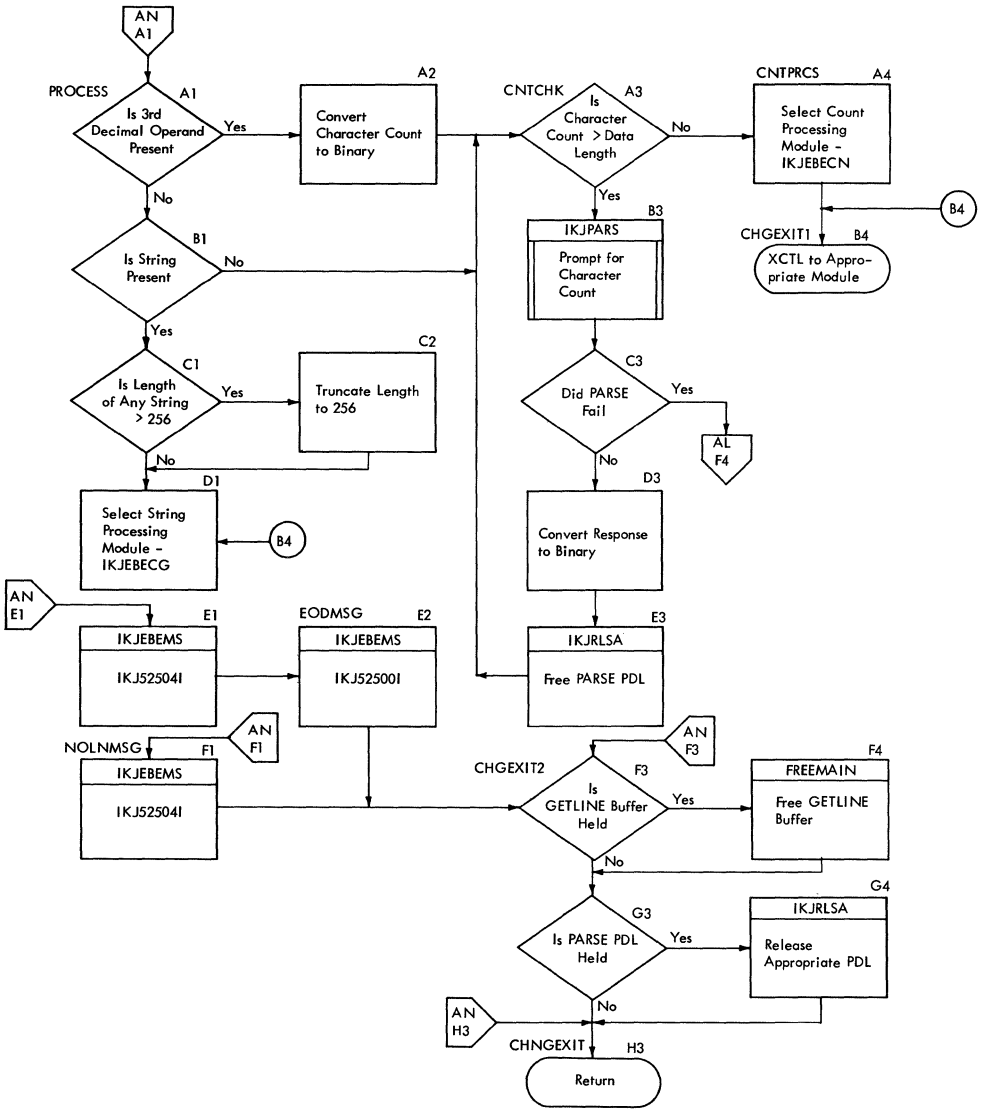


Chart AO. IKJEBECI

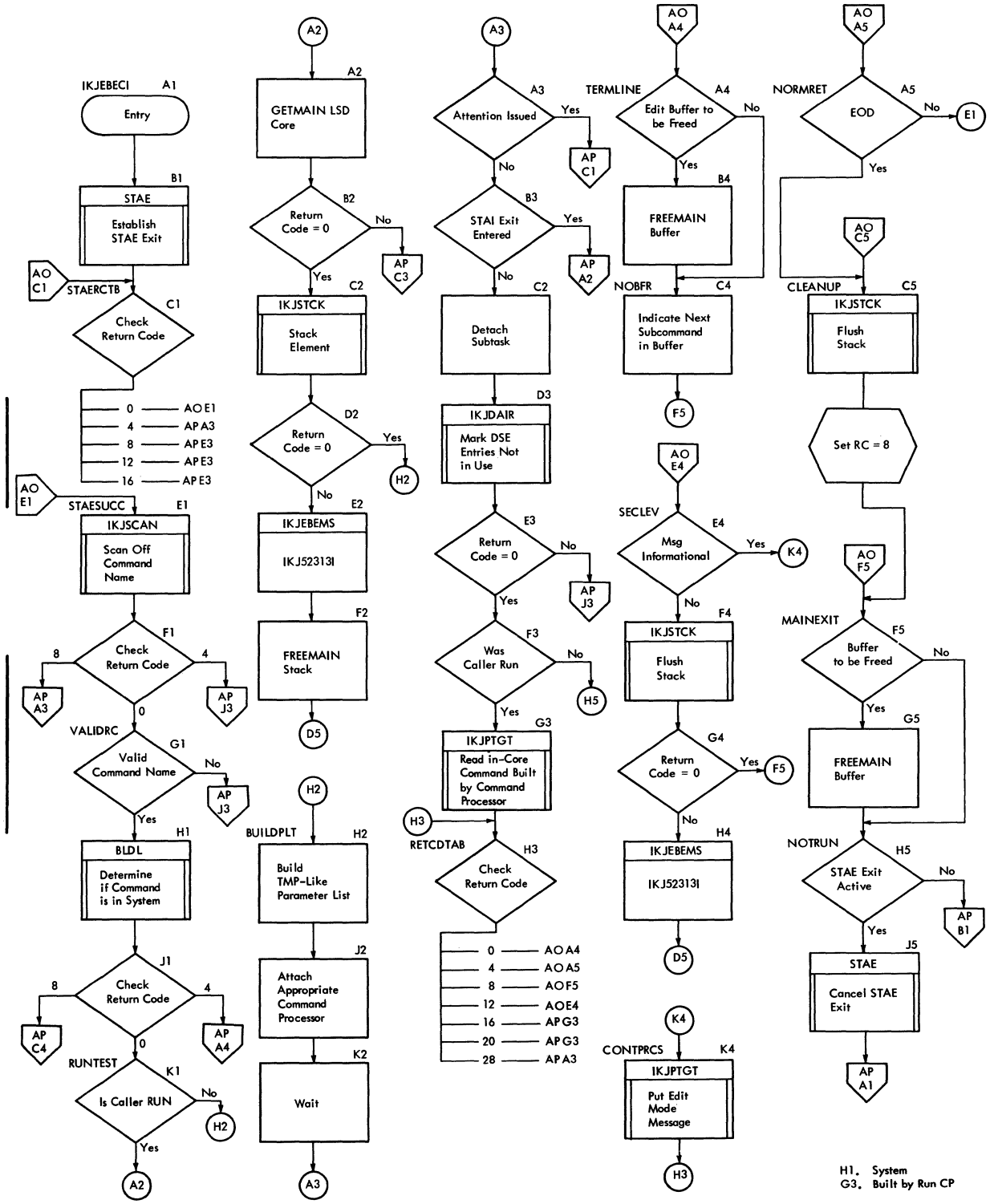


Chart AP. IKJEBECI

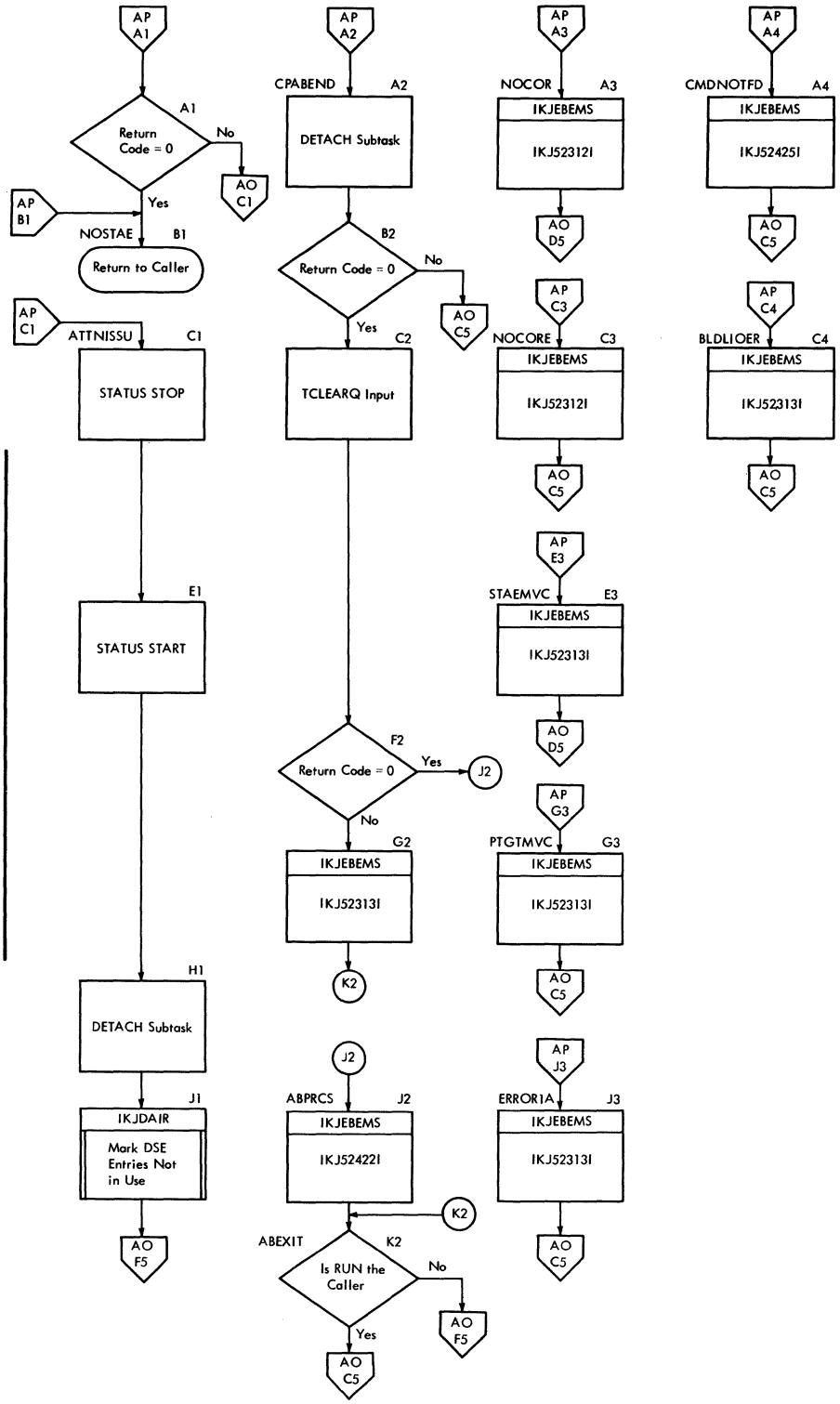


Chart AQ. IKJEBECN

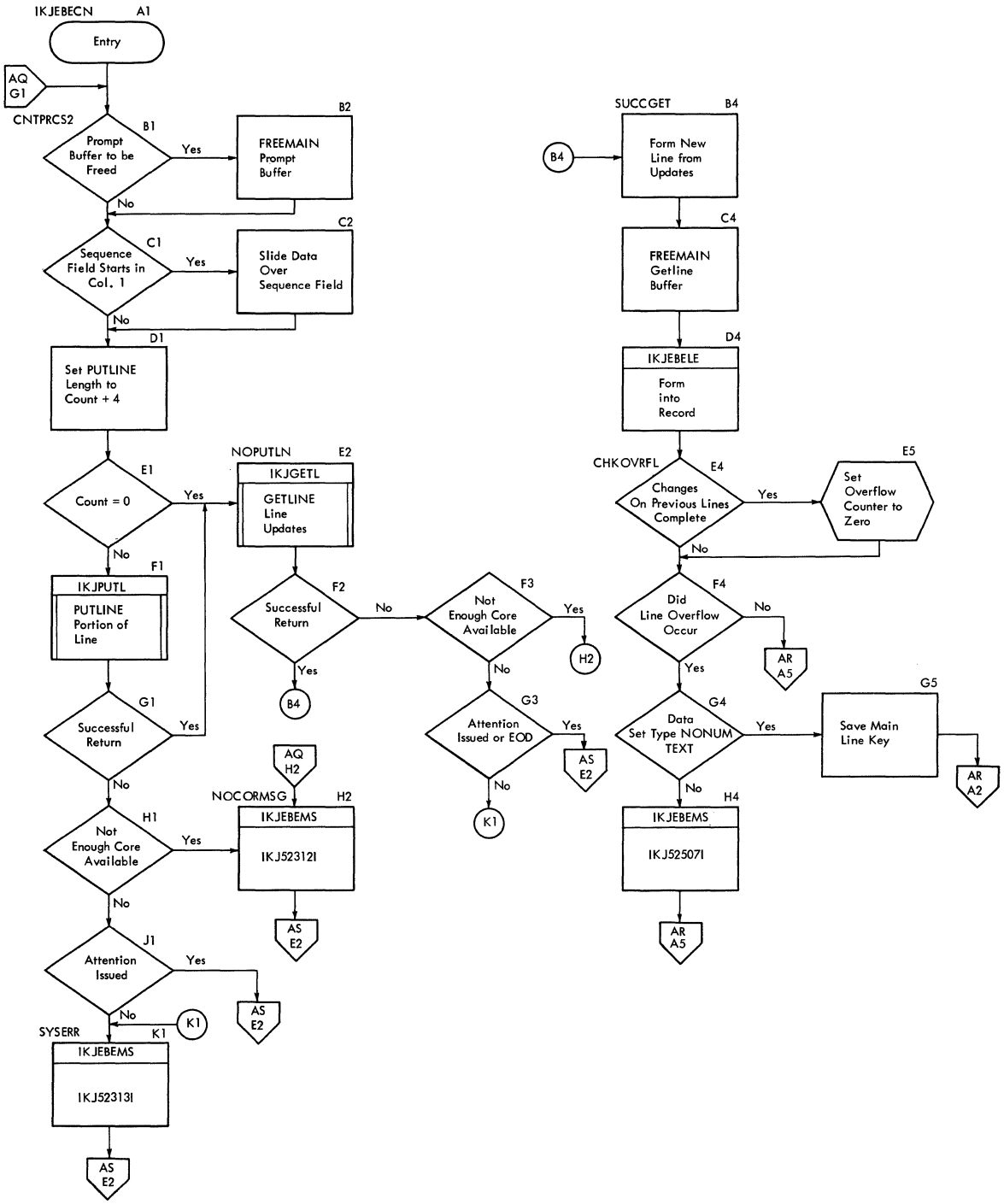


Chart AR. IKJEBECN

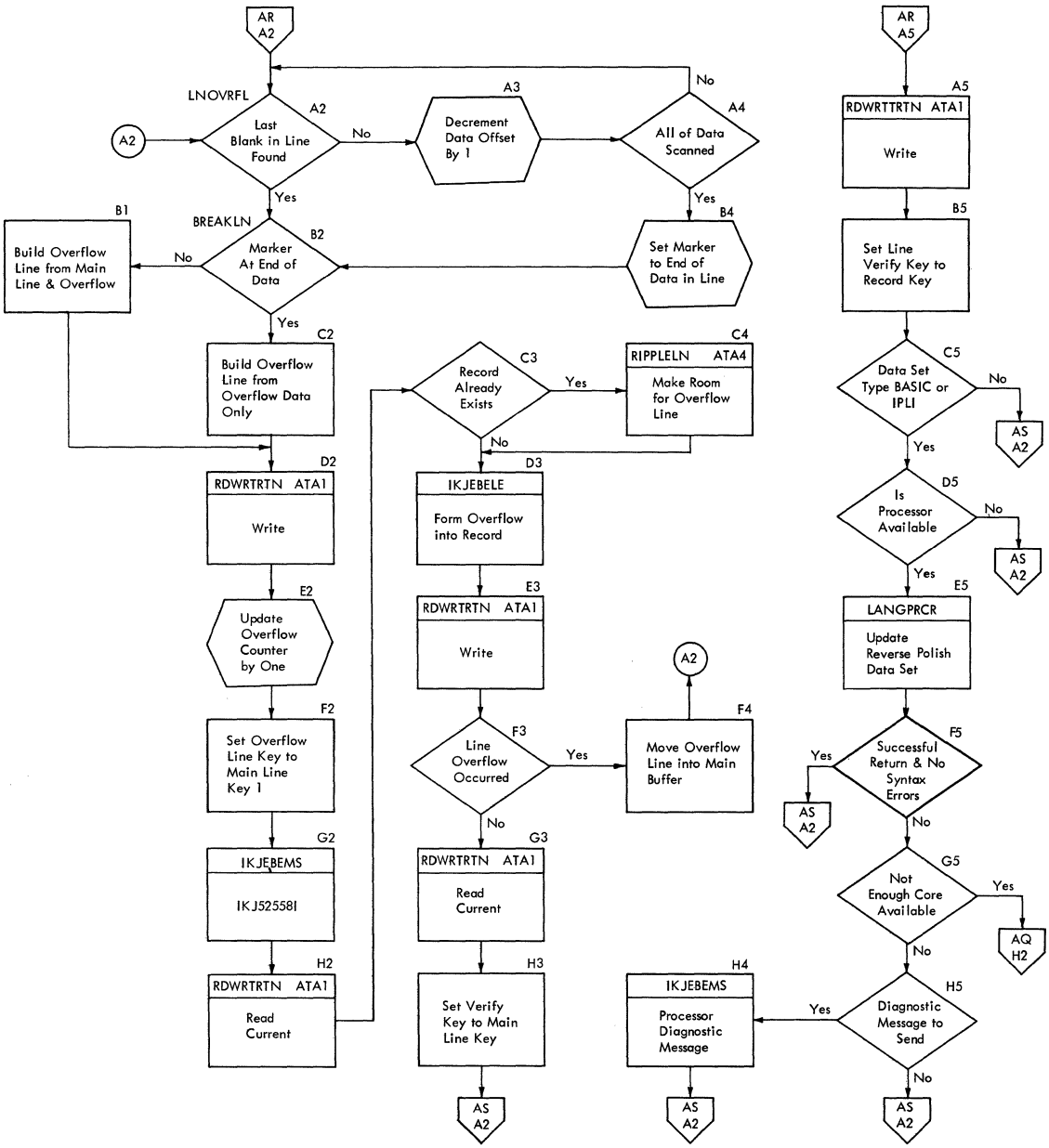


Chart AS. IKJEBECN

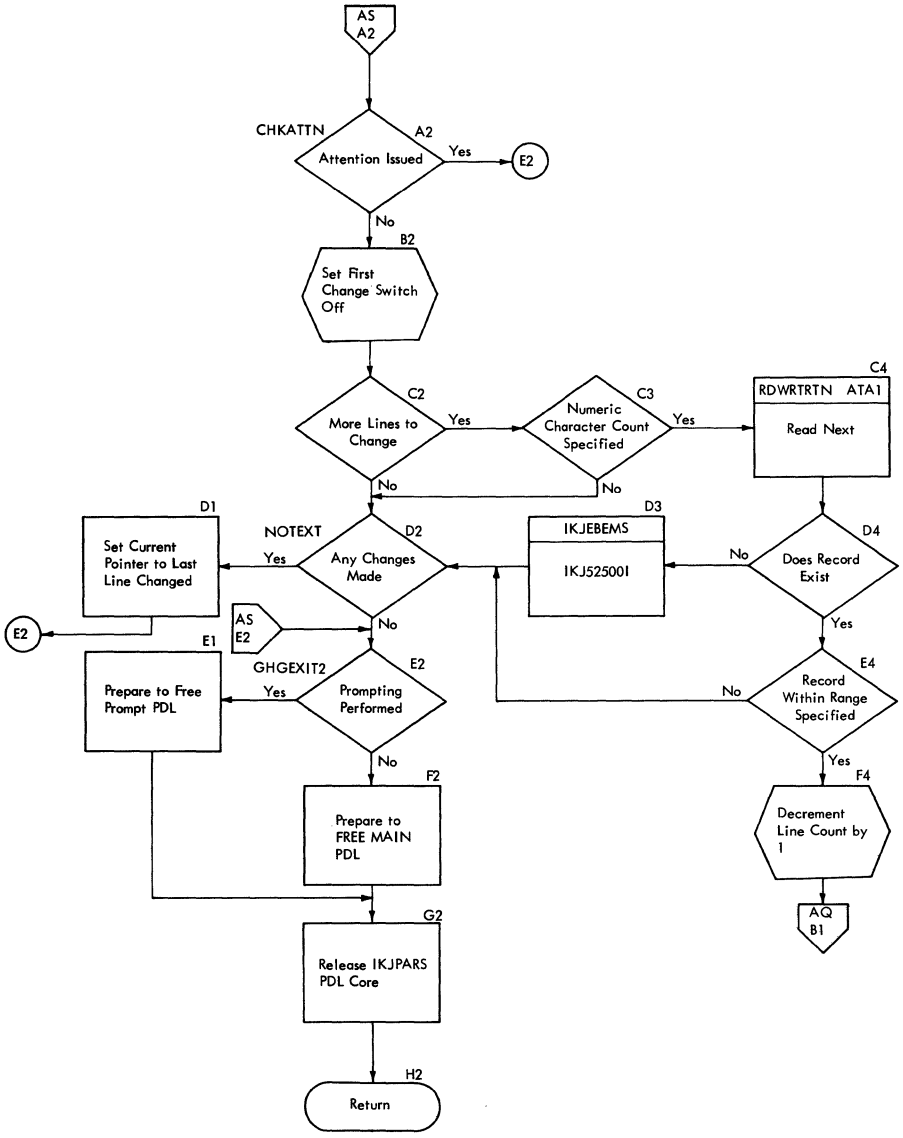
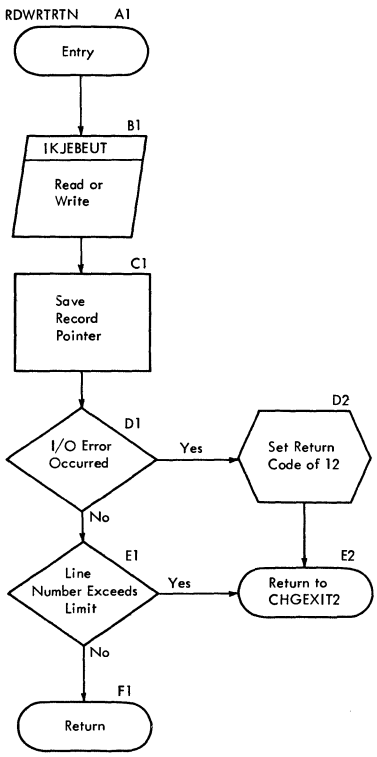


Chart AT. IKJEBECN



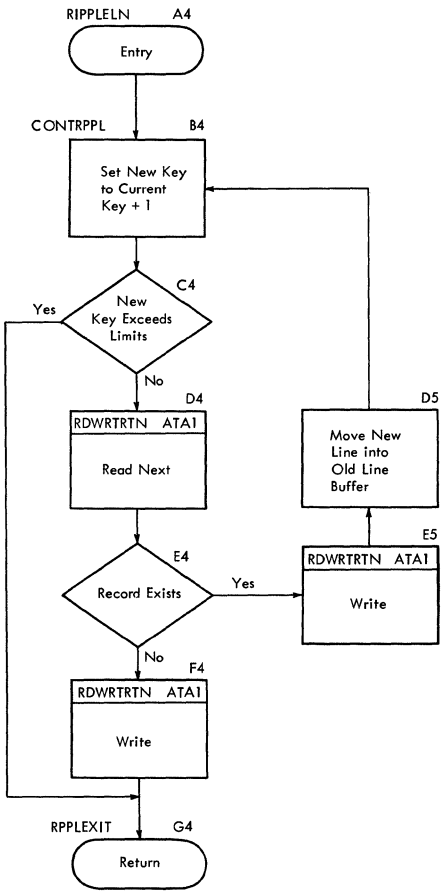


Chart AU. IKJEBECO

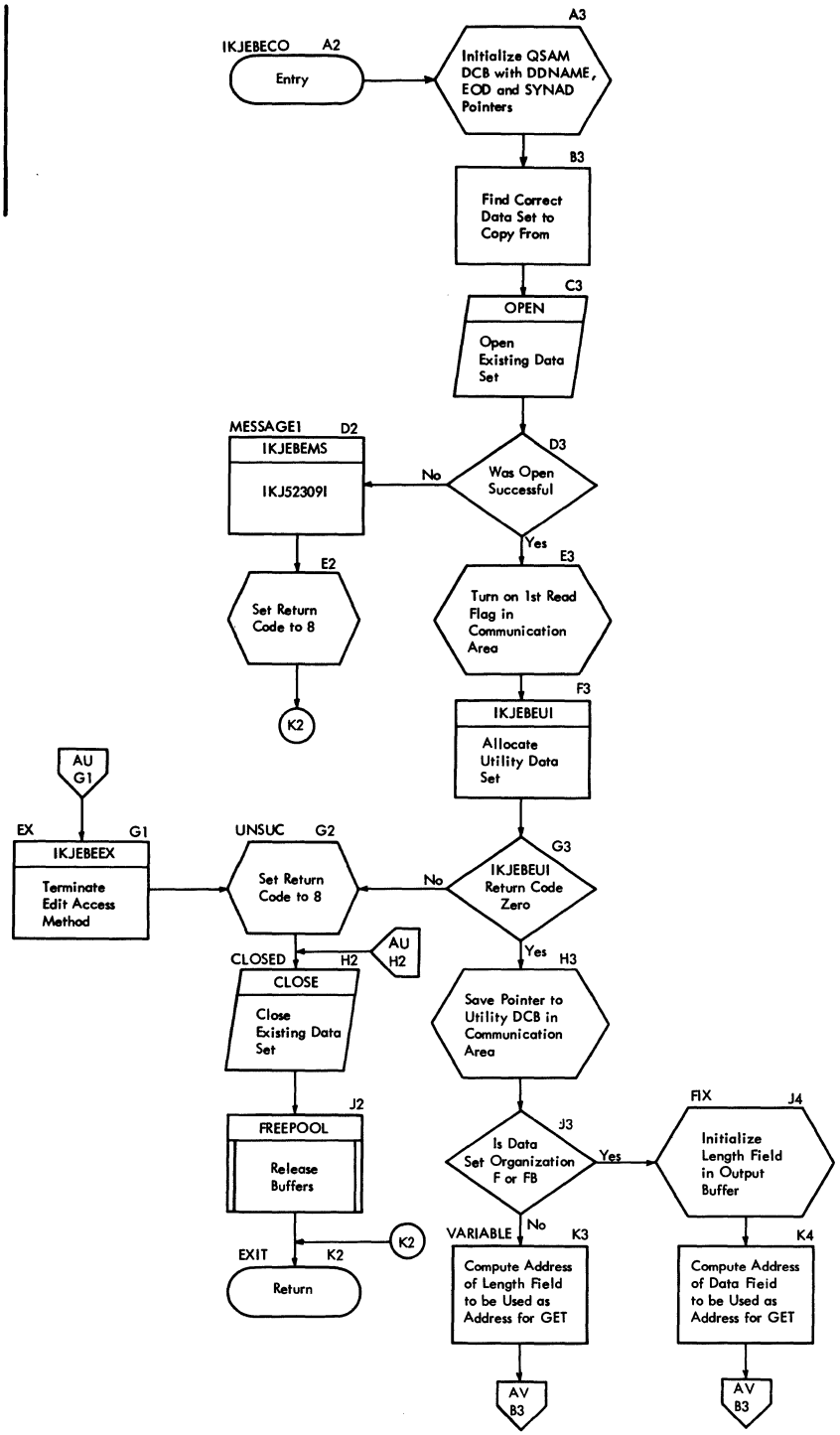


Chart AV. IKJEBECO

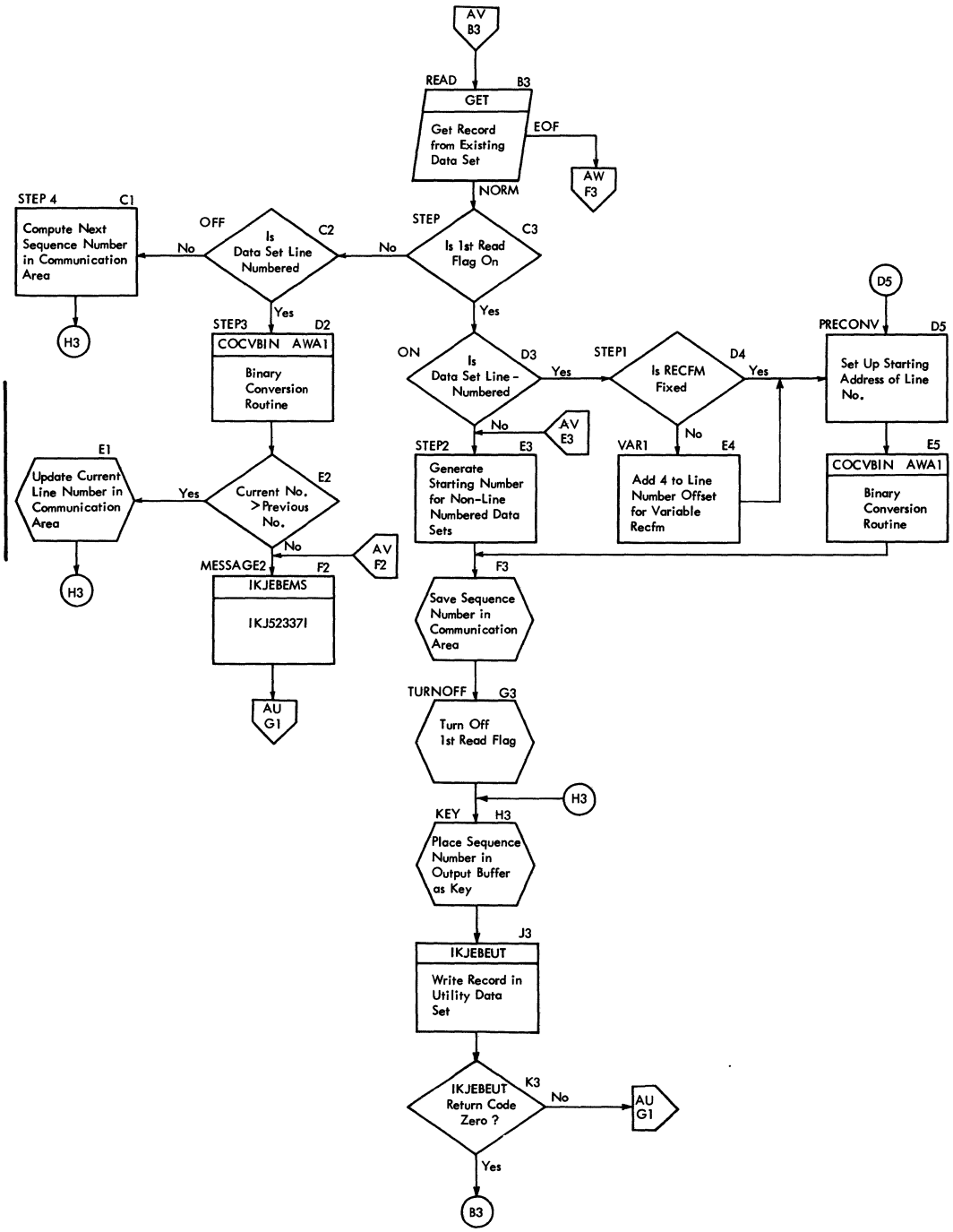


Chart AW. IKJEBECO

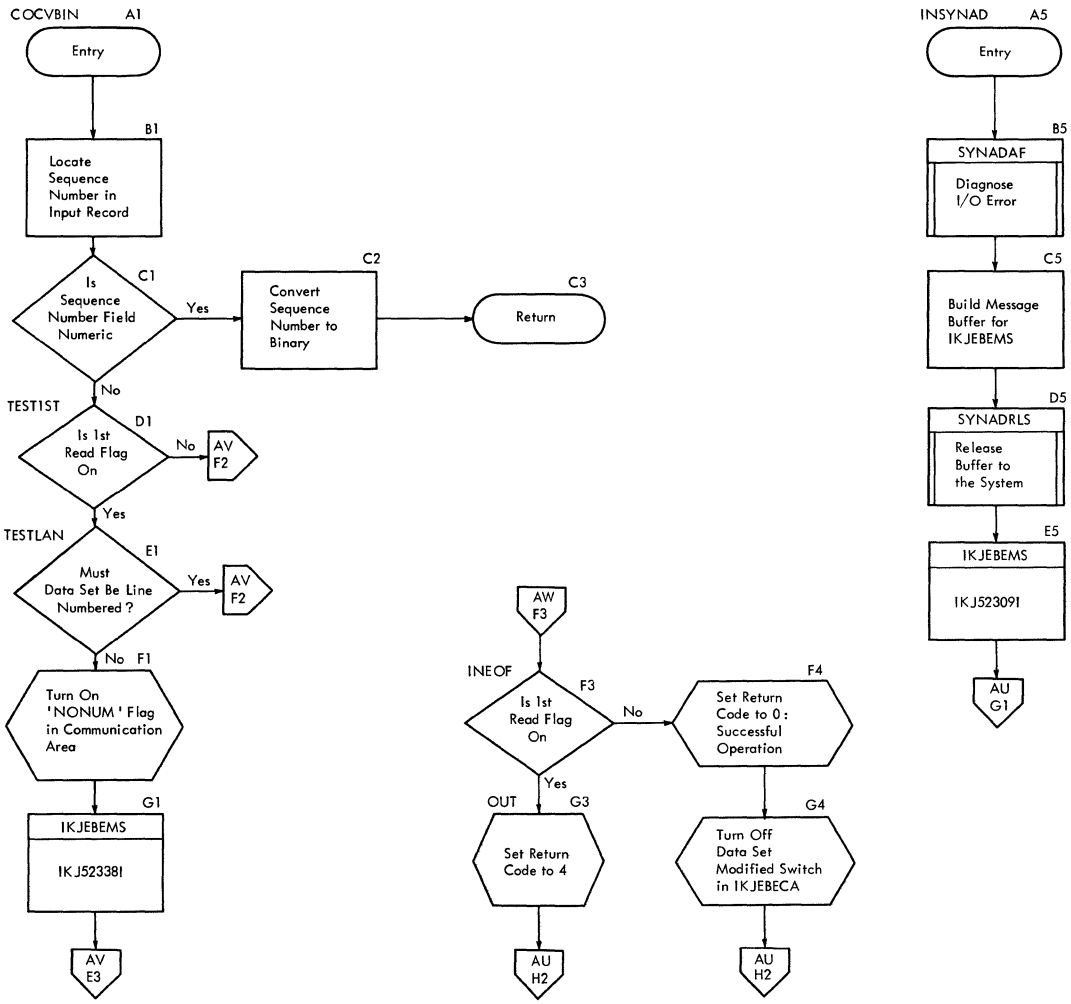


Chart AX. IKJEBEDA

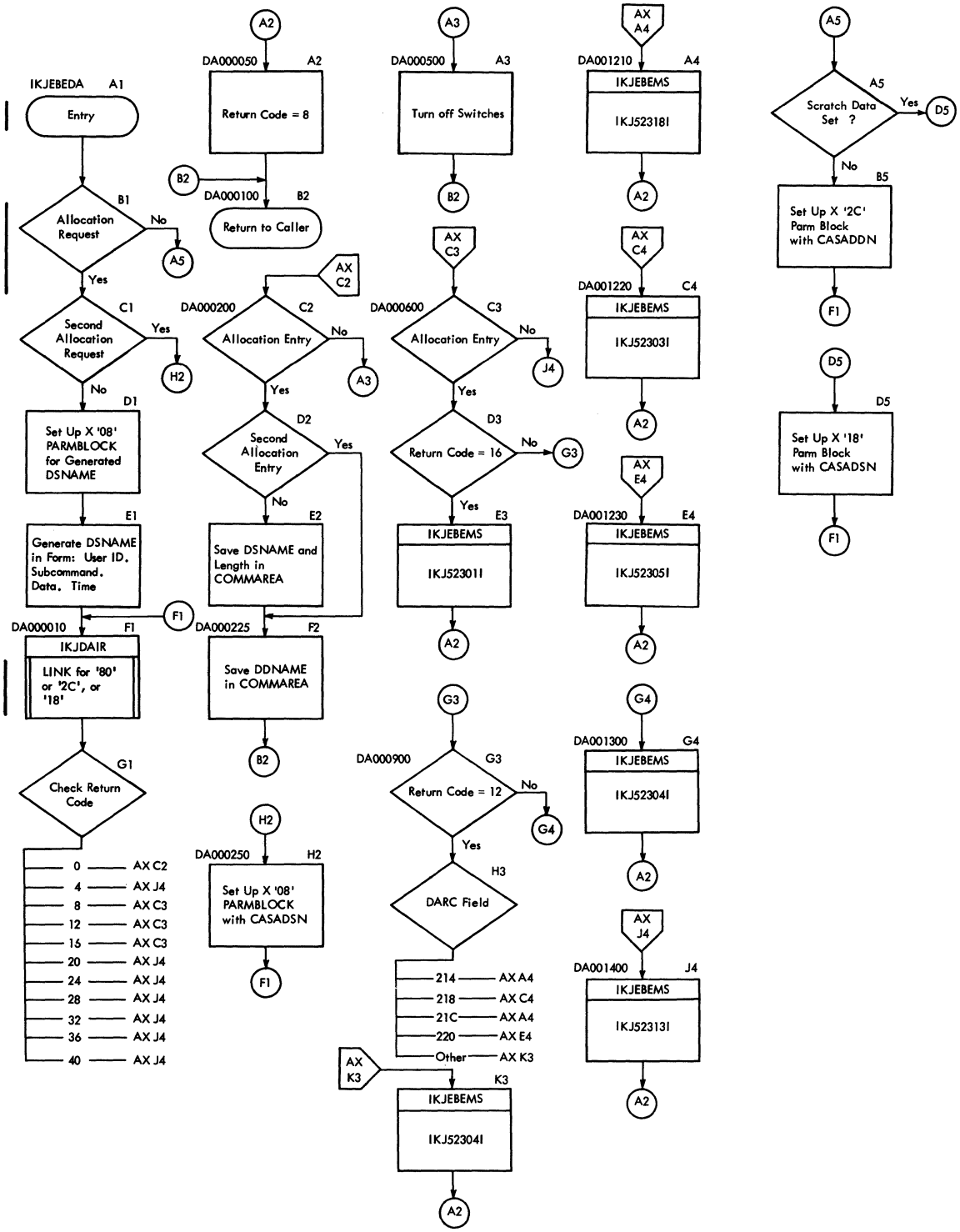


Chart AY. IKJEBEDE

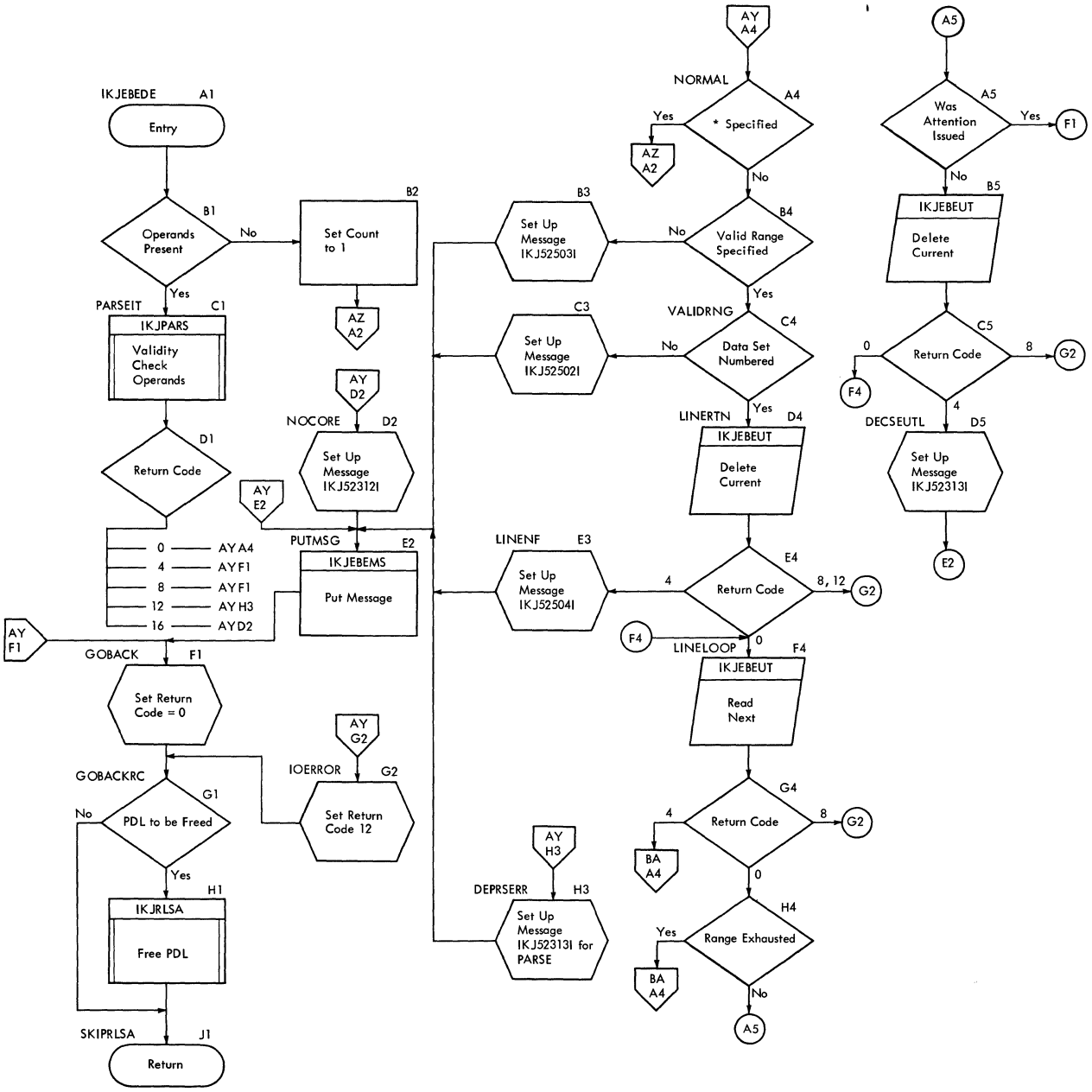


Chart AZ. IKJEBEDE

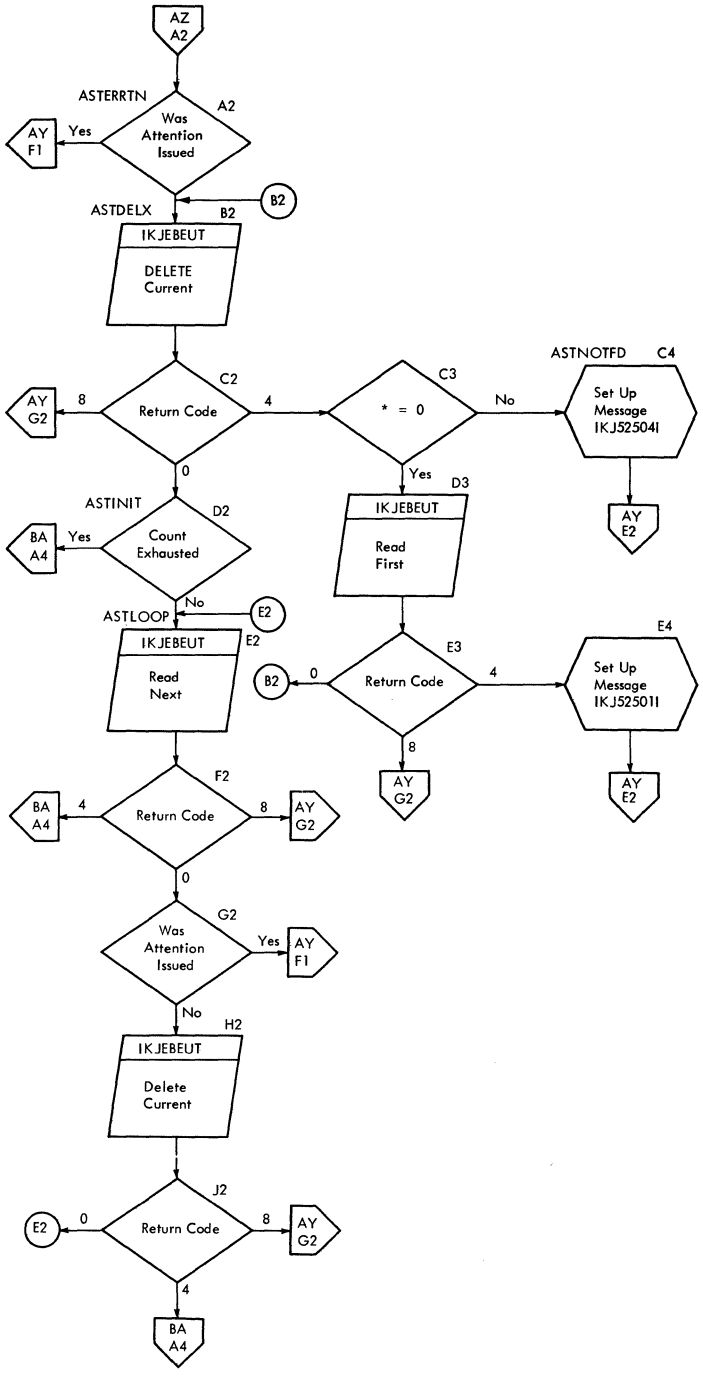


Chart BA. IKJEBEDE

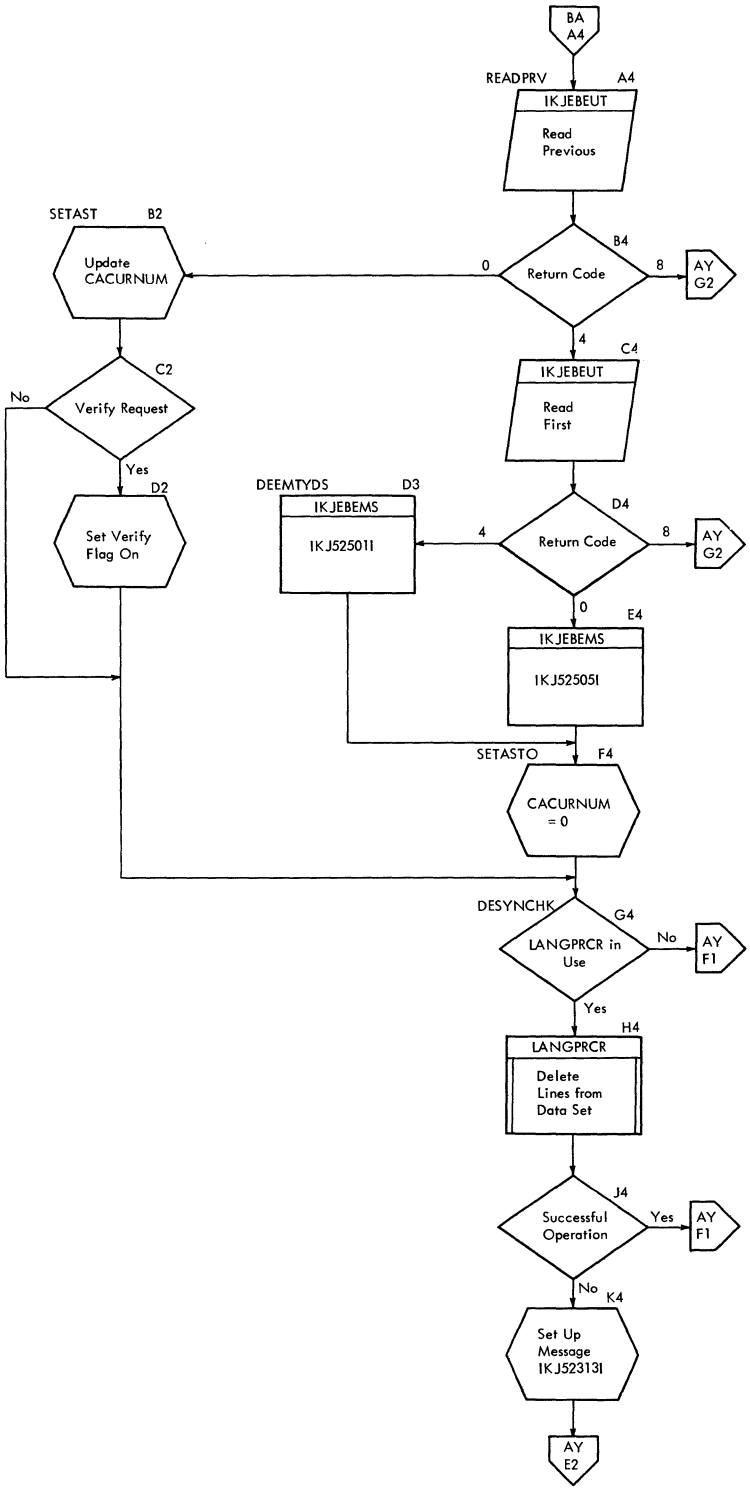


Chart BB. IKJEBEDL

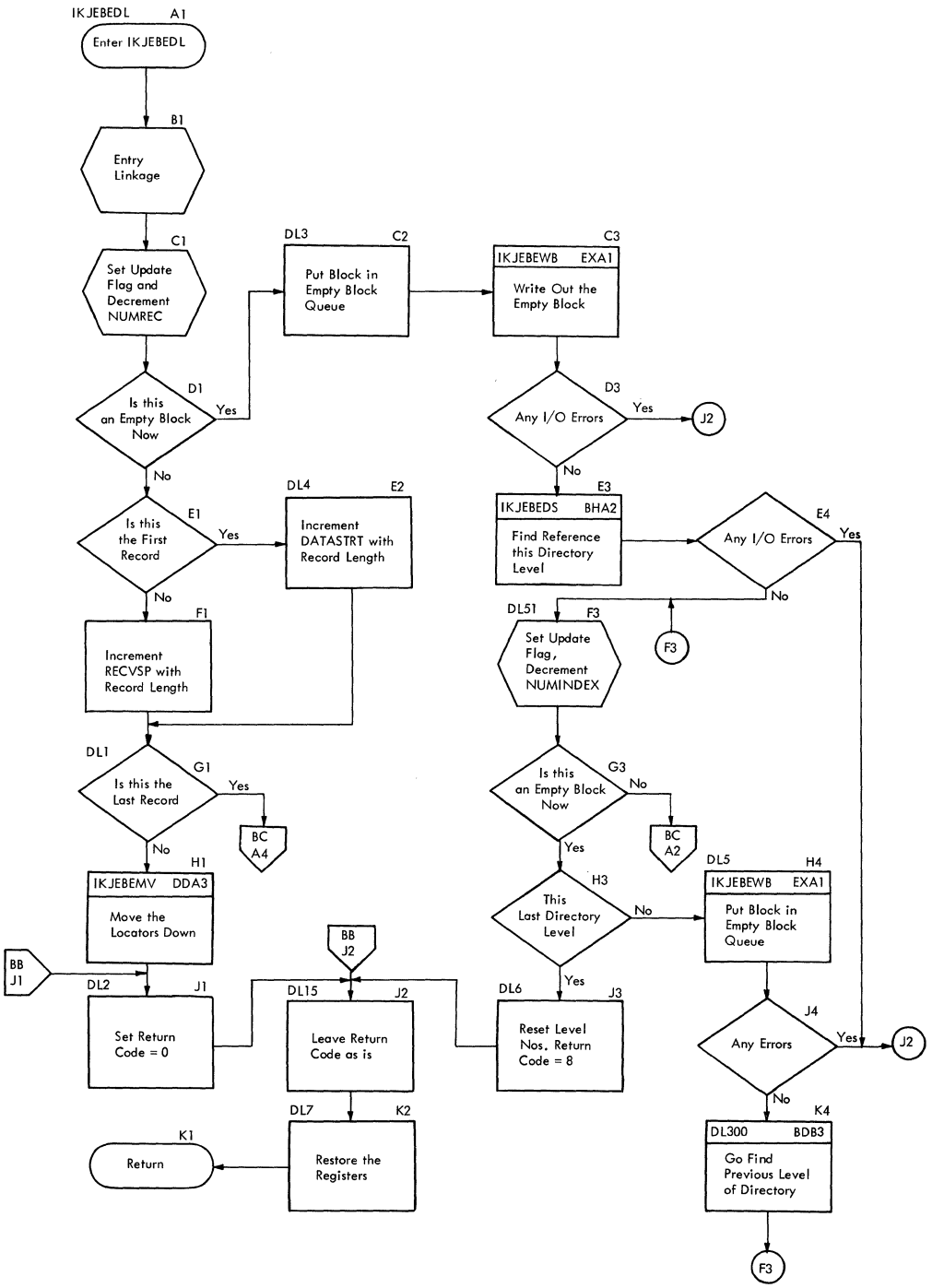


Chart BC. IKJEBEDL

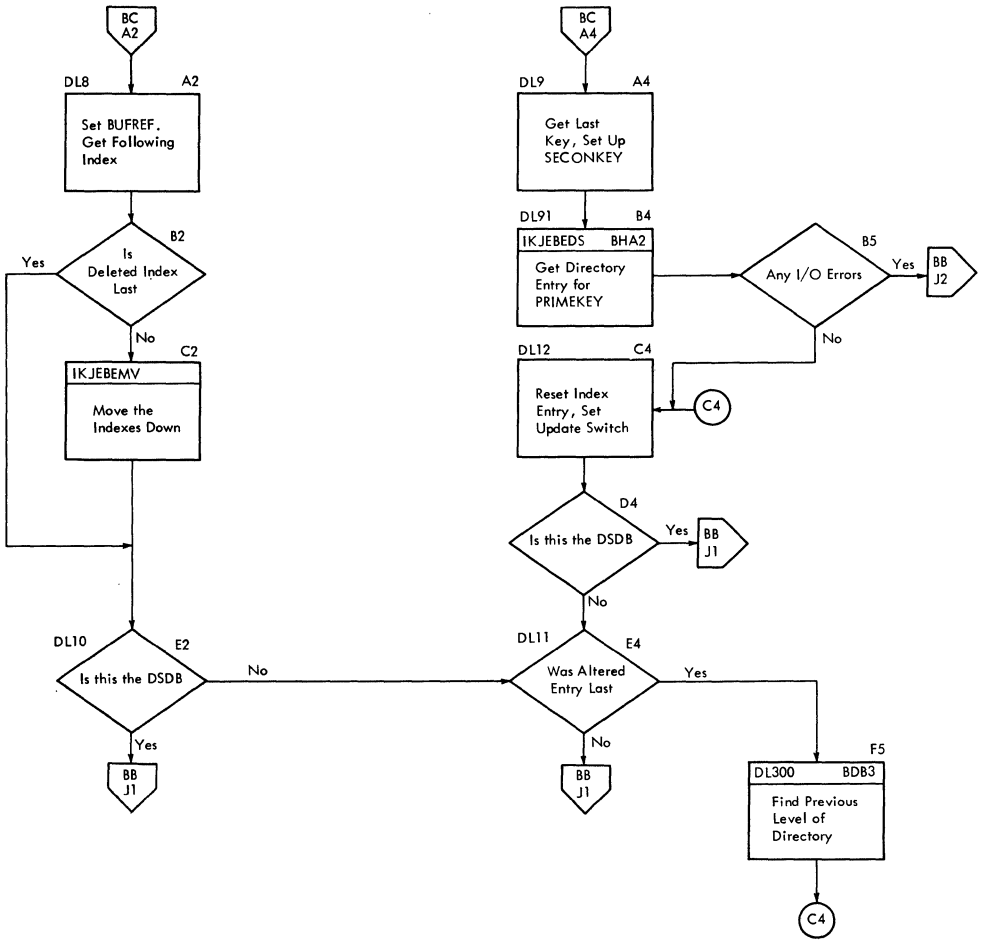


Chart BD. IKJEBEDL

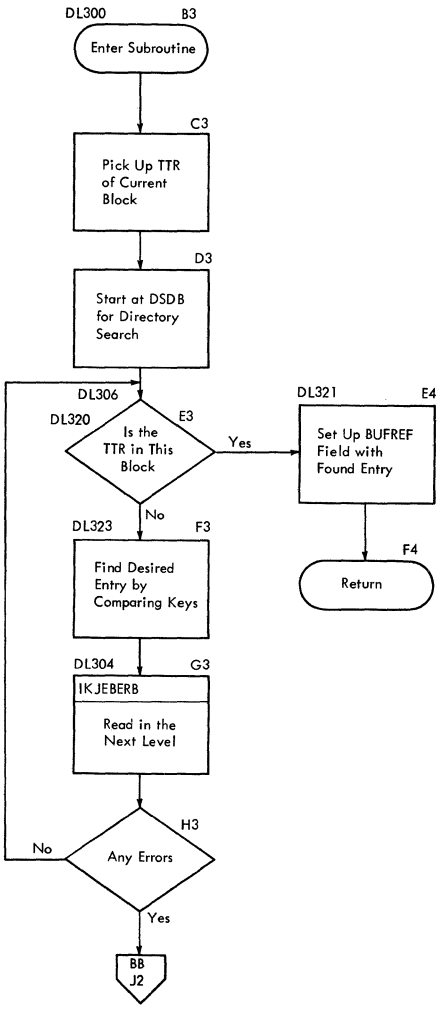


Chart BE. IKJEBEDO

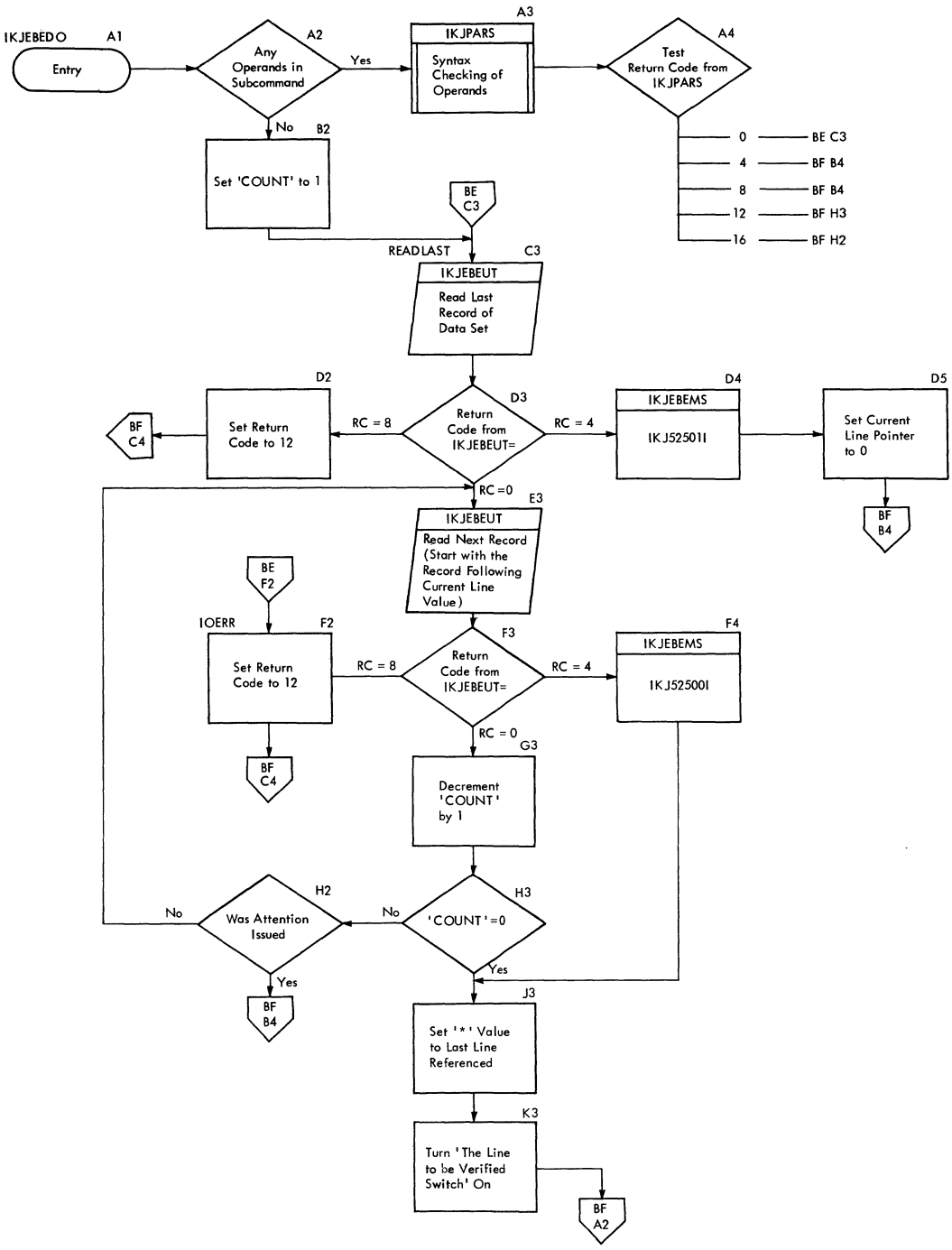


Chart BF. IKJEBEDO

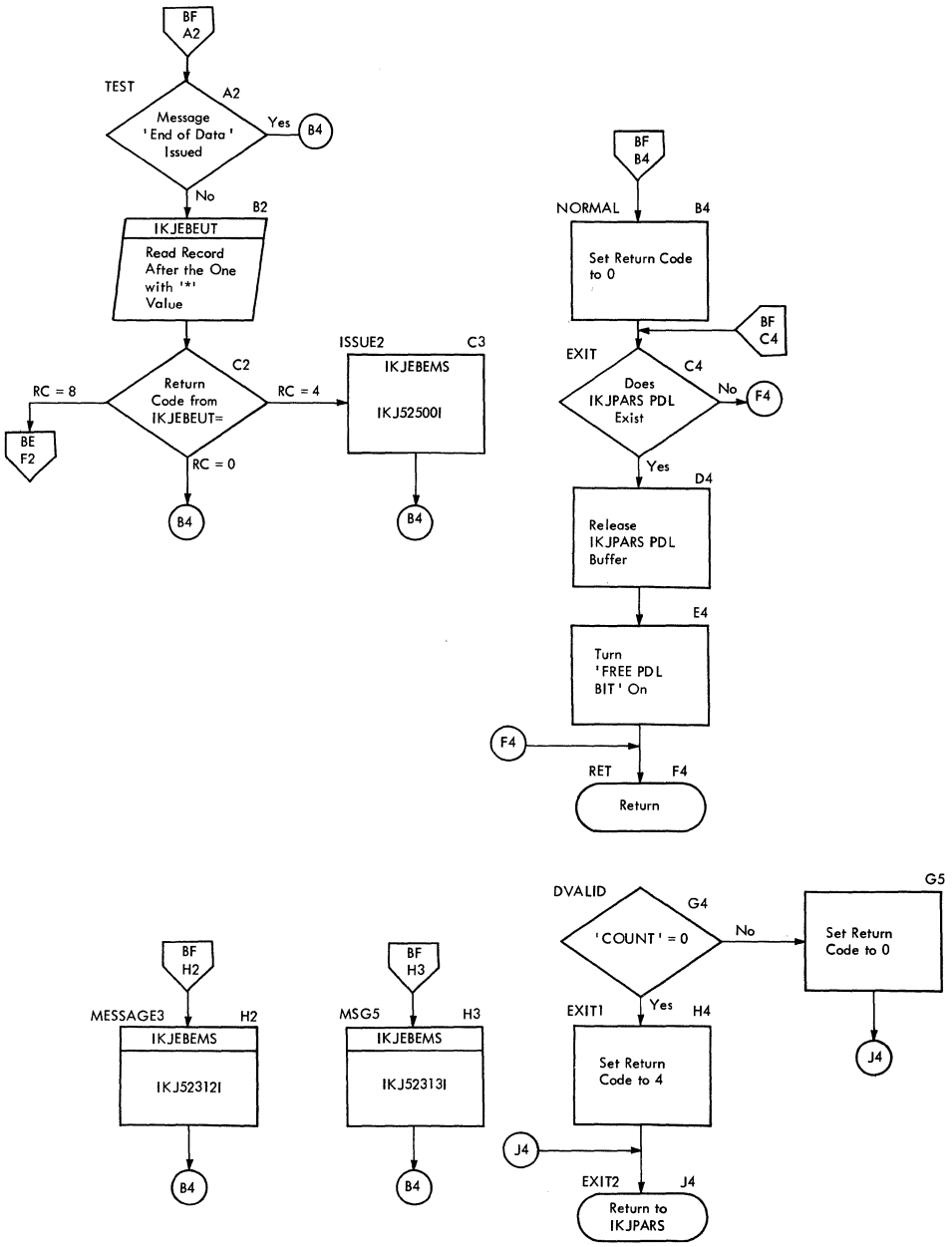


Chart BG. IKJEBEDR

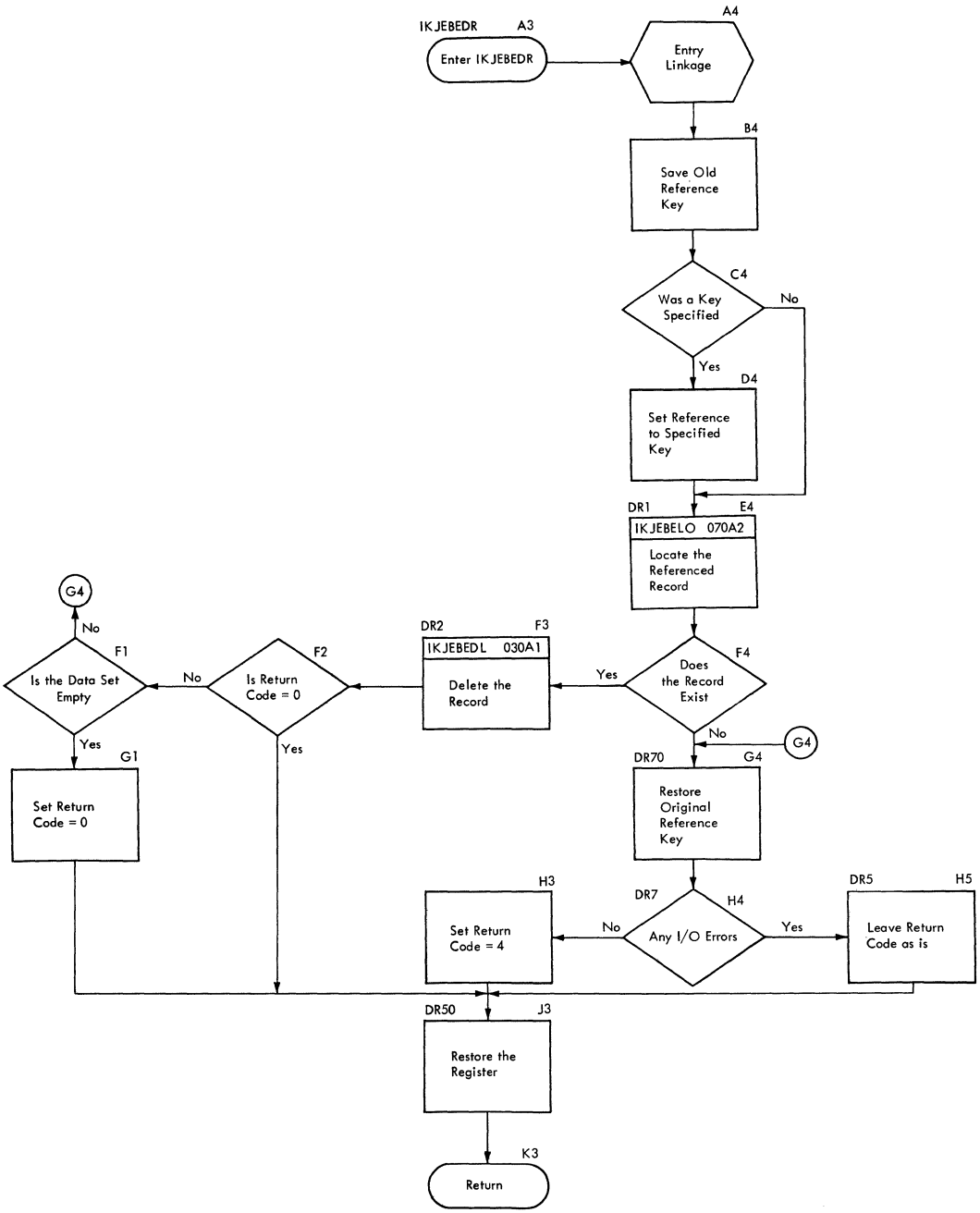


Chart BH. IKJEBEDS

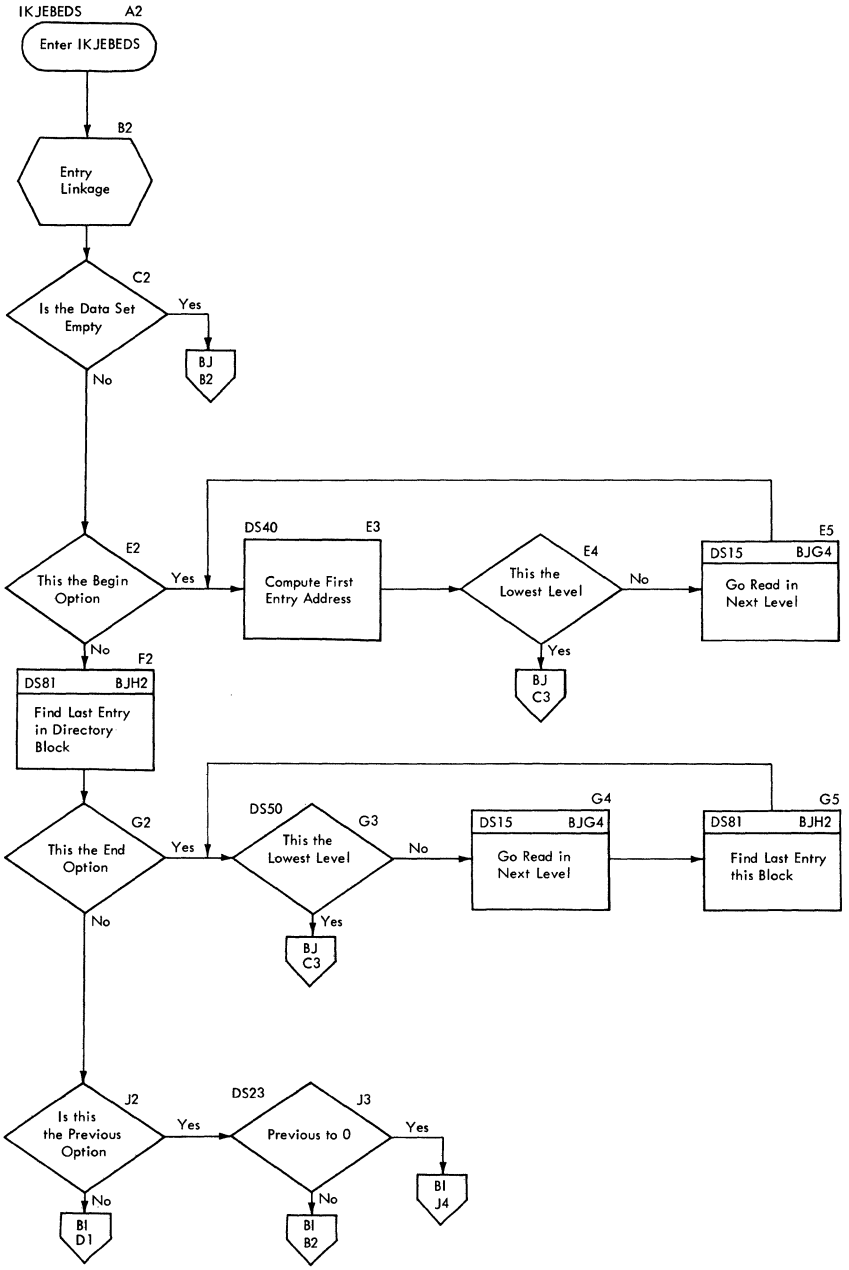


Chart BI. IKJEBEDS

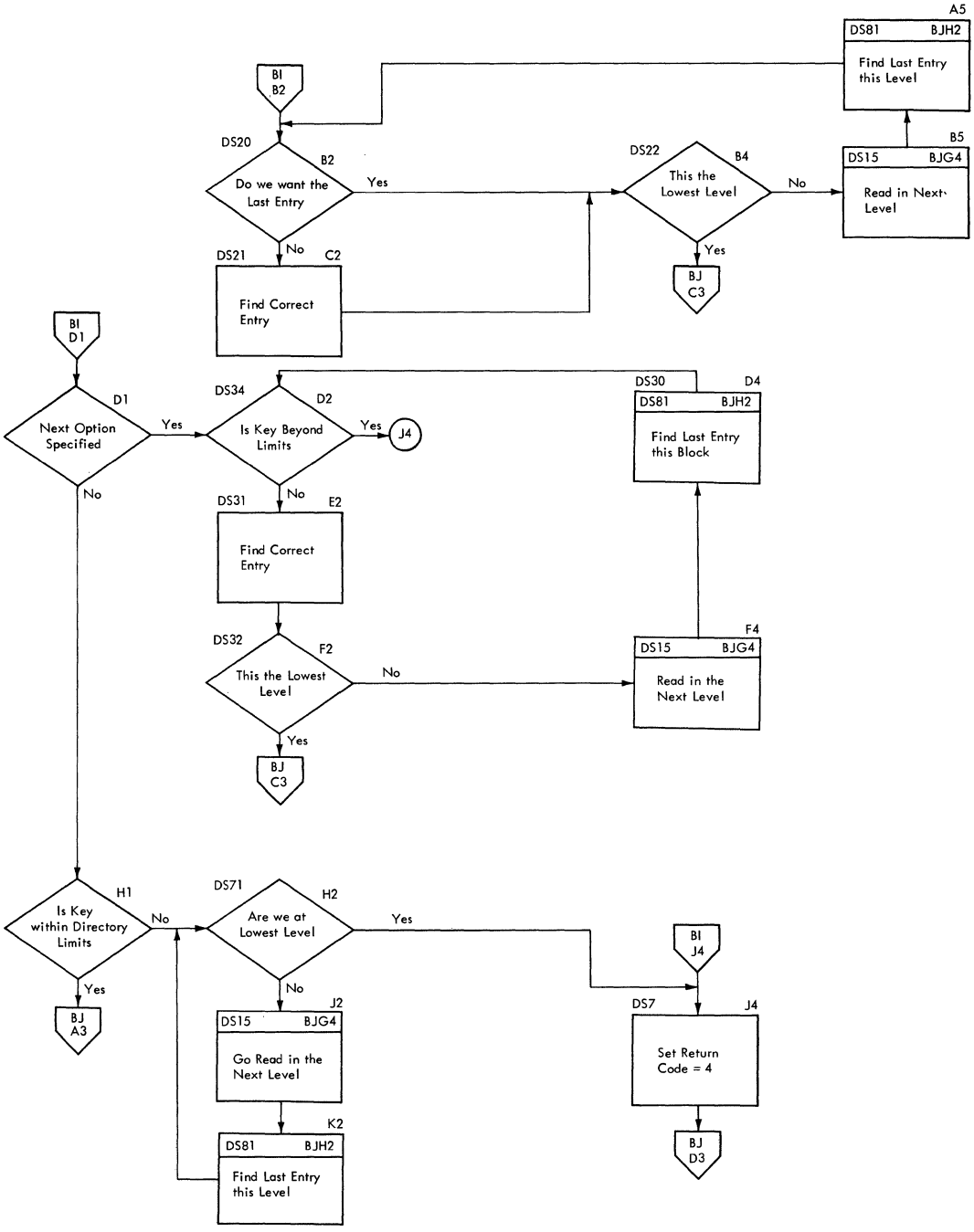
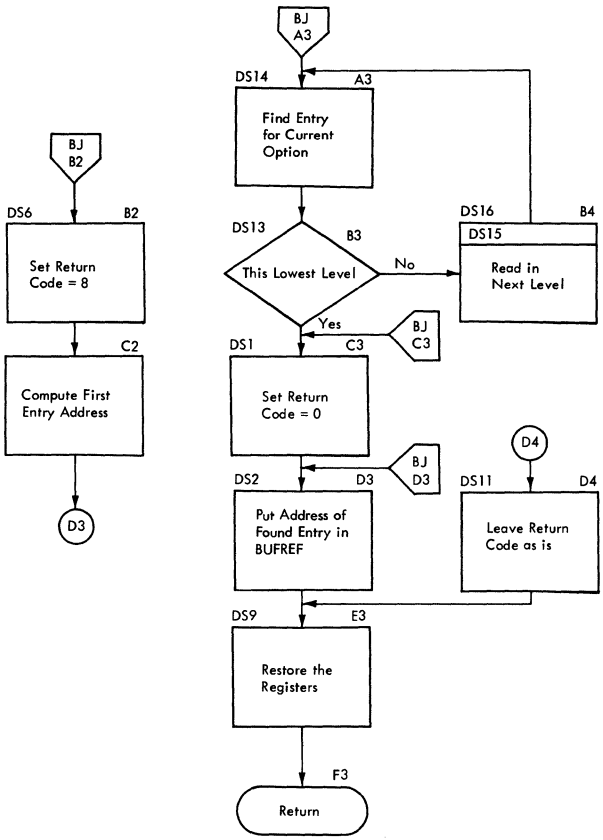


Chart BJ. IKJEBEDS



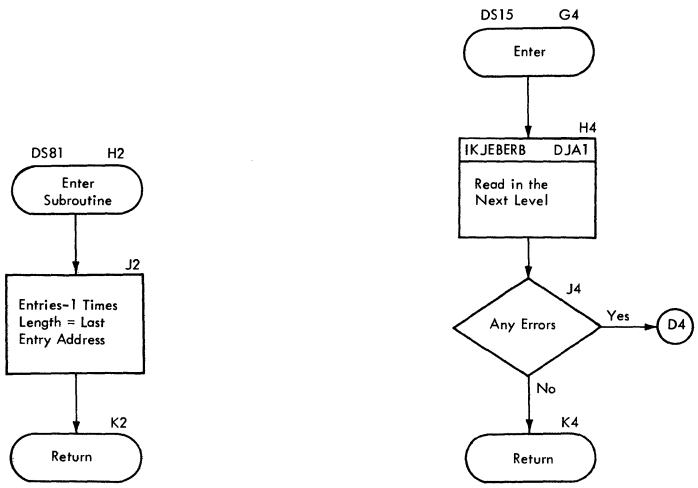
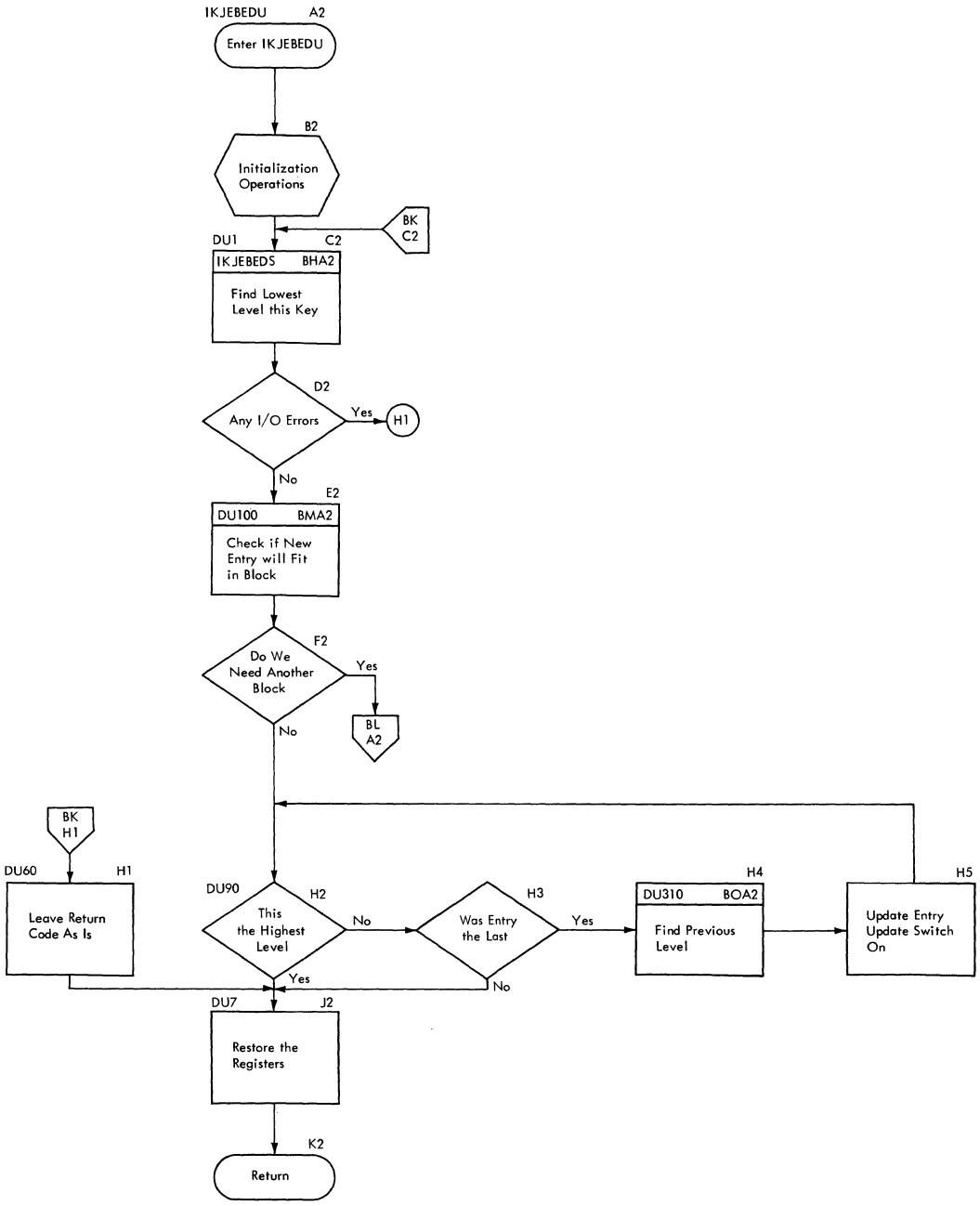


Chart BK. IKJEBEDU



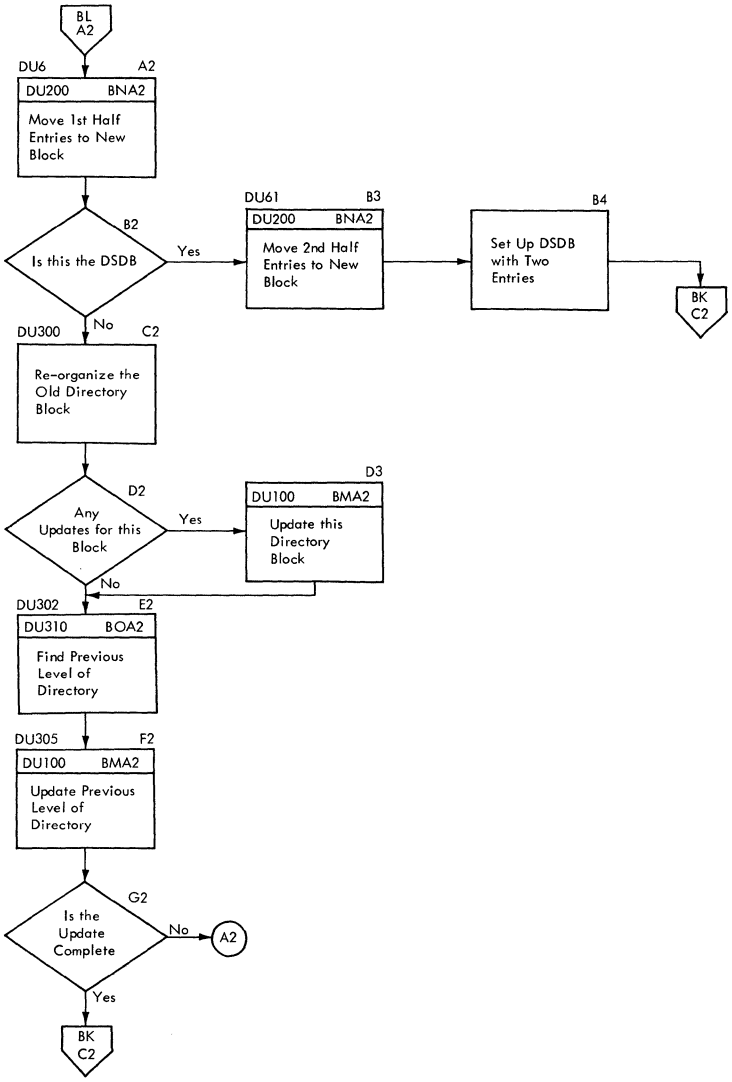

BL
A2
DU6
A2
DU200
BNA2
Move 1st Half Entries to New Block
B2
Is this the DSDB
Yes
DU61
B3
DU200
BNA2
Move 2nd Half Entries to New Block
B4
Set Up DSDB with Two Entries
BK
C2
No
DU300
C2
Re-organize the Old Directory Block
D3
D2
Any Updates for this Block
Yes
DU100
BMA2
Update this Directory Block
No
DU302
E2
DU310
BOA2
Find Previous Level of Directory
DU305
F2
DU100
BMA2
Update Previous Level of Directory
G2
Is the Update Complete
No
A2
Yes
BK
C2

Chart BM. IKJEBEDU

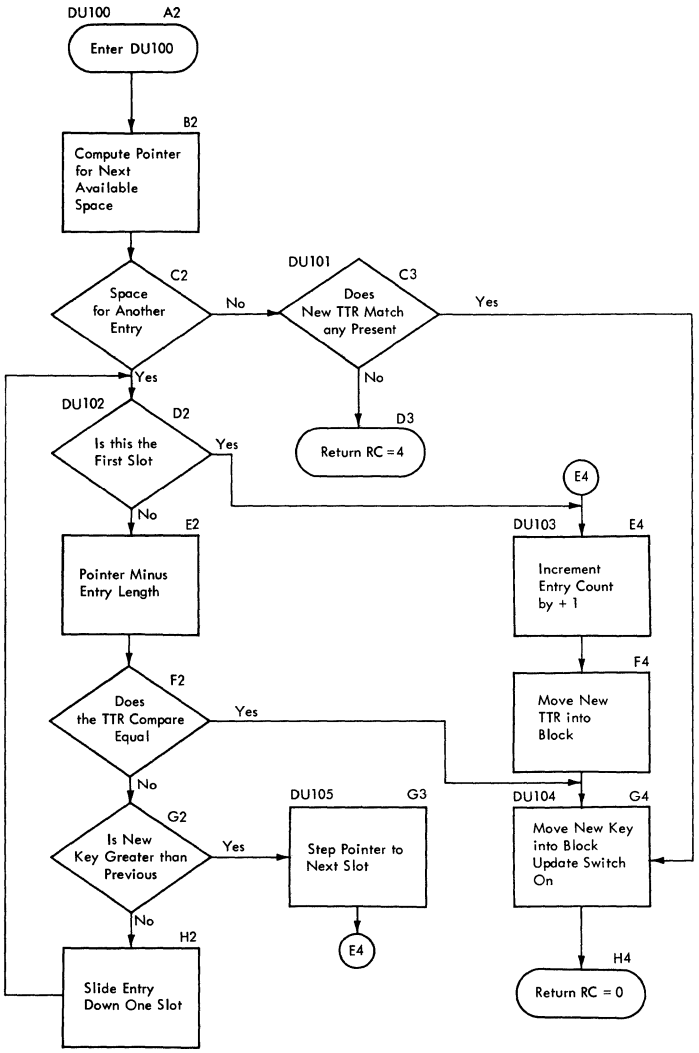


Chart BN. IKJEBEDU

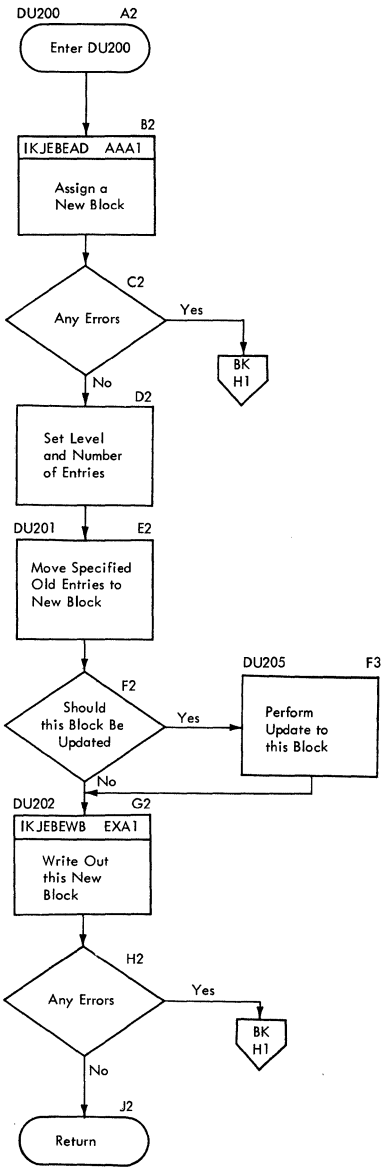


Chart BO. IKJEBEDU

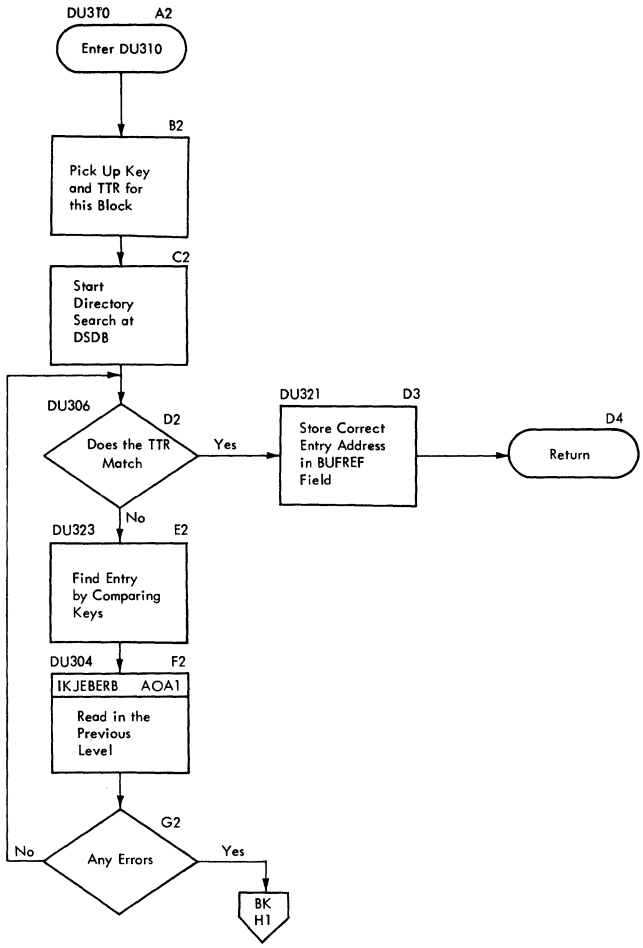


Chart BP. IKJEBEEN

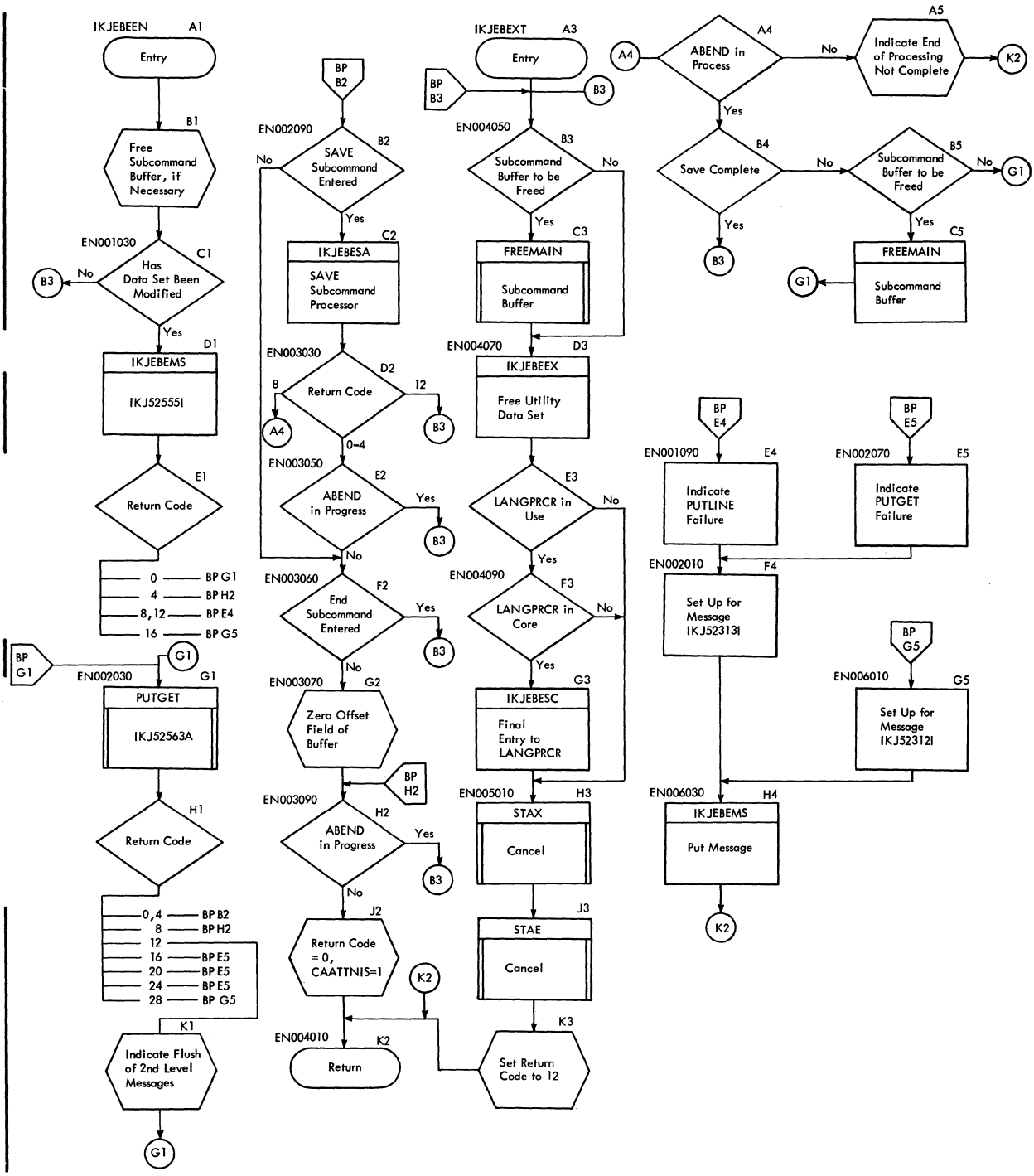


Chart BQ. IKJEBEEX

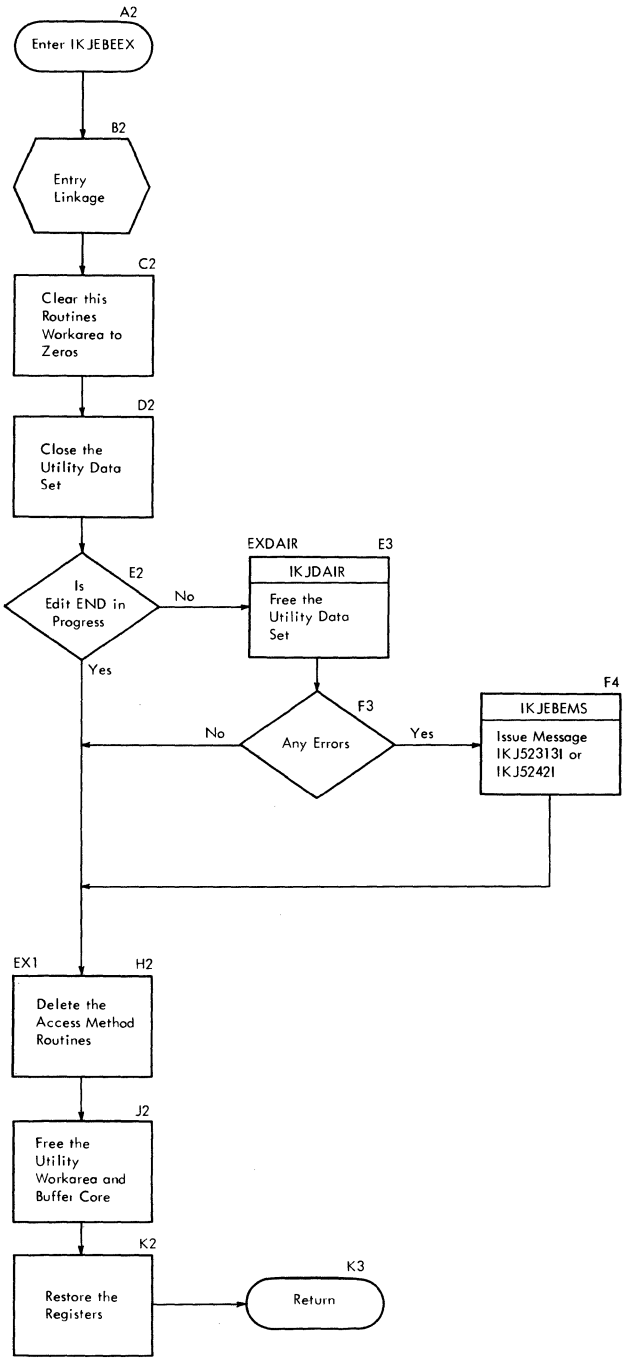


Chart BR. IKJEBEFC

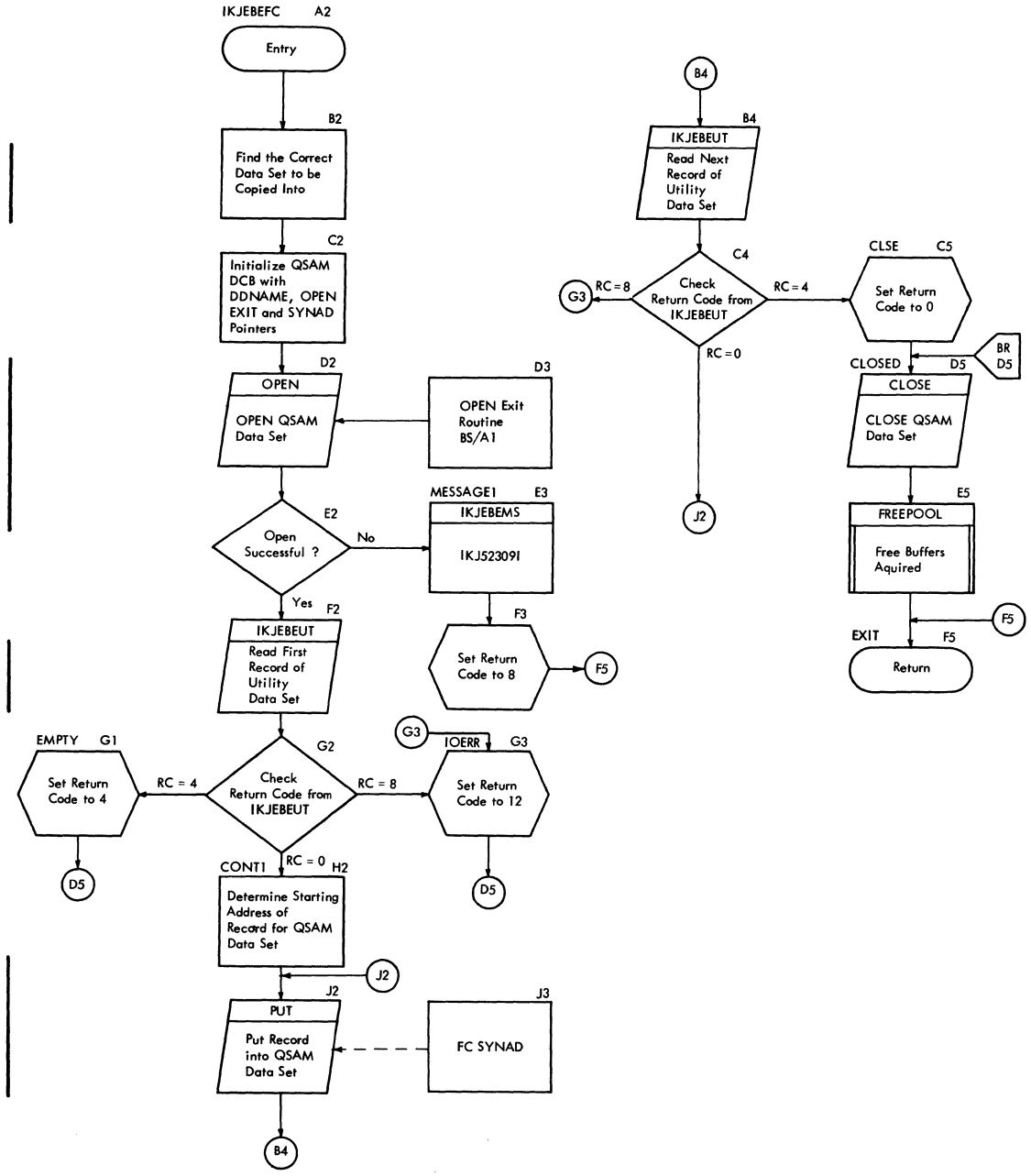


Chart BS. IKJEBEFC

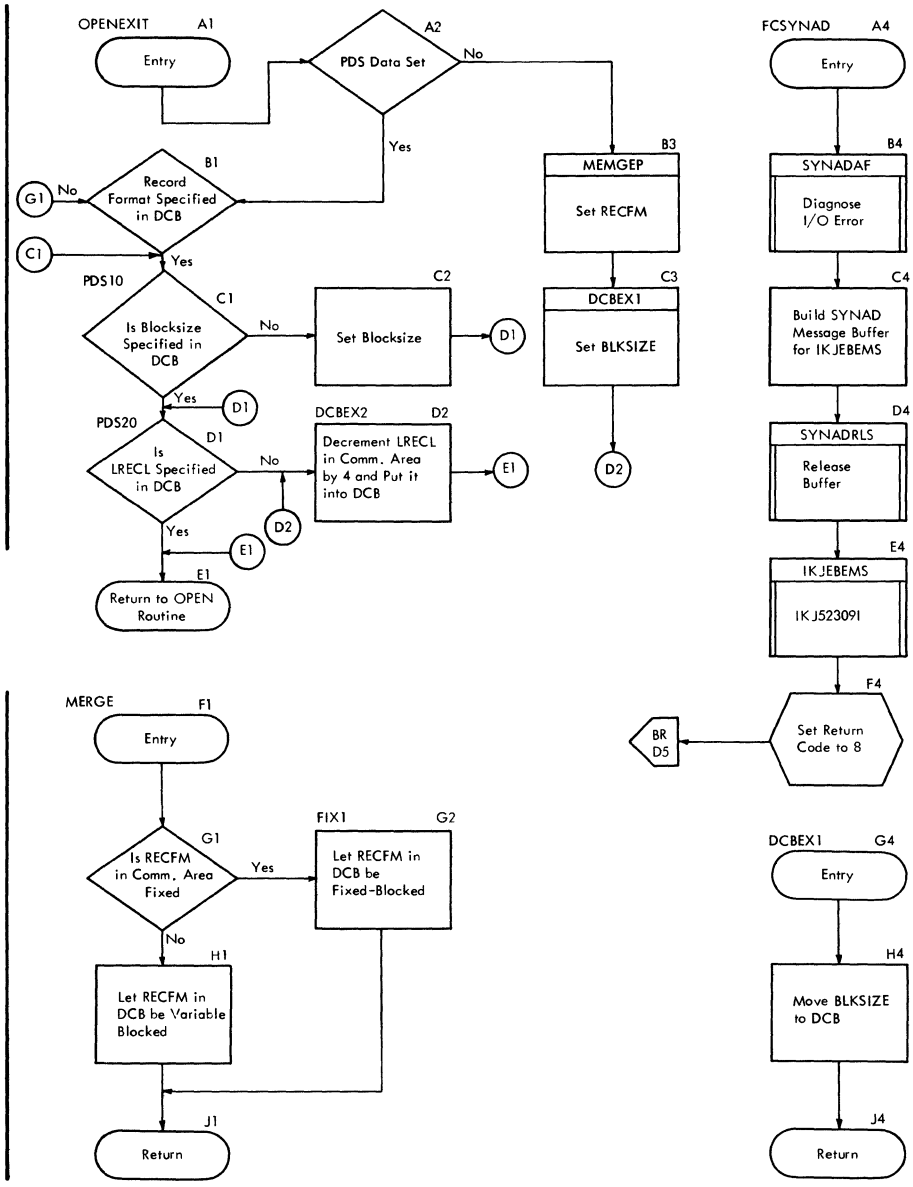


Chart BT. IKJEBEFI

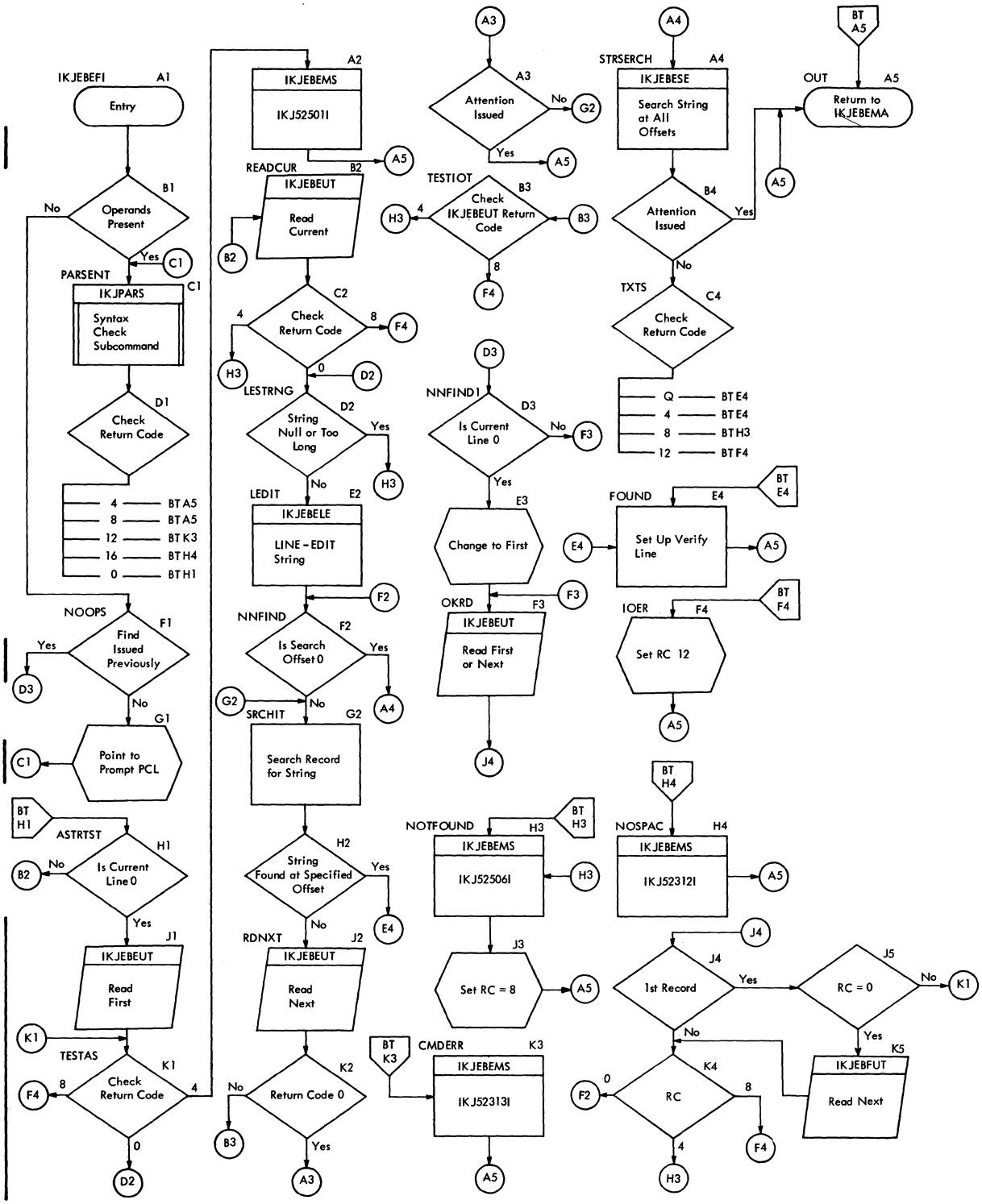


Chart BU. IKJEBEFO

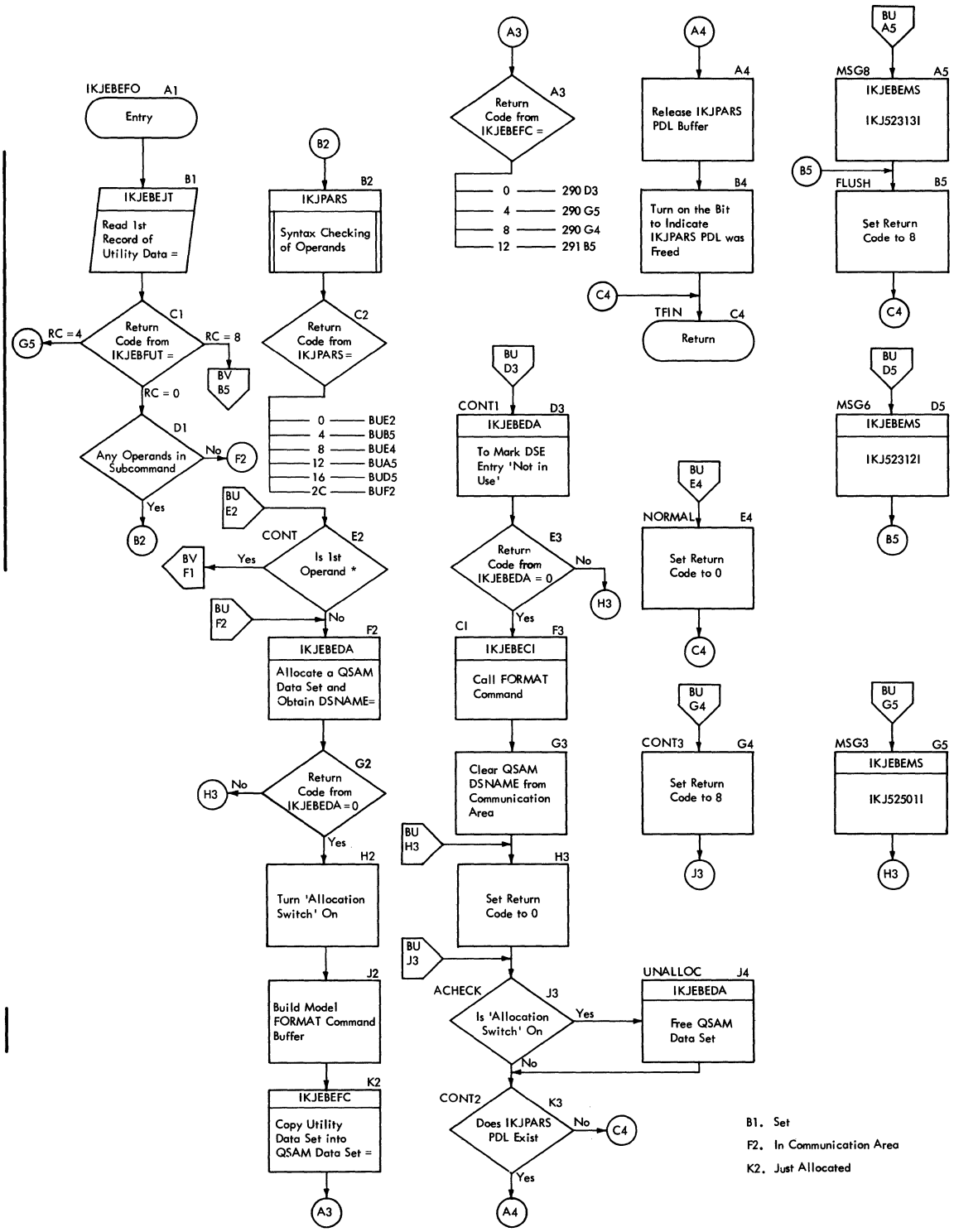


Chart BV. IKJEBEFO

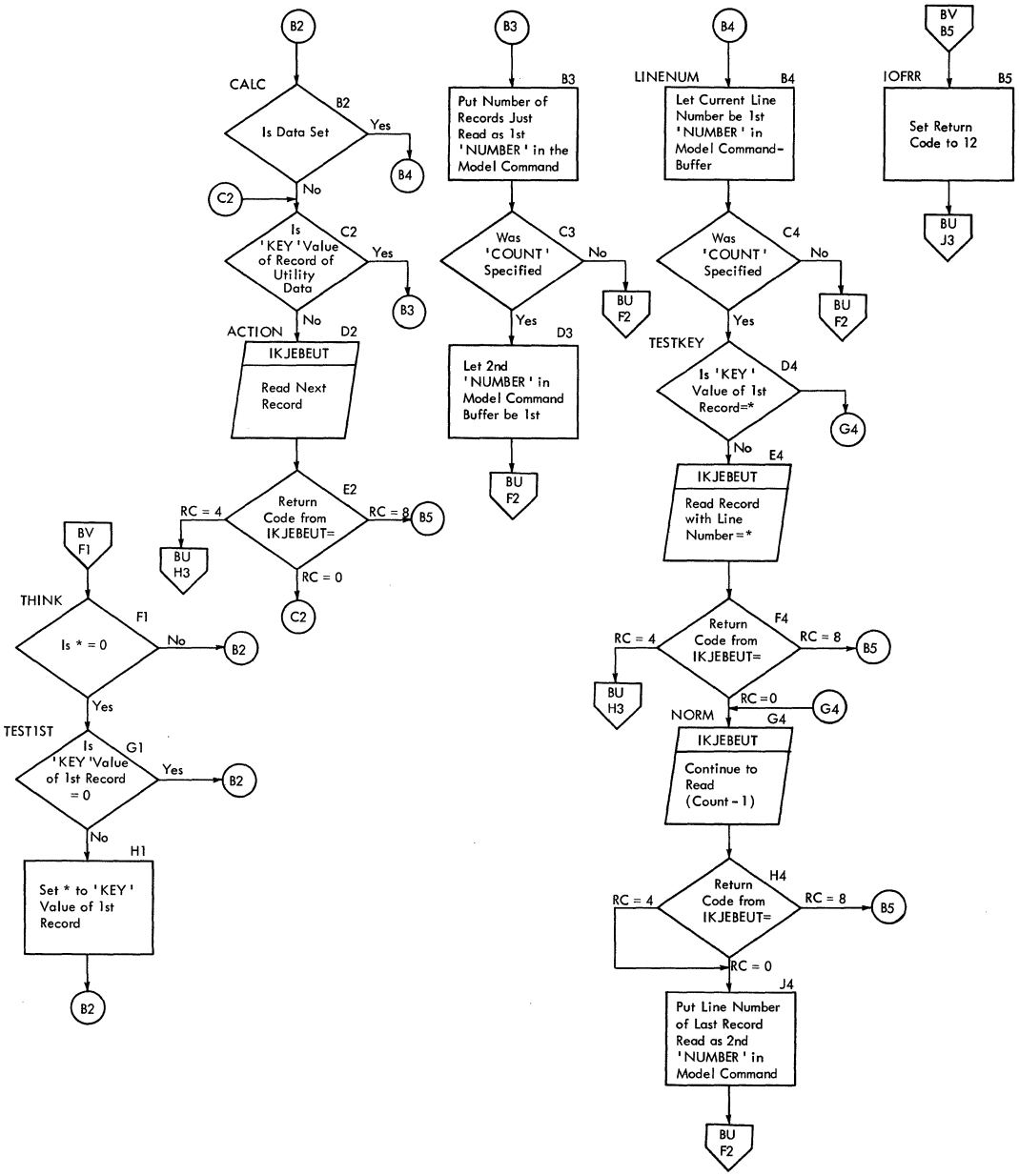


Chart BW. IKJEBEHE

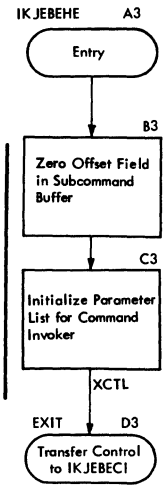


Chart BX. IKJEBEIM

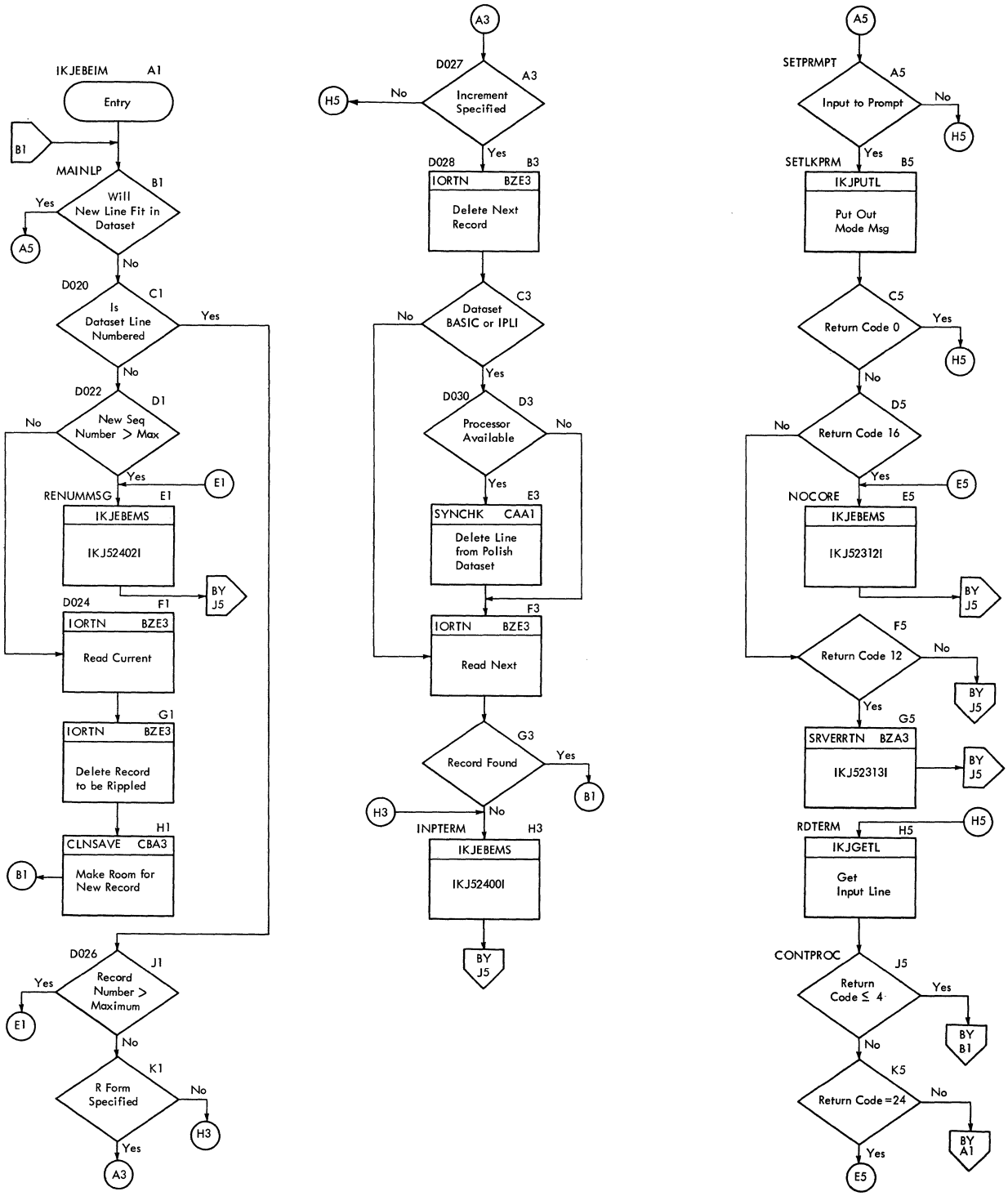


Chart BY. IKJEBEIM

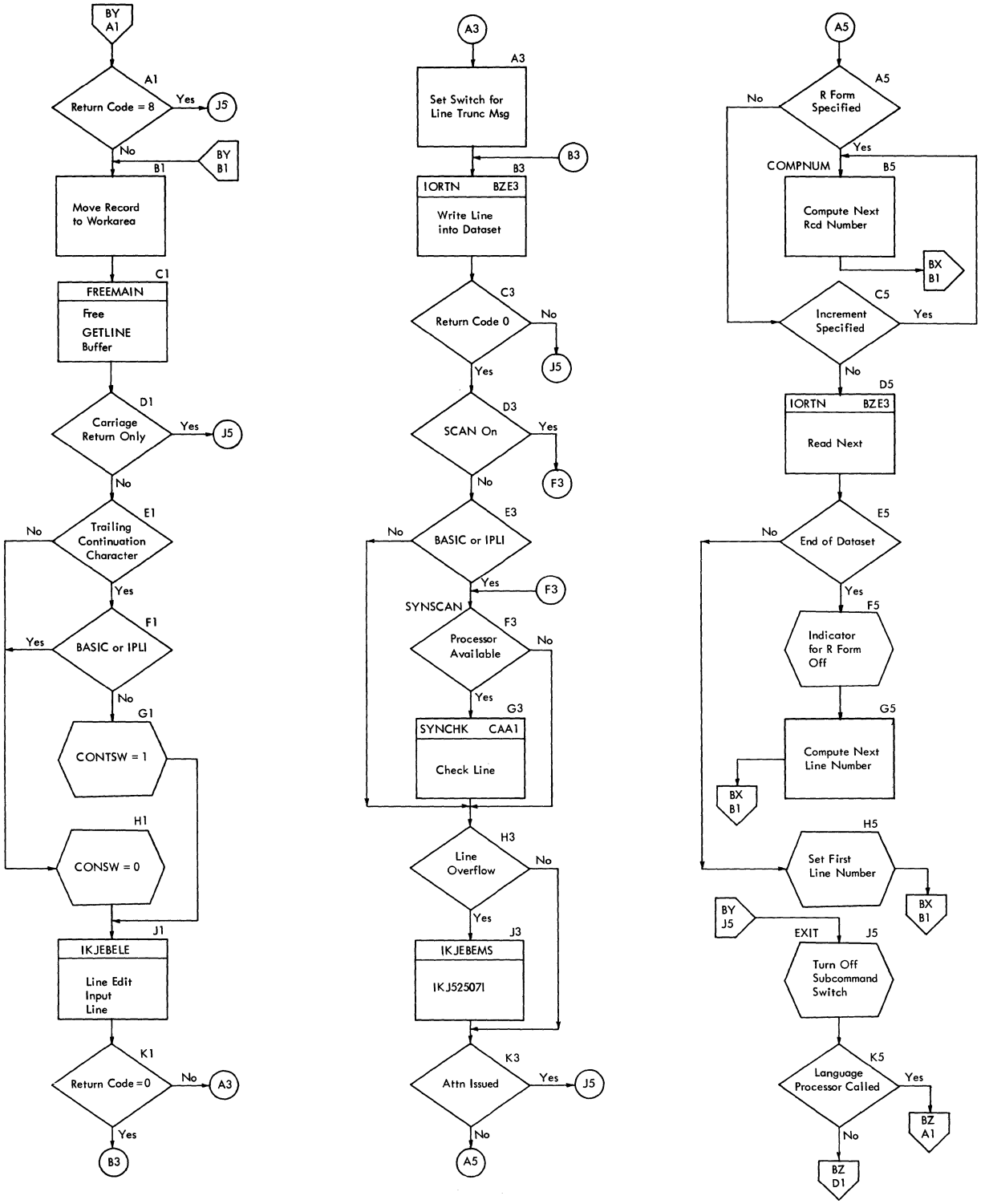


Chart BZ. IKJEBEIM

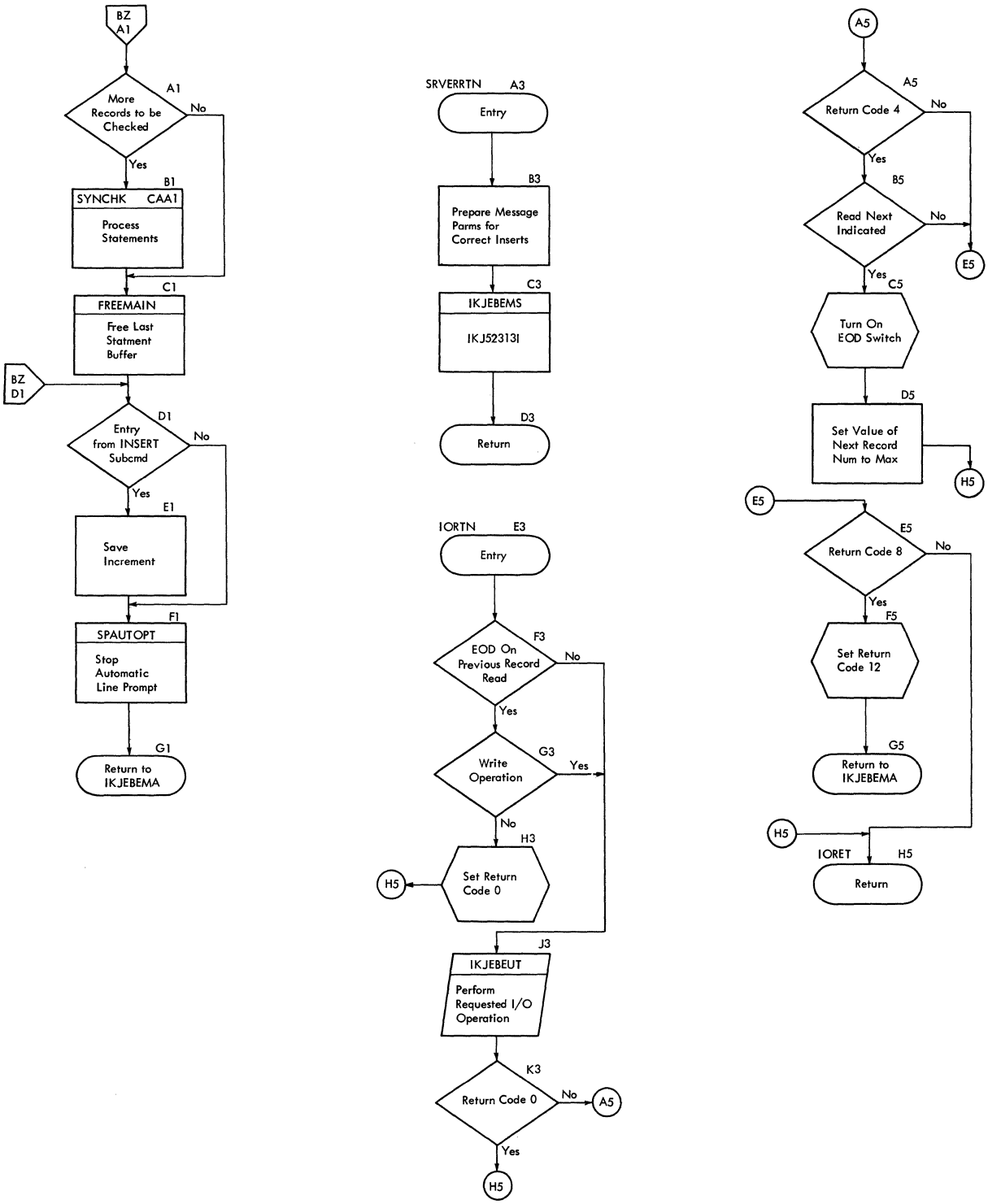


Chart CA. IKJEBEIM

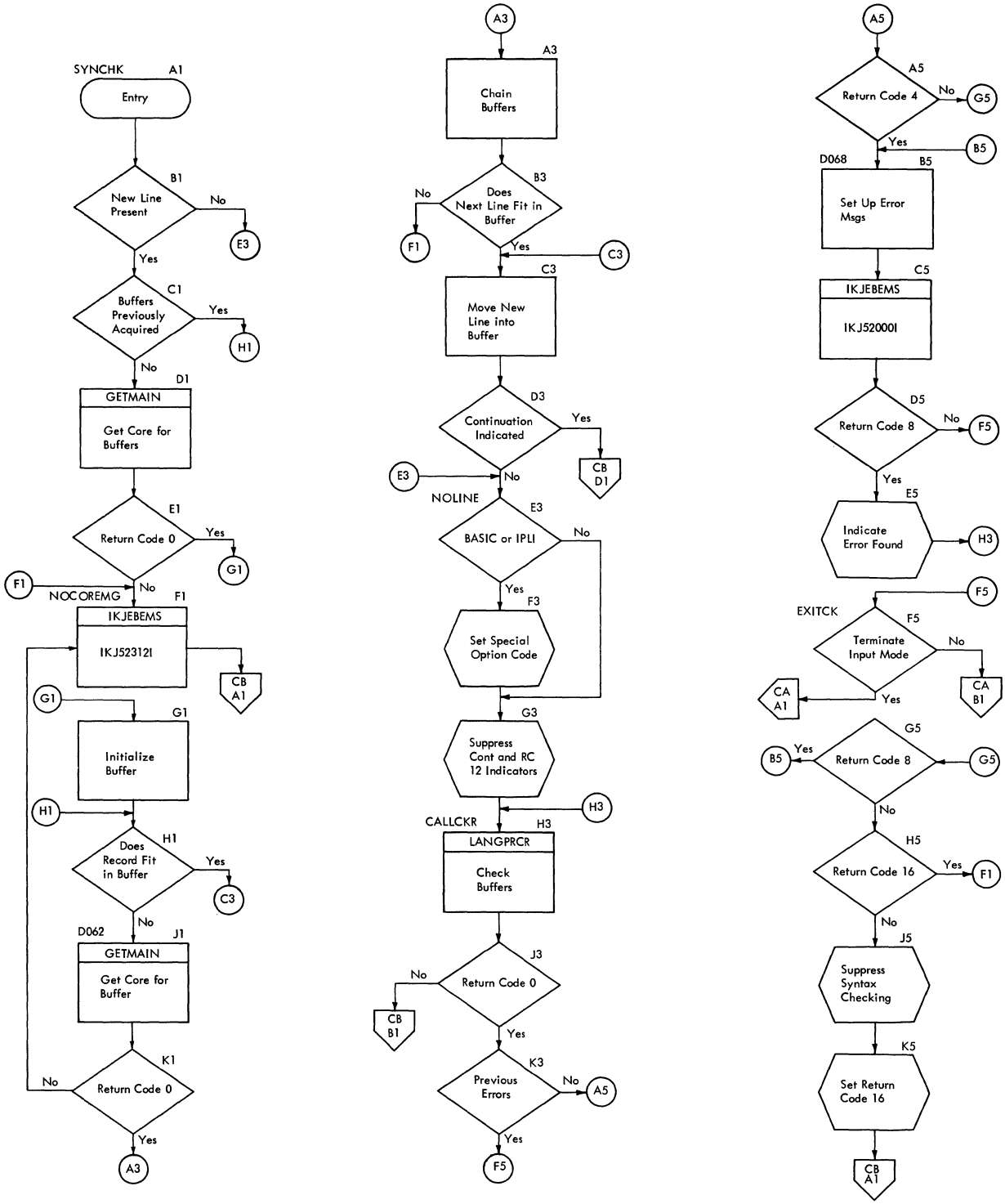


Chart CB. IKJEBEIM

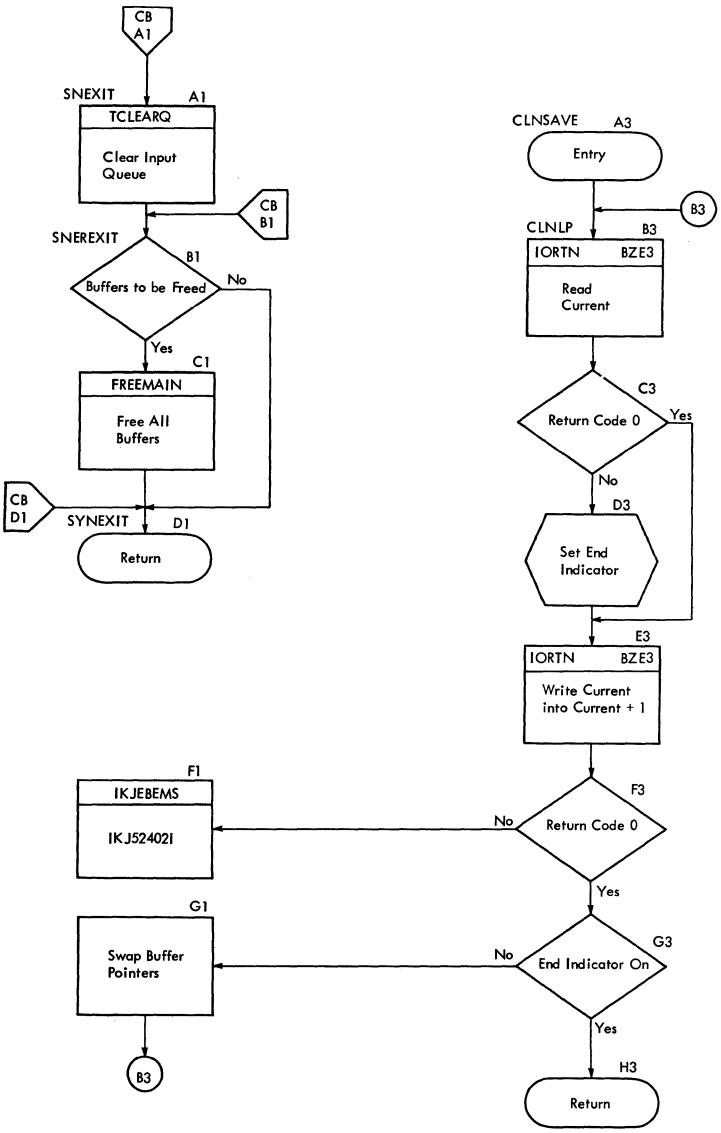


Chart CC. IKJEBEIN

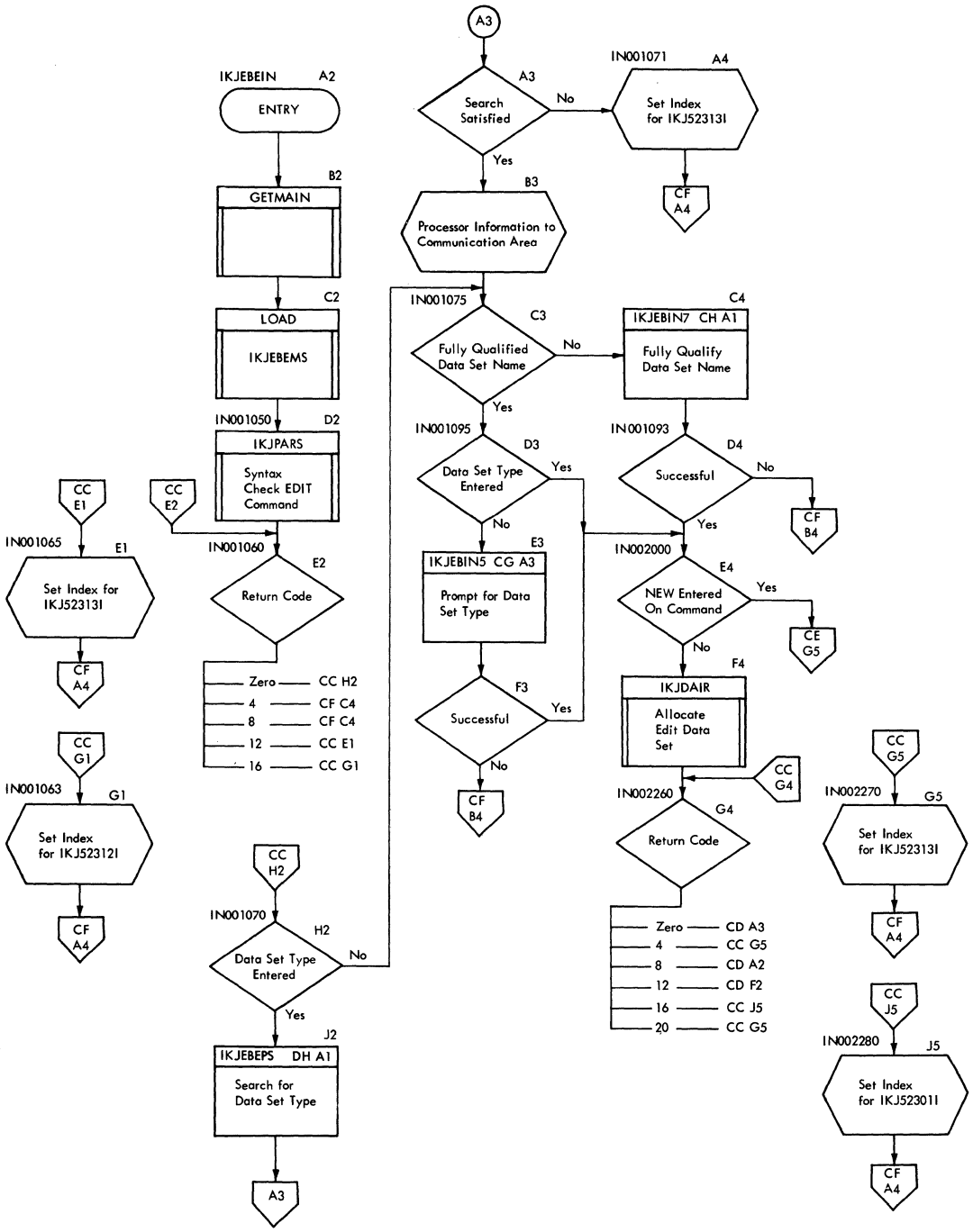


Chart CD. IKJEBEIN

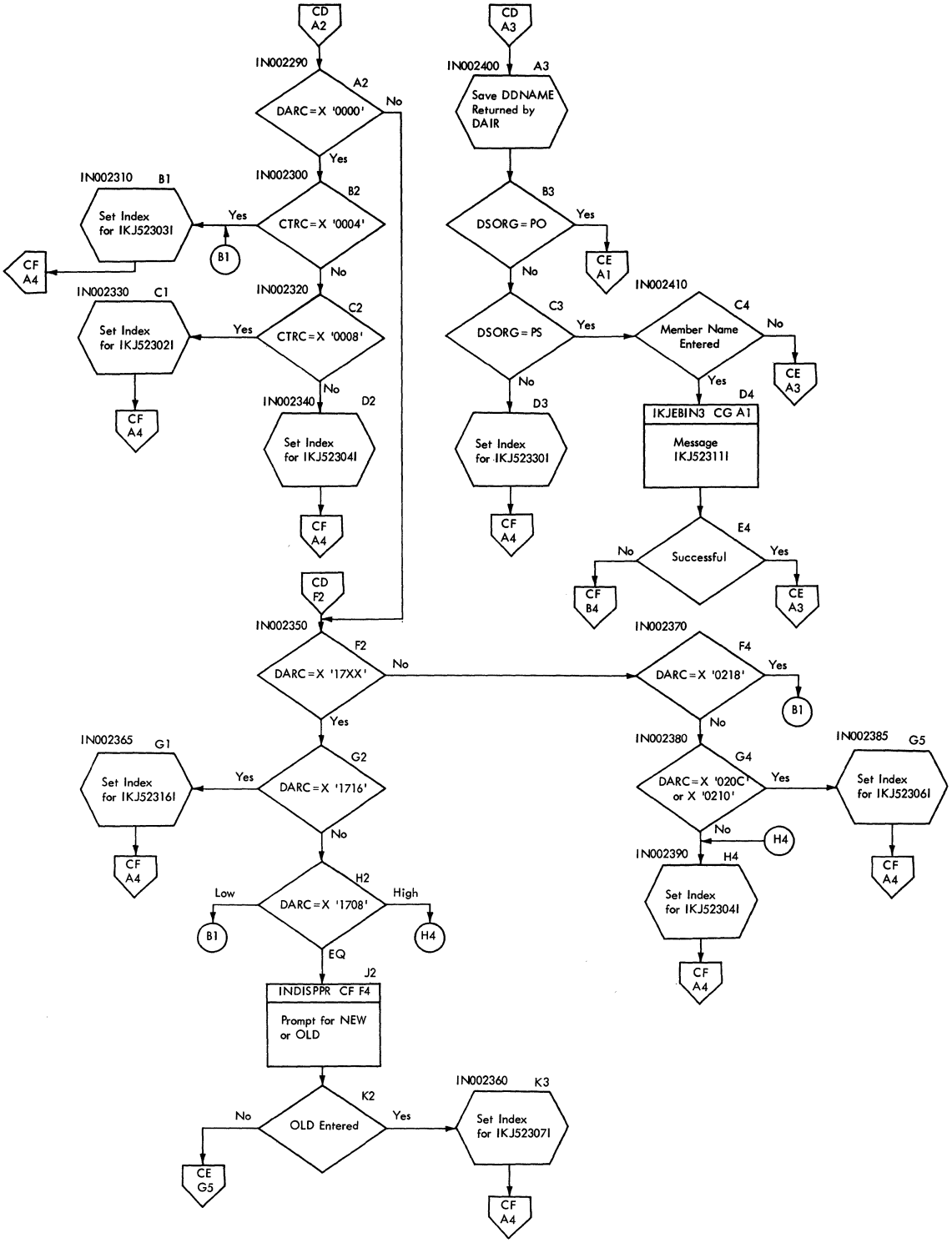


Chart CE. IKJEBEIN

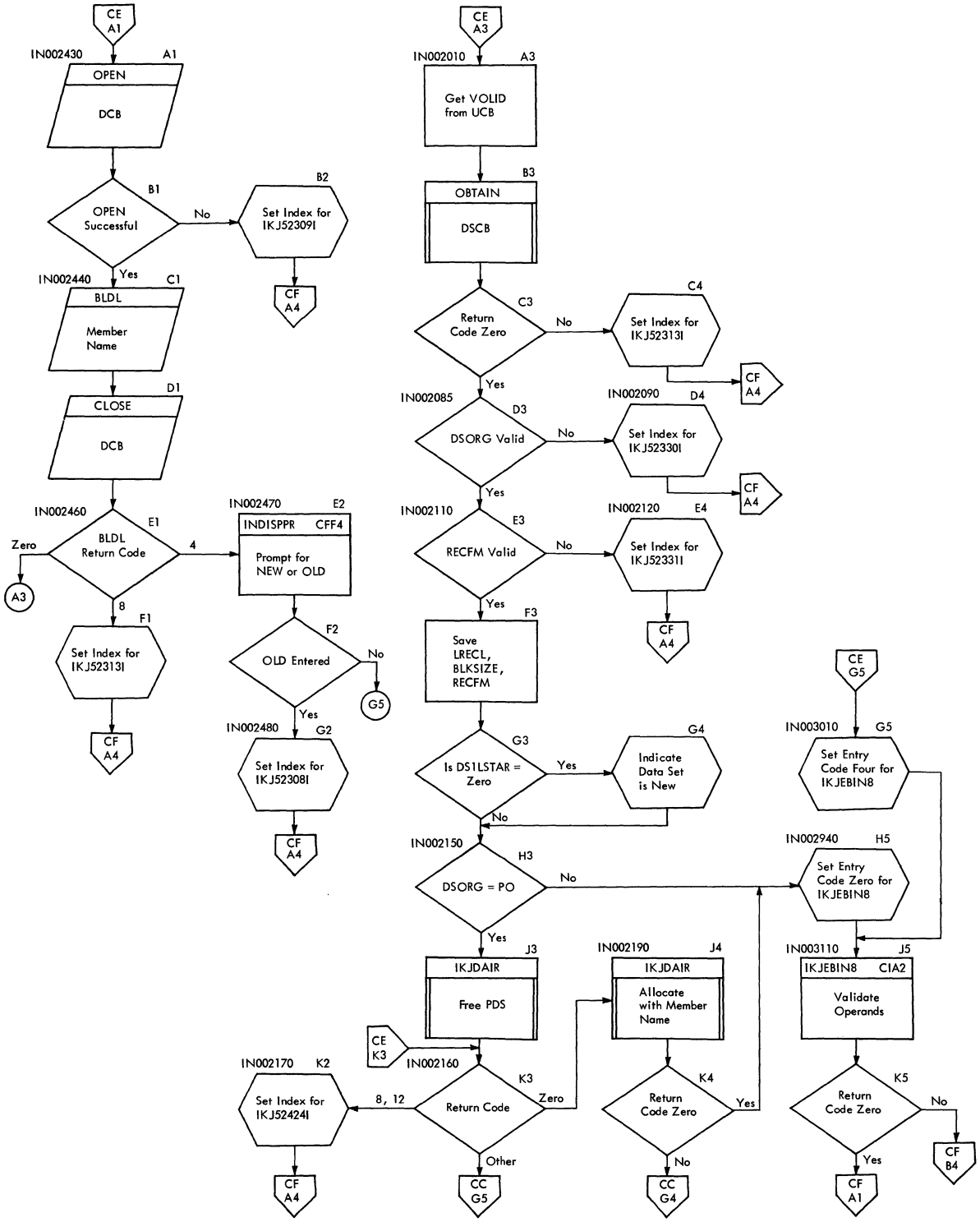


Chart CF. IKJEBEIN

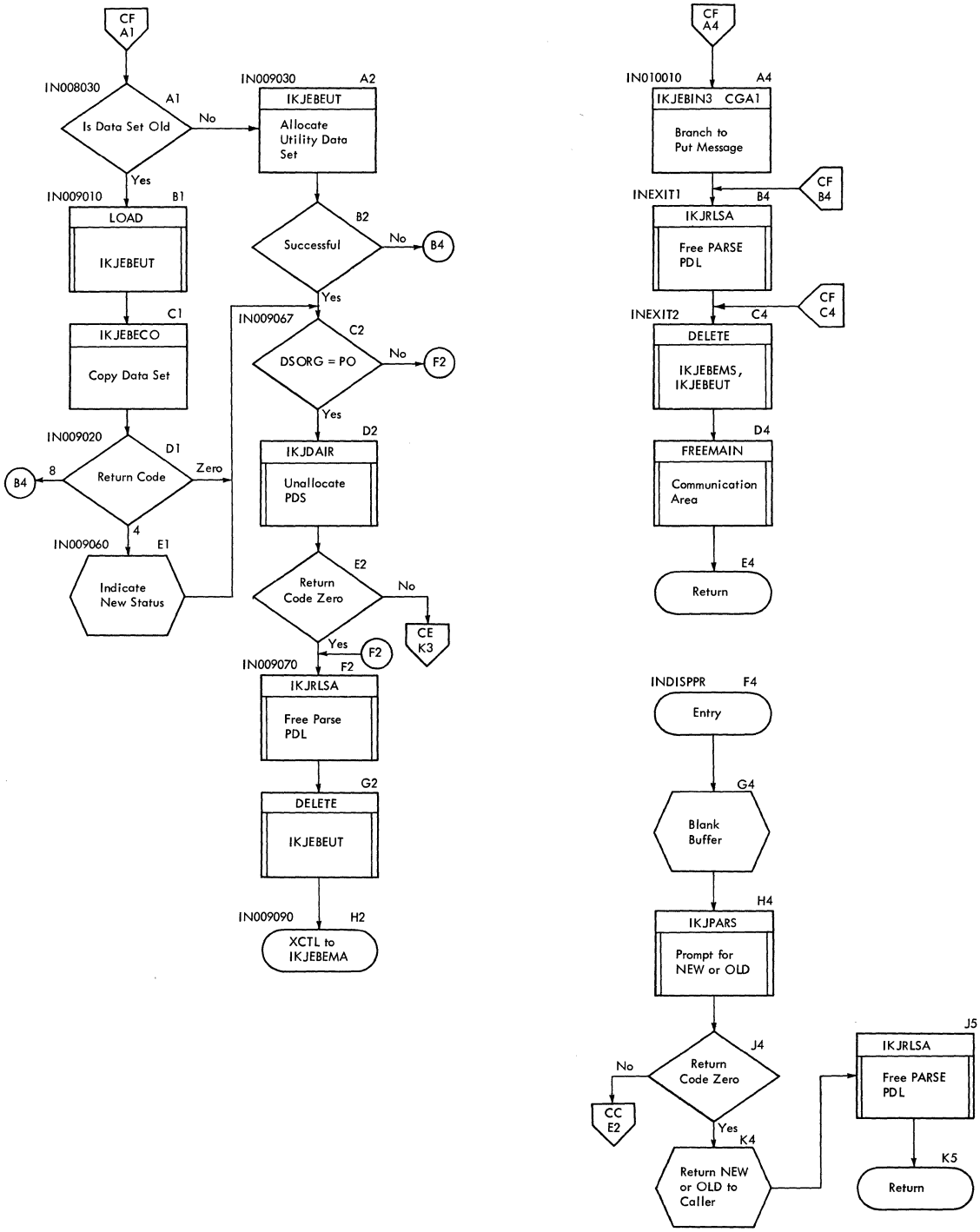


Chart CG. IKJEBEIN

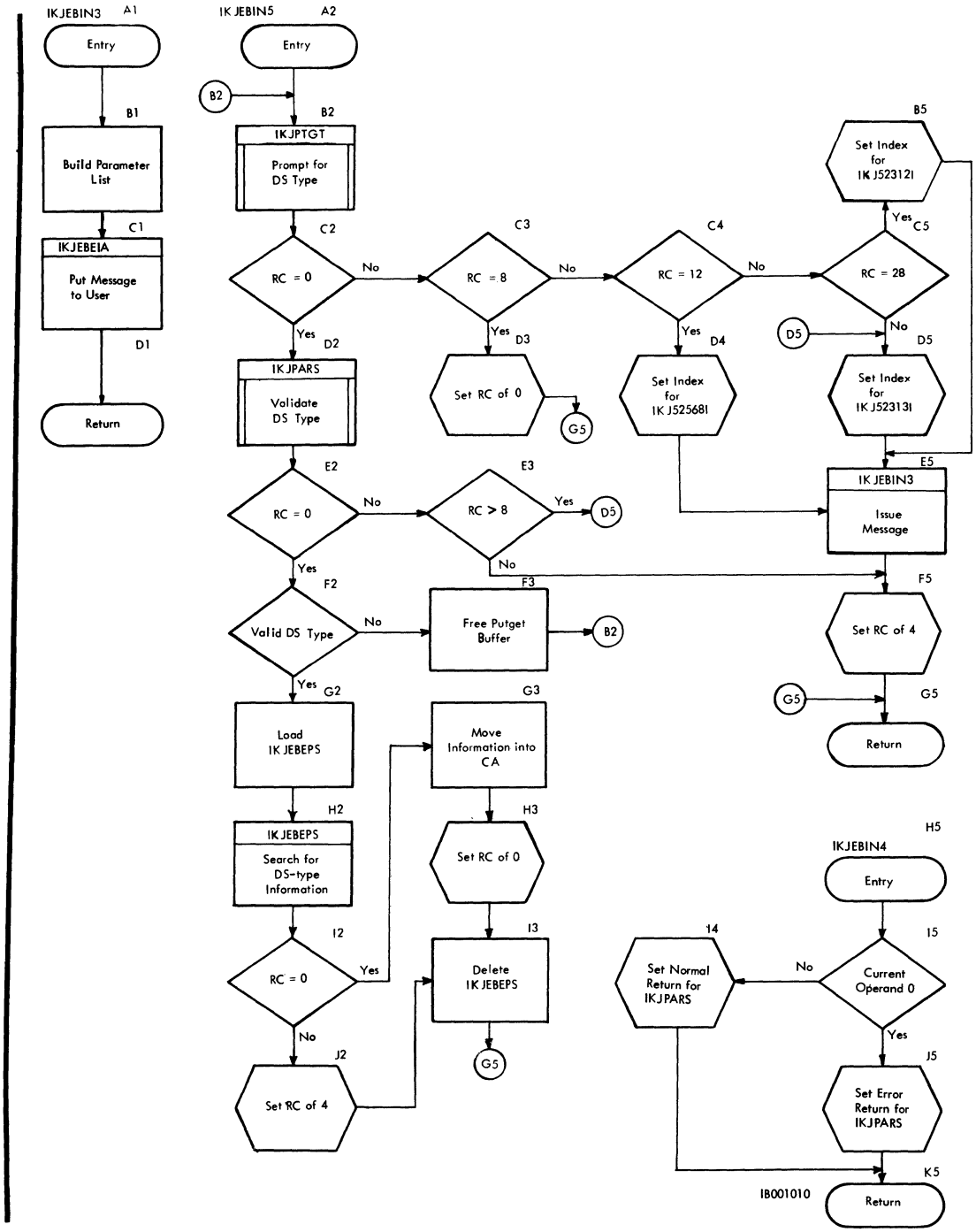


Chart CH. IKJEBEIN

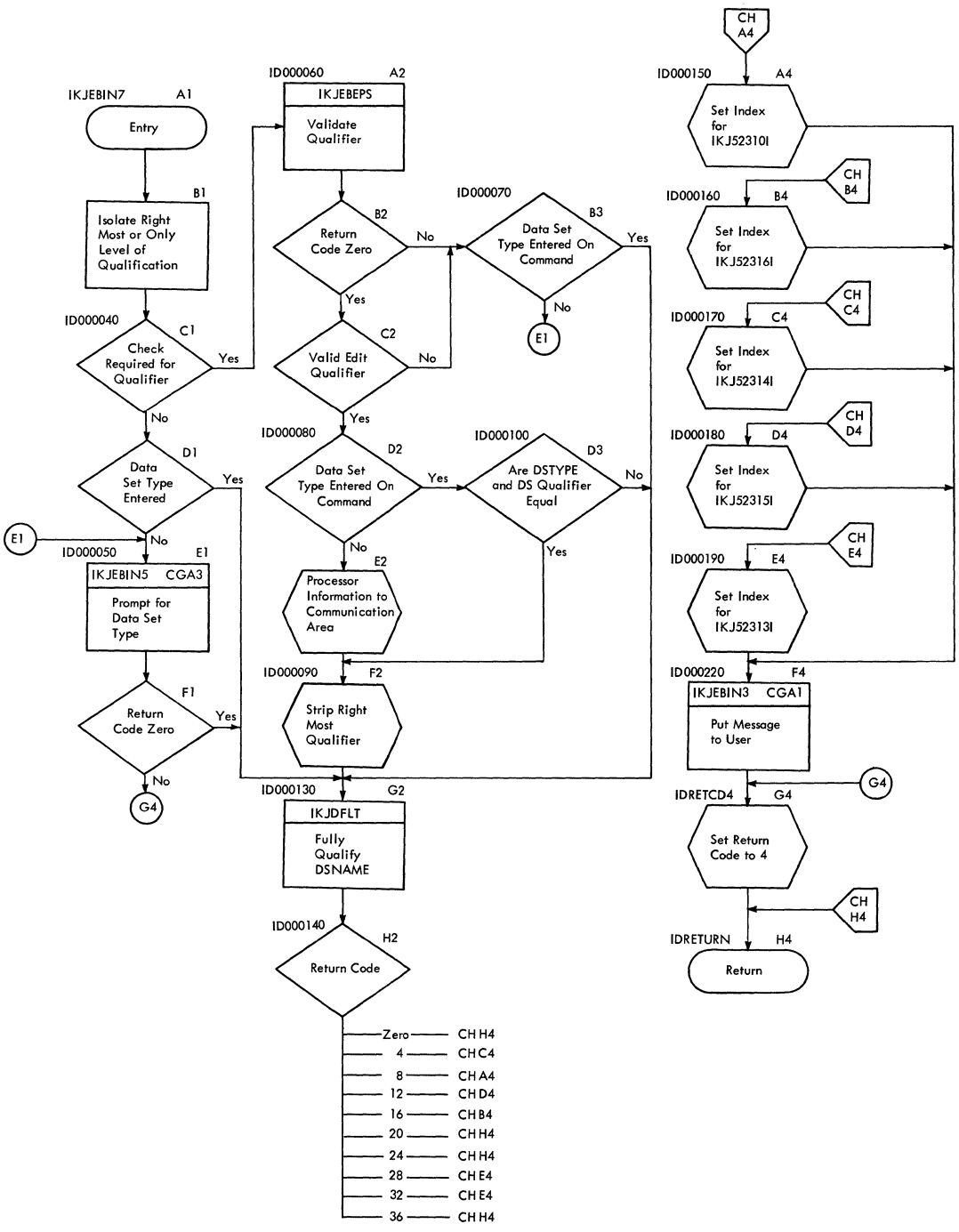


Chart CI. IKJEBEIN

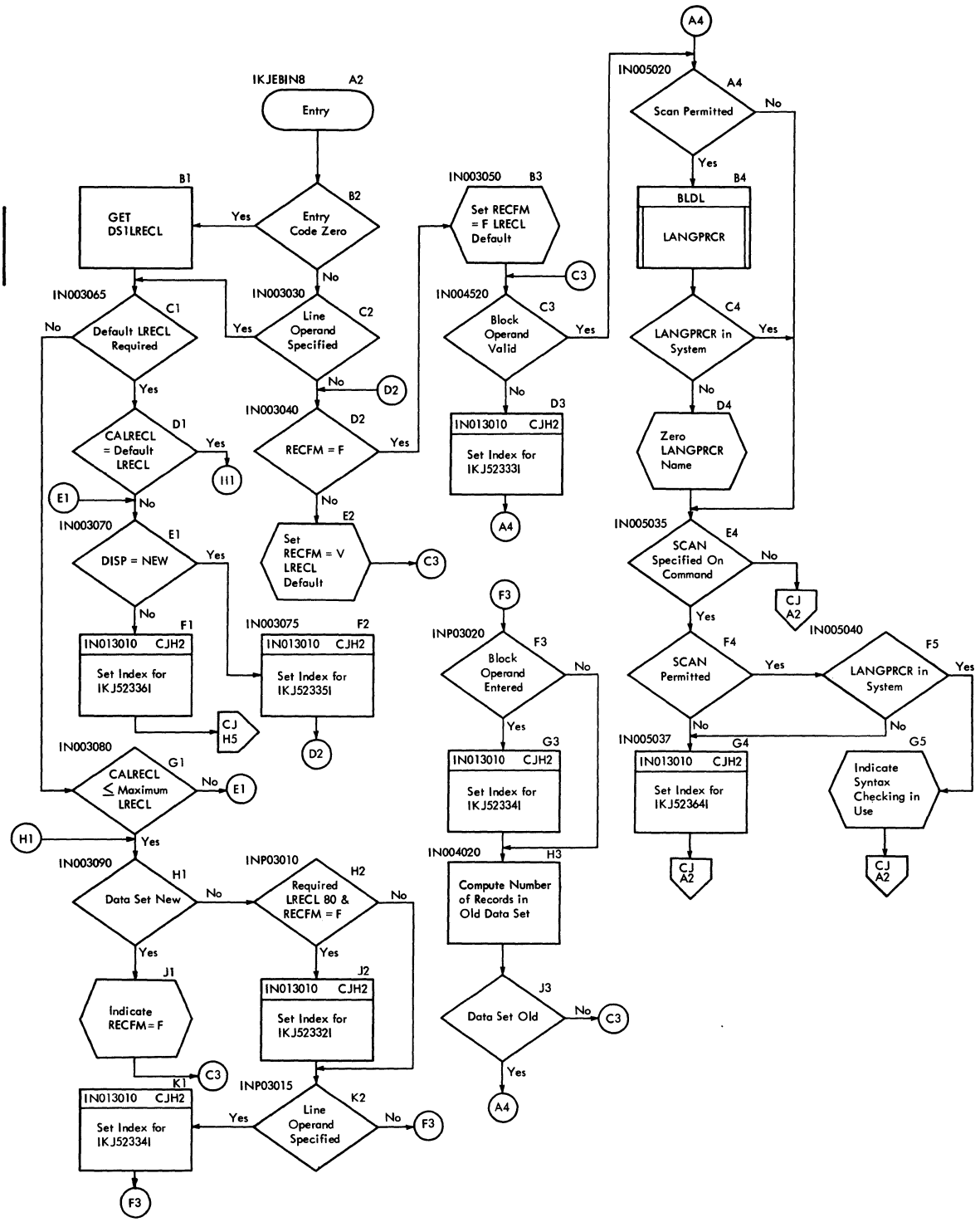


Chart CJ. IKJEBEIN

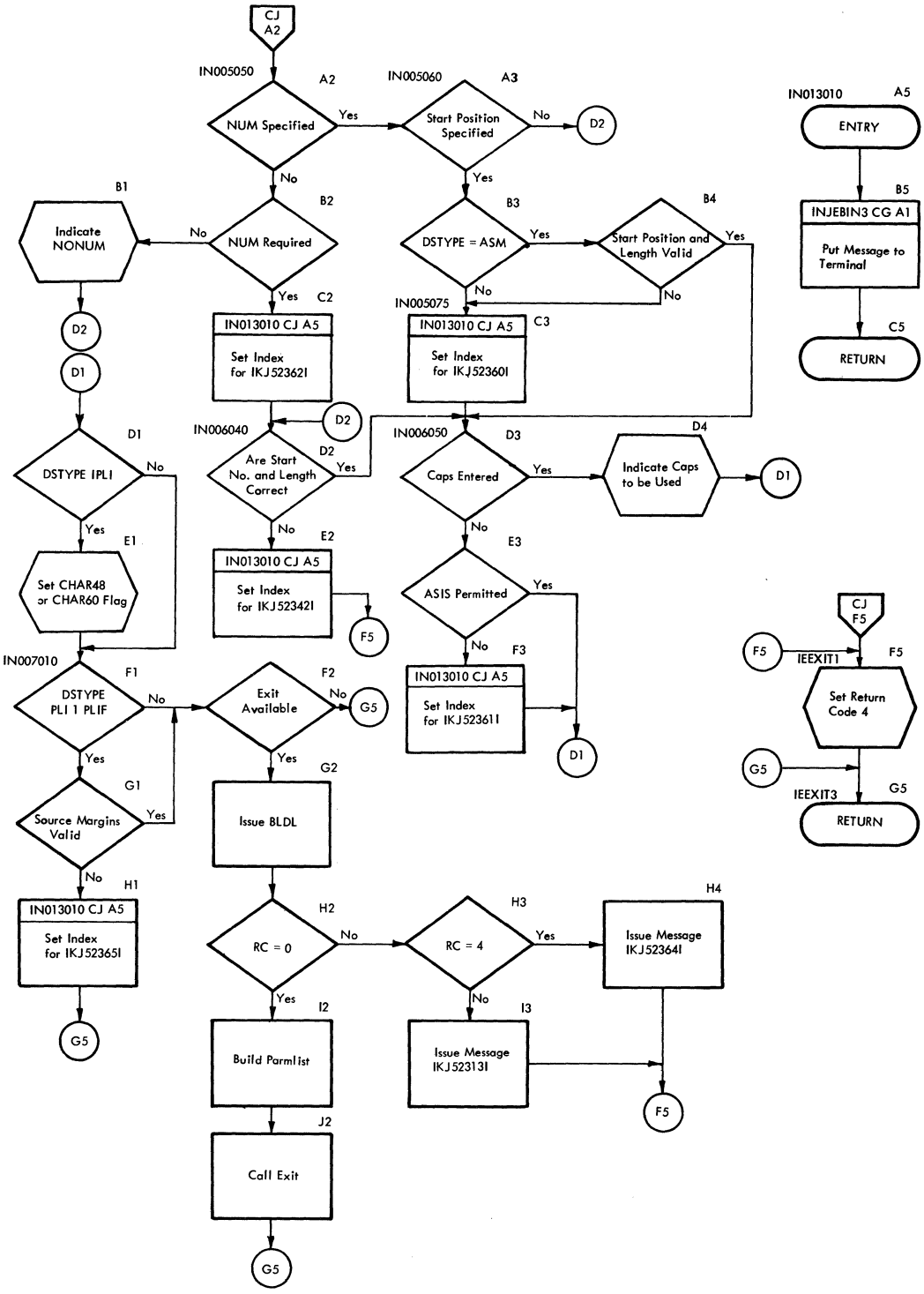


Chart CK. IKJEBEIP

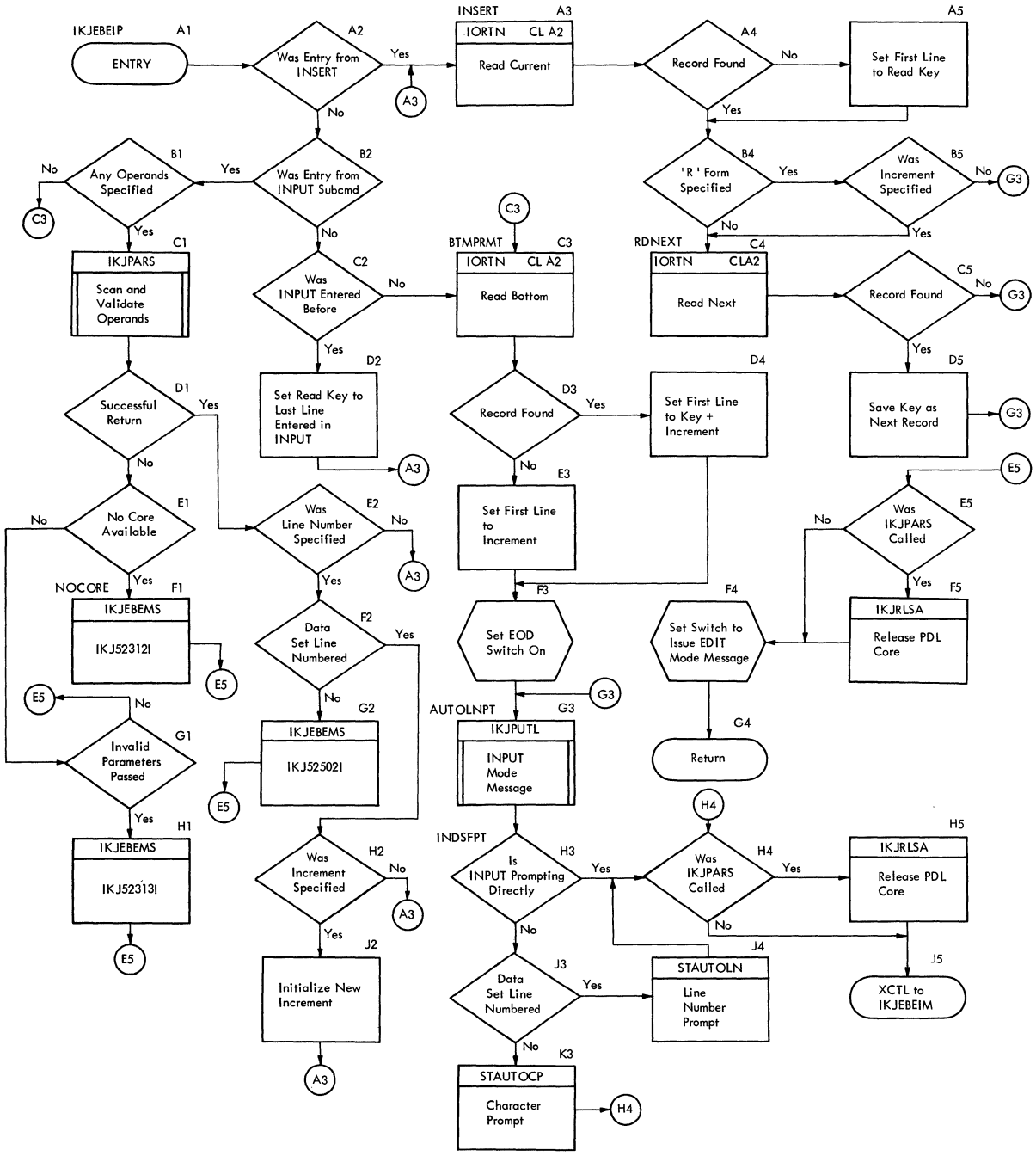


Chart CL. IKJEBEIP

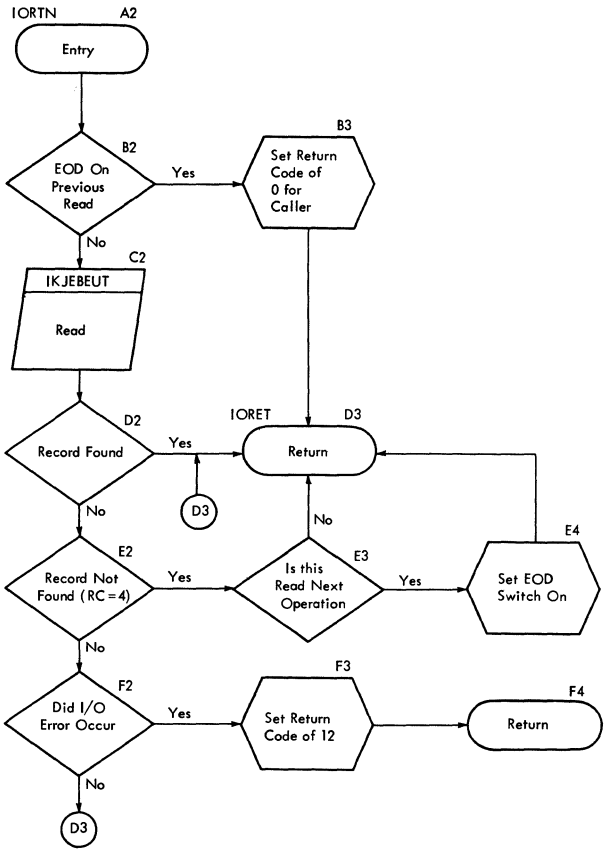


Chart CM. IKJEBEIS

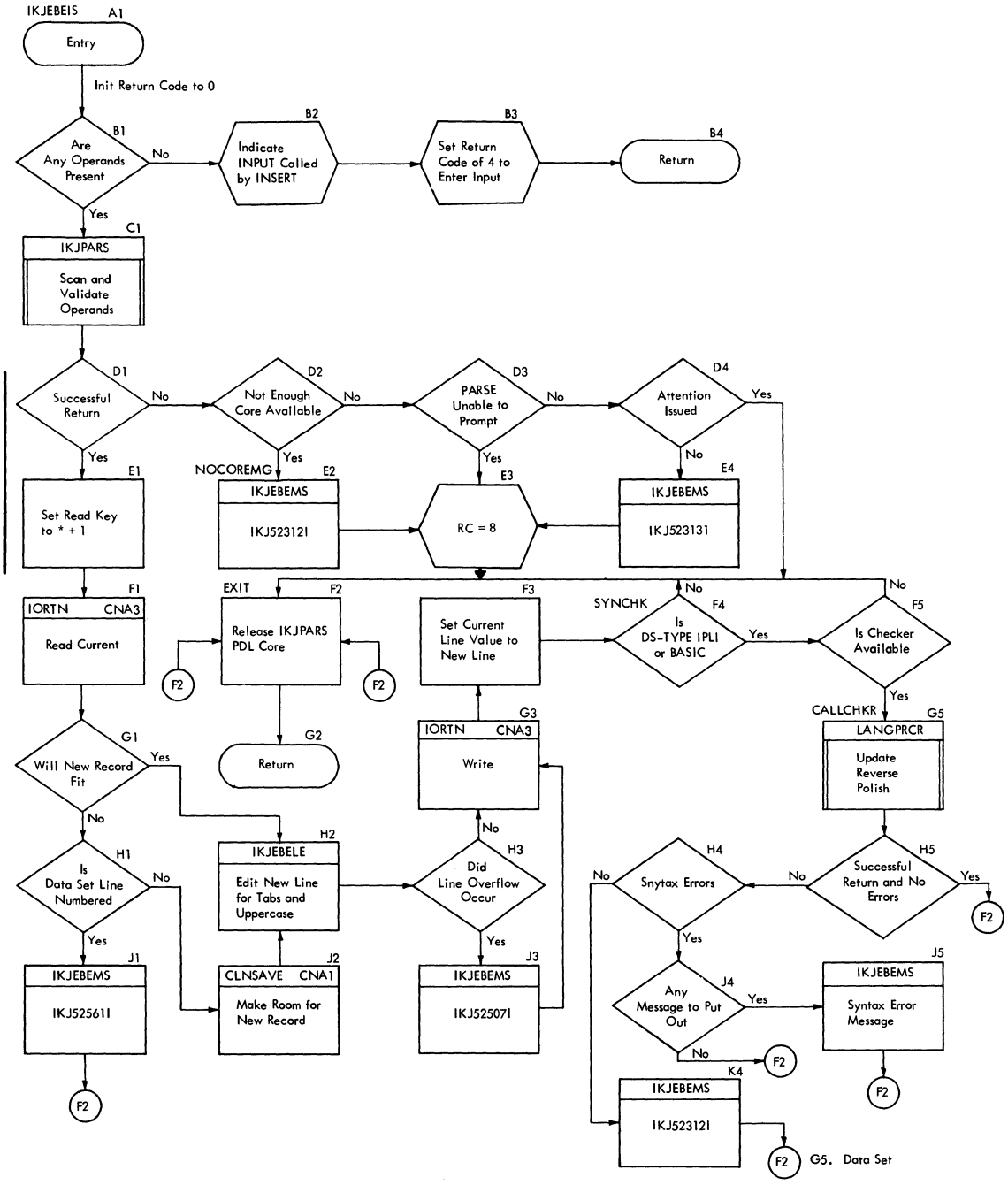


Chart CN. IKJEBEIS

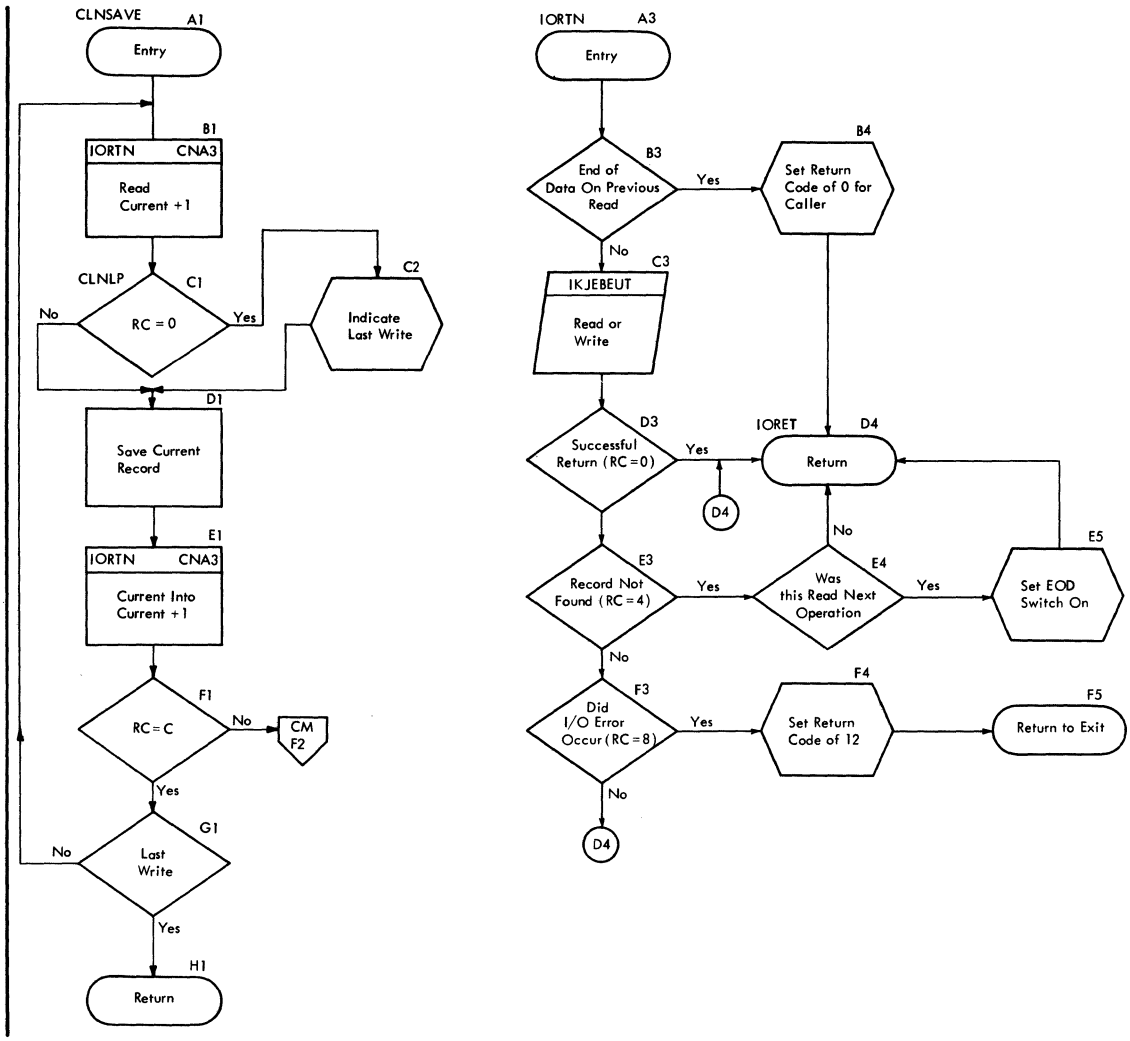
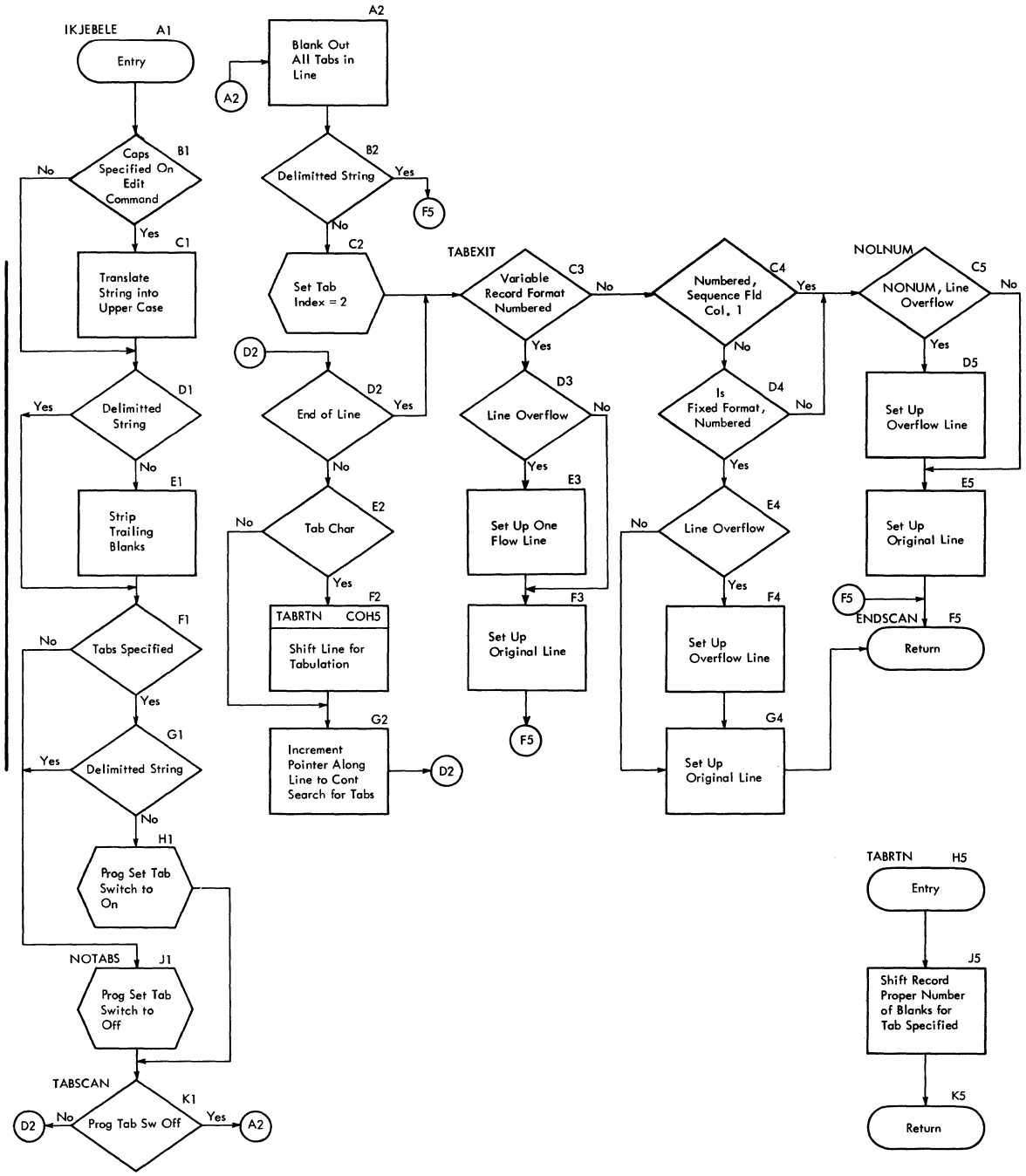


Chart CO. IKJEBELE



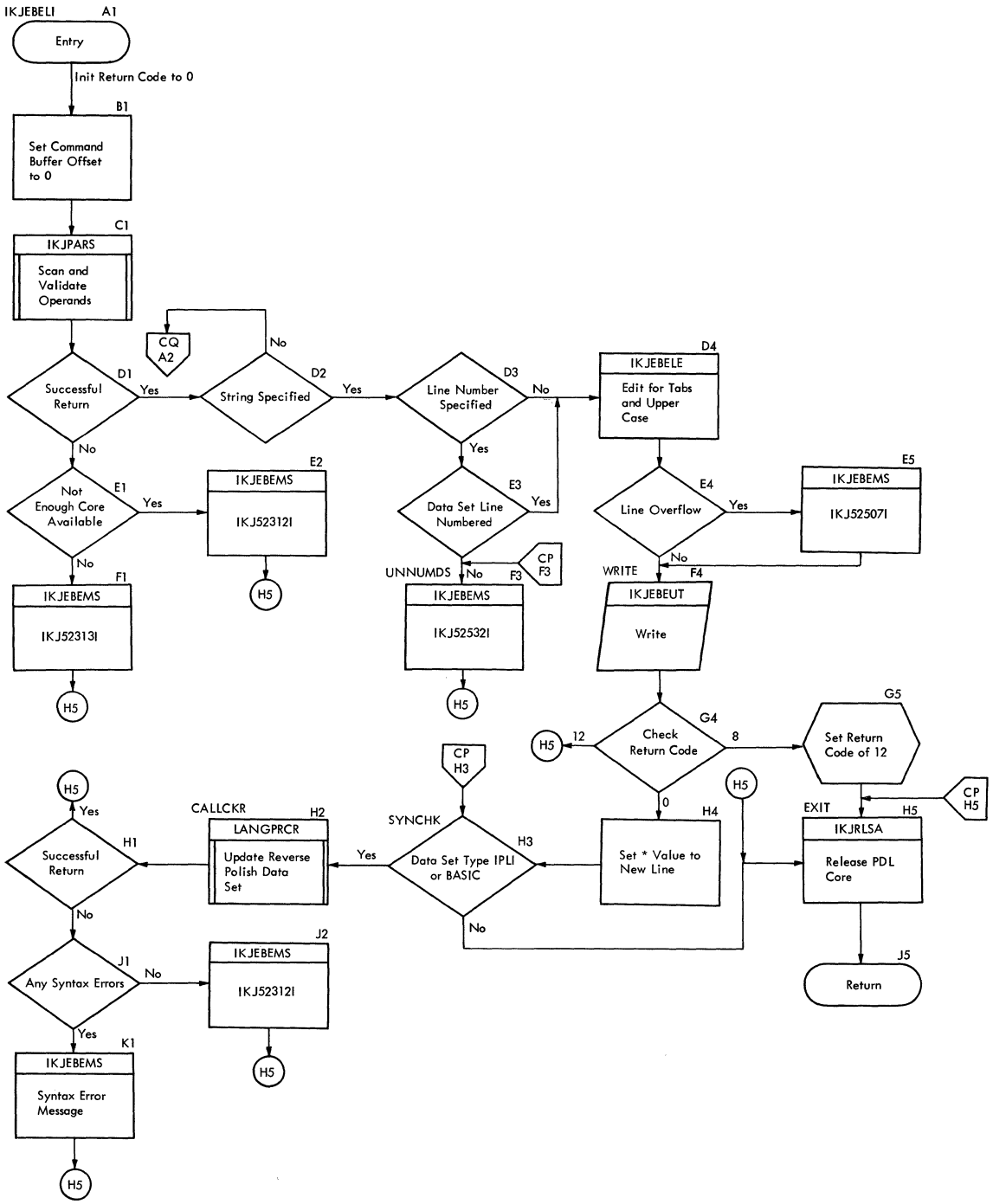
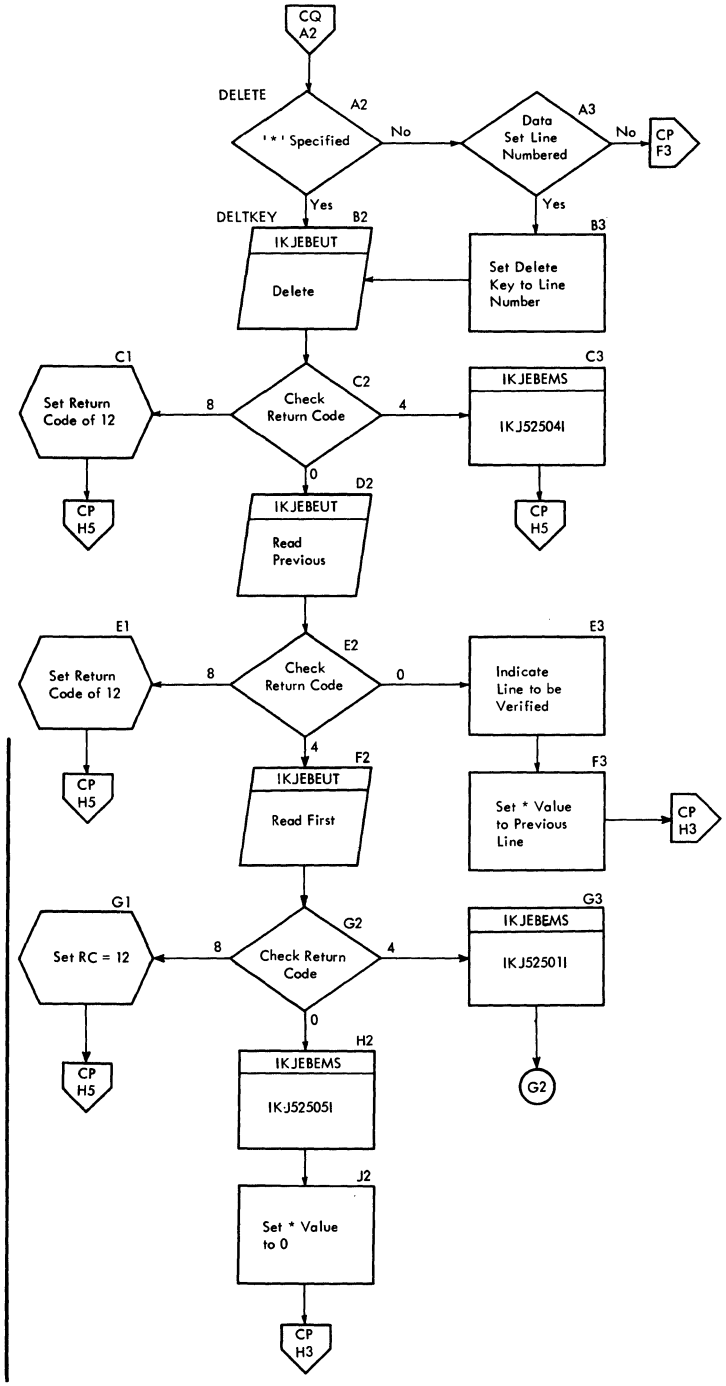

IKJEBELI
A1
Entry
Init Return Code to 0
B1
Set Command Buffer Offset to 0
C1
IKJPARS
Scan and Validate Operands
CQ A2
D1
Successful Return
Yes
No
D2
String Specified
Yes
No
D3
Line Number Specified
No
Yes
D4
IKJEBELE
Edit for Tabs and Upper Case
E1
Not Enough Core Available
Yes
No
E2
IKJEBEMS
IKJ52312I
H5
E3
Data Set Line Numbered
Yes
No
CP F3
UNNUMDS
F3
IKJEBEMS
IKJ52532I
H5
E4
Line Overflow
Yes
No
E5
IKJEBEMS
IKJ52507I
WRITE
F4
IKJEBEUT
Write
F1
IKJEBEMS
IKJ52313I
H5
G4
Check Return Code
12
8
0
H5
G5
Set Return Code of 12
CP H5
EXIT
H5
IKJRLSA
Release PDL Core
J5
Return
H4
Set * Value to New Line
CP H3
SYNCHK
H3
Data Set Type IPLI or BASIC
Yes
No
CALLCKR
H2
LANGPRCR
Update Reverse Polish Data Set
H1
Successful Return
Yes
No
H5
J1
Any Syntax Errors
No
Yes
J2
IKJEBEMS
IKJ52312I
H5
K1
IKJEBEMS
Syntax Error Message
H5

Chart CQ. IKJEBELI



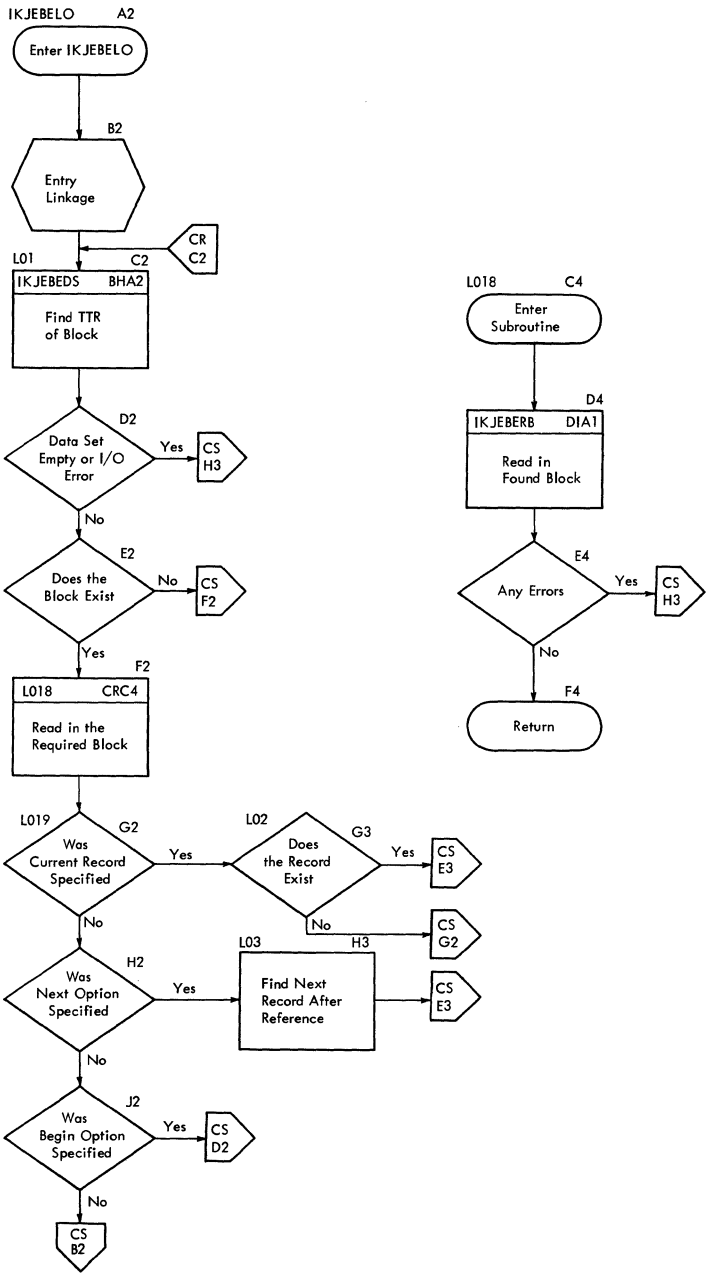

IKJEBELO A2
Enter IKJEBELO
B2
Entry Linkage
CR C2
L01 C2
IKJEBEDS BHA2
Find TTR of Block
D2
Data Set Empty or I/O Error
Yes
CS H3
No
E2
Does the Block Exist
No
CS F2
Yes
F2
L018 CRC4
Read in the Required Block
L019 G2
Was Current Record Specified
Yes
L02 G3
Does the Record Exist
Yes
CS E3
No
CS G2
No
H2
Was Next Option Specified
Yes
L03 H3
Find Next Record After Reference
CS E3
No
J2
Was Begin Option Specified
Yes
CS D2
No
CS B2
L018 C4
Enter Subroutine
D4
IKJEBERB DIA1
Read in Found Block
E4
Any Errors
Yes
CS H3
No
F4
Return

Chart CS. IKJEBELO

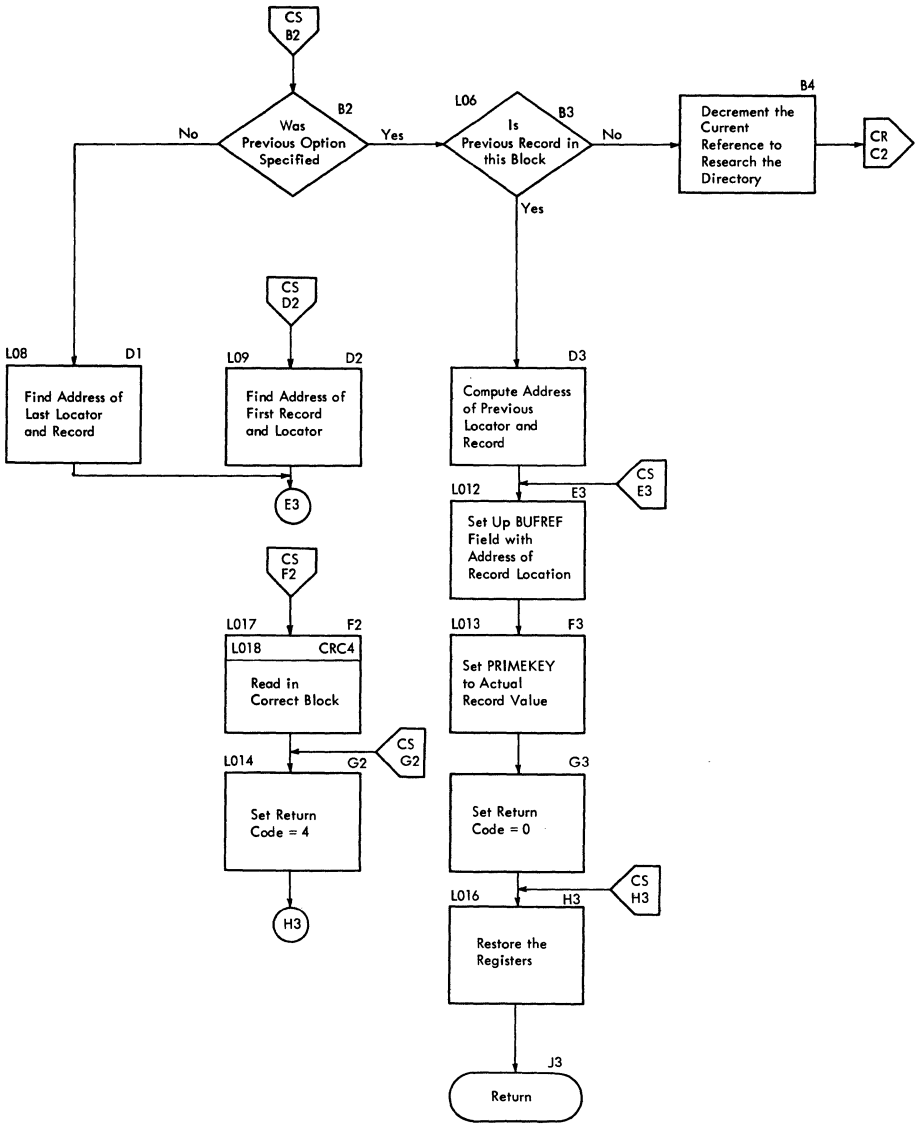


Chart CT. IKJEBELT

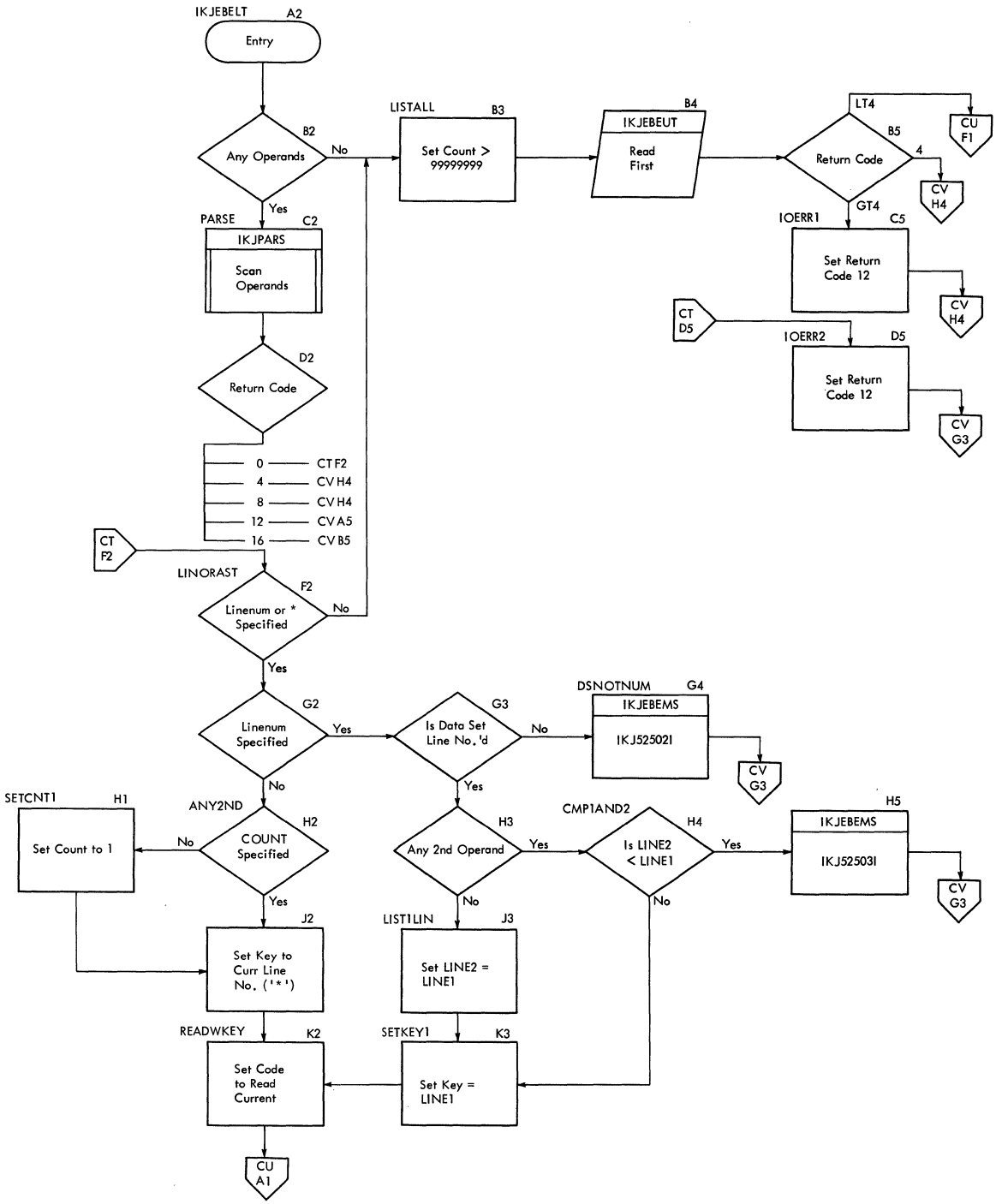


Chart CU. IKJEBELT

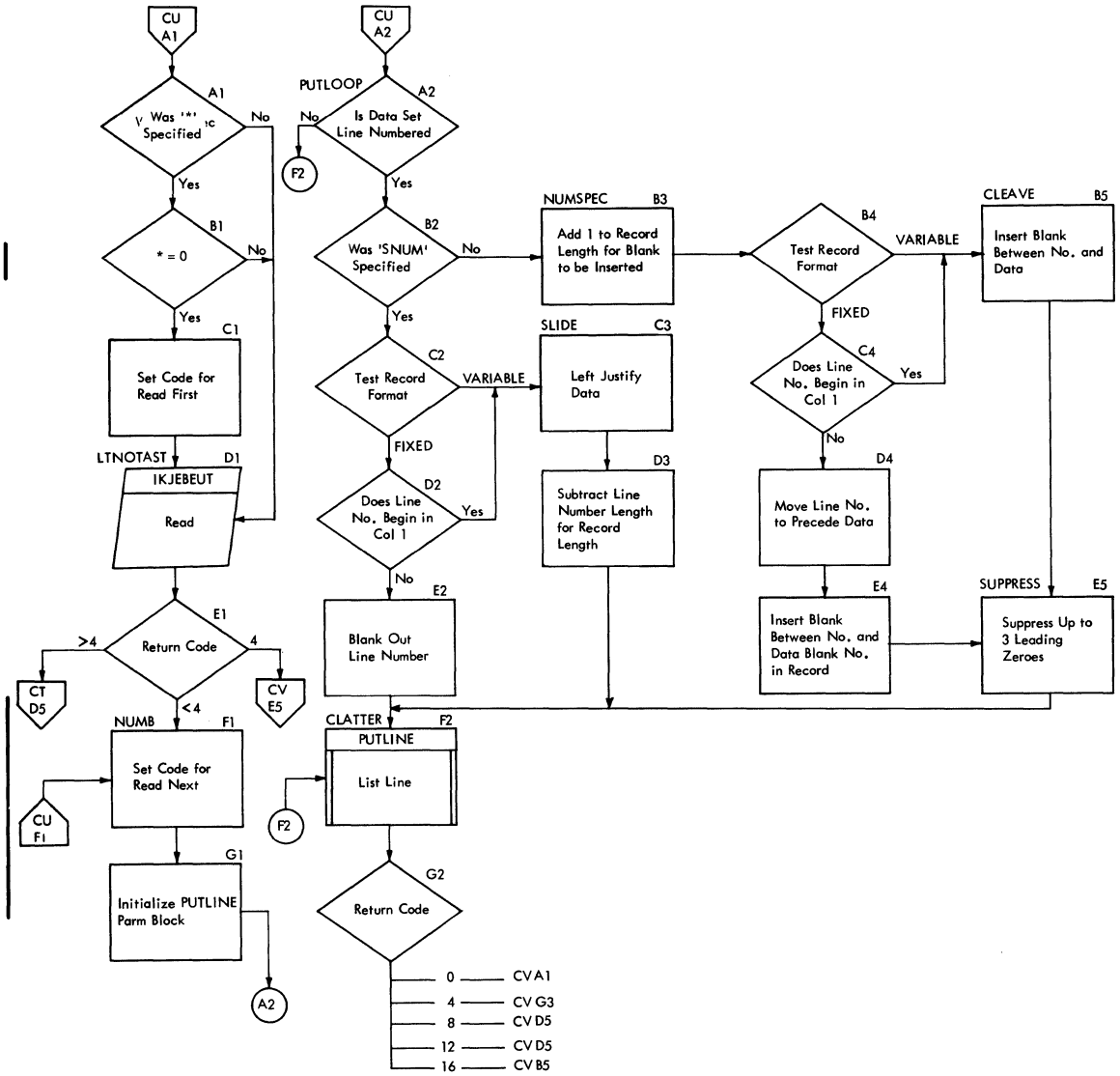


Chart CV. IKJEBELT

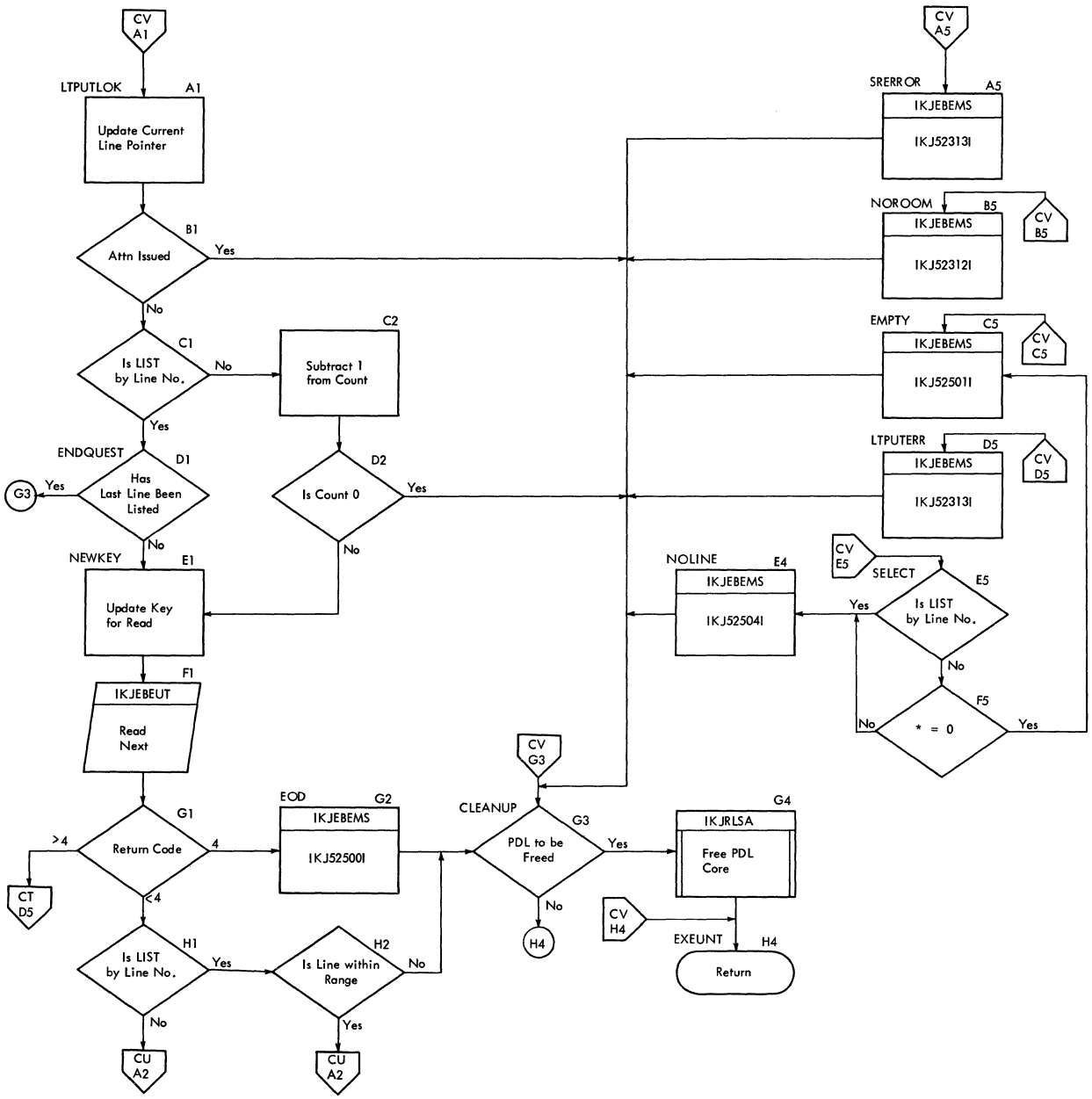


Chart CW. IKJEBEMA

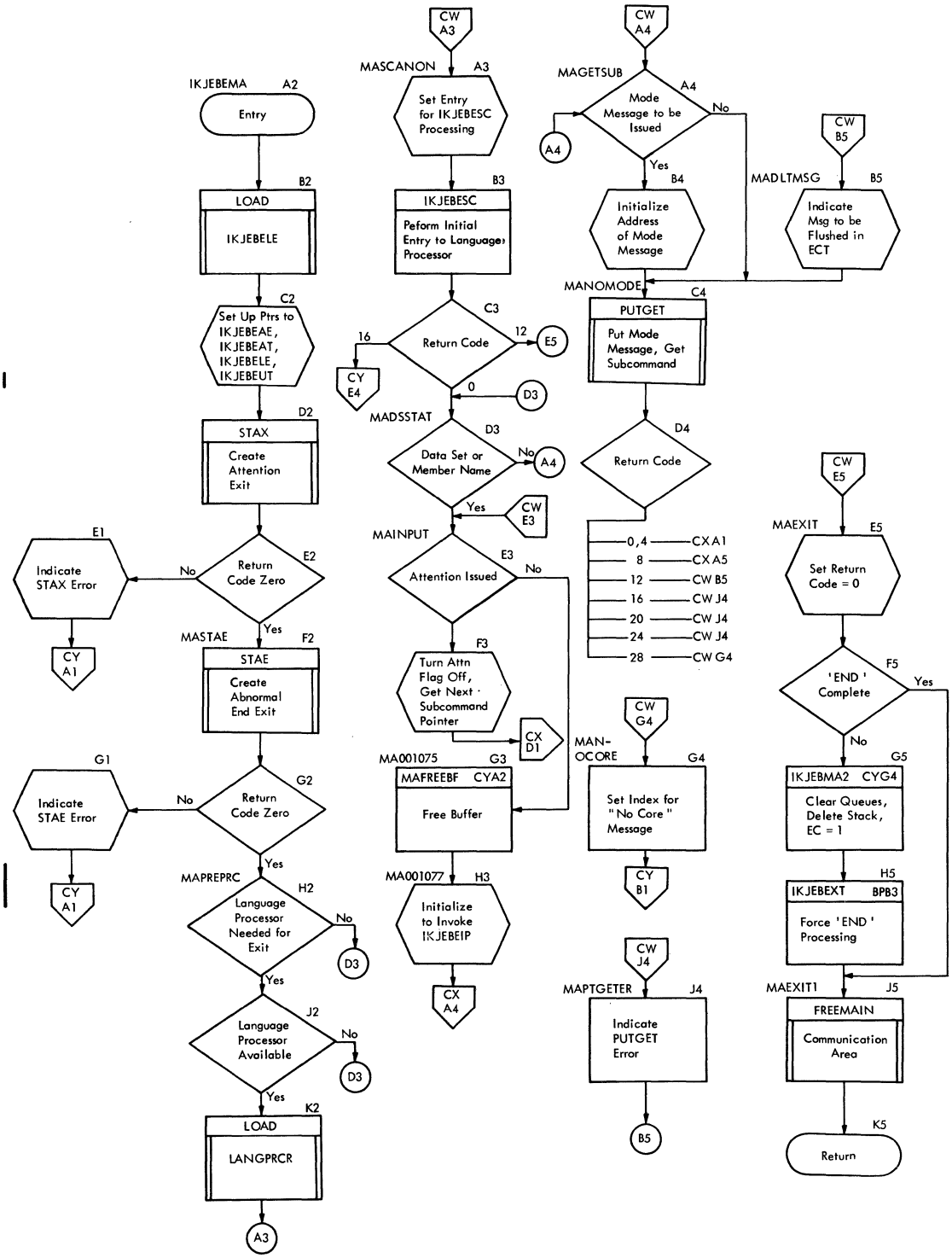


Chart CX. IKJEBEMA

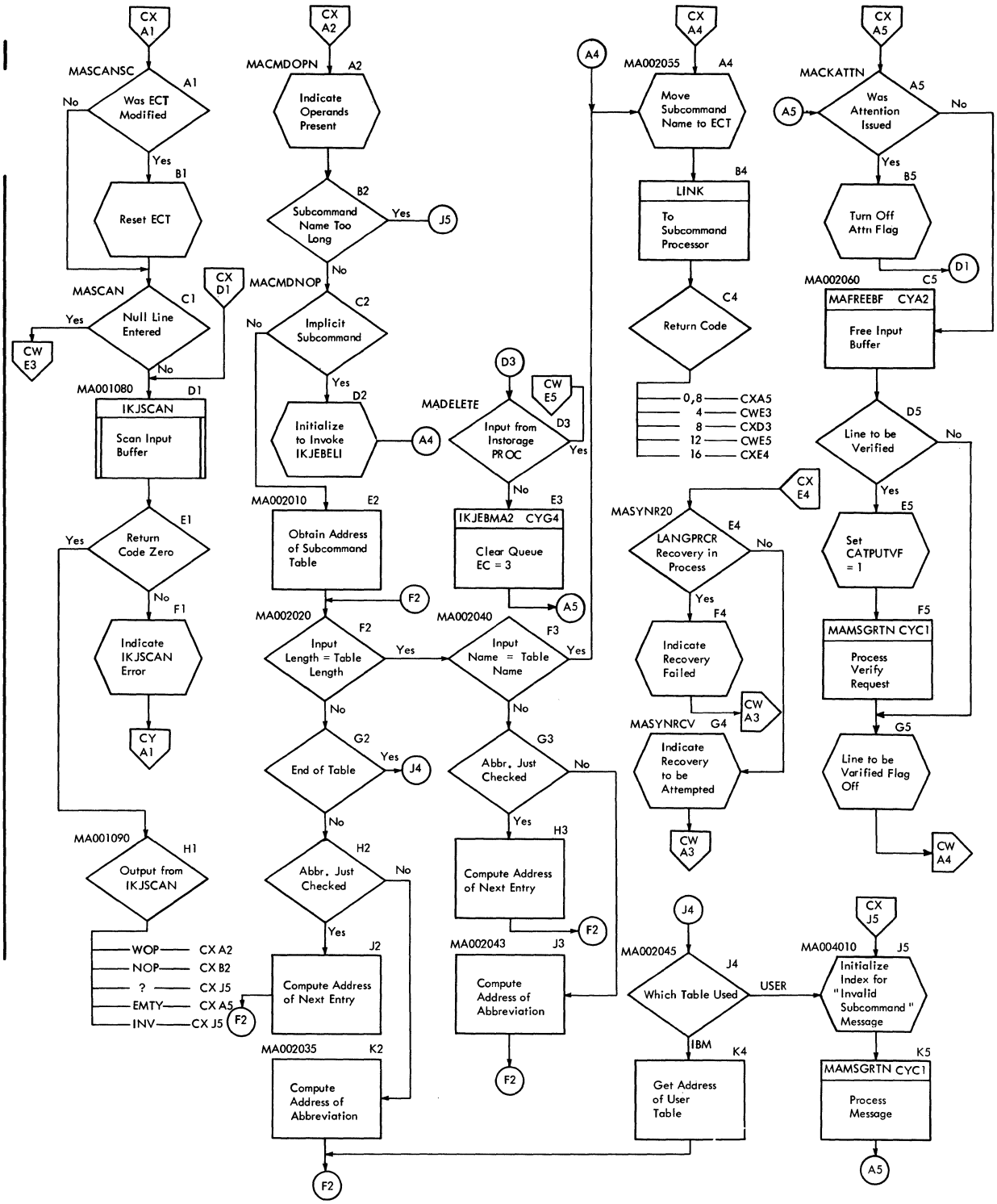


Chart CY. IKJEBEMA

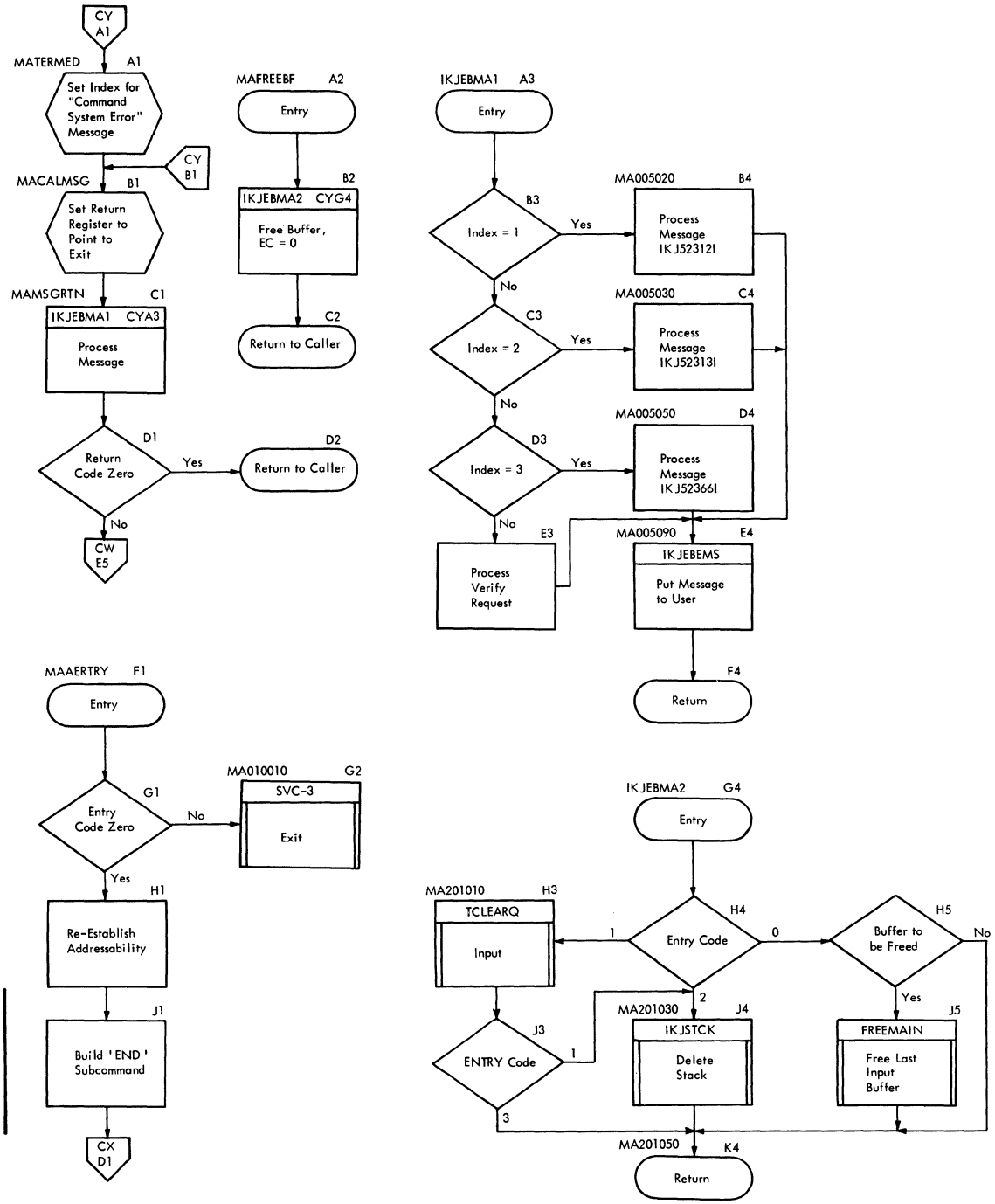


Chart CZ. IKJEBEME

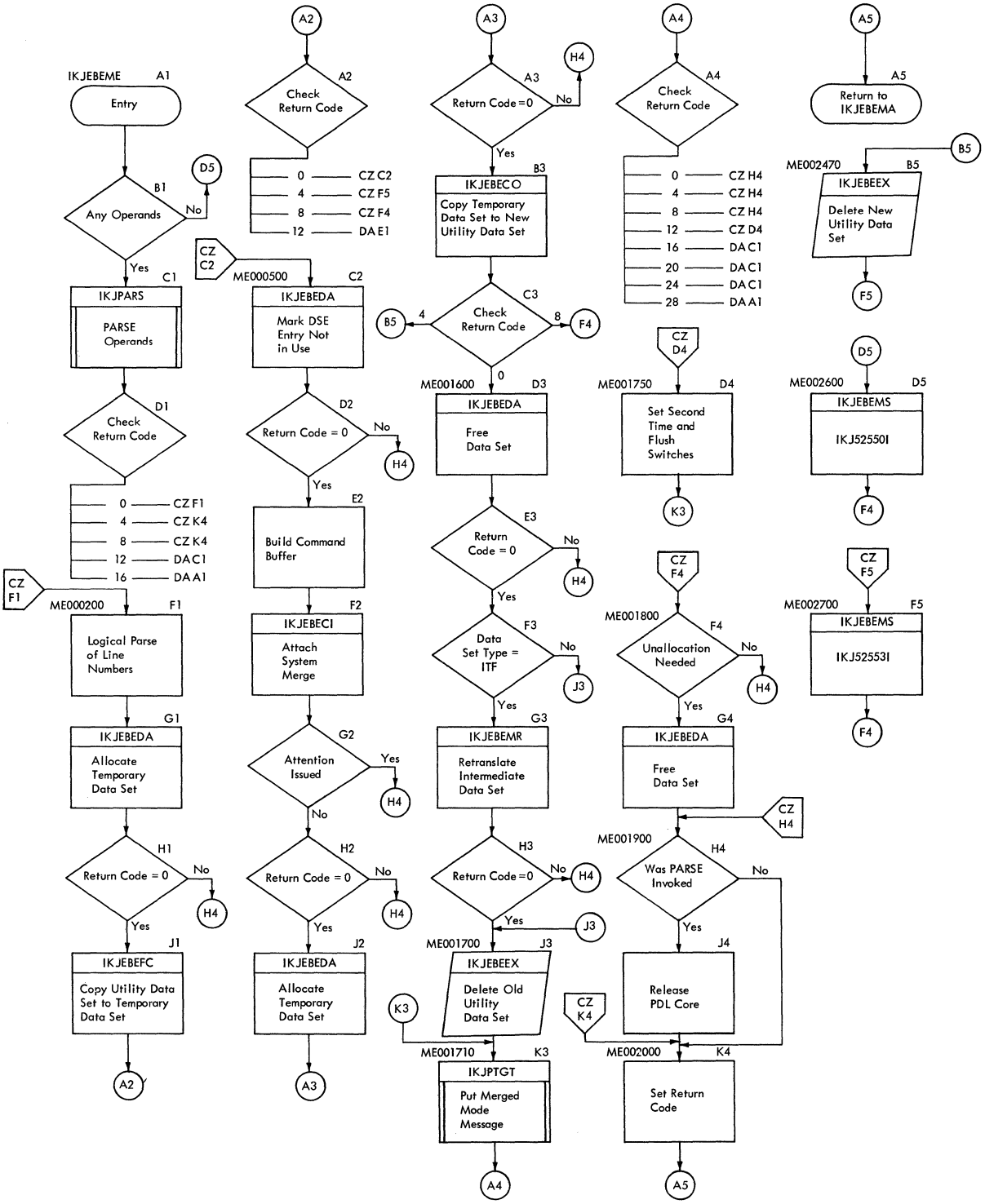


Chart DA. IKJEBEME

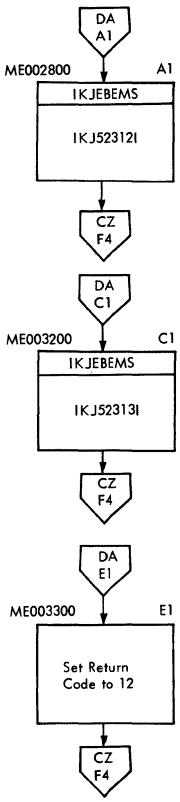


Chart DB. IKJEBEMR

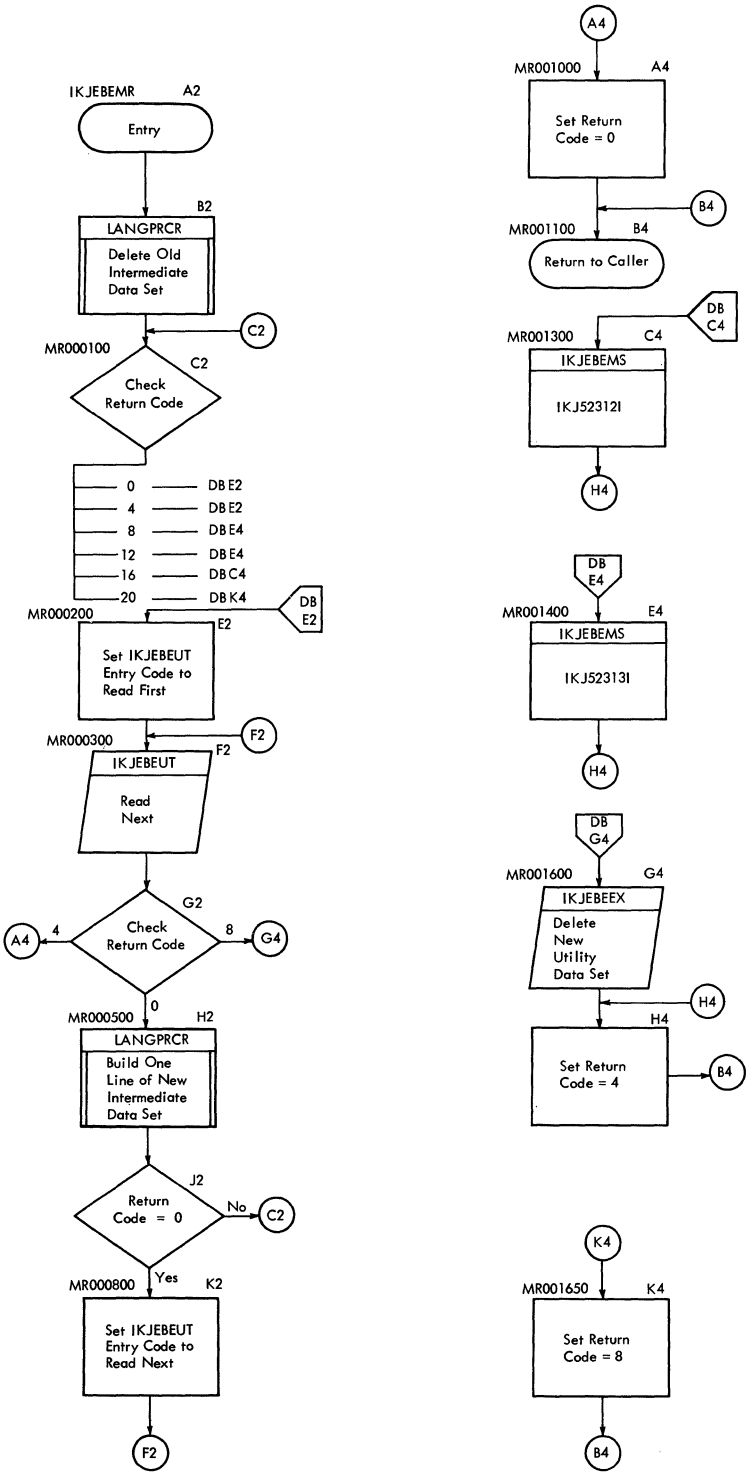


Chart DC. IKJEBEMS

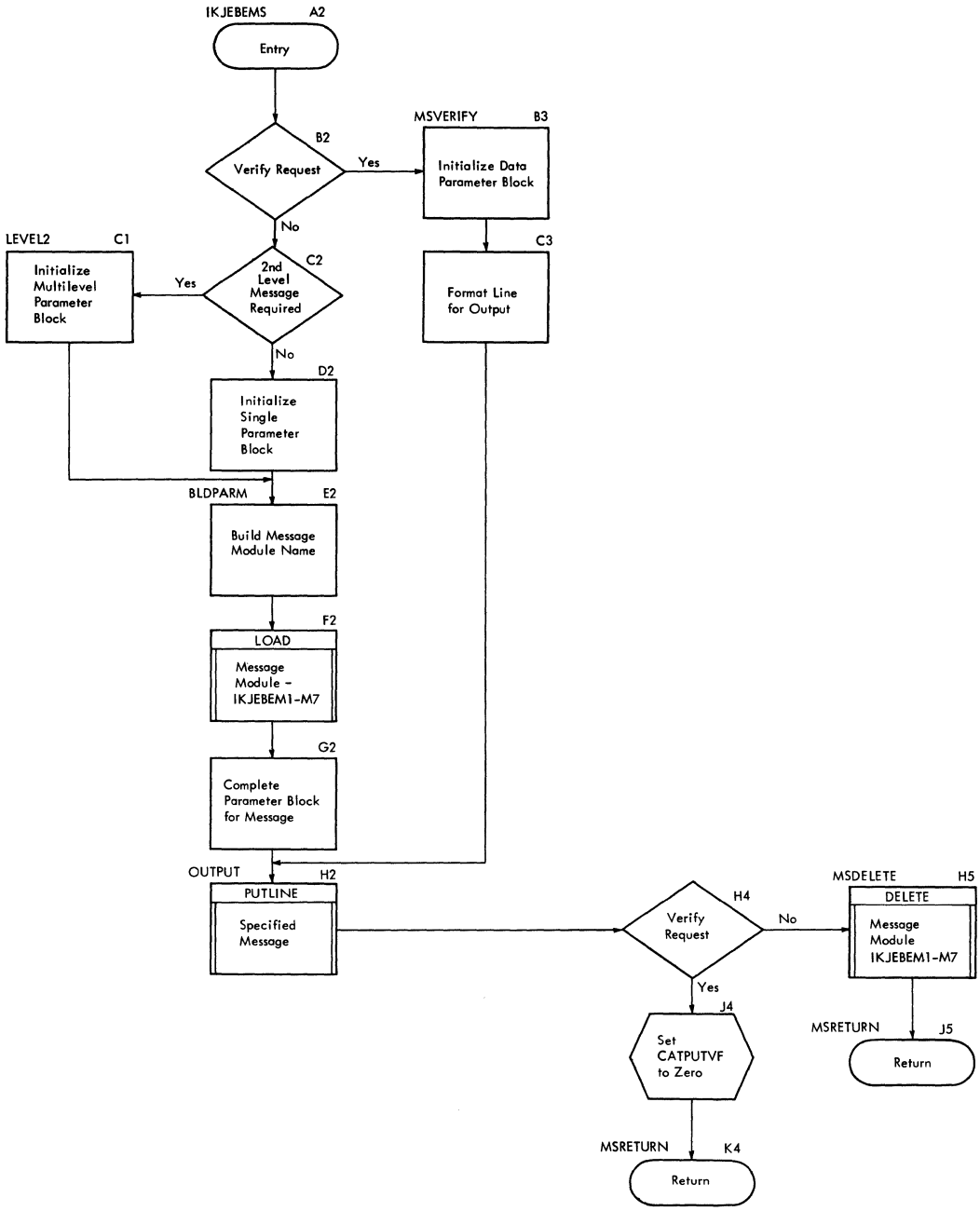


Chart DD. IKJEBEMV

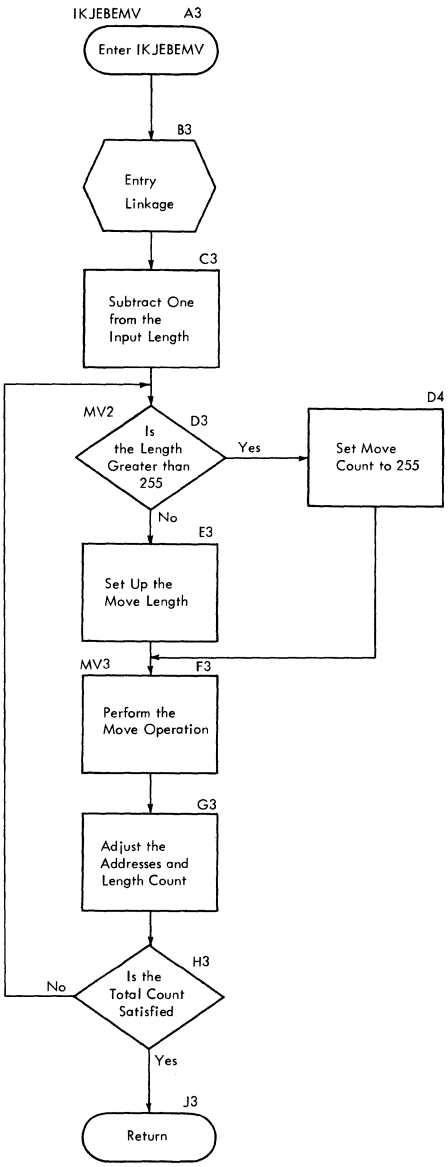


Chart DH. IKJEBEPS

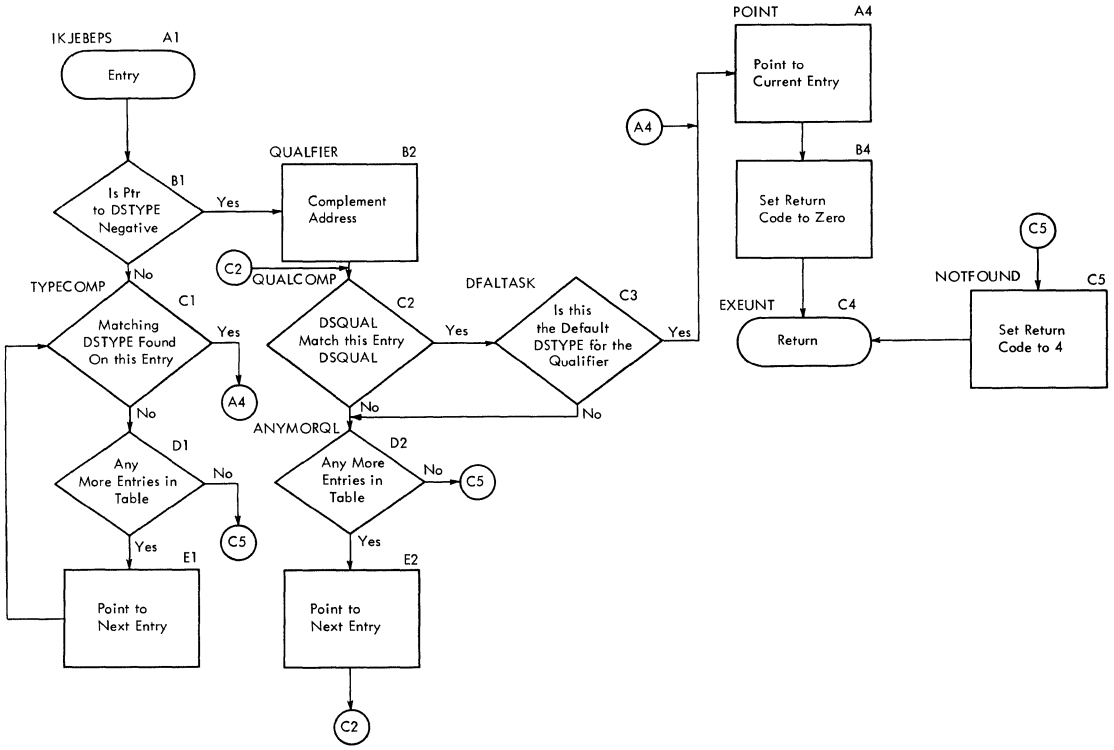


Chart DI. IKJEBERB

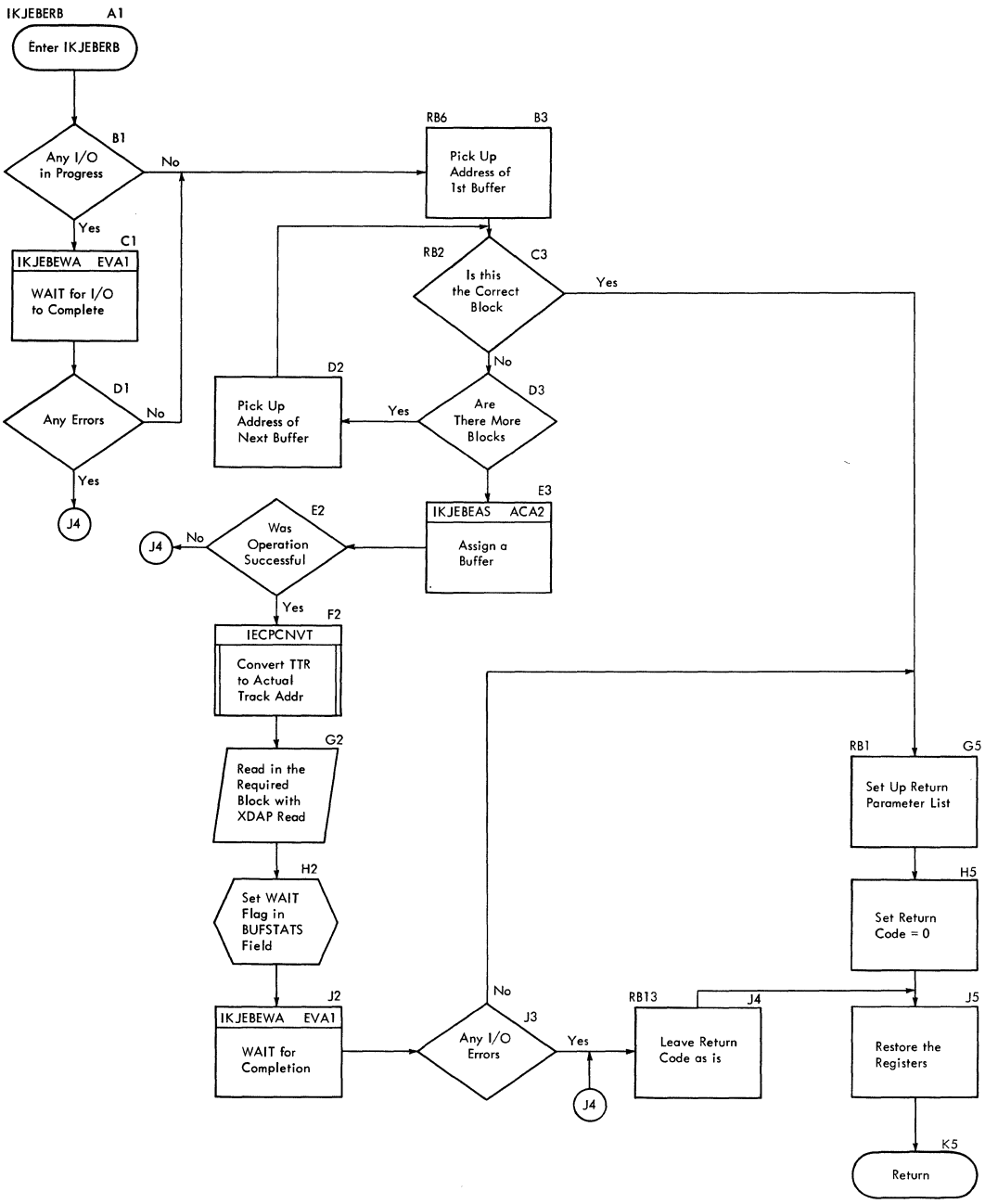


Chart DJ. IKJEBERE

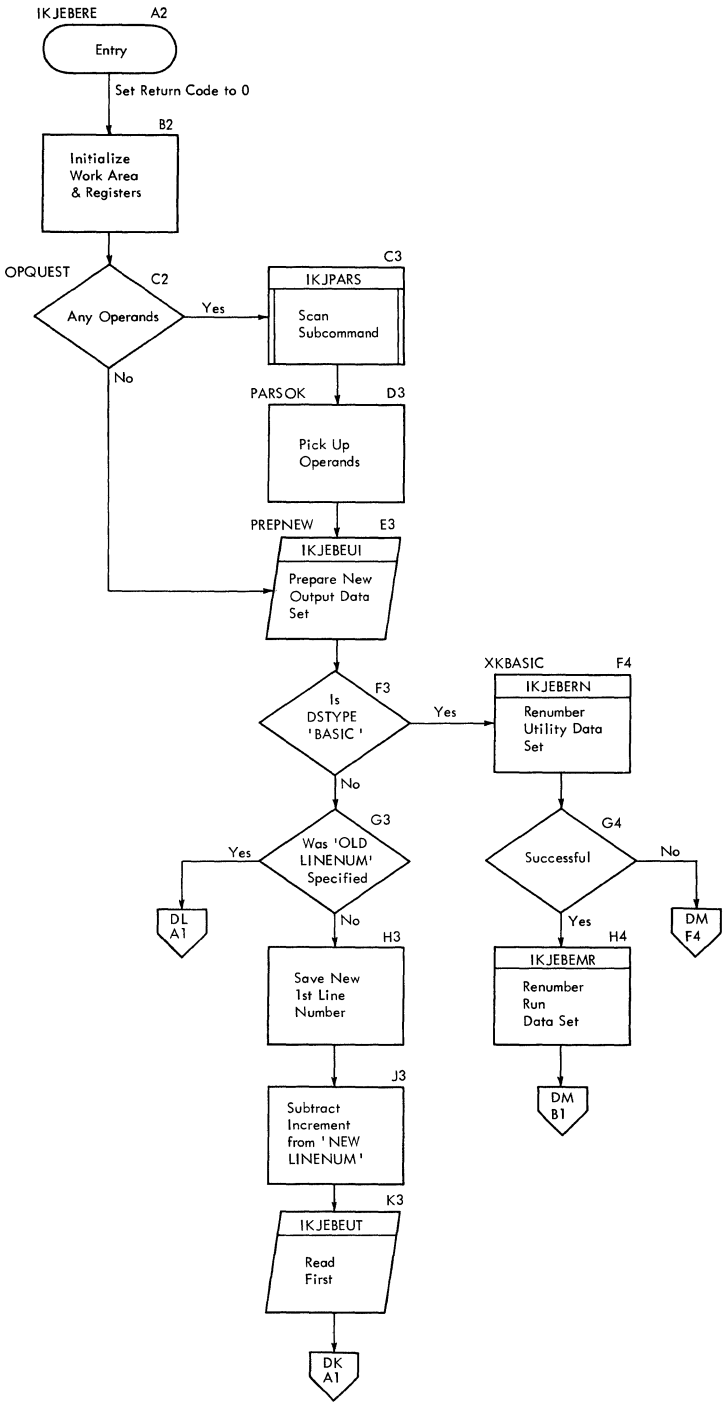


Chart DK. IKJEBERE

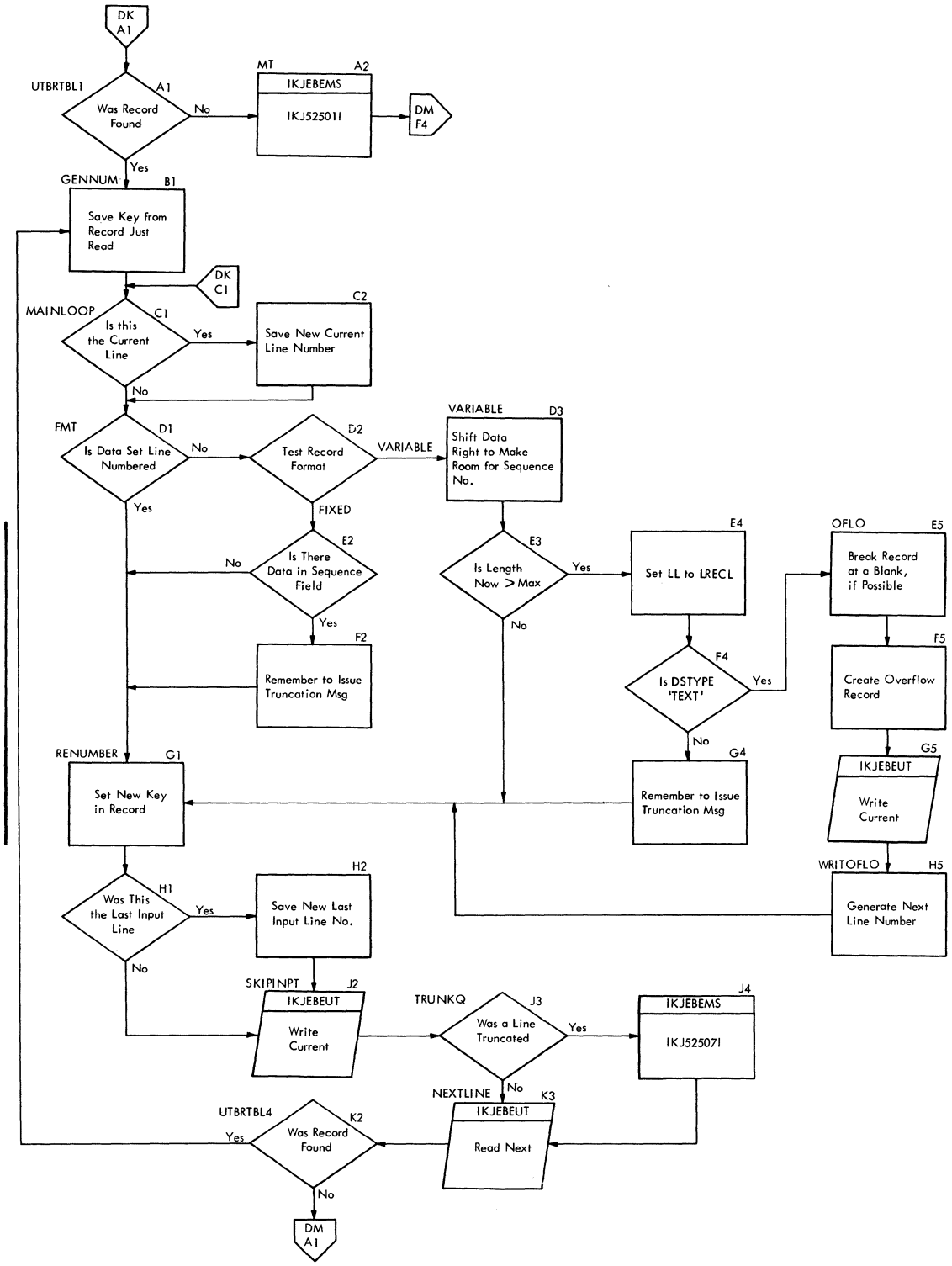


Chart DL. IKJEBERE

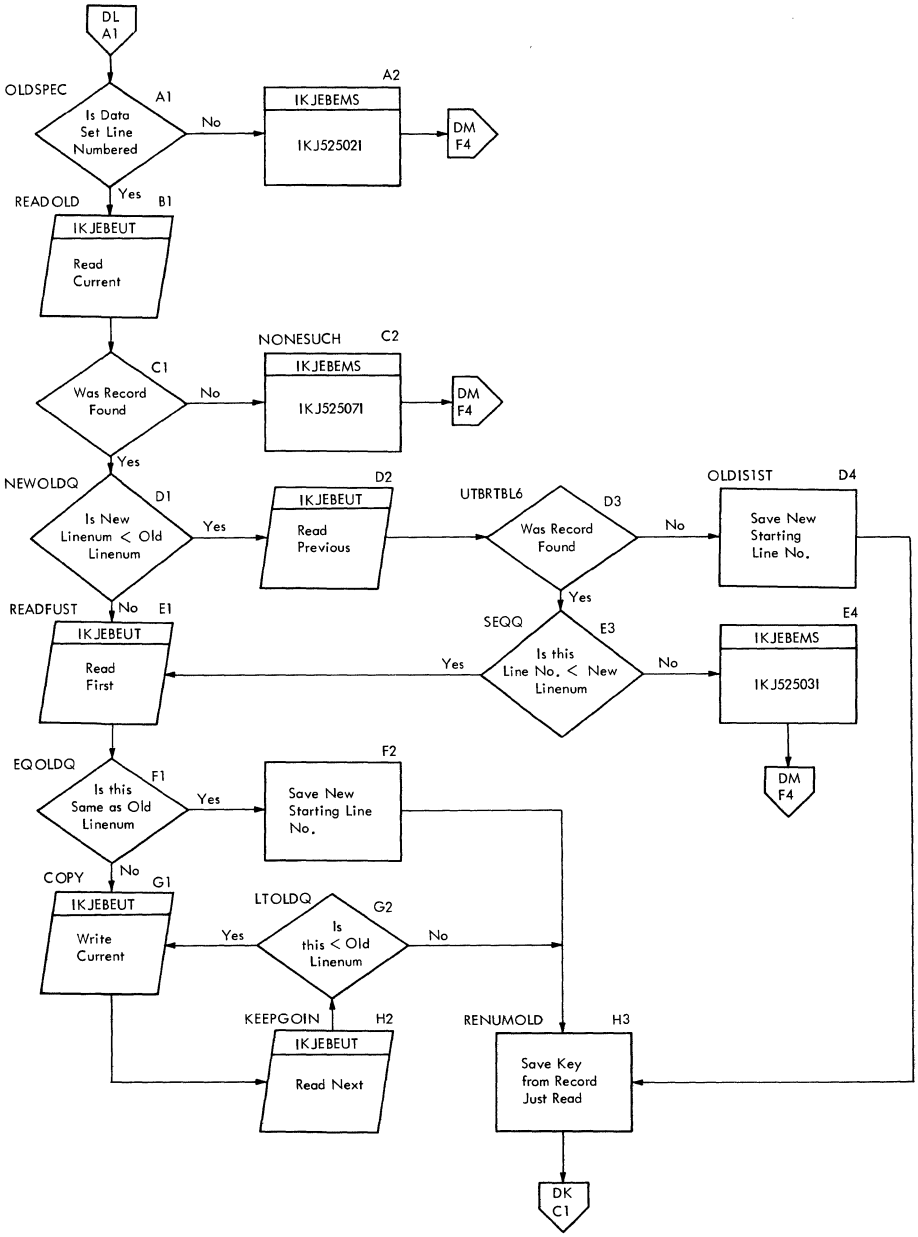


Chart DM. IKJEBERE

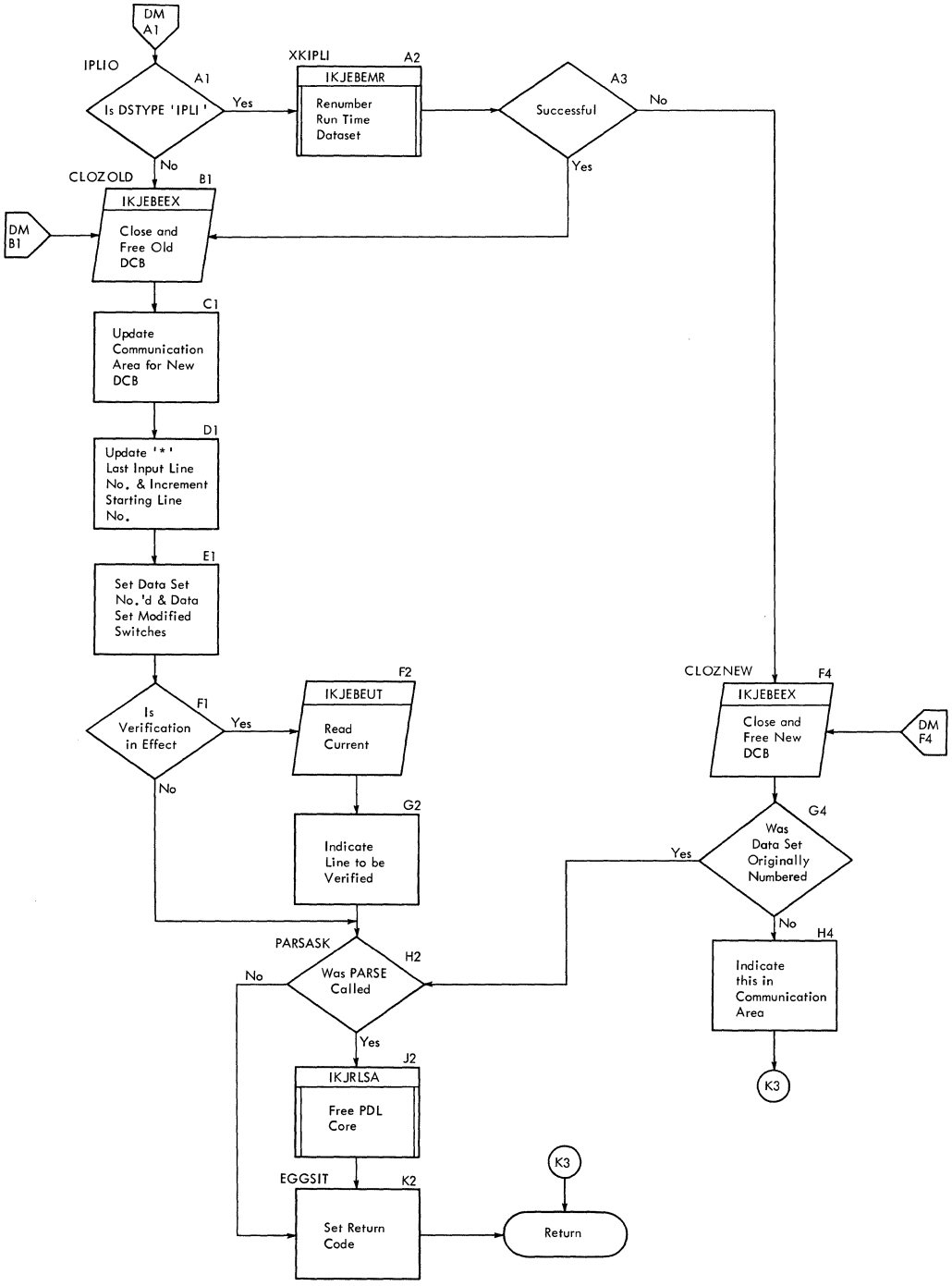
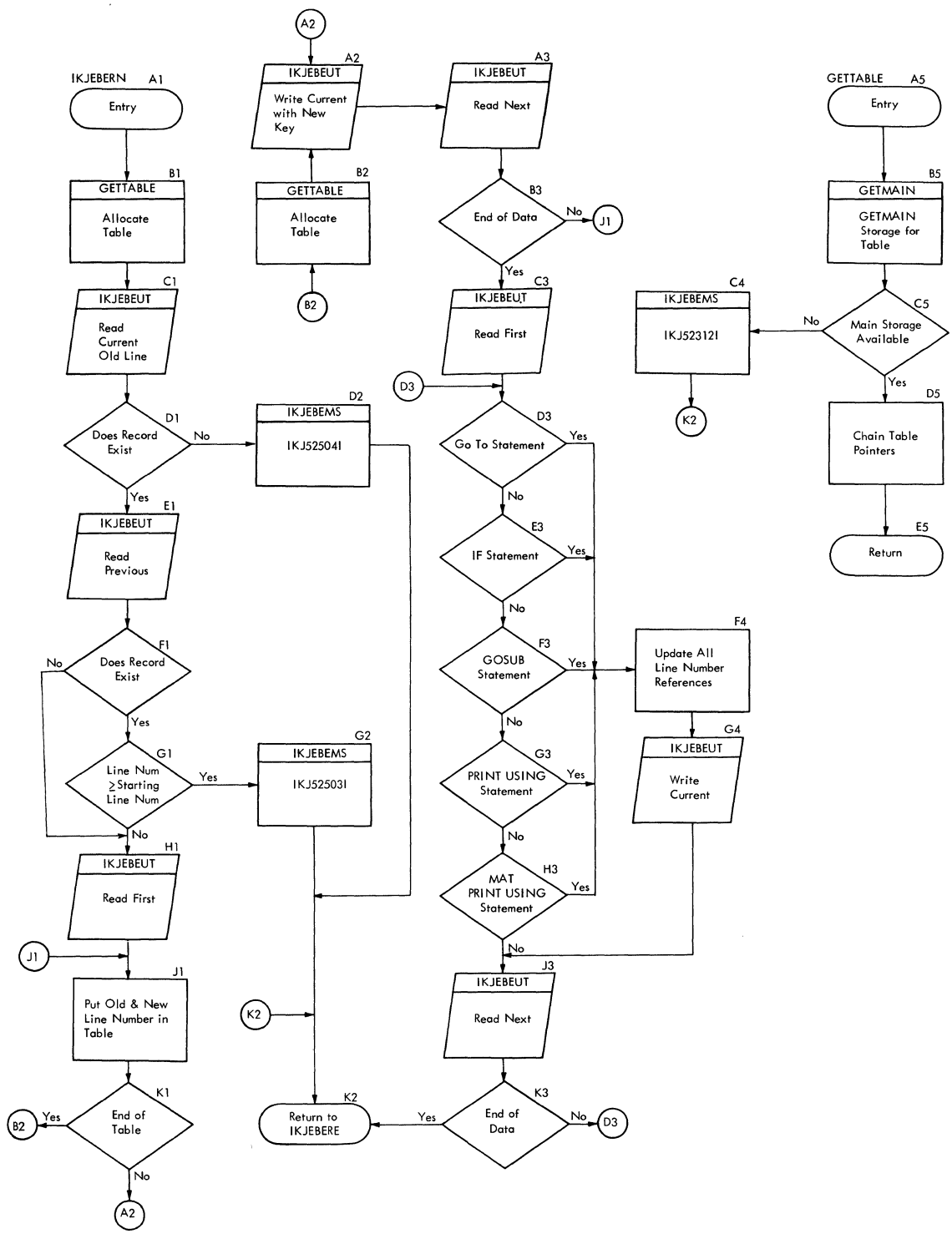


Chart DN. IKJEBERN



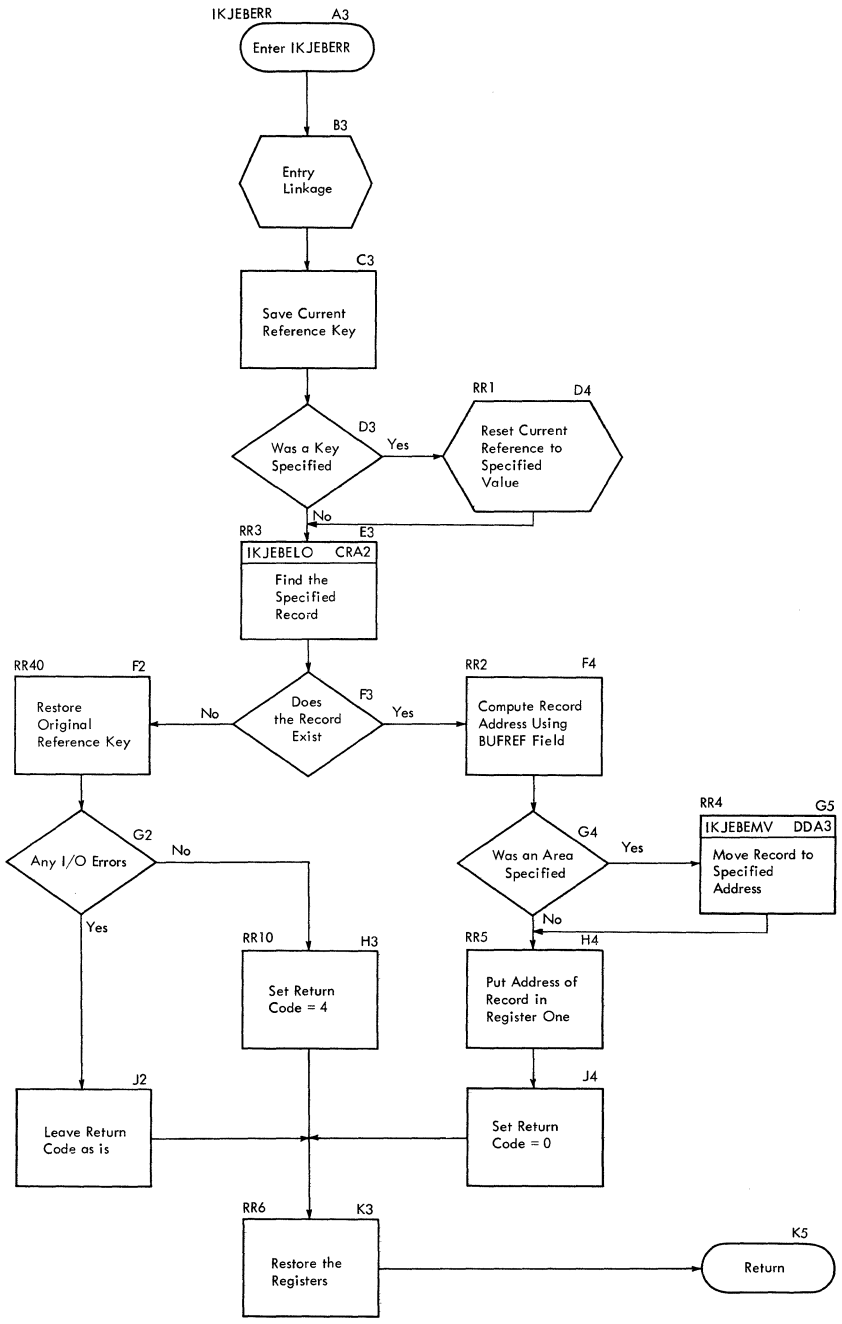

IKJEBERR
A3
Enter IKJEBERR
B3
Entry Linkage
C3
Save Current Reference Key
RR1
D4
D3
Was a Key Specified
Yes
Reset Current Reference to Specified Value
No
RR3
E3
IKJEBELO CRA2
Find the Specified Record
RR40
F2
Restore Original Reference Key
No
F3
Does the Record Exist
Yes
RR2
F4
Compute Record Address Using BUFREF Field
RR4
G5
IKJEBEMV DDA3
Move Record to Specified Address
G2
Any I/O Errors
No
Yes
G4
Was an Area Specified
Yes
No
RR10
H3
Set Return Code = 4
RR5
H4
Put Address of Record in Register One
J2
Leave Return Code as is
J4
Set Return Code = 0
RR6
K3
Restore the Registers
K5
Return

Chart DP. IKJEBERU

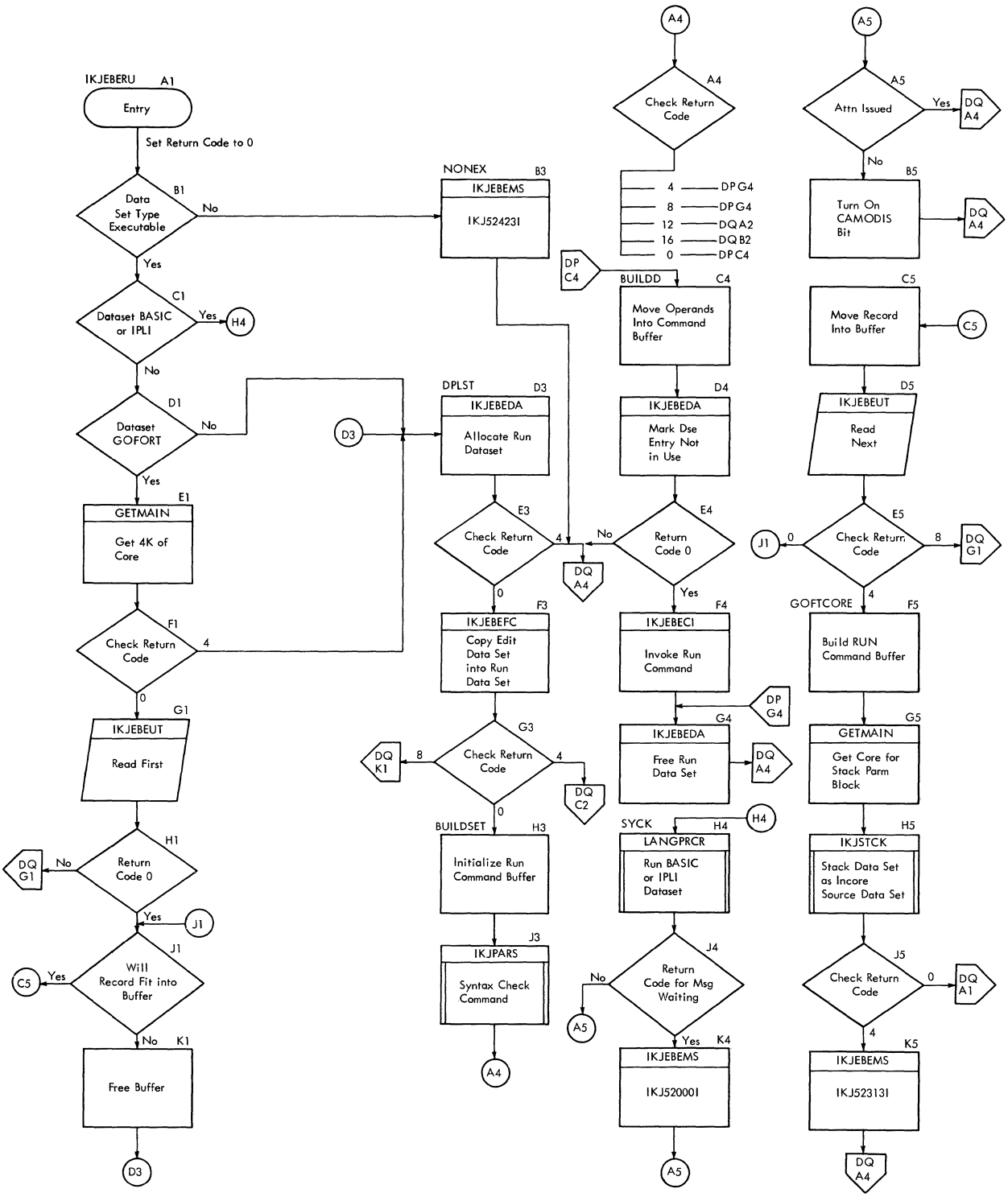


Chart DQ. IKJEBERU

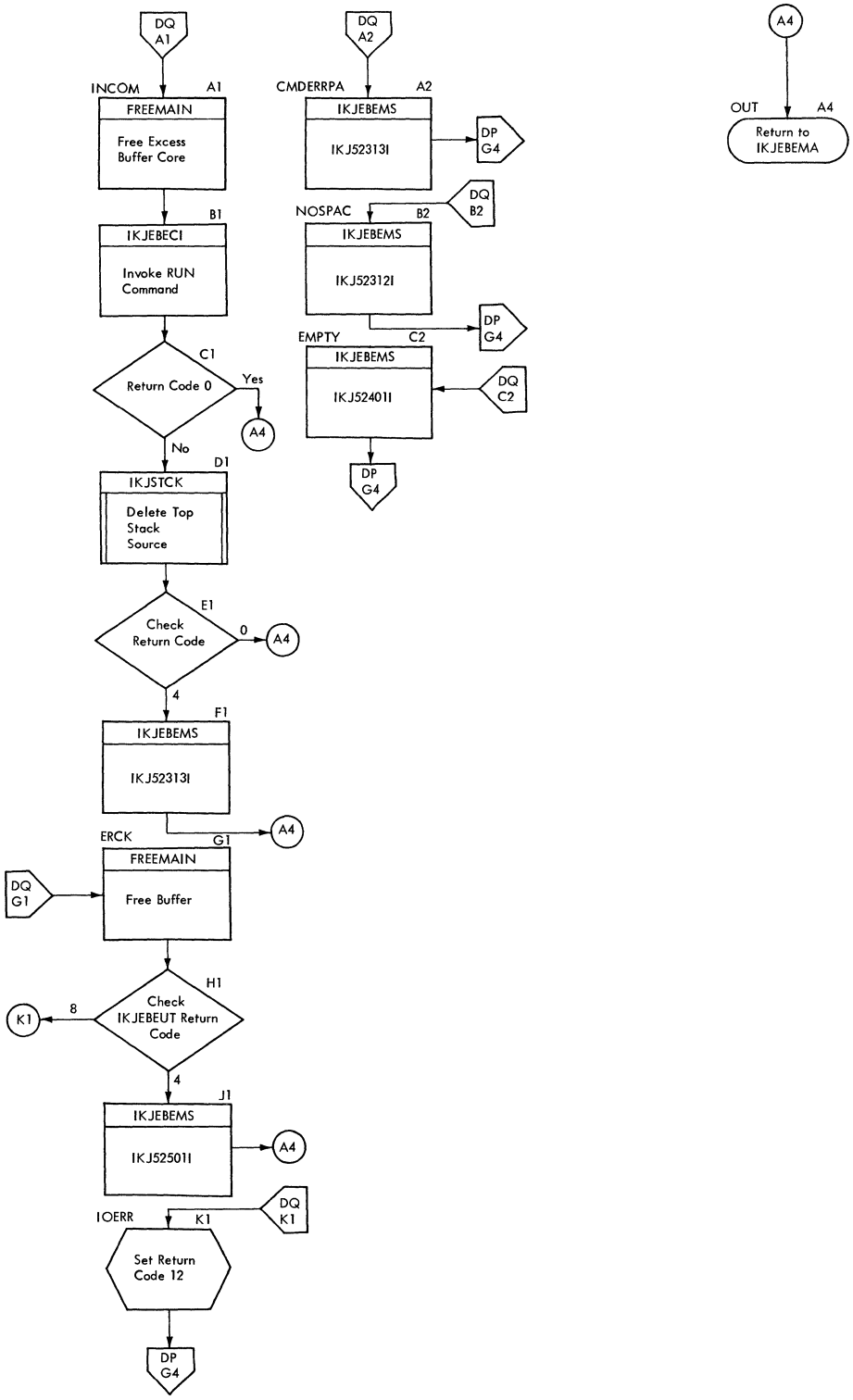


Chart DR. IKJEBESA

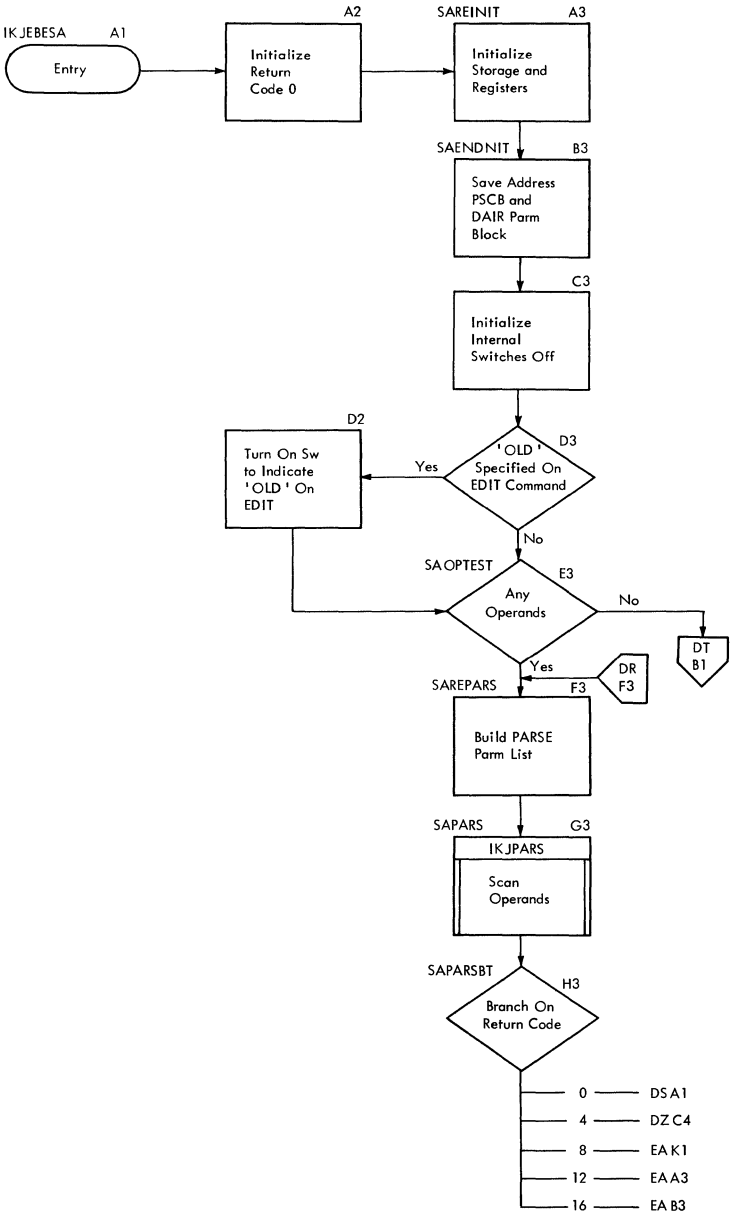


Chart DS. IKJEBESA

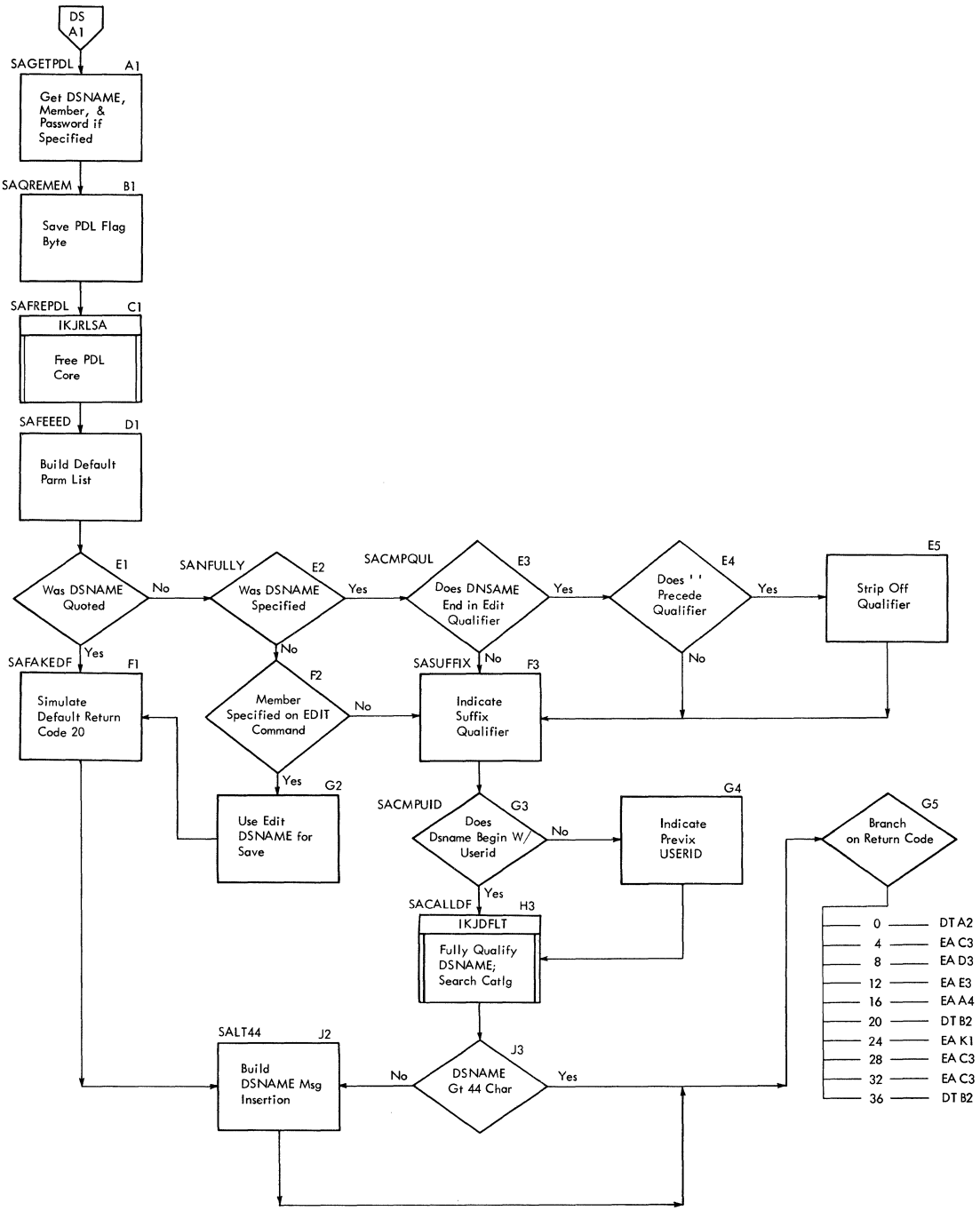


Chart DT. IKJEBESA

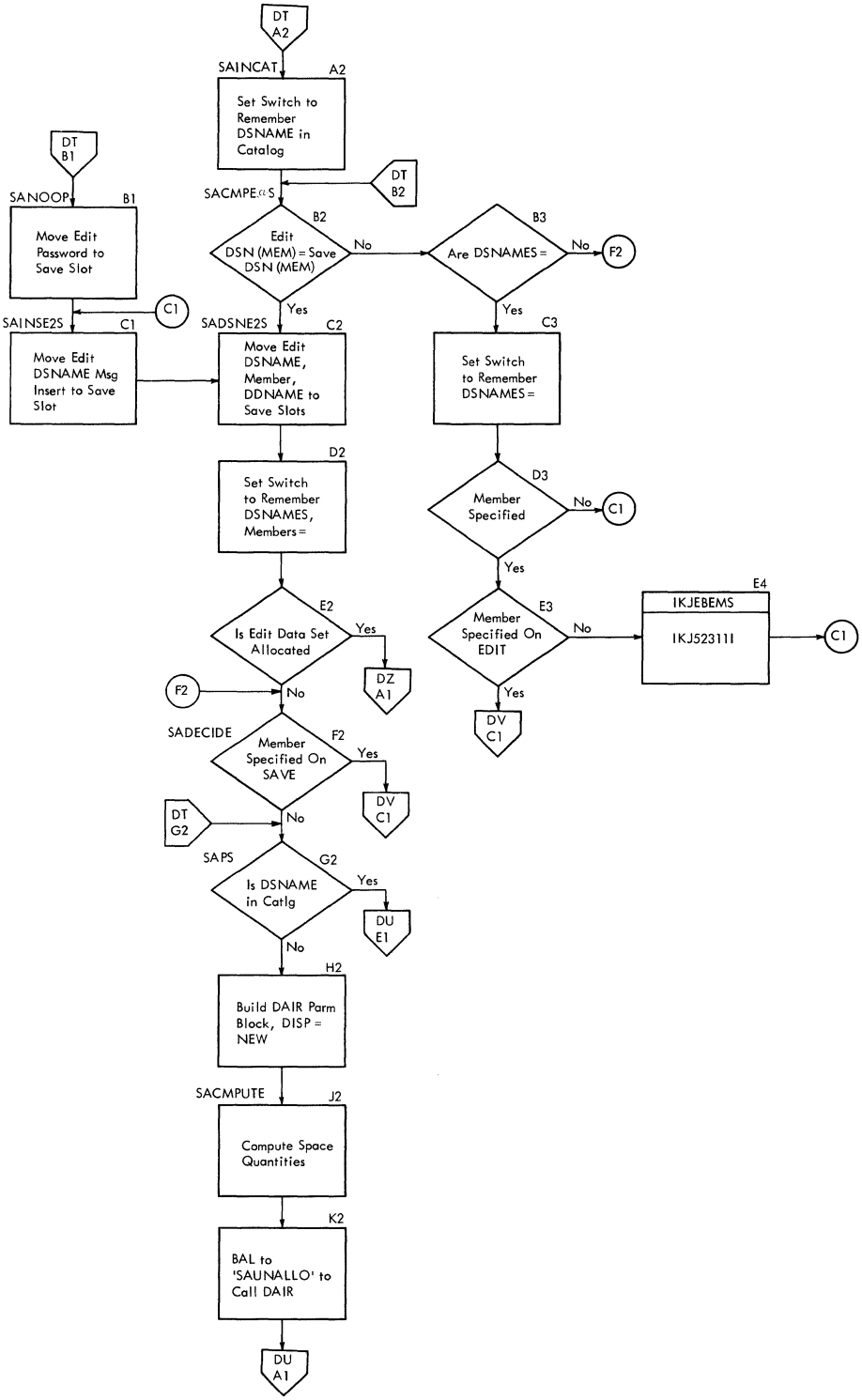


Chart DU. IKJEBESA

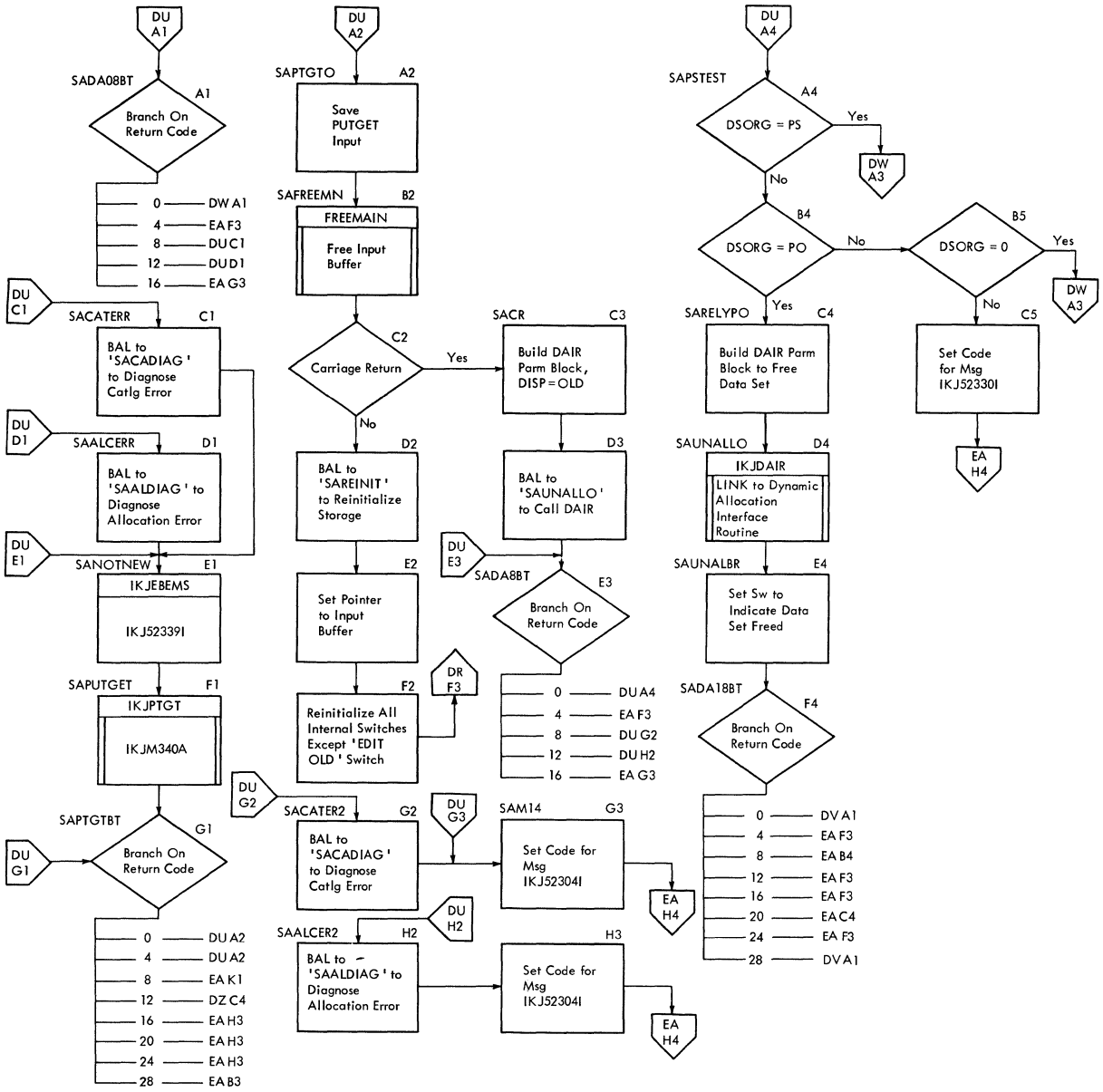


Chart DV. IKJEBESA

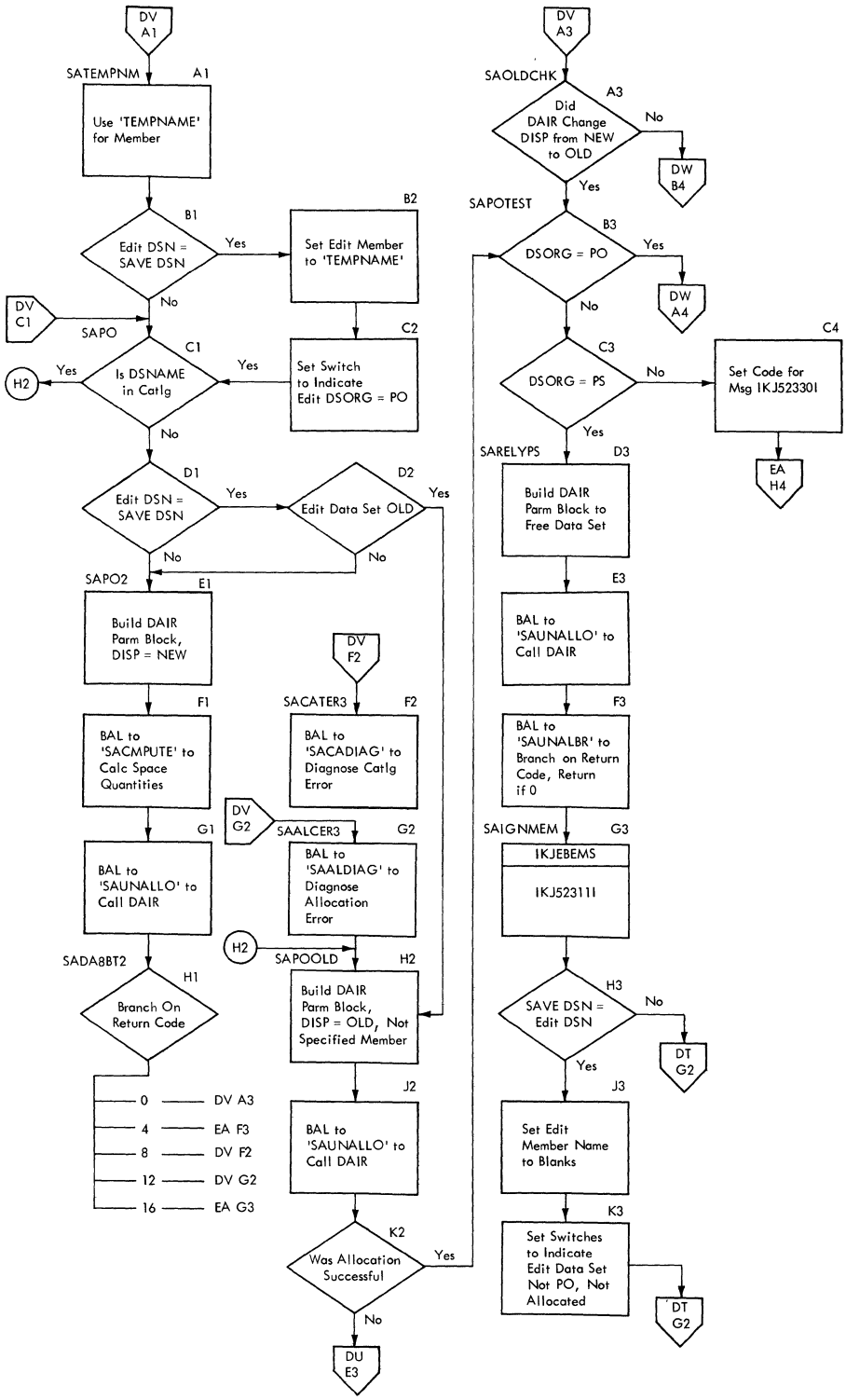


Chart DW. IKJEBESA

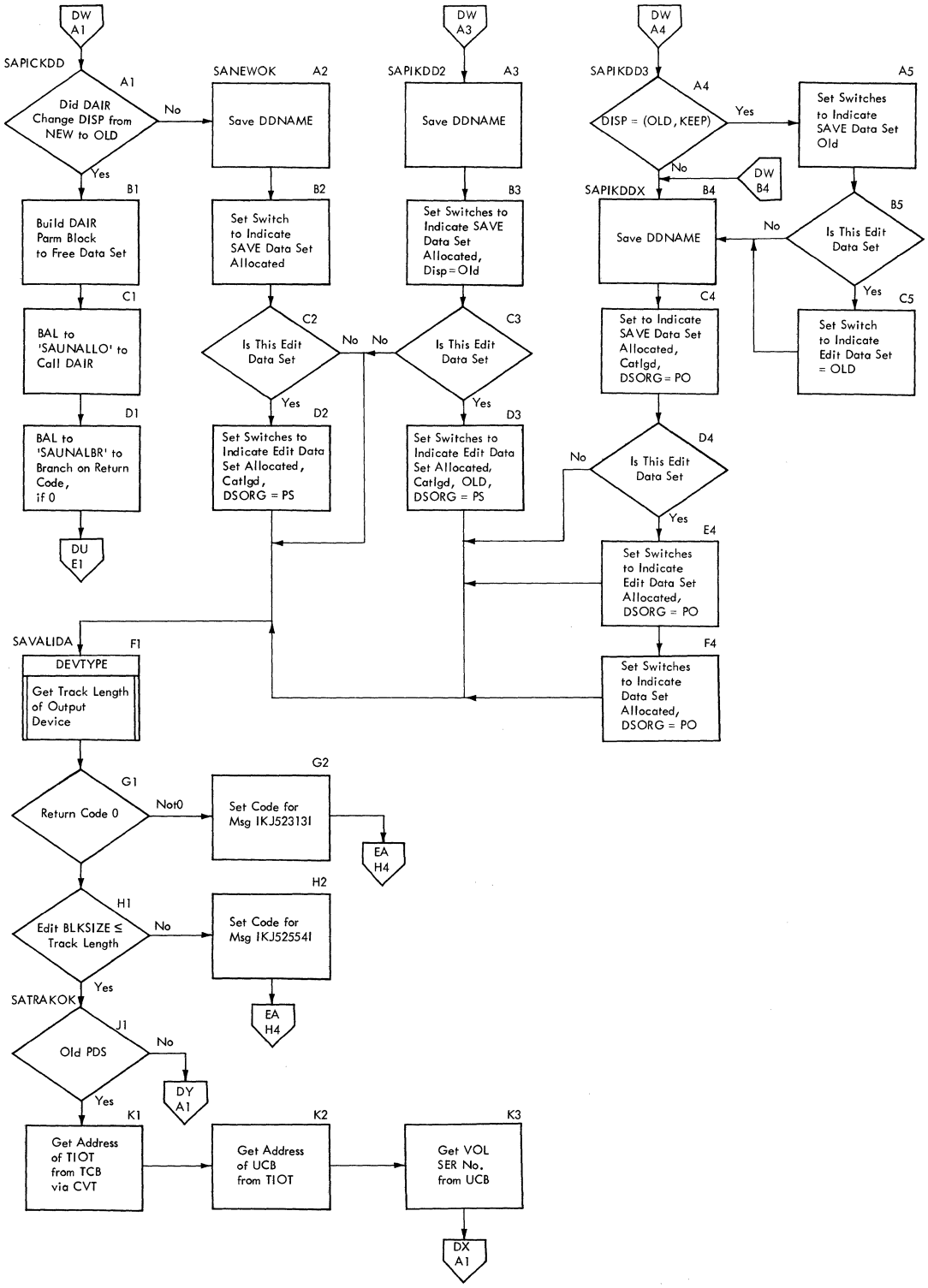


Chart DX. IKJEBESA

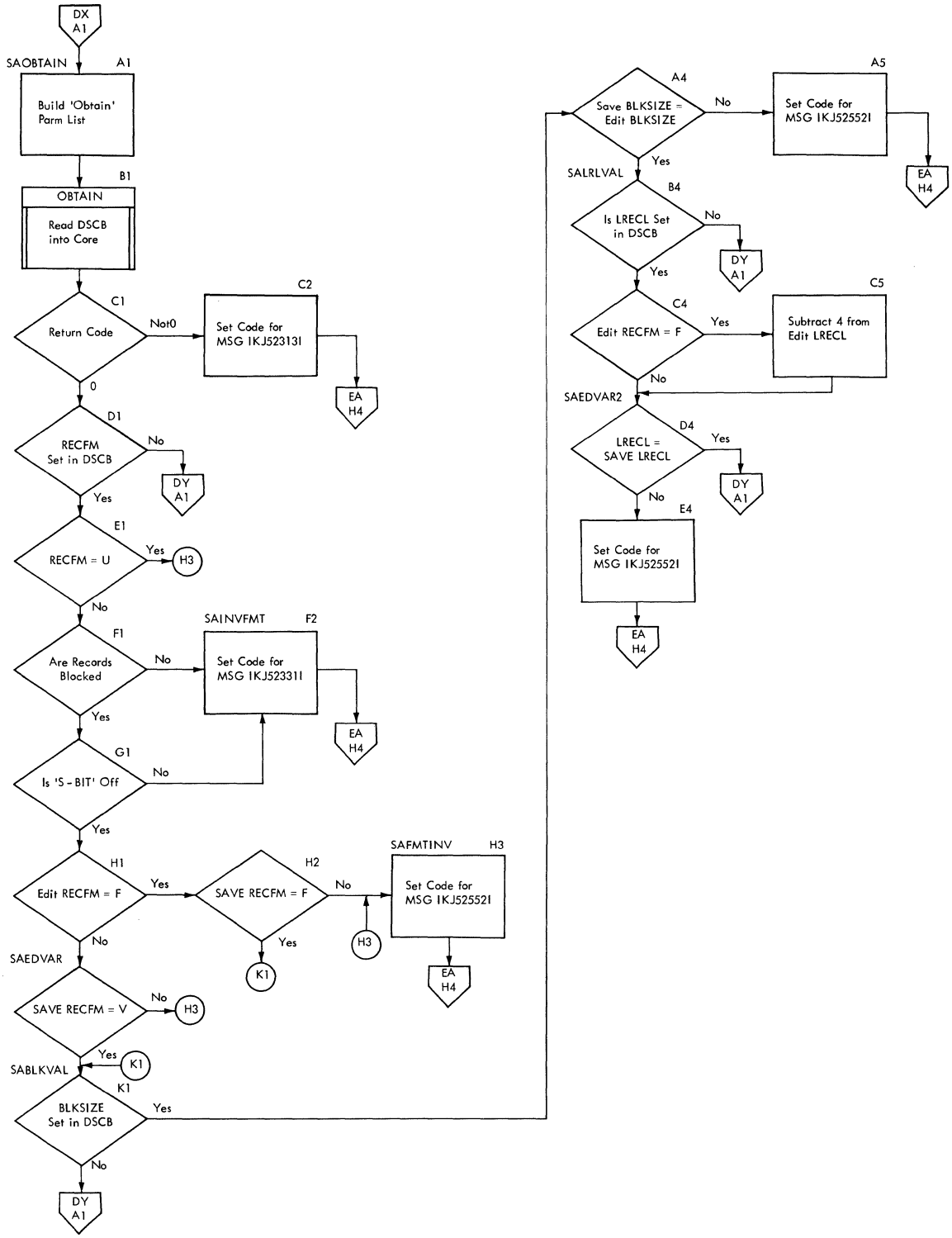


Chart DY. IKJEBESA

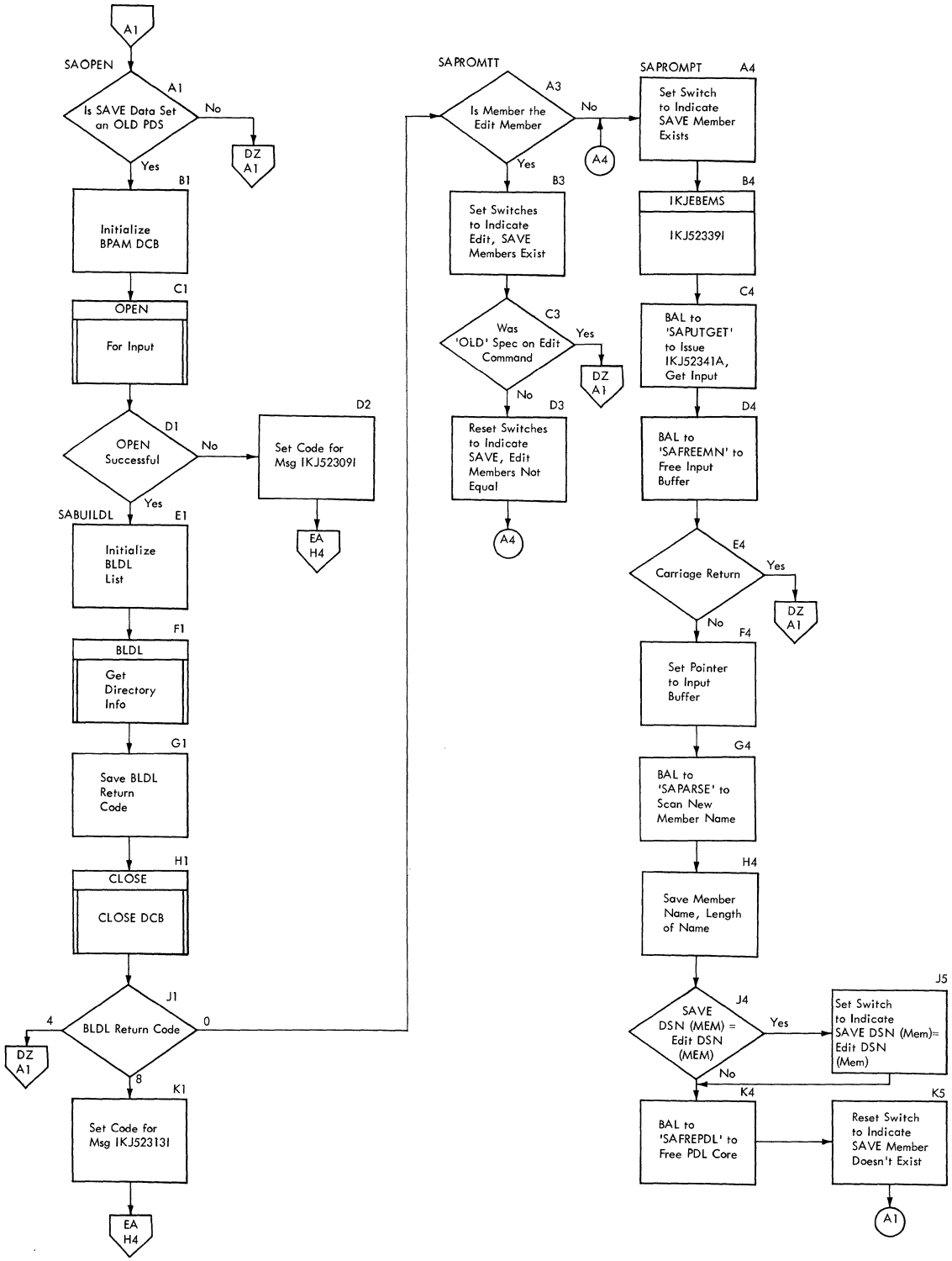


Chart DZ. IKJEBESA

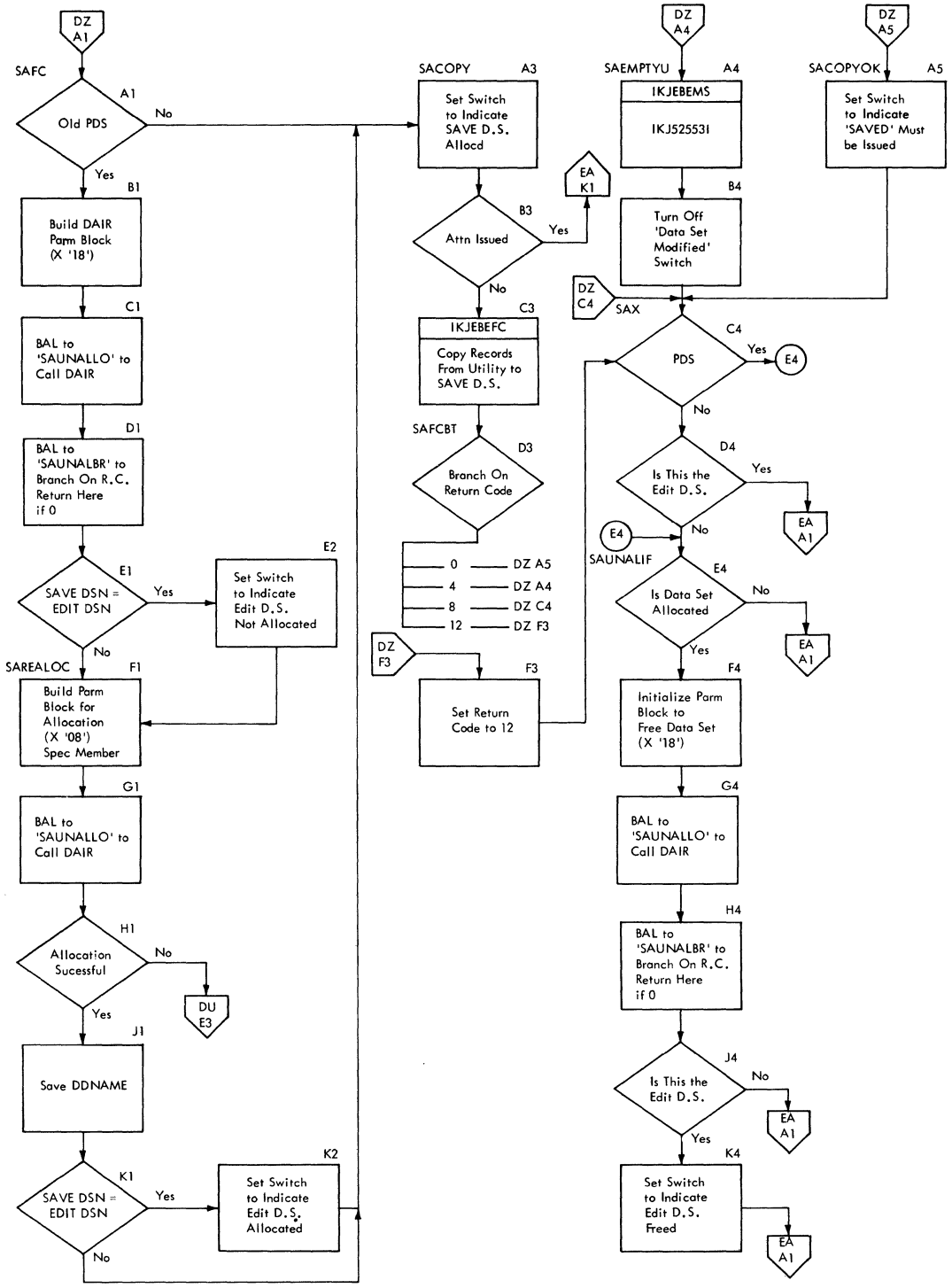


Chart EA. IKJEBESA

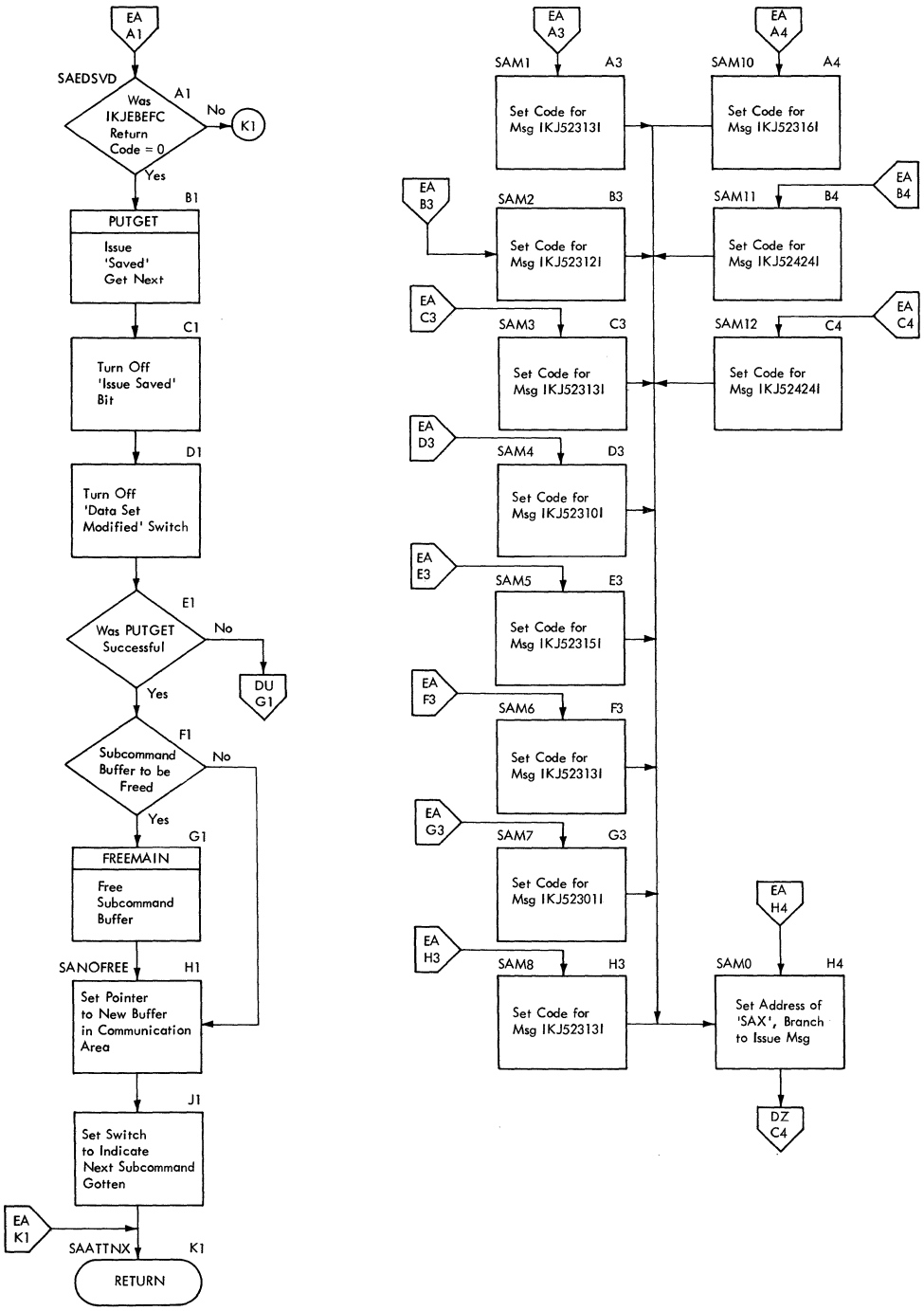


Chart EB. IKJEBESA

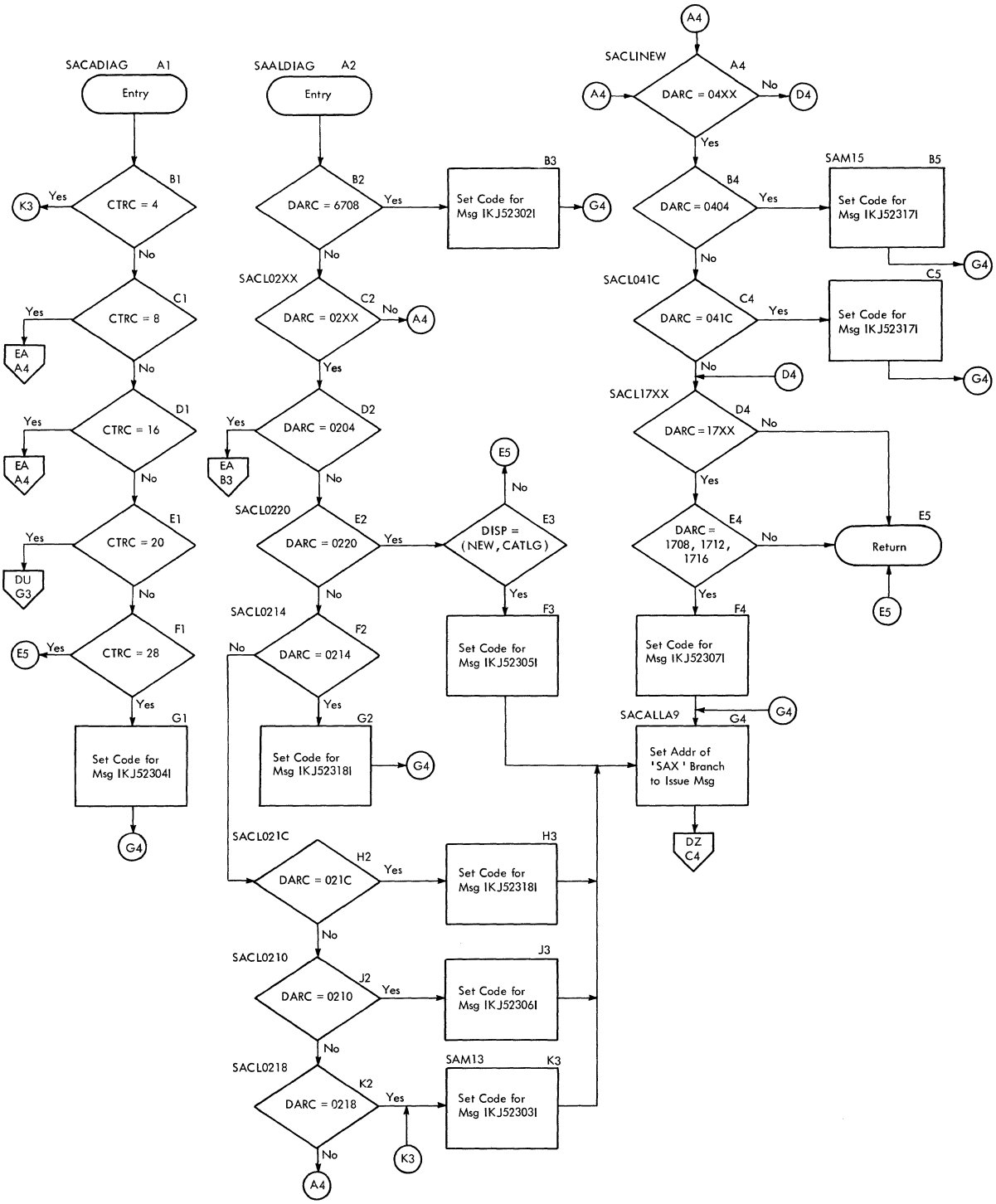


Chart EC. IKJEBESC

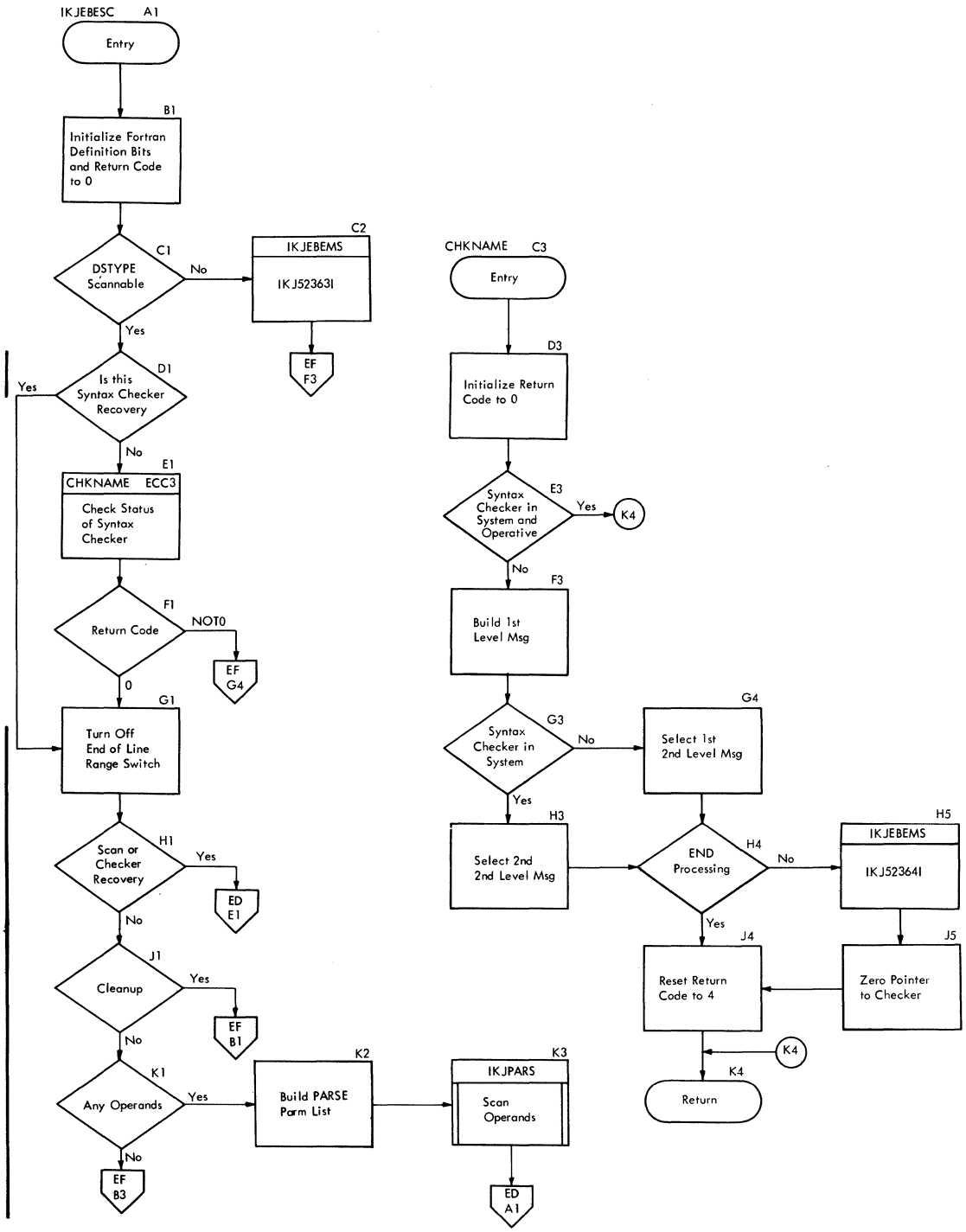


Chart ED. IKJEBESC

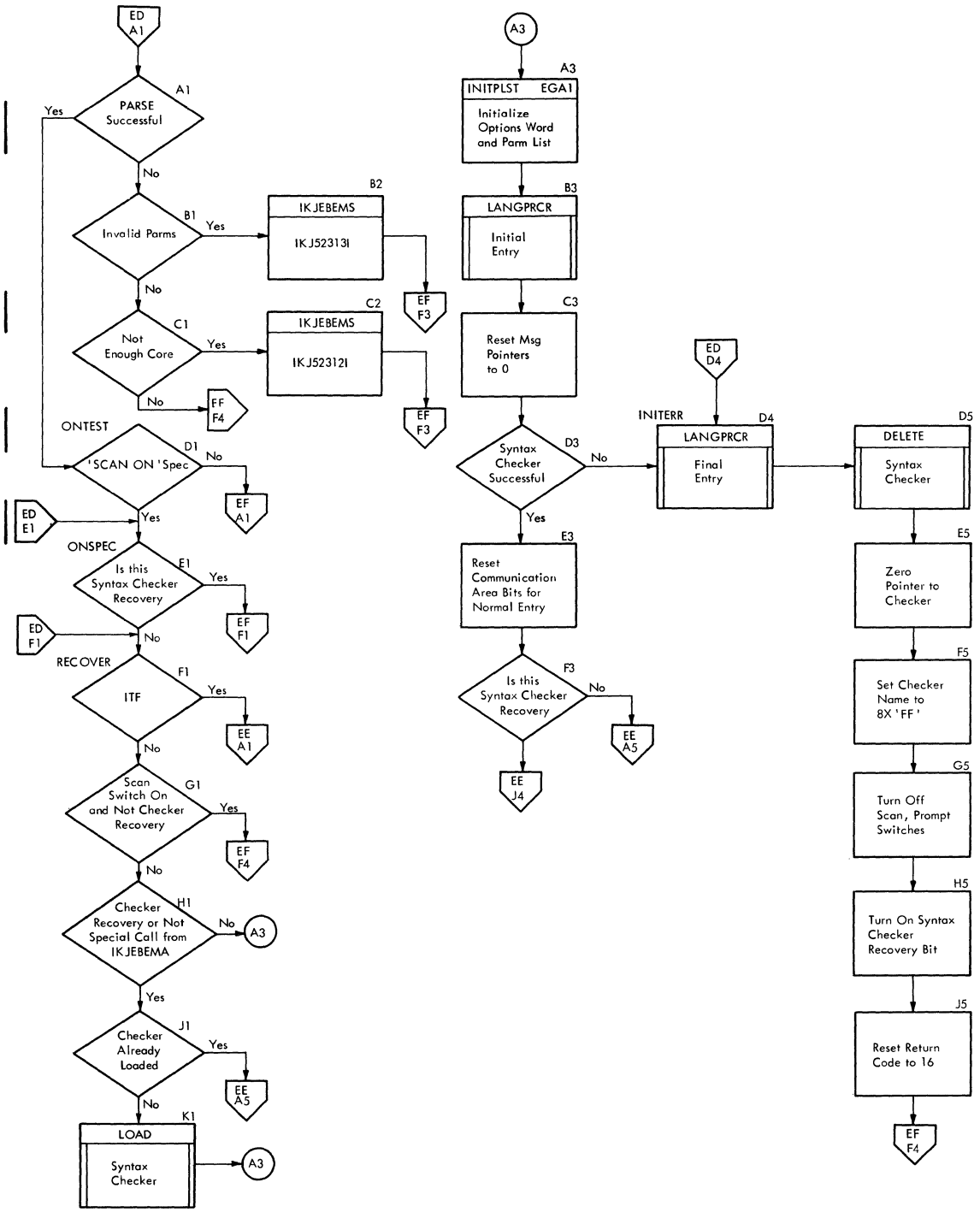


Chart EE. IKJEBESC

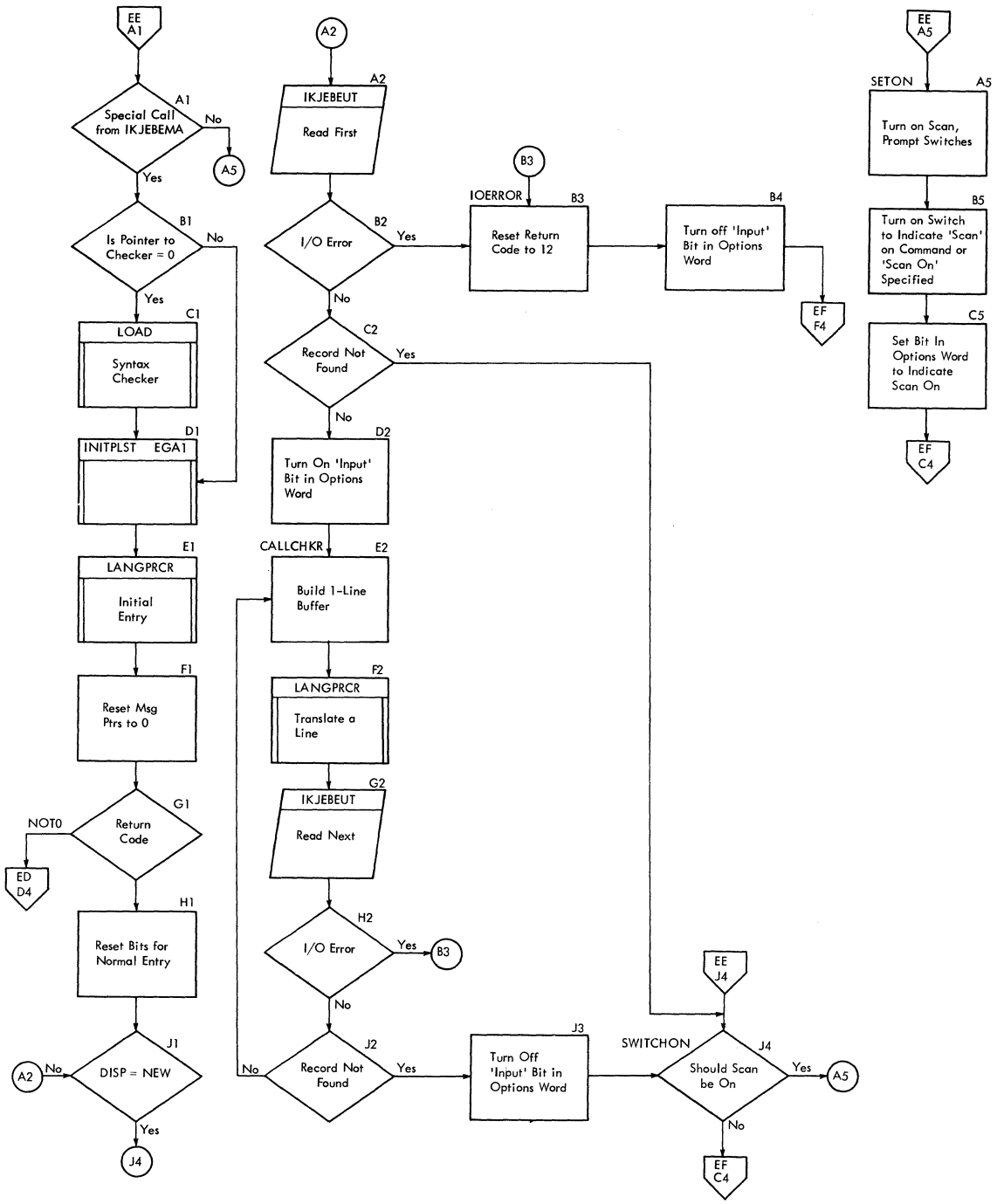
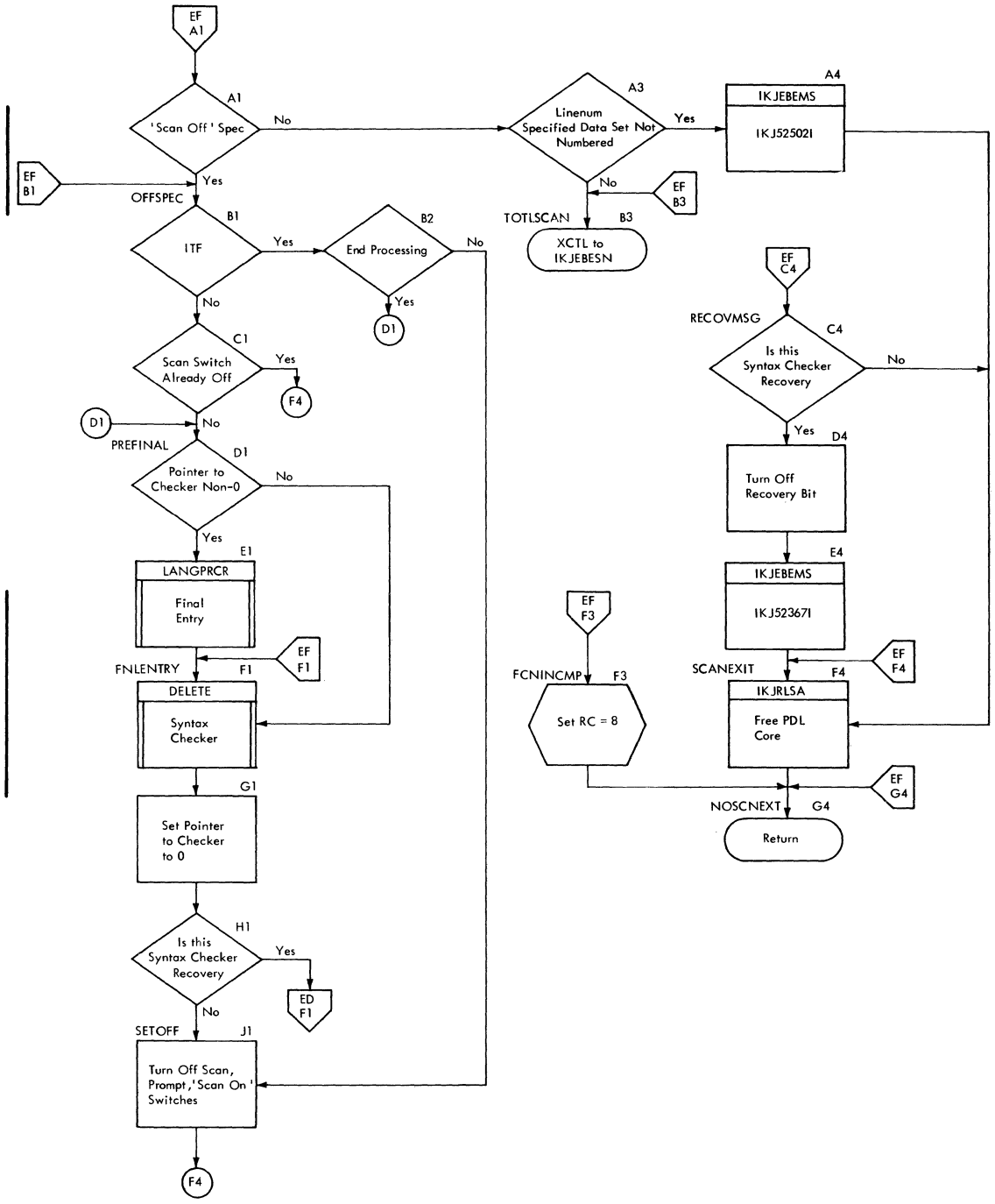


Chart EF. IKJEBESC



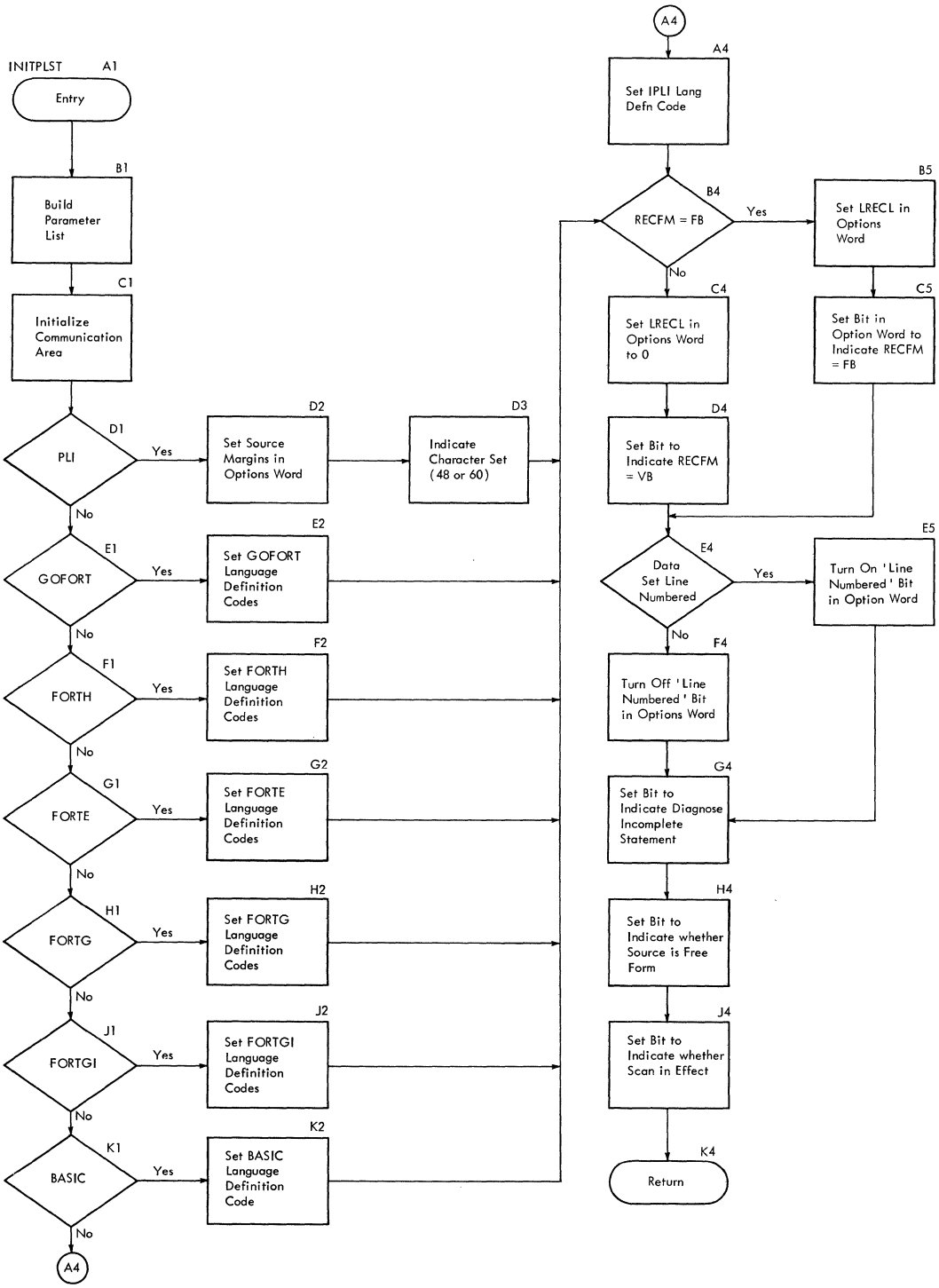

INITPLST
A1
Entry
B1
Build Parameter List
C1
Initialize Communication Area
D1
PLI
Yes
No
D2
Set Source Margins in Options Word
D3
Indicate Character Set (48 or 60)
E1
GOFORT
Yes
No
E2
Set GOFORT Language Definition Codes
F1
FORTH
Yes
No
F2
Set FORTH Language Definition Codes
G1
FORTE
Yes
No
G2
Set FORTE Language Definition Codes
H1
FORTG
Yes
No
H2
Set FORTG Language Definition Codes
J1
FORTGI
Yes
No
J2
Set FORTGI Language Definition Codes
K1
BASIC
Yes
No
K2
Set BASIC Language Definition Code
A4
A4
Set IPLI Lang Defn Code
B4
RECFM = FB
Yes
No
B5
Set LRECL in Options Word
C4
Set LRECL in Options Word to 0
C5
Set Bit in Option Word to Indicate RECFM = FB
D4
Set Bit to Indicate RECFM = VB
E4
Data Set Line Numbered
Yes
No
E5
Turn On 'Line Numbered' Bit in Option Word
F4
Turn Off 'Line Numbered' Bit in Options Word
G4
Set Bit to Indicate Diagnose Incomplete Statement
H4
Set Bit to Indicate whether Source is Free Form
J4
Set Bit to Indicate whether Scan in Effect
K4
Return
A4

Chart EH. IKJEBESE

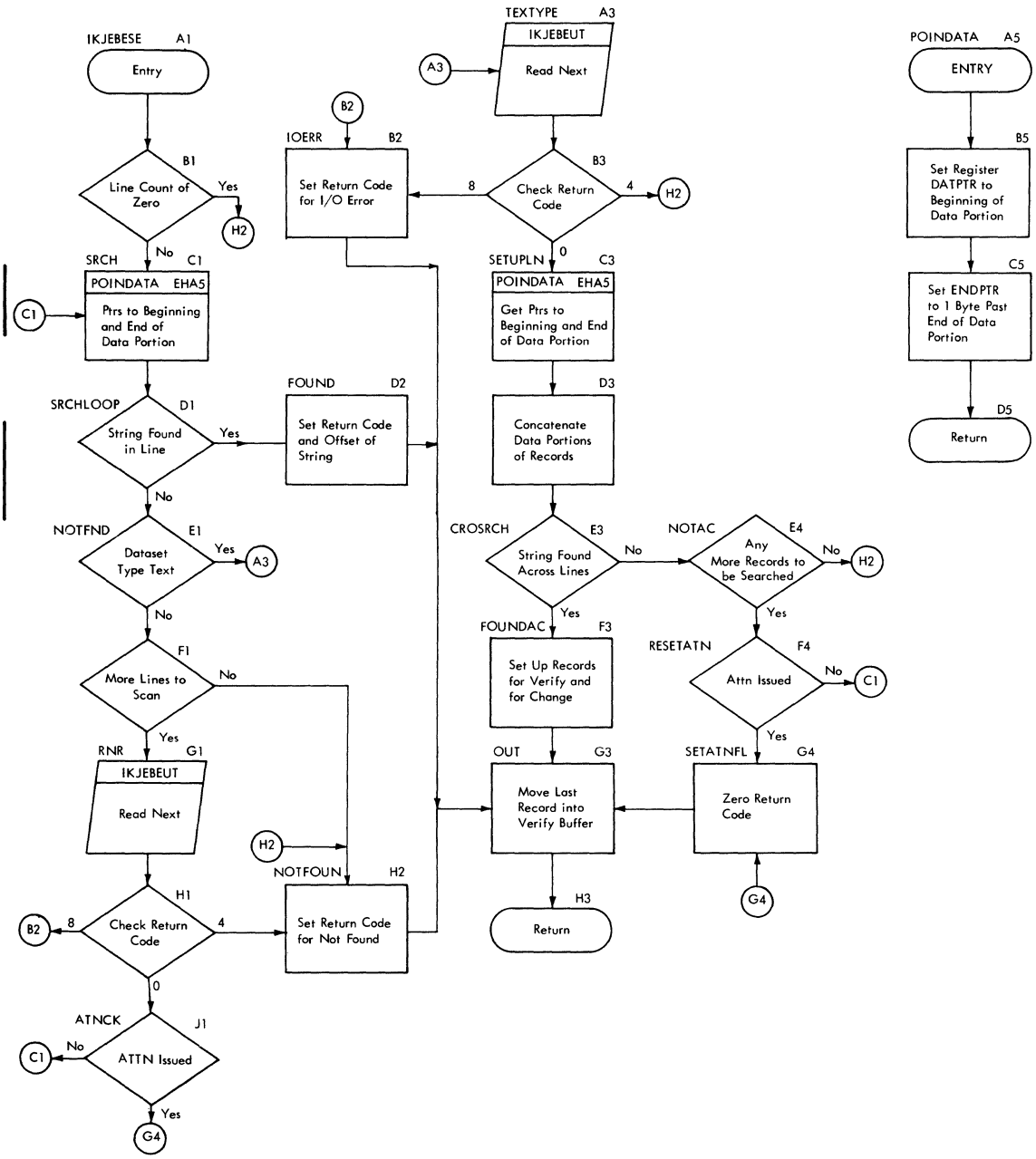


Chart EI. IKJEBESN

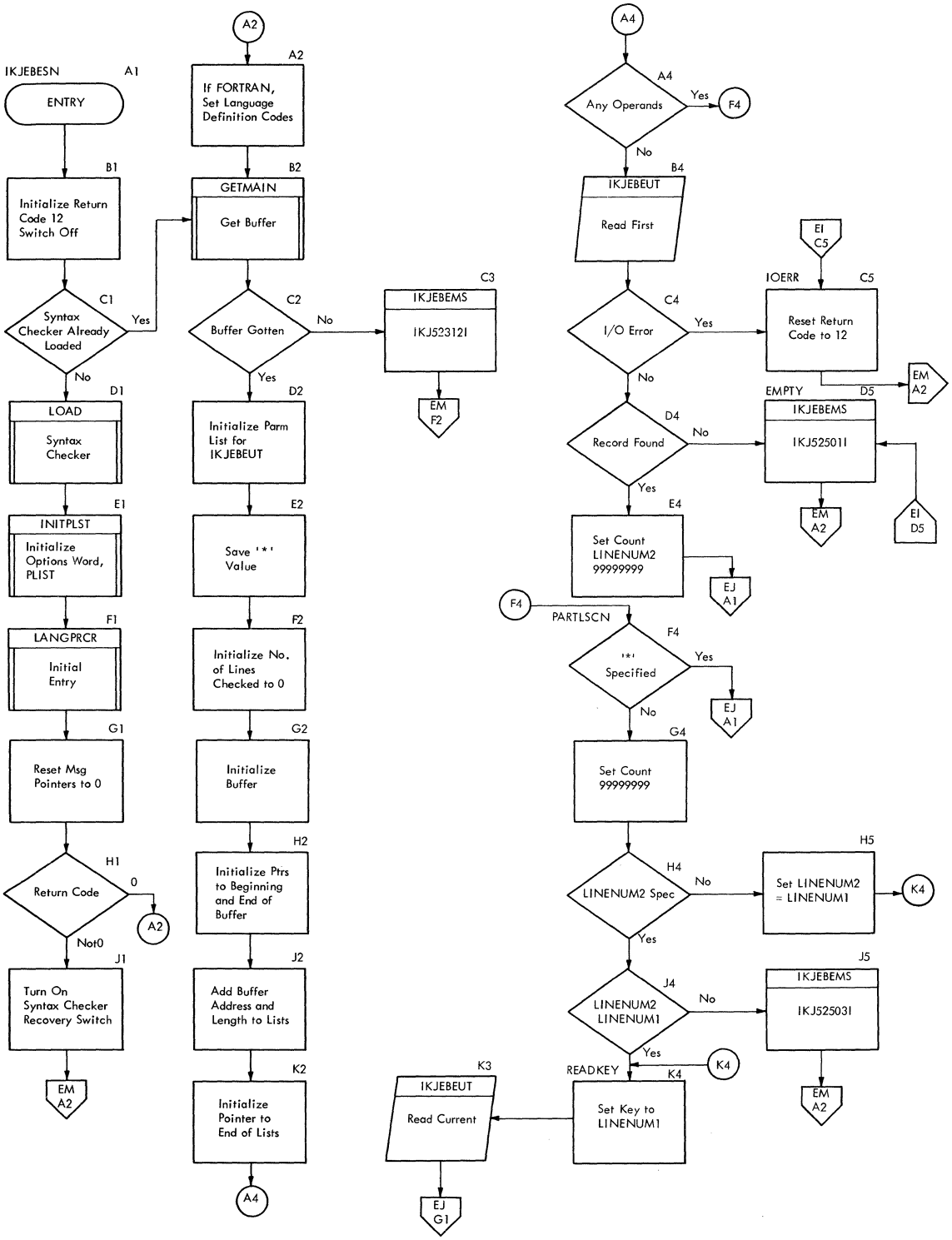


Chart EJ. IKJEBESN

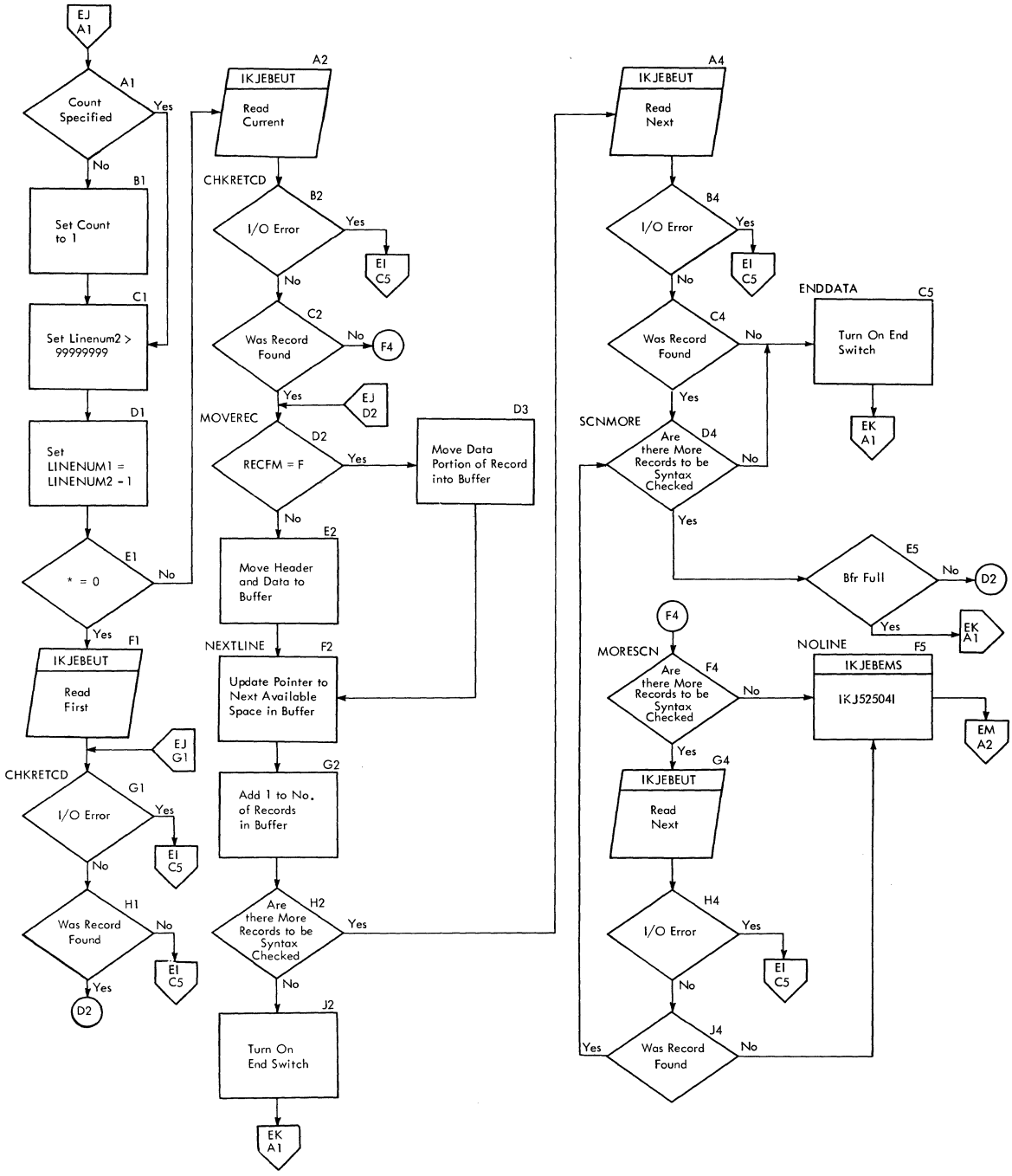


Chart EK. IKJEBESN

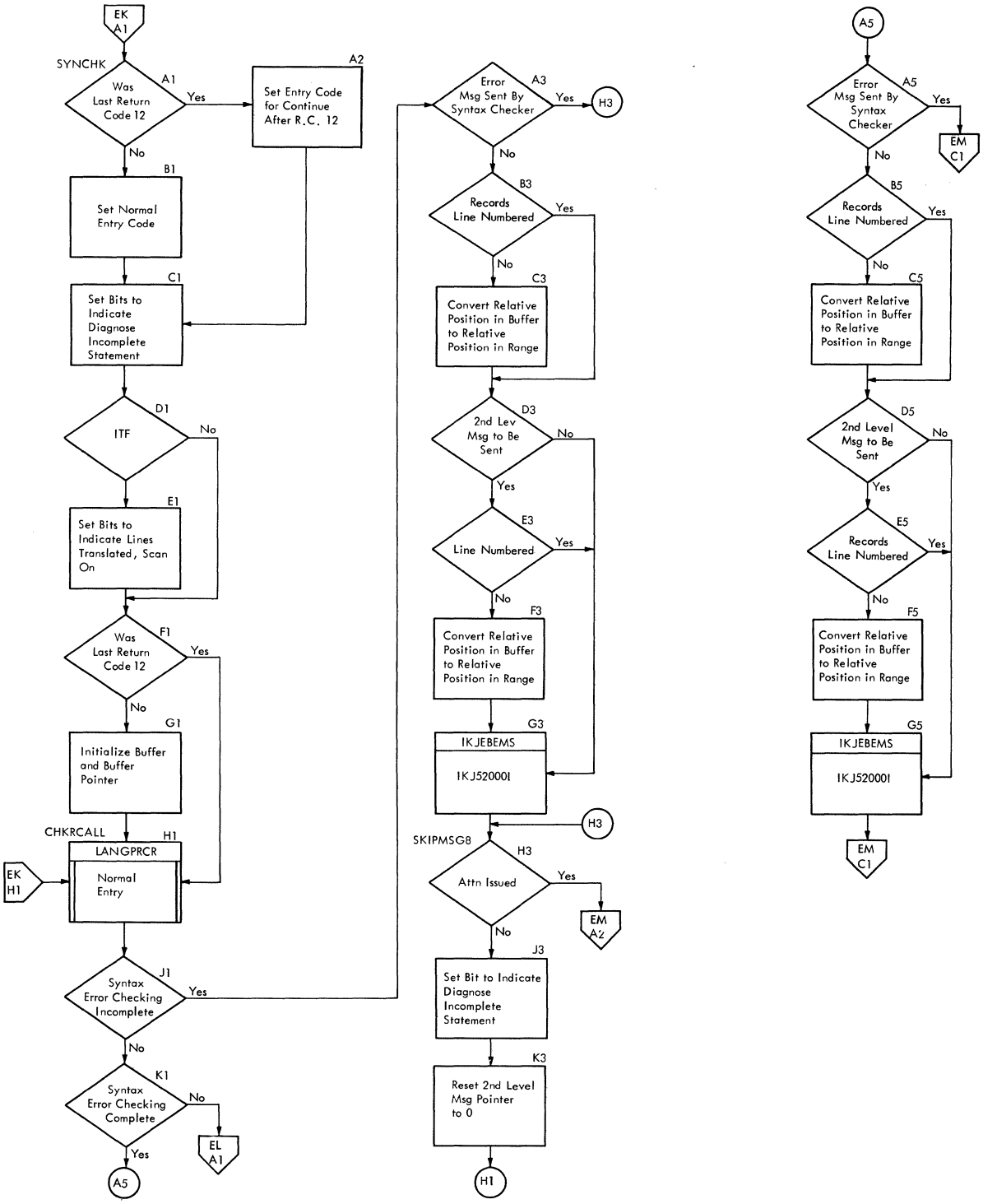


Chart EL. IKJEBESN

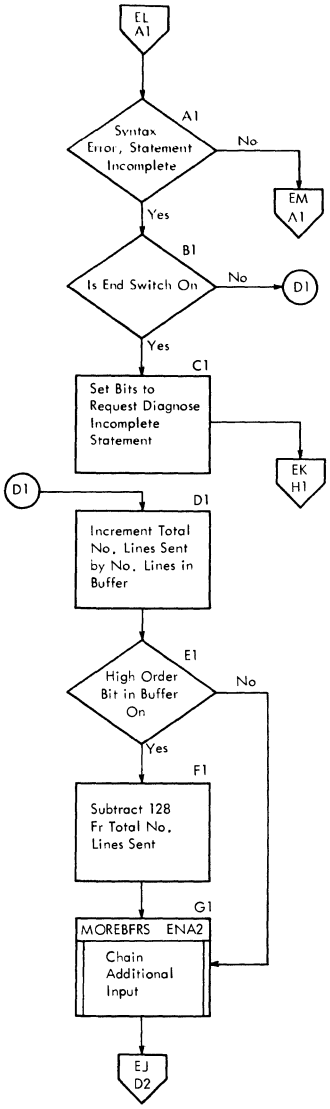


Chart EM. IKJEBESN

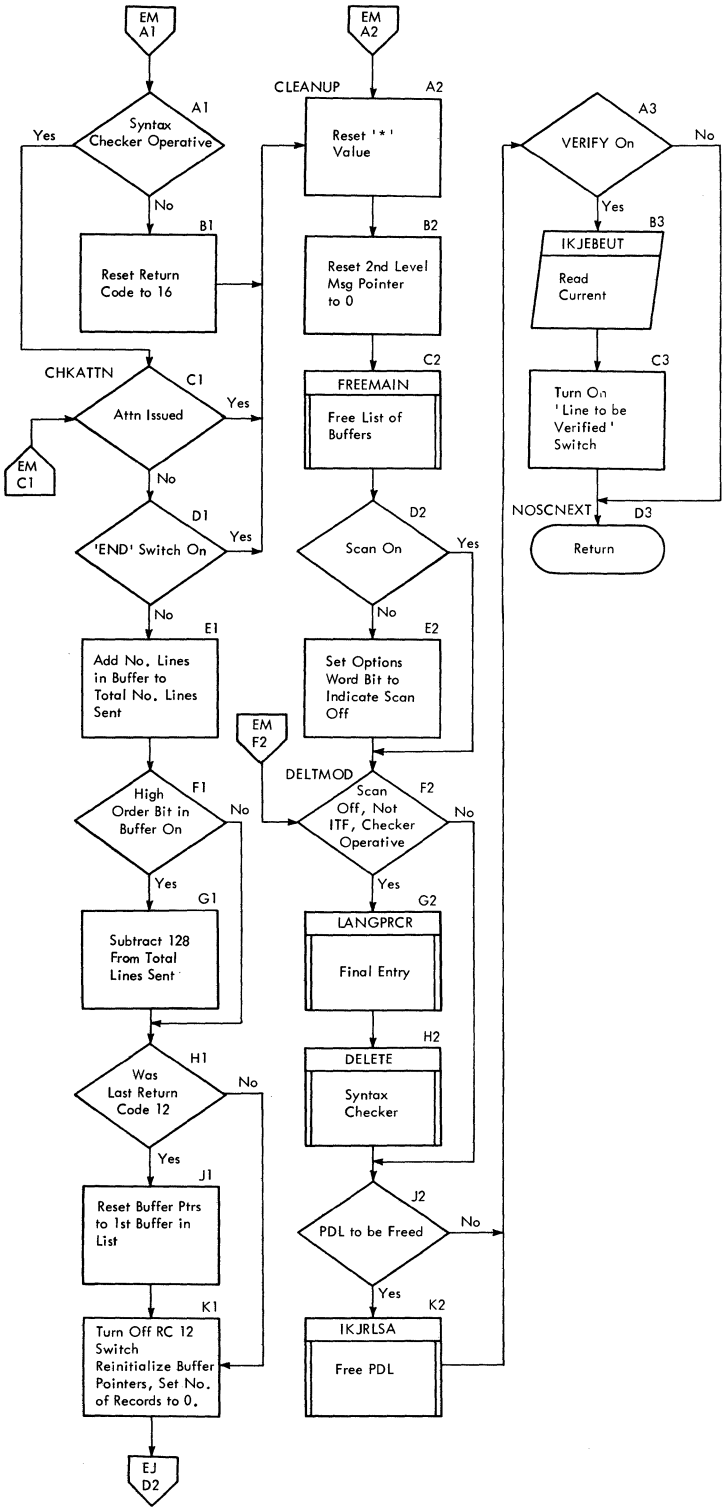
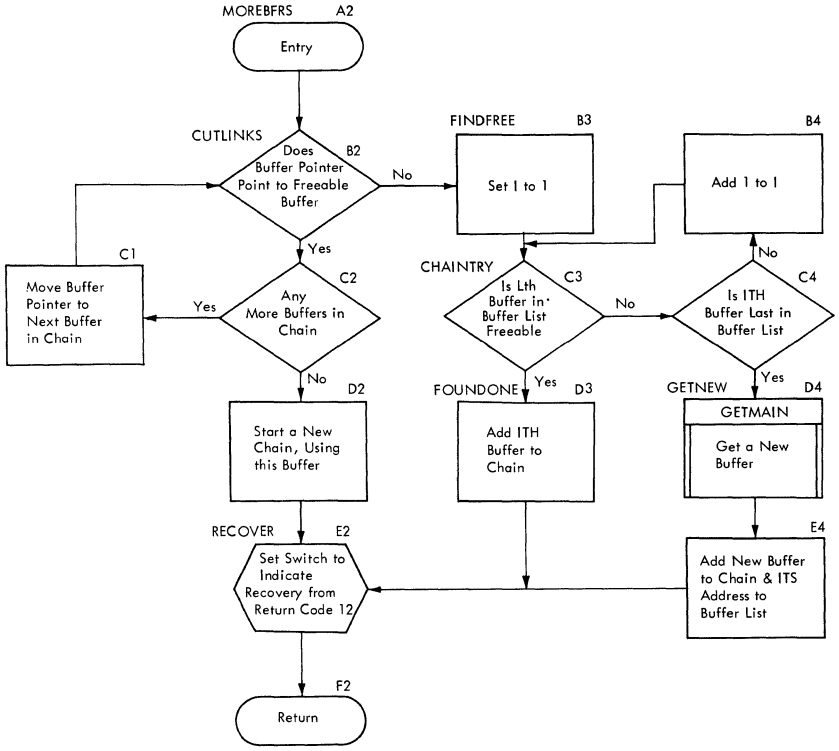


Chart EN. IKJEBESN



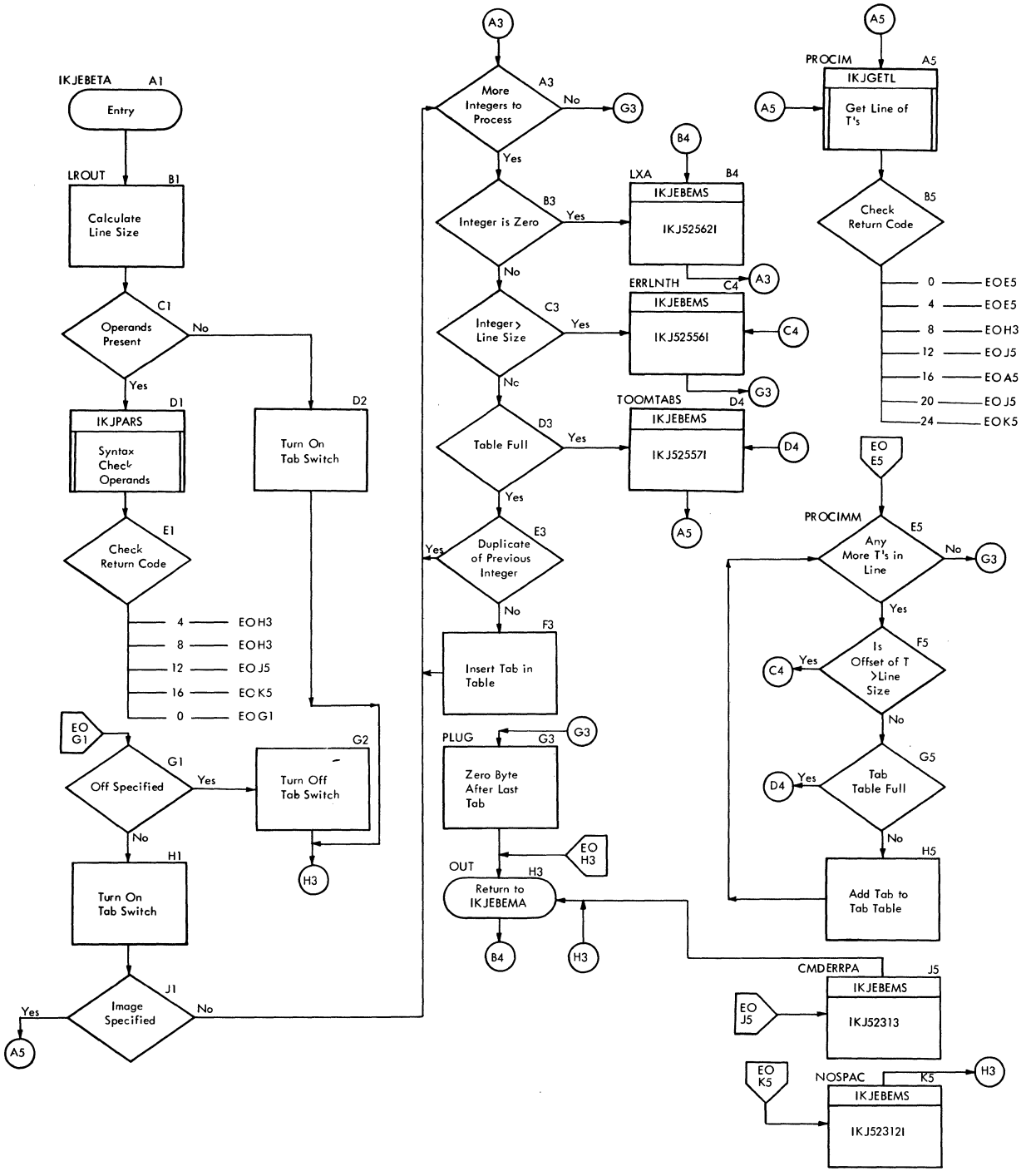

IKJEBETA A1
Entry
LROUT B1
Calculate Line Size
C1
Operands Present
No
Yes
D1
IKJPARS
Syntax Check Operands
D2
Turn On Tab Switch
E1
Check Return Code
4 EOH3
8 EOH3
12 EOJ5
16 EOK5
0 EOG1
EO G1
G1
Off Specified
Yes
G2
Turn Off Tab Switch
No
H1
Turn On Tab Switch
J1
Image Specified
Yes
No
A5
H3
A3
More Integers to Process
No
G3
Yes
B3
Integer is Zero
Yes
B4
LXA
IKJEBEMS
IKJ52562I
No
C4
A3
ERRLNTH
IKJEBEMS
IKJ52556I
C3
Integer > Line Size
Yes
C4
No
G3
D4
TOOMTABS
IKJEBEMS
IKJ52557I
D3
Table Full
Yes
D4
A5
Yes
E3
Duplicate of Previous Integer
Yes
No
F3
Insert Tab in Table
PLUG G3
G3
Zero Byte After Last Tab
EO H3
OUT H3
Return to IKJEBEMA
B4
H3
A5
PROCIM A5
IKJGETL
Get Line of T's
B5
Check Return Code
0 EOE5
4 EOE5
8 EOH3
12 EOJ5
16 EOA5
20 EOJ5
24 EOK5
EO E5
PROCIMM
E5
Any More T's in Line
No
G3
Yes
F5
Is Offset of T > Line Size
Yes
C4
No
G5
Tab Table Full
Yes
D4
No
H5
Add Tab to Tab Table
CMDERRPA J5
IKJEBEMS
IKJ52313
EO J5
EO K5
NOSPAC K5
H3
IKJEBEMS
IKJ52312I

Chart EP. IKJEBETO

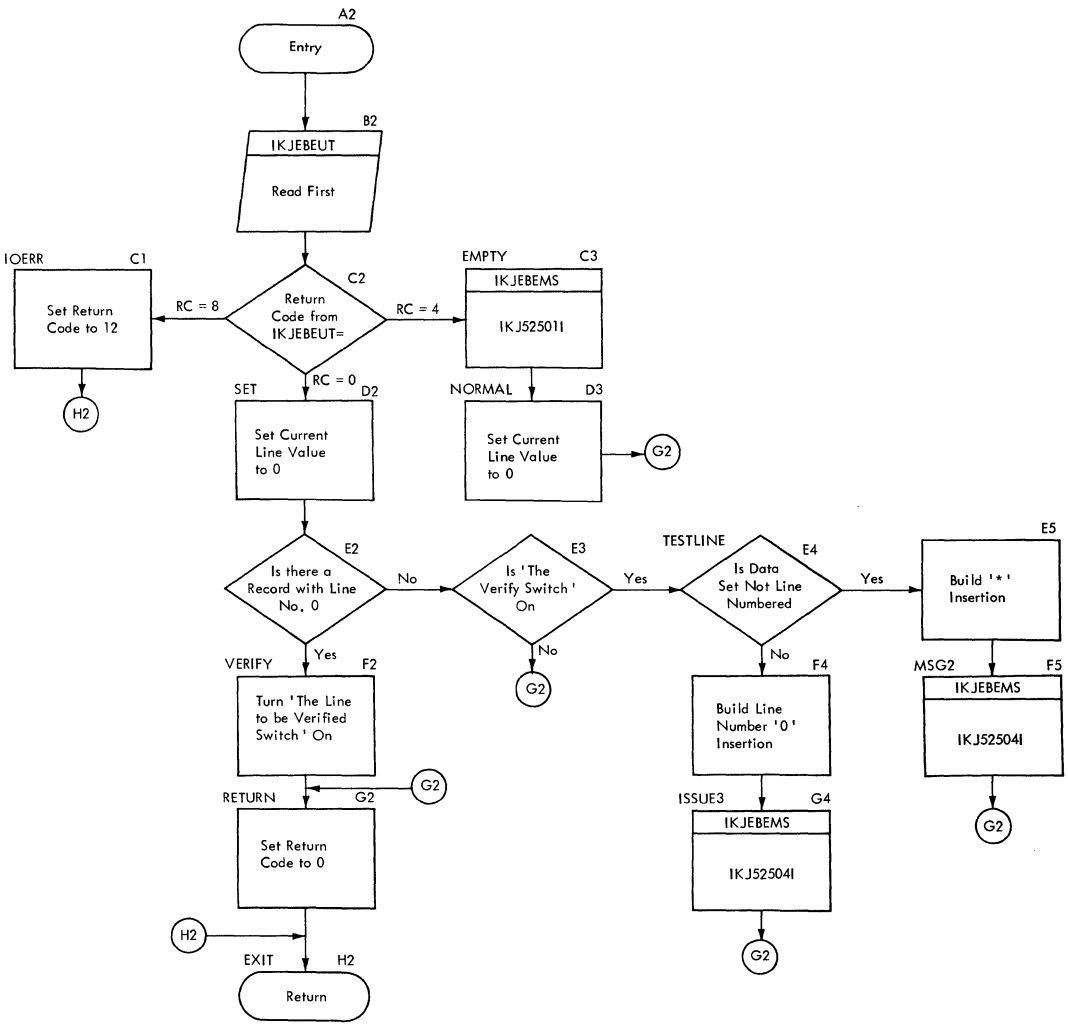


Chart EQ. IKJEBEUI

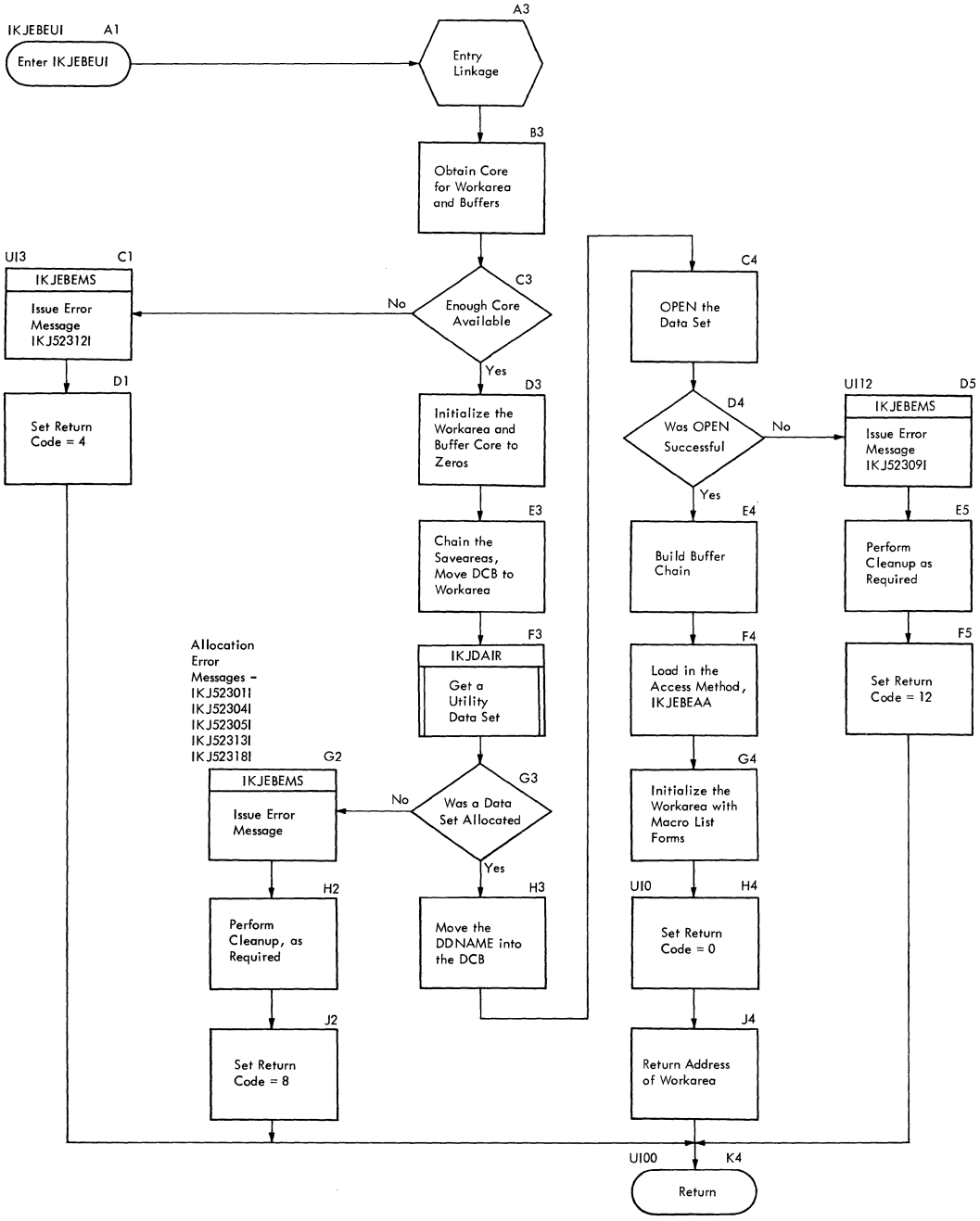


Chart ER. IKJEBEUP

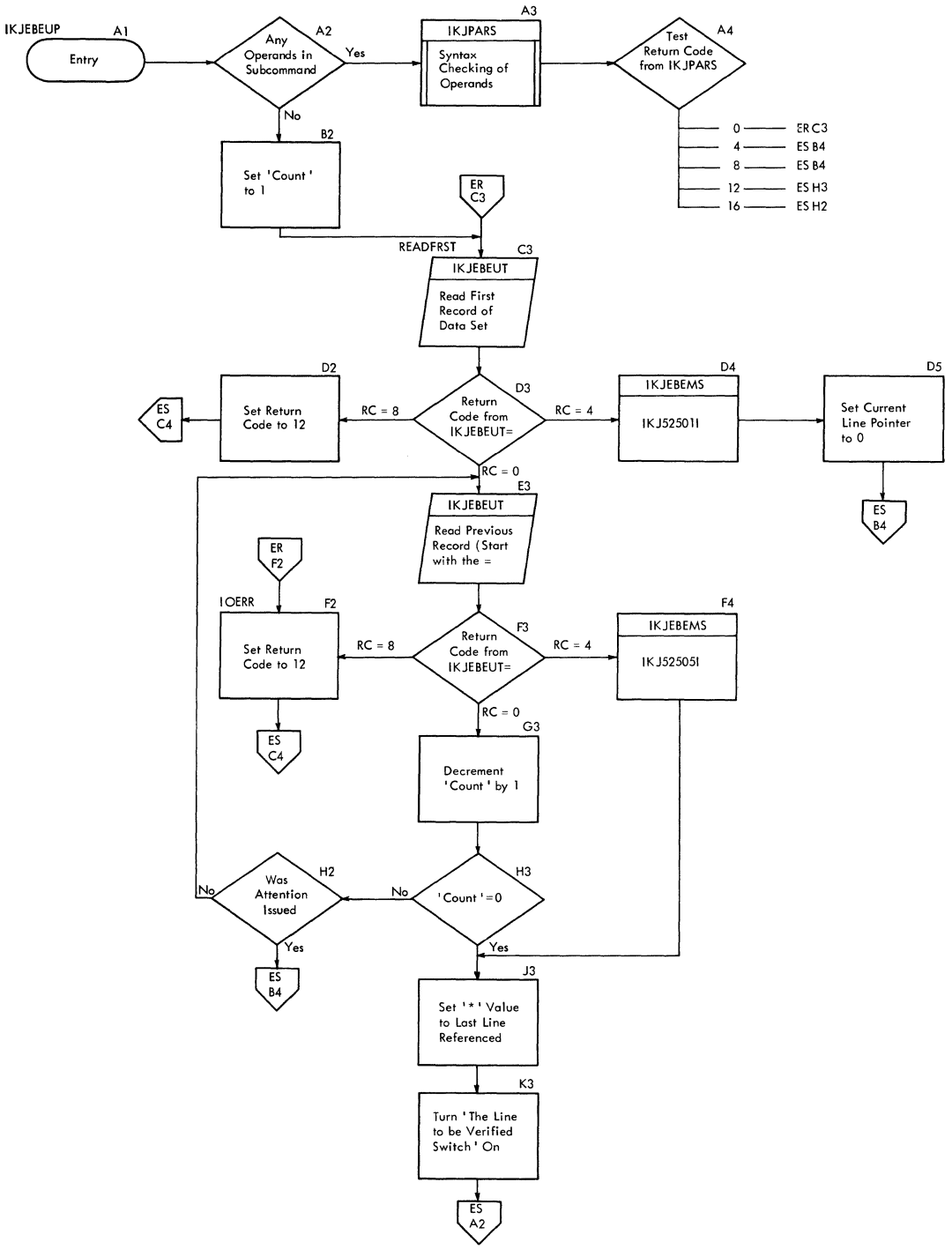


Chart ES. IKJEBEUP

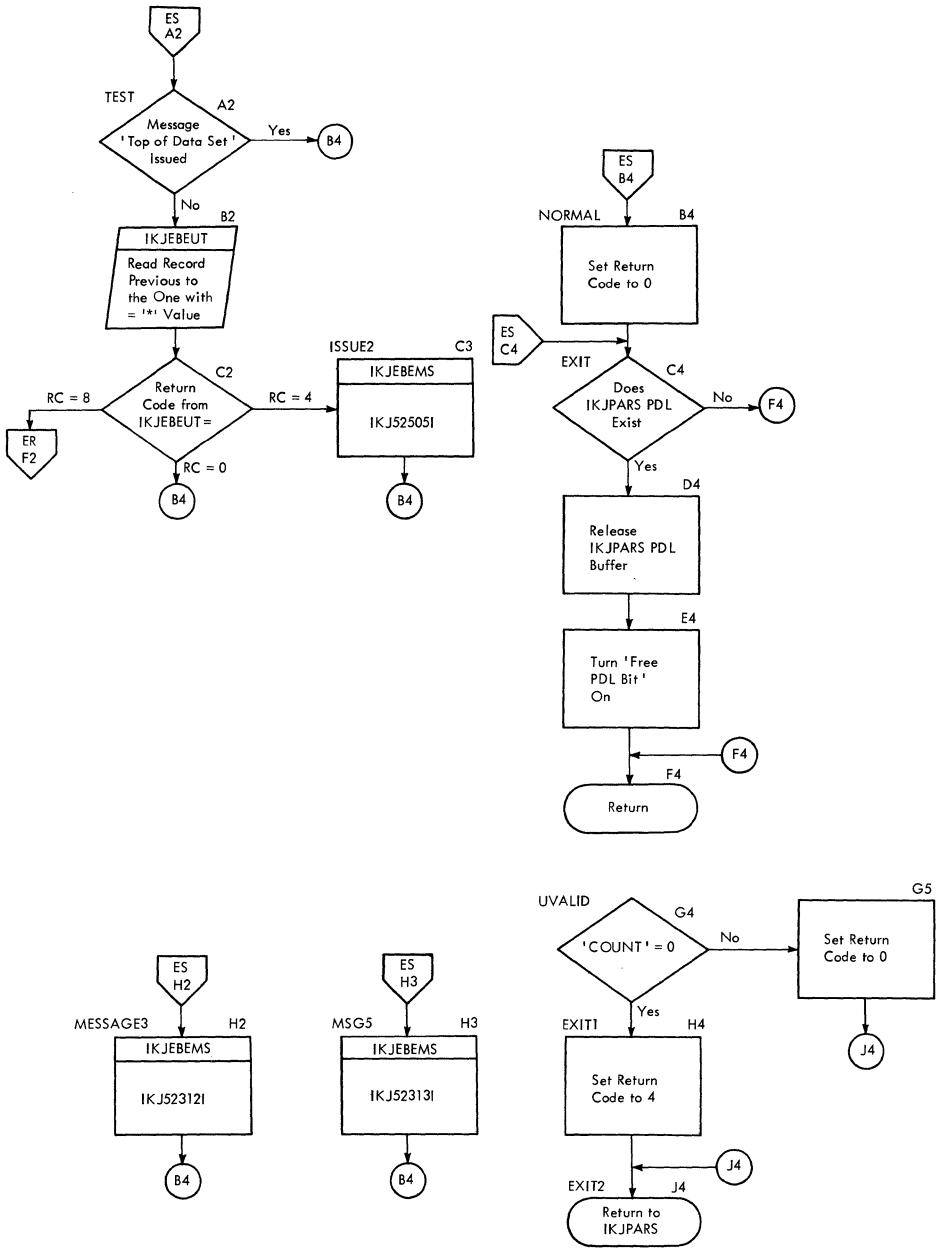


Chart ET. IKJEBEUT

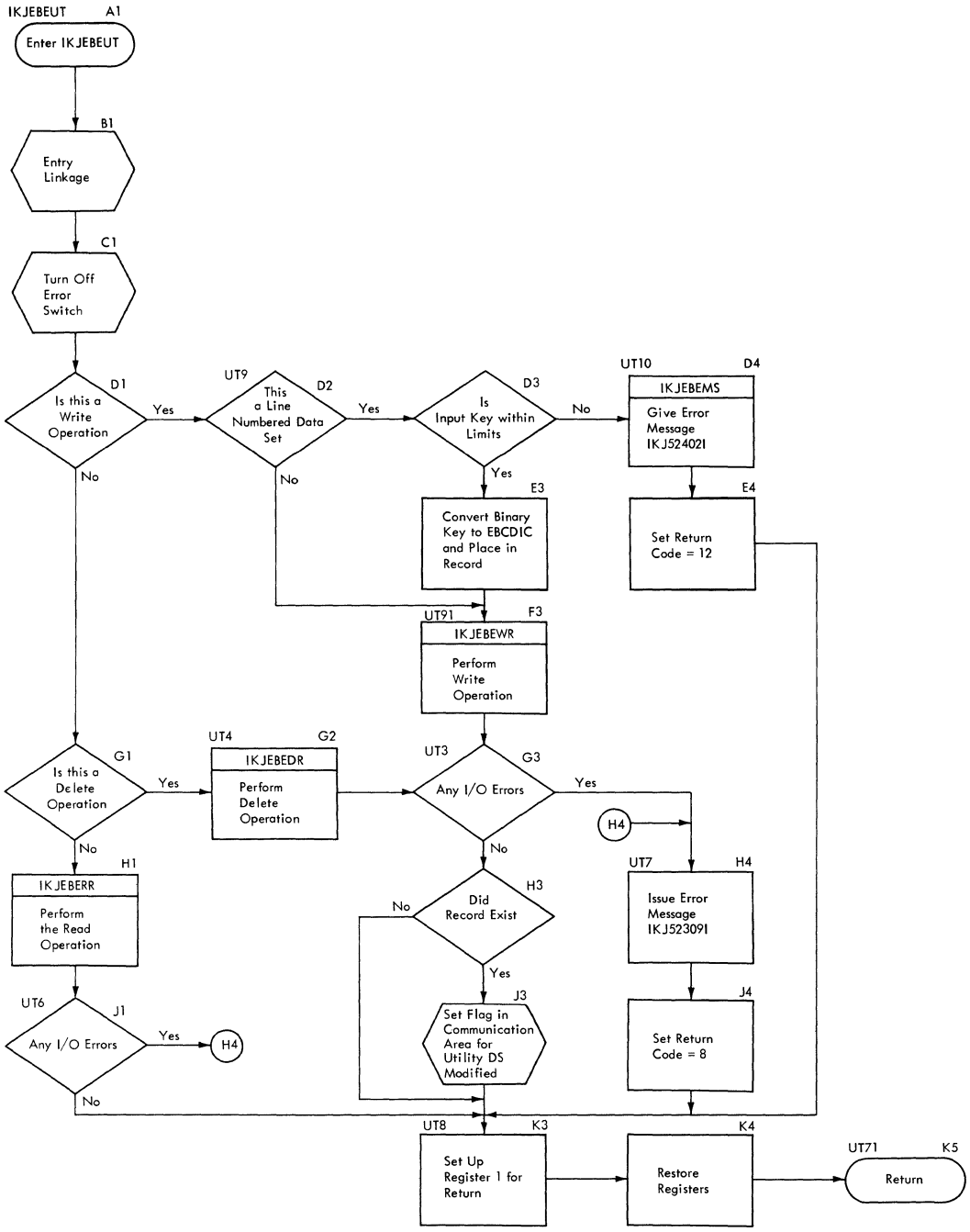


Chart EU. IKJEBEVE

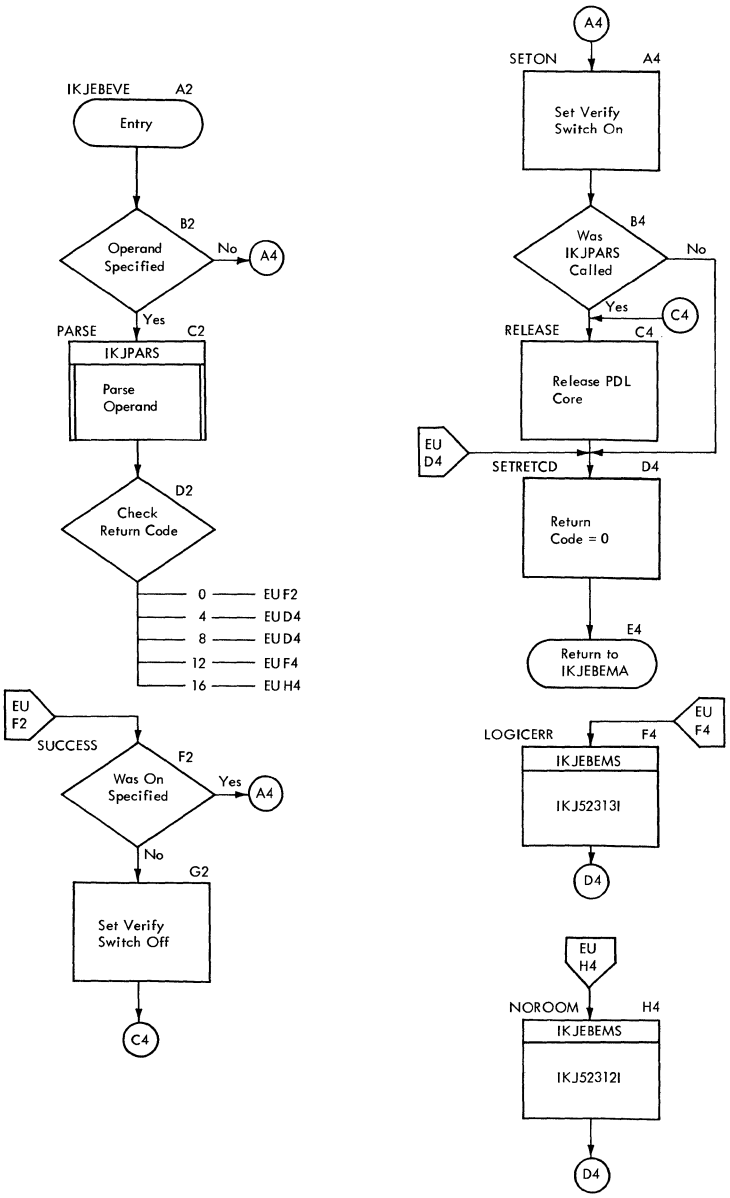


Chart EV. IKJEBEWA

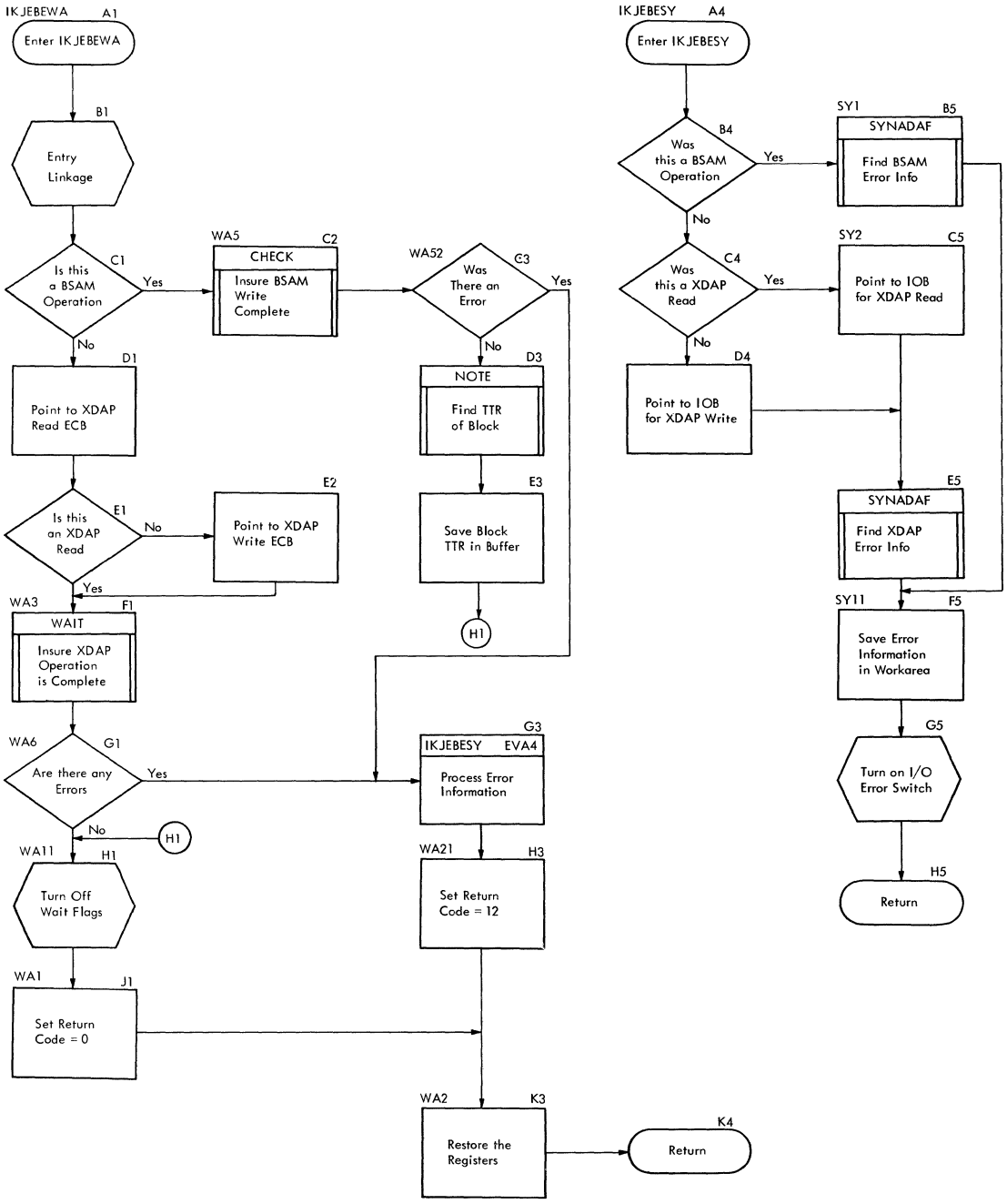


Chart EW. IKJEBEWB

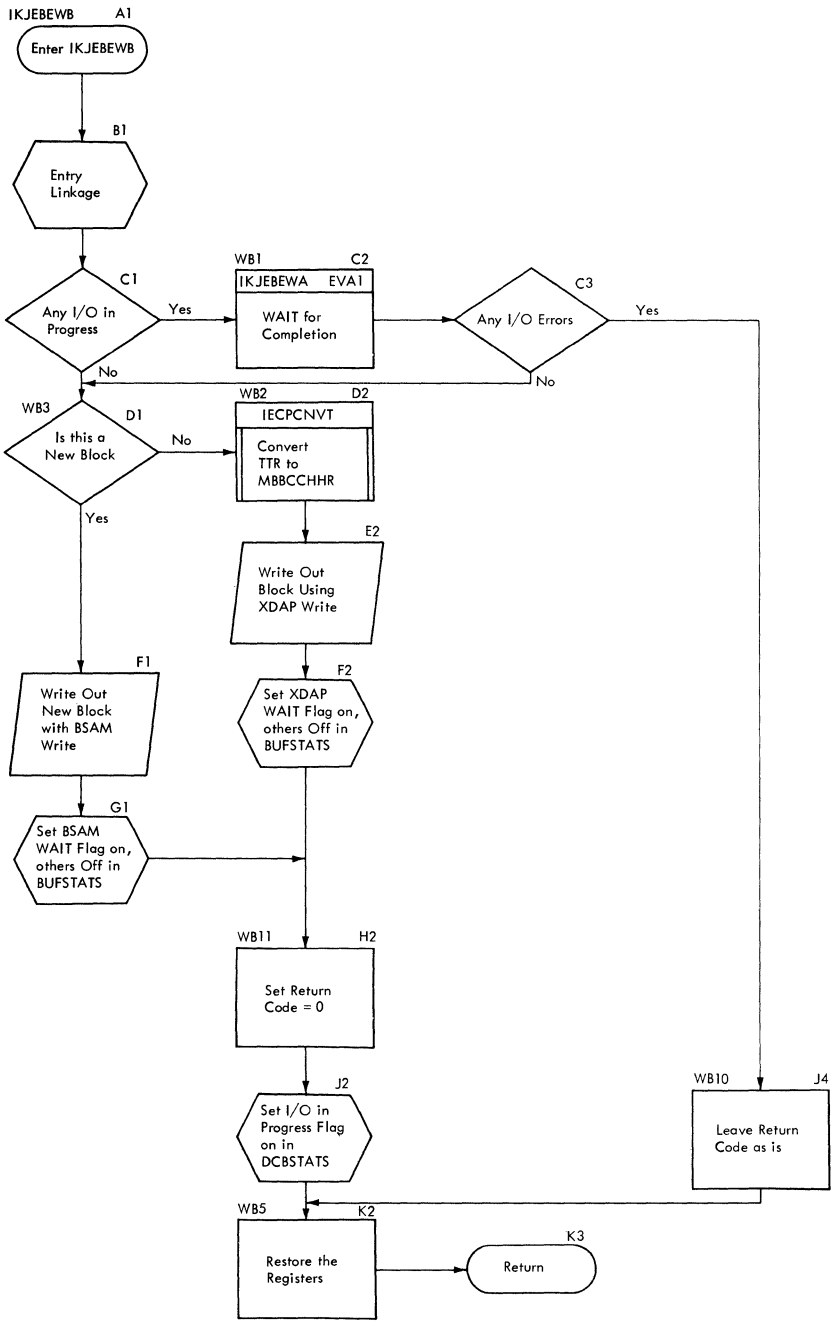


Chart EX. IKJEBEWR

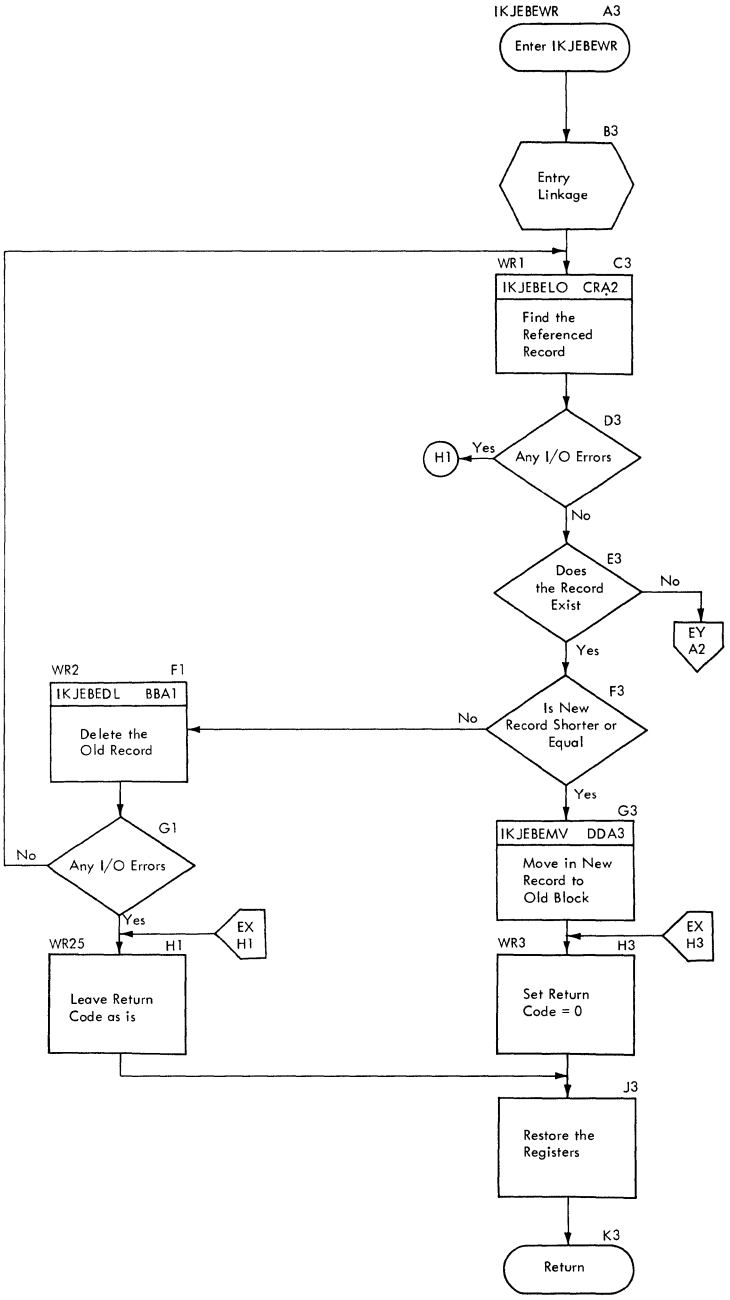


Chart EY. IKJEBEWR

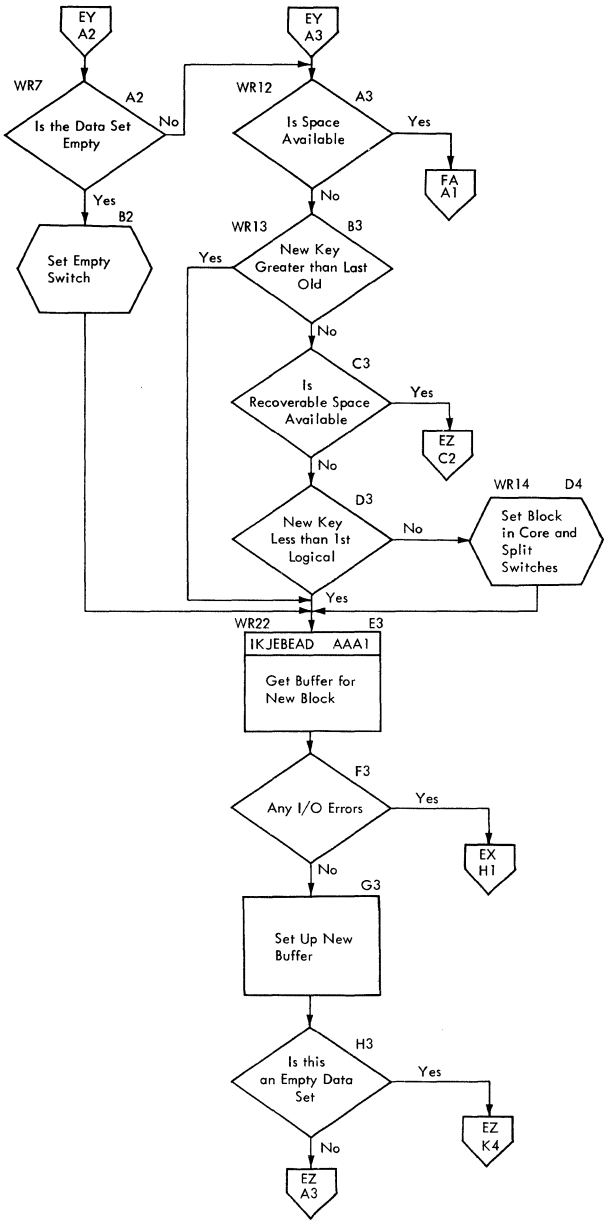


Chart EZ. IKJEBEWR

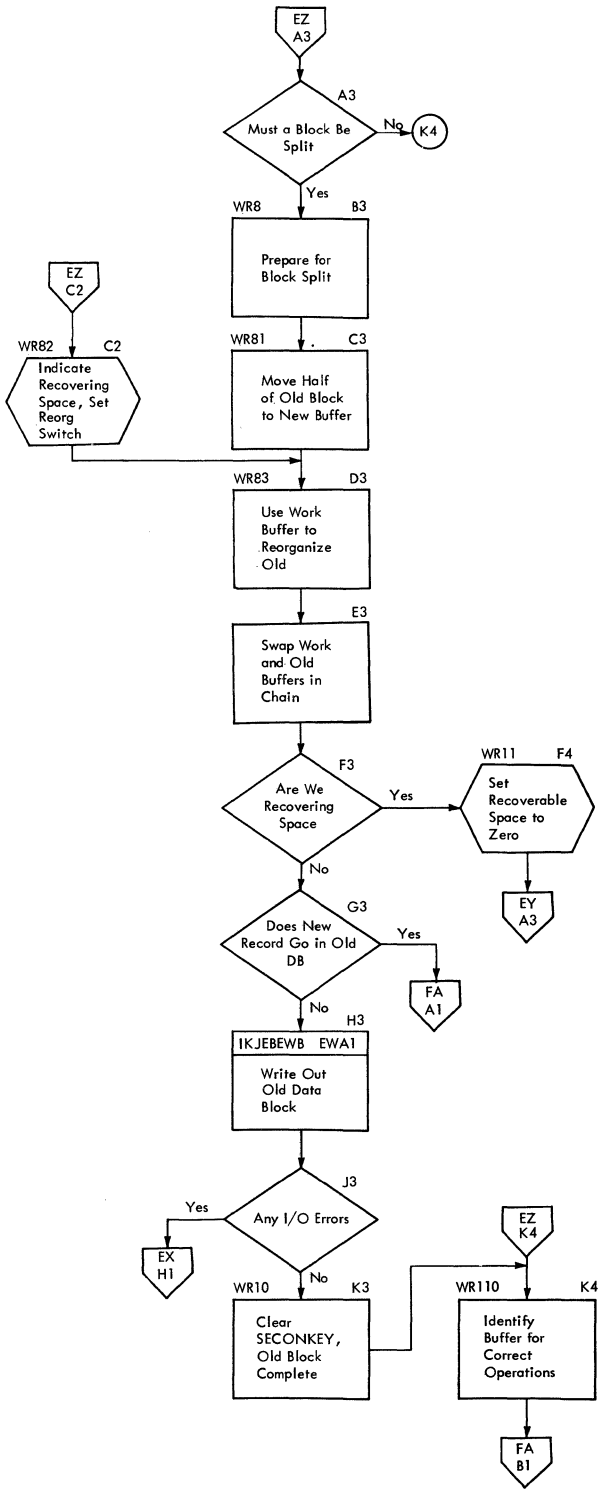
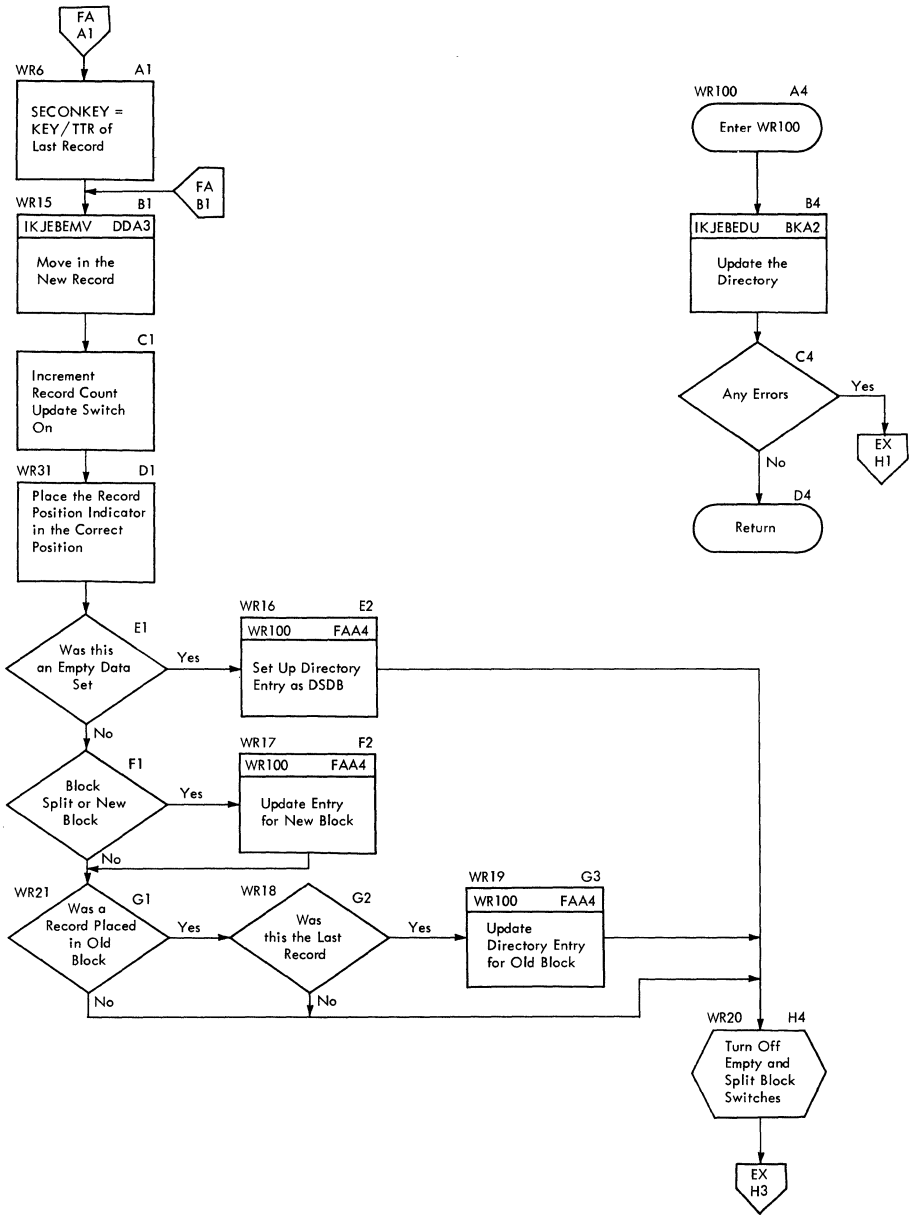


Chart FA. IKJEBEWR



Section 4: Directory

Load Module	Assembly Module	Control Section	Entry Point	MO Diagram	Flow Chart
IKJEBEAA	IKJEBEAA	IKJEBEAD	IKJEBEAD	33	AA
		IKJEBEAS	IKJEBEAS	36	AC
		IKJEBEDL	IKJEBEDL	32	BB-BD
		IKJEBEDR	IKJEBEDR	30	BG
		IKJEBEDS	IKJEBEDS	37	BH-BJ
		IKJEBEDU	IKJEBEDU	34	BK-BO
		IKJEBELO	IKJEBELO	31	CR,CS
		IKJEBEMV	IKJEBEMV	--	DD
		IKJEBERB	IKJEBERB	38	DI
		IKJEBERR	IKJEBERR	29	DO
		IKJEBEWA	IKJEBEWA	39	EV
		IKJEBEWB	IKJEBEWB	35	EW
		IKJEBEWR	IKJEBEWR	28	EX-FA
IKJEBEBO	IKJEBEBO	IKJEBEBO	IKJEBEBO	5	AE
IKJEBECG	IKJEBECG	IKJEBECG	IKJEBECG	6	AF-AK
	IKJEBESE	IKJEBESE	IKJEBESE	--	EH
IKJEBECH	IKJEBECH	IKJEBECH	IKJEBECH	5	AL-AN
		IKJEBCH1			
		IKJEBCH2			
		IKJEBCH3			
		IKJEBCH4			
IKJEBECI	IKJEBECI	IKJEBECI	IKJEBECI	--	AO,AP
			STAEEXIT	--	--
			STAIEXIT	--	--
			STAERTRY	--	--
		IKJEBCIM			
IKJEBECN	IKJEBECN	IKJEBECN	IKJEBECN	5	AQ-AT
IKJEBECO	IKJEBECO	IKJEBECO	IKJEBECO	--	AU-AW
IKJEBEDA	IKJEBEDA	IKJEBEDA	IKJEBEDA	--	AX
		IKJEBDAM			
IKJEBEDE	IKJEBEDE	IKJEBEDE	IKJEBEDE	7	AY-BA
		IKJEBDE1			
IKJEBEDO	IKJEBEDO	IKJEBEDO	IKJEBEDO	8	BE,BF
		IKJEBDO0	IKJEBDO0	--	--
IKJEBEEN	IKJEBEEN	IKJEBEEN	IKJEBEEN	--	BP
			IKJEBEXT	--	--
		IKJEBEN0			
IKJEBEEX	IKJEBEEX	IKJEBEEX	IKJEBEEX	26	BQ
IKJEBEFC	IKJEBEFC	IKJEBEFC	IKJEBEFC	--	BR,BS
IKJEBEFI	IKJEBEFI	IKJEBEFI	IKJEBEFI	9	BT
		IKJEBFI0			
		IKJEBFI1			
	IKJEBESE	IKJEBESE	IKJEBESE	--	EH
IKJEBEFO	IKJEBEFO	IKJEBEFO	IKJEBEFO	10	BU,BV
		IKJEBF00			
IKJEBEHE	IKJEBEHE	IKJEBEHE	IKJEBEHE	11	BW
IKJEBEIM	IKJEBEIM	IKJEBEIM	IKJEBEIM	12	BX-CB

Load Module	Assembly Module	Control Section	Entry Point	MO Diagram	Flow Chart
IKJEBEIN*1	IKJEBEIN	IKJEBEIN	IKJEBEIN	1	CC-CJ
		IKJEBIN1	IKJEBIN1	--	--
		IKJEBIN2	IKJEBIN2	--	--
		IKJEBIN3	IKJEBIN3	--	--
		IKJEBIN4	IKJEBIN4	--	--
		IKJEBIN5	IKJEBIN5	--	--
		IKJEBIN6	IKJEBIN6	--	--
		IKJEBIN7	IKJEBIN7	--	--
		IKJEBIN8	IKJEBIN8	--	--
IKJEBEIP	IKJEBEIP	IKJEBEIP	IKJEBEIP	--	CK,CL
		IKJEBXT1	IKJEBXT1	--	--
		IKJEBIP1			
IKJEBEIS	IKJEBEIS	IKJEBEIS	IKJEBEIS	13	CM,CN
		IKJEBIS1			
IKJEBELE	IKJEBELE	IKJEBELE	IKJEBELE	--	CO
IKJEBELI	IKJEBELI	IKJEBELI	IKJEBELI	14	CP,CQ
		IKJEBLI1			
IKJEBELT	IKJEBELT	IKJEBELT	IKJEBELT	15	CT-CV
		IKJEBLT1			
IKJEBEMA	IKJEBEMA	IKJEBEMA	IKJEBEMA	2	CW-CY
			MAAERTRY	--	--
		IKJEBMA1	IKJEBMA1	--	--
		IKJEBMA2	IKJEBMA2	--	--
		IKJEBMA8			
	IKJEBMA9*2	IKJEBMA9			
	IKJEBEAE	IKJEBEAE	IKJEBEAE	--	AB
	IKJEBEAT	IKJEBEAT	IKJEBEAT	4	AD
	IKJEBEUT	IKJEBEUT	IKJEBEUT	27	ET
IKJEBEME	IKJEBEME	IKJEBEME	IKJEBEME	16	CZ,DA
		IKJEBMEN			
		IKJEBME0			
IKJEBEMR	IKJEBEMR	IKJEBEMR	IKJEBEMR	--	DB
		IKJEBMRM			
IKJEBEMS	IKJEBEMS	IKJEBEMS	IKJEBEMS	--	DC
IKJEBEM1	IKJEBEM1	IKJEBEM1			
IKJEBEM2	IKJEBEM2	IKJEBEM2			
IKJEBEM3	IKJEBEM3	IKJEBEM3			
IKJEBEM4	IKJEBEM4	IKJEBEM4			
IKJEBEM5	IKJEBEM5	IKJEBEM5			
IKJEBEM6	IKJEBEM6	IKJEBEM6			
IKJEBEM7	IKJEBEM7	IKJEBEM7			
IKJEBEPS	IKJEBEPS	IKJEBEPS	IKJEBEPS	--	DH
		IKJEBEPD	IKJEBEPD	--	--
IKJEBERE	IKJEBERE	IKJEBERE	IKJEBERE	18	DJ-DM
		IKJEBRE4	IKJEBRE4	--	--
IKJEBERN	IKJEBERN	IKJEBERN	IKJEBERN	--	DN

*1 EDIT and E are aliases for the load module IKJEBEIN.

*2 The user has the option to assemble the EDIT program with IKJEBMA9 as an assembly module.

Load Module	Assembly Module	Control Section	Entry Point	MO Diagram	Flow Chart
IKJEBERU	IKJEBERU	IKJEBERU	IKJEBERU	19	DP,DQ
		IKJEBRU0			
IKJEBESA	IKJEBESA	IKJEBESA	IKJEBESA	20	DR-EB
		IKJEBSA1	IKJEBSA1	--	--
		IKJEBSA2	IKJEBSA2	--	--
		IKJEBSA8	IKJEBSA8	--	--
		IKJEBSA9	IKJEBSA9	--	--
IKJEBESC	IKJEBESC	IKJEBESC	IKJEBESC	21	EC-EG
		IKJEBSCI			
IKJEBESN	IKJEBESN	IKJEBESN	IKJEBESN	21	EI-EN
IKJEBETA	IKJEBETA	IKJEBETA	IKJEBETA	22	EO
		IKJEBTAO			
IKJEBETO	IKJEBETO	IKJEBETO	IKJEBETO	23	EP
IKJEBEUI	IKJEBEUI	IKJEBEUI	IKJEBEUI	26	EQ
IKJEBEUP	IKJEBEUP	IKJEBEUP	IKJEBEUP	24	ER,ES
		IKJEBUP0			
IKJEBEUT	IKJEBEUT	IKJEBEUT	IKJEBEUT	27	ET
IKJEBEVE	IKJEBEVE	IKJEBEVE	IKJEBEVE	25	EU
		IKJEBVEP			

Section 5: Data Areas

This section describes the formats of:

- The data areas and control blocks built and used by the EDIT program.
- The parameters lists passed to the EDIT program.
- The parameters lists that the EDIT program passes to the various TSO service routines and to the syntax checkers.

EDIT Communication Area (IKJEBECA)

The EDIT Communication Area is built by the EDIT initialization routine (IKJEBEIN). It is used by all of the modules of the EDIT program except for the Access Method service routines (IKJEBEAA). Fields marked with superscripts are described in more detail following the description of the EDIT Communication Area.

Disp dec	Disp hex	Field Name	Size Bytes	Contents
0	0	CAPTTMP	4	Addr. of TMP Parameter List
4	4		4	Reserved storage
8	8	CAPTAE	4	Addr. of IKJEBEAE
12	C	CAPTAT	4	Addr. of IKJEBEAT
16	10	CAPTLE	4	Addr. of IKJEBELE
20	14	CAPTMS	4	Addr. of IKJEBEMS
24	18	CAPTUT	4	Addr. of IKJEBEUT
28	1C	CAPTMSGM	4	Addr. of IN-CORE message module
32	20	CAPTRTRY	4	Addr. of STAE re-entry routine
36	24	CAPRSPDL*[1]	0	IKJPARS PDL flag byte
36	24	CAPTPRSD	4	Addr. of IKJPARS PDL
40	28	CASCBFFL*[2]	0	Subcommand buffer flags
40	28	CAPTIBFR	4	Addr. of subcommand input buffer
44	2C	CAPTSCMD	4	Addr. of subcommand last entered
48	30	CASCMDLN	2	Length of subcommand name
50	32		2	Reserved storage
52	34	CAPTCDCB	4	Addr. of current utility work area (UTILWORK)

(Part 1 of 5)

Disp dec	Disp hex	Field Name	Size Bytes	Contents
56	38	CAPTPDCB	4	Addr. of new utility work area (UTILWORK)
60	3C	CAUTILNO	4	Number of records in utility data set
64	40	CAPTCORE	4	Addr. of GETMAIN area
68	44	CACORELN	4	Length of GETMAIN area
72	48	CAPTCHK	4	Addr. of syntax checker or language processor
76	4C		44	Reserved storage
120	78	CAATTN*3	4	Attention ECB
124	7C	CACFLAG	0	Control flags
124	7C	CACFLAG1*4	1	Control flag #1
125	7D	CACFLAG2*5	1	Control flag #2
126	7E	CACFLAG3*6	1	Control flag #3
127	7F	CACFLAG4*7	1	Control flag #4
128	80	CACFLAG5	1	Control flag #5 Bits 0-7 reserved
129	81	CACFLAG6*8	1	Control flag #6
130	82	CAPLILFM	1	PLI Left source margin
131	83	CAPLIRTM	1	PLI right source margin
132	84		20	Reserved storage
152	98	CADSTYPE	8	Data set type keyword
160	A0	CADSQUAL	8	Data set name qualifier
168	A8	CABLKS	2	Default block size
170	AA	CALINE	1	Line number offset
171	AB	CALENGTH	1	Line number length
172	AC	CATABS	12	Tabsetting values
184	B8	CASYNAME	8	Syntax checker name
192	C0	CADSCODE*9	1	Data set type code
193	C1	CADSATTR*10	1	Data set attributes, Byte 1
194	C2	CADSATR2*11	1	Data set attributes, Byte 2
195	C3	CARECFMD	1	Record format default
196	C4	CAFLRLDF	2	F format lrecl default

(Part 2 of 5)

Disp dec	Disp hex	Field Name	Size Bytes	Contents
198	C6	CAFLRLMX	2	F format lrecl maximum
200	C8	CAVLRLDF	2	V format lrecl default
202	CA	CAVLRLMX	2	V format lrecl maximum
204	CC	CAULRLDF	2	U format lrecl default
206	CE	CAULRLMX	2	U format lrecl maximum
208	D0	CACHKOPT	2	Syntax checker option word, Bytes 0 and 1
210	D2	CAPRNAME	8	Prompter name
218	DA	CAEXTNAM	8	User exit name
226	E2		6	Reserved storage
232	E8	CALRECL	2	Data length plus control word
234	EA		2	Reserved storage
236	EC	CAEDFLAG*[13]	1	Control flag for EDIT data set
237	ED		1	Reserved storage
238	EE	CAEDDSNL	2	Length of DSNAME
240	F0	CAEDDSN	44	Dsname of EDIT data set
284	11C	CAEDMEMB	8	Member name for EDIT data set
292	124	CAEDDDN	8	DDname for EDIT data set
300	12C	CAEDPSWD	8	Password for EDIT data set
308	134	CAEDTSIZ	4	Size of old EDIT data set
312	138	CADSNPTR	4	Pointer to next insertion record
316	13C	CADSNLEN	2	Length of next insertion record
318	13E	CADSNOFF	2	Offset in message to next insertion record
320	140	CADSNREC	56	EDIT data set name insertion
376	178	CASAFLAG*[14]	1	Control flag for SAVE data set
377	179		1	Reserved storage
378	17A	CASADSNL	2	Length of SAVE dsname
380	17C	CASADSN	44	Dsname of SAVE data set

(Part 3 of 5)

Disp dec	Disp hex	Field Name	Size Bytes	Contents
424	1A8	CASAMEMB	8	Member name for SAVE data set
432	1B0	CASADDN	8	DDname for SAVE data set
440	1B8	CASAPSWD	8	Password for SAVE data set
448	1C0	CASTNUM	4	Starting line number
452	1C4	CANXTREC	4	Next record key for input mode
456	1C8	CACURNUM	4	Current line number, '*'
460	1CC	CAINCRE	4	Line number increment
464	1D0	CAIMLLNO	4	Last line number used in input mode
468	1D4	CAIMLINC	4	Last increment used in input mode
472	1D8		4	Reserved storage
476	1DC	CAINSAVE	4	Last line number in Input mode when INSERT was used.
480	1E0		16	Reserved storage
496	1F0	CASYNLST	0	Syntax checker parameter list
496	1F0	CASYNBFR	4	Addr. of first buffer in chain
500	1F4	CASYNPWA	4	Addr. of work area
504	1F8	CASYNPTO	4	Addr. of option word
508	1FC	CASYNECD	1	Syntax checker entry code
509	1FD	CASYNWAP	3	Addr. of checker work area
512	200	CASYNMS1	4	Addr. of first error message
516	204	CASYNMS2	4	Addr. of record and chained messages
520	208	CASYNTEM	4	Temporary storage for checker
524	20C	CASYNCD1	1	Option word code #1
525	20D	CASYNCD2	1	Option word code #2
526	20E	CASYNRCL	1	Record length for fixed record length only. For variable record length value of this field is zero
527	20F	CASYNSW*15	1	Bit switches

(Part 4 of 5)

Disp dec	Disp hex	Field Name	Size Bytes	Contents
528	210	CAPTUPT	4	Add. of UPT
532	214	CAPTECT	4	Addr. of ECT
536	218	CAPTECB	4	Addr. of EDIT ATTENTION ECB
540	21C	CASRPLST	16	TMP Service Routine parameter list
556	22C	CASTAXPL	20	STAX parameter list
576	240	CASTAEPL	20	STAE parameter list
596	254	CAMAWKA	32	Main controller work area
628	274	CAMSWKA	100	Message selection work area
728	2D8	CASRWKA	200	Service routine work area
928	3A0	CAMODEMG	0	Insertion record for command name
928	3A0	CAMODEIS	4	Number of insertions
932	3A4	CAMODEPT	4	Address of insertion text
936	3A8	CAMODELN	2	Length of insertion record
938	3AA	CAMODEOF	2	Offset in msg for insertion
940	3AC	CAMODETX	12	Insertion text
952	CB8	CAATNWKA	112	Attention exit work area
1064	428		124	Reserved
1188	4A4	CAFIBFR	260	FIND buffer
1448	5A8	CASCWKA	336	Subcommand work area
1784	6F8	CABFRPL	528	Buffer pool
2312	908	CATEMPBF	528	Temporary buffer pool available to all EDIT subcommands and service routines
2840	B18	CASVAREA	720	Chained save areas
3560	DE8	CANXTSVA	4	Addr. of next save area to be used
3564	DEC		20	Reserved
3584	E00	CADSNPT2	4	Pointer to next insertion record
3588	E04	CADSNLN2	2	Length of this next insertion record, including header
3590	E06	CADSNOF2	2	Offset, in message, to insertion
3592	E08	CADSNRC2	56	SAVE data set name message insertion

CAPRSPDL*[1]

The following table indicates the values against which CAPRSPDL is compared.

CAPRSPDL is the IKJPARS PDL byte. Note: Bits 1-7 are reserved.

Symbolic Name	Size	Value	Setting of Bit 0 - Meaning
CAFREEDL	1 Bit	80 hex	0 - PDL requires FREEMAIN 1 - PDL does not exist

CASCBFFL*[2]

The following table indicates the values against which CASCBFFL is compared.

CASCBFFL contains the subcommand buffer flags. Note: Bits 1-7 are reserved.

Symbolic Name	Size	Value	Setting of Bit 0 - Meaning
CAOPERND	1 Bit	80 hex	0 - subcommand operands do not exist 1 - subcommand operands do exist

CAATTN*[3]

The following table indicates the values against which CAATTN is compared.

CAATTN is the ATTENTION ECB. Note: Bit 0 and bits 2-7 are reserved.

Symbolic Name	Size	Value	Setting of Bit 1 - Meaning
CAATTNIS	1 Bit	40 hex	1 - attention not issued 0 - attention issued

CACFLAG1*[4]

The following table indicates the values against which CACFLAG1 is compared.

CACFLAG1 is control flag #1.

Symbolic Name	Size	Value	Meaning
CALNTOVF	1 Bit	80 hex	Line to be verified? 1 - Yes 0 - No
CAVRFYSW	1 Bit	40 hex	Verify switch 1 - On 0 - Off
CAPROMPT	1 Bit	20 hex	Prompt switch 1 - On 0 - Off
CASCANSW	1 Bit	10 hex	Scan switch 1 - On 0 - Off
CAINITSC	1 Bit	08 hex	Spec. call of SCAN? 1 - Yes 0 - No
CAENDSC	1 Bit	04 hex	SCAN called by 'END'? 1 - Yes 0 - No
CACAPS	1 Bit	02 hex	1 - CAPS 0 - ASIS
CANONUM	1 Bit	01 hex	1 - NONUM 0 - NUM

CACFLAG2*[5]

The following table indicates the values against which CACFLAG2 is compared.

CACFLAG2 is control flag #2. Note: Bits 4-7 are reserved.

Symbolic Name	Size	Value	Meaning
CADSMODS	1 Bit	80 hex	Data set modified? 1 - Yes 0 - No
CARECFM	1 Bit	40 hex	1 - FIXED format 0 - VARIABLE format
CASCANON	1 Bit	20 hex	1 - 'SCAN' 0 - 'NOSCAN'
CAMODMSG	1 Bit	10 hex	0 - mode message not to be issued 1 - issue EDIT mode message

CACFLAG3*[6]

The following table indicates the values against which CACFLAG3 is compared.

CACFLAG3 has been equated to CAIMFLG.

CACFLAG3 is control flag #3 indicating the flags used by INPUT.
Note: Bits 6-7 are reserved.

Symbolic Name	Size	Value	Meaning
CAIMPT	1 Bit	80 hex	1 - Prompt 0 - No prompt
CAIMINS	1 Bit	40 hex	1 - INSERT Processing 0 - NOT INSERT Processing
CAIMSC	1 Bit	20 hex	1 - INPUT Entered from carriage return 0 - Entered from other than carriage return
CAIMIR	1 Bit	10 hex	1 - I FORM 0 - R FORM
CAIMCIN	1 Bit	08 hex	1 - Increment specified 0 - Increment not specified
CAIMSFPT	1 Bit	04 hex	1 - INPUT will prompt 0 - TCAM will prompt

CACFLAG4*[7]

The following table indicates the value against which CACFLAG4 is compared.

CACFLAG4 is control flag #4. Note: Bits 5-7 are reserved.

Symbolic Name	Size	Value	Meaning
CAFINDIS	1 Bit	80 hex	1 - FIND issued 0 - FIND not issued
CAPTGTBF	1 Bit	40 hex	Buffer to be freed at exit from subcommand? 1 - Yes 0 - No
CATPUTVF	1 Bit	20 hex	Verify line to be printed at terminal? 1 - Yes 0 - No
CAABEND	1 Bit	10 hex	1 - ABEND is in progress 0 - ABEND not in progress
CASCRC20	1 Bit	08 hex	1 - Syntax checker recovery in progress 0 - Syntax checker recovery not in progress.
CAINPROC	1 Bit	04 hex	1 - Command procedure is current input source 0 - Terminal is current input source

CACFLAG6*[8]

Control flag #6. Note: Bits 3-7 are reserved.

Symbolic Name	Size	Value	Meaning
CAFREE	1 Bit	80 hex	GOFORT statement format 1 - FREE 0 - FIXED
CACHAR48	1 Bit	40 hex	PL1 48-character set? 1 - Yes 0 - No
CACHAR60	1 Bit	20 hex	PL1 60-character set? 1 - Yes 0 - No

CADSCODE*[9]

The following table indicates the values against which CADSCODE is compared.

CADSCODE contains the data set type code.

Symbolic Name	Size [1]	Value	Meaning
CANOTYPE	1 byte	00 hex	No data set type entered
CAPLIF	1 byte	01 hex	PL/IF data set type
CAFORTE	1 byte	02 hex	FORTRAN E compiler type
CAFORTG	1 byte	03 hex	FORTRAN G compiler type
CAFORTH	1 byte	04 hex	FORTRAN H compiler type
CATEXT	1 byte	05 hex	Text data set type
CADATA	1 byte	06 hex	Data data set type
CACLIST	1 byte	07 hex	Control list data set type
CACNTL	1 byte	08 hex	Control data set type
CAASM	1 byte	15 hex	Assembler data set type
CACOBOL	1 byte	16 hex	COBOL data set type
CAFORTGI	1 byte	17 hex	FORTRAN GI compiler type
CAGOFORT	1 byte	1F hex	GOFORT data set type
CABASIC	1 byte	20 hex	Basic data set type
CAIPLI	1 byte	21 hex	IPL/I data set type
CAPLI	1 byte	22 hex	PL/1 data set type
CAEDTTYP	1 byte	32 hex	Maximum value for data set type that can be edited

[1] Note: Each symbolic name refers to a bit-setting configuration of the entire byte.

CADSATTR*10

The following table indicates the values against which CADSATTR is compared.

CADSATTR contains the data set attributes. Note: Bit 7 is reserved.

Symbolic Name	Size	Value	Meaning
CARUN	1 Bit	80 hex	Executable under EDIT? 1 - Yes 0 - No
CASCAN	1 Bit	40 hex	Syntax checking allowed? 1 - Yes 0 - No
CACAPSRQ	1 Bit	20 hex	CAPS required? 1 - Yes 0 - No
CACAPSDF	1 Bit	10 hex	1 - CAPS default 0 - ASIS default
CADSCONT	1 Bit	08 hex	Continuation remains in record? 1 - Yes 0 - No
CALNNUM	1 Bit	04 hex	Data set must be line numbered? 1 - Yes 0 - No
CALRECLX	1 Bit	02 hex	LRECL default required? 1 - Yes 0 - No

CADSATR2*11

The following indicates the value with its corresponding symbolic name against which CADSATR2 is compared.

CADSATR2 contains data set attributes. Note: Bits 2-7 are reserved.

Symbolic Name	Size	Value	Meaning
CALINTAB	1 Bit	80 hex	Line number length in tab value? 1 - Yes 0 - No
CADSNDEF	1 Bit	40 hex	DSTYPE is data set name qualifier default? 1 - Yes 0 - No
CARUNDS	1 Bit	10 hex	Prompter accepts in-core source 1 - Yes 0 - No
CAOBJGEN	1 Bit	20 hex	Is an object data set generated? 1 - Yes 0 - No

CARECFMD*[12]

The following table indicates the values against which CARECFMD is compared.

CARECFMD contains record format codes. Note: Bits 2-7 are reserved.

Symbolic Name	Size [1]	Value	Meaning
CARECFMF	2 Bits	80 hex	Fixed record format
CARECFMV	2 Bits	40 hex	Variable record format
CARECFMU	2 Bits	C0 hex	Undefined record format

1 Note: Each symbolic name refers to a bit-setting configuration of bits 0 and 1.

CAEDFLAG*[13]

The following table indicates the values against which CAEDFLAG is compared.

CAEDFLAG is the control flag for the EDIT data set.

Symbolic Name	Size	Value	Meaning
CAEDITDS	1 Bit	80 hex	1 - EDIT data set 0 - SAVE data set
CAEDFNCP	1 Bit	40 hex	Final copy to be performed? 1 - Yes 0 - No
CAEDINCP	1 Bit	20 hex	Initial copy to be performed? 1 - Yes 0 - No
CAEDDISP	1 Bit	10 hex	1 - DISP = OLD 0 - DISP = NEW
CAEDMEM	1 Bit	08 hex	Member exists? 1 - Yes 0 - No
CAEDDSOR	1 Bit	04 hex	1 - DSORG = PO 0 - DSORG = PS
CAEDUNCG	1	02 hex	1 - UNCATLG 0 - CATLG
CAEDALOC	1	01 hex	Data set allocated? 1 - Yes 0 - No

CASAFLAG*14

The following table indicates the values against which CASAFLAG is compared.

CASAFLAG is the control flag for the SAVE data set.

Symbolic Name	Size	Value	Meaning
CASAVEDS	1 Bit	80 hex	1 - EDIT data set 0 - SAVE data set
CASAFNCP	1 Bit	40 hex	Final copy to be performed? 1 - Yes 0 - No
CASAINCP	1 Bit	20 hex	Initial copy to be performed? 1 - Yes 0 - No
CASADISP	1 Bit	10 hex	1 - DISP = OLD 0 - DISP = NEW
CASAMEM	1 Bit	08 hex	Member exists? 1 - Yes 0 - No
CASADSOR	1 Bit	04 hex	1 - DSORG = PO 0 - DSORC = PS
CASAUNCG	1 Bit	02 hex	1 - UNCATLG 0 - CATLG
CASAALOC	1 Bit	01 hex	Data set allocated? 1 - Yes 0 - No

CASYNSW*15

The following table indicates the values against which CASYNSW is compared.

CASYNSW contains the list switches. Note: Bits 6 and 7 are reserved.

Symbolic Name	Size	Value	Meaning
CASYNLN	1 Bit	40 hex	Line numbered? 0 - Yes 1 - No
CASYNIS	1 Bit	10 hex	Diagnose incomplete statements? 0 - Yes 1 - No
CASYNRFM	1 Bit	08 hex	0 - FIXED format 1 - VARIABLE format
CASYNSF	1 Bit	04 hex	0 - Standard form 1 - Free form
CASYNML	1 Bit	02 hex	0 - LMSG 1 - SMSG
CASYNSCN	1 Bit	01 hex	0 - SCAN 1 - NOSCAN

CAMAWKA *16

The following table describes the structure of CAMAWKA.

Symbolic Name	Size	Value	Meaning
	7 Bytes	NA	Work Area
MACFLAGS	1 Byte		Control Flags (next 8 bits)
MAECTMOD	1 Bit	80 hex	ECT modified to delete second-level msg
MAABBREV	1 Bit	40 hex	Subcommand name/abbreviation flag
MAENDPRC	1 Bit	20 hex	END processing complete
	5 Bits		Reserved.
MACFLAG2	1 Byte		Control flags (next 8 bits)
MATABLE1	1 bit	80 hex	IBM/user table indicator
	7 Bits		Reserved
	2 Bytes	NA	Reserved

Processor Data Table (IKJEBEPD)

The processor data table is the second Csect of the IKJEBEPS load module. It contains processor-dependent information (attributes) and is searched by the processor data table search routine (IKJEBEPS).

Disp dec.	Disp hex.	Field Name	Field Size	Contents
0	0	CADSTYPE	8	Data set type keyword
8	8	CADSQUAL	8	Data set name qualifier
16	10	CABLKS	2	Default block size
18	12	CALINE	1	Line number offset
19	13	CALENGTH	1	Line number length
20	14	CATABS	12	Tabsetting values
32	20	CASYNAME	8	Syntax checker name
40	28	CADSCODE	1	Data set type code
41	29	CADSATTR	1	Data set attributes, Byte 1
42	2A	CADSATR2	1	Data set attributes, Byte 2
43	2B	CARECFMD*[12]	1	Default record format
44	2C	CAFLRLDF	2	F format LRECL default
46	2E	CAFLRLMX	2	F format LRECL maximum
48	30	CAVLRLDF	2	V format LRECL default
50	32	CAVLRLMX	2	V format LRECL maximum
52	34	CAULRLDF	2	U format LRECL default
54	36	CAULRLMX	2	U format LRECL maximum
56	38	CACHKOPT	2	Syntax checker option word, Bytes 0 and 1
58	3A	CAPRNAME	8	Prompter name
66	42	CAEXTNAM	8	User exit name
74	4A		6	Reserved storage

The following tables present processor data table field values for the various data set types.

Table 33. Field Values for Data Set Types (1 of 4)

Field Name	CADSTYPE	CADSQUAL	CABLKS	CALINE	CALENGTH	CATABS
Field Values	PLIF	PLI	400	73	8 'FF'	2 72 0
	FORTE	FORT	400	73	8 'FF'	7 72 0
	FORTG	FORT	400	73	8 'FF'	7 72 0
	FORTH	FORT	400	73	8 'FF'	7 72 0
	TEXT	TEXT	1680	1	8 'FF'	5 10 15 20 30 40 0
	DATA	DATA	1680	73	8 'FF'	10 20 30 40 50 60 0
	CLIST	CLIST	1680	1	8 'FF'	10 20 30 40 50 60 0
	CNTL	CNTL	1680	73	8 'FF'	10 20 30 40 50 60 0
	ASM	ASM	1680	73	8 'FF'	10 16 31 72 0
	COBOL	COBOL	400	1	6 'FF'	8 12 72 0
	FORTGI	FORT	400	73	8 'FF'	7 72 0
	GOFORT	FORT	1680	1	8 'FF'	7 72 0
	BASIC	BASIC	1680	1	8 'FF'	10 20 30 40 50 60 0
	IPLI	IPLI	1680	1	8 'FF'	10 20 30 40 50 60 0
	PLI	PLI	400	1	8 'FF'	5 10 15 20 25 30 35
						40 45 50 0
	User-defined Other*1			73	8 'FF'	10 20 30 40 50 60 0

Unusable under EDIT-(See Table 34)

Table 33. Field Values for Data Set Types (2 of 4)

Field Name	CADSTYPE	CASYNAME	CADSCODE*2	CADSATTR*3	CADSATR2*3	CARECFMD*4
Field Values	PLIF	PLIFSCAN	'01'	'70'	'00'	X'80'
	FORTF	IPDSNEXC	'02'	'72'	'00'	X'80'
	FORTG	IPDSNEXC	'03'	'72'	'00'	X'80'
	FORTH	IPDSNEXC	'04'	'72'	'00'	X'80'
	TEXT	0	'05'	'40'	'40'	X'40'
	DATA	0	'06'	'10'	'40'	X'80'
	CLIST	0	'07'	'3C'	'40'	X'40'
	CNTL	0	'08'	'32'	'40'	X'80'
	ASM	0	'15'	'B2'	'60'	X'80'
	COBOL	0	'16'	'B2'	'E0'	X'80'
	FORTGI	IPDSNEXC	'17'	'F2'	'20'	X'80'
	GOFORT	IPDSNEXC	'1F'	'DC'	'40'	X'40'
	BASIC	IKJNC211	'20'	'F4'	'40'	X'40'
	IPLI	IKJNC211	'21'	'F4'	'40'	X'40'
	PLI	PLISCAN	'22'	'D0'	'40'	X'40'
	User-Defined Other*1	0	'22'-'36'	'10'	'10'	X'80'

*1Unusable under EDIT-see Table 34.
*2See Table 35 for explanation of values.
*3See Table 36 for explanation of values.
*4See Table 37 for explanation of values.

Table 33. Field Values for Data Set Types (3 of 4)

Field Name	CADSTYPE	CAFLRLDF	CAFLRLMX	CAVLRLDF	CAVLRLMX	CACHKOPT	CAEXTNAM
Field Values	PLIF	80	100	0	0	'0248'	0
	FORTE	80	80	0	0	'0101'	0
	FORTG	80	80	0	0	'0202'	0
	FORTH	80	80	0	0	'0002'	0
	TEXT	0	255	255	255	'0000'	0
	DATA	80	255	0	255	'0000'	0
	CLIST	0	255	255	255	'0000'	0
	CNTL	80	80	0	0	'0000'	0
	ASM	80	80	0	0	'0000'	0
	COBOL	80	80	0	0	'0000'	0
	FORTGI	80	80	0	0	'0402'	0
	GOFORT	0	255	255	255	'0302'	0
	BASIC	0	120	120	120	'0100'	0
	IPLI	0	120	120	120	'0000'	0
	PLI	0	100	104	104	'0248'	0
	User-defined	80	255	0	255	'0000'	0
	Other*1					'0000'	0

*1 Unusable under EDIT - See Table 34

Table 33. Field Values for Data Set Types (4 of 4)

Field Name	CADSTYPE	CAULRLDF	CAULRLMX	CAPRNAME
Field Values	PLIF	0	0	0
	FORTE	0	0	0
	FORTG	0	0	0
	FORTH	0	0	0
	TEXT	0	0	0
	DATA	0	0	0
	CLIST	0	0	0
	CNTL	0	0	0
	ASM	0	0	ASM
	COBOL	0	0	COBOL
	FORTGI	0	0	FORT
	GOFORT	0	0	GOFORT
	BASIC	0	0	IKJNC211
	IPLI	0	0	IKJNC211
	PLI	0	0	PLI
	User-Defined	0	0	0
	Other*1			

*1Unusable under EDIT - See Table 34

Table 34. Field Values for Data Set Types Not Used With EDIT (1 of 4)

Field Name	CADSTYPE	CADSQUAL	CABLKS	CALINE	CALENGTH	CATABS
Field Values	STEX	STEX	0	0	0	'00'
	OBJ	OBJ	0	0	0	'00'
	LIST	LIST	3556	1	8	'00'
	LOAD	LOAD	0	0	0	'00'
	LINKLIST	LINKLIST	81	0	0	'00'
	LOADLIST	LOADLIST	81	0	0	'00'
	TESTLIST	TESTLIST	1695	0	0	'00'
	OUTLIST	OUTLIST	0	0	0	'00'

Table 34. Field Values for Data Set Types Not Used With EDIT (2 of 4)

Field Name	CADSTYPE	CASYNAME	CADSCODE*1	CADSATTR*2	CADSATR2*2	CARECFMD*3
Field Values	STEX	0	'33'	'00'	'40'	0
	OBJ	0	'34'	'00'	'40'	0
	LIST	0	'35'	'00'	'40'	'40'
	LOAD	0	'36'	'00'	'40'	0
	LINKLIST	0	'37'	'40'	'80'	'80'
	LOADLIST	0	'38'	'40'	'80'	'80'
	TESTLIST	0	'39'	'00'	'40'	'40'
	OUTLIST	0	'3A'	'00'	'40'	0

*1See Table 35 for explanation of values.
*2See Table 36 for explanation of values.
*3See Table 37 for explanation of values.

Table 34. Field Values for Data Set Types Not Used With EDIT (3 of 4)

Field Name	CADSTYPE	CAFLRLDF	CAFLRLMX	CAVLRLMX	CAVLRLDF
Field Values	STEX	0	0	0	0
	OBJ	0	0	0	0
	LIST	0	0	148	148
	LOAD	0	0	0	0
	LINKLIST	81	81	0	0
	LOADLIST	81	81	0	0
	TESTLIST	0	0	1695	1695
	OUTLIST	0	0	0	0

Table 34. Field Values for Data Set Types Not Used With EDIT (4 of 4)

Field Name	CADSTYPE	CAULRLDF	CAULRLMX	CAPRNAME
Field Values	STEX	0	0	0
	OBJ	0	0	0
	LIST	0	0	0
	LOAD	0	0	0
	LINKLIST	0	0	0
	LOADLIST	0	0	0
	TESTLIST	0	0	0
	OUTLIST	0	0	0

Table 35. Explanation of CADSCODE Field Values

Range of codes (base 10, inclusive)	Meaning
1-20	"Standard" EDIT data set types
21-30	"Standard" EDIT data set types, except for having prompters which are IBM Program Products
31-50	Data set types which are not IBM Type I, and/or whose compilers and syntax checkers are IBM Program Products
51-255	Non-EDITable data set types

Table 36. Explanation of CADSATTR and CADSATR2 Field Values

CADSATTR, Bit 0: Data set is executable under EDIT 1: Syntax checking is available 2: CAPS attribute is required 3: CAPS is a default attribute 4: Continuation-character must remain in record 5: Data set must be line-numbered 6: Default LRECL is required
CADSATR2, Bit 0: Line number length is included in tab value 1: Data set type keyword is the default for this data set name qualifier 2: Prompter for data set type generates an object data set 3: Prompter for data set type accepts in-core source input

Table 37. Explanation of CARECFMD Field Values

Setting	Meaning
X'40'	Variable format is default.
X'80'	Fixed format is default.
X'C0'	Undefined format is default.

IBM- and User-Supplied Tables of Subcommands

This table describes the formats of IKJEBMA8 and IKJEBMA9 which are the csects of the controller routine (IKJEBEMA) that contain the lists of available EDIT subcommands.

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	None	1	Length of subcommand name (m)
1	1	None	m	Subcommand name
m+1	m+1	None	1	Length of subcommand name abbreviation (n)
m+2	m+2	None	n	Subcommand name abbreviation
m+2 +n	m+2 +n	None	8	Subcommand processor load module name

EDIT Access Method Work Area (UTILWORK)

Disp dec	Disp hex	Field Name	Size Bytes	Contents
0	0	UTILDCB	88	BSAM dcb
88	58	VTABLE	4	Pointer to vector table which contains addresses of Access Method modules
92	5C	DCBBUFIO	4	Address of buffer being written
96	60	DCBBUFAD	4	Pointer to first buffer in chain
100	64	DCBWRKAD	4	Address of work buffer
104	68	DCBEBQX	4	Pointer to first empty block in chain
108	6C	DCBRECNO	4	Number records in utility data set
112	70	DCBNLEV	2	Level number of dsdb; see NLEV field in dsdb
114	72		1	Reserved
115	73	DCBSTATS*[1]	1	Status of data set
116	74	PRIMEKEY	4	Current reference key
120	78	SAVEKEY	4	Temporary storage of key
124	80	SECONKEY	8	Key/TTR of old block
132	88	THIRDKEY	8	Key/TTR of new block
140	90	PARM1	4	First word of parameter list
144	94	PARM2	4	Second word of parameter list
148	98	PARM3	4	Third word of parameter list
152	9C	XDAPWLST	68	List form of XDAP write
220	E0	WRITLIST	20	List form of BSAM write
240	F4	XDAPRLST	68	List form of XDAP read
308	138	TEMPAREA	128	Temporary work area
436	1B8	SAVEAREA	432	Register save area

DCBSTATS*[1]

The following table indicates the meaning of the bit settings for the DCBSTATS byte.

Bit On	Meaning
0	Block must be re-organized.
1	Block to be split.
3	I/O in progress.
6	Empty data set.
7	I/O error.

EDIT Access Method Control Blocks

DATA BLOCKS

The following table describes the data block control fields and their contents.

Data Block Format

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	NUMREC	2	Number of records in the data block.
2	2	DATASTRT	2	Pointer to where next record inserted into data block should go.
4	4	RECVSP	2	Number of bytes of recoverable, unused space between the position pointed to by DATASTRT and the end of the data block.
6*	6*	Locator entry	2	Pointer to a record; the locator entries are in ascending sequence based on the key of the record to which they point.

*Lowest-key entry

DIRECTORY BLOCKS AND LOWER-LEVEL DIRECTORY BLOCKS

The following table describes the control fields and contents of the directory block and the lower-level directory blocks.

Directory Block Format

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	NLEV	2	DSDB - Number of LLDB levels LLDB - Level number of this LLDB A zero-level block always points to a data block.
2	2	NUMINDEX	2	Number of entries in block.
4*	4*	Index entry	7	A 4-byte binary key value and a 3-byte TTR; the 3-byte TTR points to a data block or to a lower-level directory block; the 4-byte key indicates the highest key value in that data block or lower-level directory block.

*First entry

BUFFERS

The following table describes the buffer control fields and their contents.

Buffer Format

Disp dec.	Disp hex.	Field Name	Field Size	Contents
0	0	BUFCHAIN	4	Address of the next buffer in the chain, or zero if this buffer is the last in the chain.
4	4	BUFSTATS	*1	Flags indicating the status of the buffer. See * below.
5	5	BBKCHAIN	3	Address of previous buffer in chain, or zero if this buffer is the first in the chain.
8	8	BUFTTR	4	TTR of the physical block which was last read into this buffer, or into which this buffer is to be written.
12	C	BUFREF	4	Address of index entry or record locator last found.

*Note:

Bit on	Meaning
0	Buffer updated
1	Block in buffer
4	New block
5	BSAM operation
6	XDAP read
7	XDAP write

Command Buffer (CBUF)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	CBUFLNG	2	Length of Command Buffer.
2	2	CBUFOFF	2	Offset to first unscanned byte of data field.
		CBUFDATA	VAR	Variable length data field containing a command and its operands.

Command Processor Parameter List (CPPL)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	CPPLBUF	4	Pointer to the Command Buffer (CBUF).
4	4	CPPLUPT	4	Pointer to the User Profile Table (UPT).
8	8	CPPLPSCB	4	Pointer to the Protected Step Control Block (PSCP).
12	C	CPPLECT	4	Pointer to the Environment Control Table (ECT).

Command Scan Parameter List (CSPL)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	CSPLUPT	4	Address of the User Profile Table (UPT).
4	4	CSPLECT	4	Address of the Environment Control Table (ECT).
8	8	CSPLECB	4	Address of the Event Control Block (ECB).
12	C	CSPLFLAG	4	Address of a Flag Word set as follows: X'00' - syntax check command name X'80' - do not syntax check command name
16	10	CSPLCSOA	4	Address of the Command Scan Output Area. (CSOA)
20	14	CSPLCBUF	4	Address of Command Buffer (CBUF).

Command Scan Output Area (CSOA)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	CSOACNM	4	Address of command name (0 if invalid).
4	4	CSOALNM	2	Length of command name.
6	6	CSOAFLG	1	Flags set as follows:
		CSOAVWP		X'80' - valid with parameters
		CSOAVNP		X'40' - valid, no parameters
		CSOAQM		X'20' - questionmark
		CSOANOC		X'10' - no command name
		CSOABAD		X'08' - invalid command name
		-------	1	Reserved (0).

DAIR Parameter Block (DAPB08)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	DA08CD	2	Entry Code X'0008'.
2	2	DA08FLG	2	Flags set on return, as follows: Bit Meaning when set 0 Data set allocated but secondary error occurred. Register 15 contains error return code. 1-15 Reserved (0).
4	4	DA08DARC	2	Error return code from Dynamic Allocation routines.
6	6	DA08CTRC	2	Error return code from Catalog Management routines.
8	8	DA08DSN	4	Address of DSNAME Buffer. The format of the DSNAME Buffer is as follows: Byte Contents 0-1 The length, in bytes, of the DSNAME. 2-45 The DSNAME, left justified, and padded to the right with blanks.
12	0C	DA08DDNM	8	DDNAME for the data set. If a specific DDNAME is not required, this field must contain 8 blanks; the DDNAME to which the data is allocated will be placed in this field.
20	14	--------	8	Unit Name desired.
28	1C	--------	8	Serial Number desired. Only the first 6 bytes are significant. If the serial number is less than 6 bytes, it must be padded to the right with blanks. If the serial number is omitted, the entire field must contain blanks.
36	24	--------	4	Block size requested. The average record length desired.
40	28	--------	4	Primary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.

(Part 1 of 3)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
44	2C	--------	4	Secondary space quantity desired. The high order byte must be zero, the low order three bytes contain the space quantity desired. If the quantity is omitted, the entire field must be set to zero.
48	30	--------	4	Directory quantity desired. The high order byte must be zero; the low order three bytes contain the number of directory blocks desired. If the quantity is omitted, the entire block field must be set to zero.
52	34	--------	8	Member name of a partitioned data set. If the name has less than 8 characters, it must be padded to the right with blanks. If the name is omitted, the entire field must contain blanks.
60	3C	--------	8	Password for the data set. If the password has less than 8 characters, it must be padded to the right with blanks. If the password is omitted, the entire field must contain blanks. Bit settings that indicate the status of the data set, as follows: Bit Meaning when set 0-3 Reserved (0) 4 SHR 5 NEW 6 MOD 7 OLD
69	45	--------	1	Bit settings that indicate the normal disposition of the data set, as follows: Bit Meaning when set 0-3 Reserved (0) 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG
70	46	--------	1	Bit settings that indicate the abnormal disposition of the data set, as follows: Bit Meaning when set 0-3 Reserved (0) 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG

(Part 2 of 3)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
71	47	--------	1	Bit settings that control the operations to be performed by DAIR, as follows: Bit Meaning when set 0-1 Specifies the type of units desired for the space parameters, as follows: '01'B units are in average block length '10'B units are in TRKS '11'B units are in CYLS 2 Prefix userid to DSNAME 3 RLSE is desired 4 Data set is to be permanently allocated; not be unallocated until specifically requested. 5 DUMMY data set is desired. 6-7 Reserved (0).
72	48	--------	3	Reserved (0).
75	4B	--------	1	Bit settings on return that indicate the organization of the data set, as follows: Bit Meaning when set 0 Indexed Sequential (IS) 1 Physical Sequential (PS) 2 Direct Organization (DO) 3-5 Reserved (0) 6 Partitioned Organization (PO) 7 Unmoveable

(Part 3 of 3)

DAIR Parameter Block (DAPB18)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	DA18CD	2	Entry Code X'0018'
2	2	DA18FLG	2	Flags set on return, as follows: Bit Meaning when set 0-1 Reserved (0) 2 Prefix 3-15 Reserved (0)
4	4	DA18DARC	2	Error return code from Dynamic Allocation routines.
6	6	DA18CTRC	2	Error return code from Catalog Management routines.
8	8	DA18DSN	4	Address of the DSNAME buffer. The format of the DSNAME Buffer is as follows: Byte Contents 0-1 The length, in bytes, of the DSNAME. 2-45 The DSNAME, left justified, and padded to the right with blanks.
12	C	DA18DDN	8	DDNAME of the data set to be unallocated.
20	14	--------	8	Member name of a partitioned data set. If the name has less than 8 characters, it must be padded to the right with blanks. If the password is omitted, the entire field must contain blanks.
28	1C	--------	2	SYSOUT class. An alphabetic or numeric character. If SYSOUT is not specified, this field must contain blanks.
30	1E	--------	1	Bit settings that indicate the normal disposition of the data set, as follows: Bit Meaning when set 0-3 Reserved (0) 4 KEEP 5 DELETE 6 CATLG 7 UNCATLG

(Part 1 of 2)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
31	1F	DA18CTRL	1	Bit settings that control DAIR operations, as follows: Bit Meaning when set 0-1 Specifies the type of units desired for the space parameters, as follows: '0I'B units are in average block length '10'B units are in TRKS '11'B units are in CYLS 2 Prefix userid to DSNAME 3 RLSE is desired 4 Data set is to be permanently allocated; not be unallocated until specifically requested. 5 DUMMY data set is desired. 6-7 Reserved (0).
32	20	--------	8	The jobname for enqueuing SYSOUT data sets. If the jobname is omitted, the jobname will be taken from the TIOT.

(Part 2 of 2)

GETLINE Parameter Block (GTPB)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	PARMCNTL	2	Control flags. Bit settings that indicate the operation to be performed, as follows: Bit Meaning when set 0-2 Reserved (0). PARMPHYS 3 The line of input is a physical line, if zero, it is a logical line. PARMTERM 4 The source of input is the terminal; if zero, it is as pointed to by the top element on the input stack. 5-15 Reserved (0).
2	2	PARMTGT	2	TGET Options. Bit settings, as follows: Bit Meaning when set 0 Always set for TGET. 1-2 Reserved (0). 3 NOWAIT was specified; if zero, WAIT was specified. 4-6 Reserved (0). 7 ASIS was specified, if zero.

PUTGET Parameter (PGPB)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	PTGPCNTL	2	PUTGET Options. Bit settings that indicate the output operations to be performed by PUTGET, as follows:
				Bit Meaning when set
		PTGPBTO		0 Reserved (0).
		PTGPPUT		1 Reserved (0).
		PTGPDTMS		2 Always zero for PUTGET.
		PTGPSNGT		3 Single-level format.
		PTGPMLIN		4 Always zero for PUTGET.
		PTGPMLEV		5 Multi-level format.
		PTGPIFOR		6 Always zero for PUTGET.
		PTGPPRMT		7 Prompt message.
		PTGPMODE		8 Mode message.
		PTGPDMND		9 Reserved (0).
		PTGPFORM		10 Reserved (0).
		PTGPBYPS		11 Bypass processing. The typing element will not leave an impression on the paper, but the message will be received from the terminal.
		PTGPUNUS		12-15 Reserved (0).
2	2	PTGPTPUT	2	TPUT Options Field. Bit settings that indicate the TPUT options requested, as follows:
				Bit Meaning when set
				0 Always zero for TPUT. 1-2 Reserved (0). 3 NOWAIT processing requested. 4 Hold processing requested. 5 BREAKIN processing requested. 6 CONTROL processing requested. 7 ASIS processing requested. 8-15 Reserved (0).
4	4	PARMAOUT	4	The address of the Output Line Descriptor (OLD) for the message.

(Part 1 of 2)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
8	8	PTGGCNTL	2	GET Options. Bit settings that indicate the input operations to be performed by PUTGET, as follows: Bit Meaning when set
		PTGGBITO		0 Reserved (0).
		PTGGGET		1 Not used in PUTGET.
		PTGGPHYS		2 Not used in PUTGET.
		PTGGTERM		3 Demand from terminal.
		PTGGBRSV		4-15 Reserved (0).
10	A	PTGGTGET	2	TGET options. Bit settings that indicate the TGET options requested, as follows: Bit Meaning when set 0 Always set for TGET. 1-2 Reserved (0). 3 NOWAIT processing requested. 4-6 Reserved (0). 7 ASIS processing requested. 8-15 Reserved (0).
12	C	PTGGADIN	4	The address of the input buffer in which the line is placed.

(Part 2 of 2)

PARSE Parameter List (PPL)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	PPLUPT	4	Address of the User Profile Table (UPT)
4	4	PPLECT	4	Address of the Environment Control Table (ECT).
8	8	PPLECB	4	Address of the Event Control Block.
12	C	PPLPCL	4	Address of the Parameter Control List (PCL).
16	10	PPLANSP	4	Address of the place where Parse will place the address of the Parameter Description List (PDL).
20	14	PPLCBUF	4	Address of the Command Buffer (CBUF).
24	18	PPLWORD	4	Reserved for the installation's use.

PUTLINE Parameter Block (PTPB)

Disp Dec.	Disp Hex.	Field Name	Bytes Size	Contents
0	0	PARMCNTL	2	Control flags. Bit settings, as follows: Bit Meaning when set 0 Reserved (0). 1 Put line(s) to the terminal. 2 Line contains data. If 0, line contains message. 3 Line is single-level or single-line. 4 Multi-line format. 5 Multi-level format. 6 Informational message. 7 Reserved (0). 8 Reserved (0). 9 Reserved (0). 10 Format only. (Do not put out message to the terminal). 11-15 Reserved (0).
		PARMPUTO		
		PARMPUT		
		PARMDTMS		
		PARMSNGL		
		PARMMLIN		
		PARMMLEV		
		PARMIFOR		
		PARMPRMT		
		PARMMODE		
		PARMDMND		
		PARMFORM		
		PARMUNUS		
2	2	PARMTPUT	2	TPUT Options field. Bit settings that indicate the TPUT options requested, as follows: Bit Meaning when set 0 Always zero for TPUT. 1-2 Reserved (0). 3 NOWAIT processing requested. 4 HOLD processing requested. 5 BREAKIN processing requested. 6 CONTROL processing requested. 7 ASIS processing requested. 8-15 Reserved (0).
4	4	PARMAOUT	4	The address of Output Line Descriptor (OLD) for a message or the address of the fullword header for data.
8	8	PARMAFRM	4	The address of the formatted line if "format only" was specified. Otherwise, the field contains zeroes.

Syntax Checker Control Blocks

The following tables describe the contents of the control blocks pointed to by the syntax checker parameter list. The topic "Syntax Checking" in Section 2: Method of Operation discusses EDIT's use of the syntax checkers.

BUFFER

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	C	1	Number of records in buffer (maximum-127); bit zero is set to 1 when the syntax checker has scanned all records in the buffer.
1	1	Chain	3	Address of next buffer; set to zero if this is last buffer in chain.
4	4	Record	Variable	Line or lines of source input data to be syntax checked; can be fixed- or variable-length, numbered or unnumbered.

COMMAND INTERFACE (DELETE SUBCOMMAND)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	WORD1	4	Reserved.
4	4	STARTV	4	Binary value of starting line number.
8	8	STOPV	4	Binary value of ending line number.

COMMAND INTERFACE (RUN SUBCOMMAND)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents
0	0	WORD1	4	Reserved.
4	4	CMDPTR	4	Pointer to command (if high-order byte is X'80', operands are present).
8	8	TMPPRM	4	Pointer to Terminal Monitor Program parameter list.

SYNTAX CHECKER COMMUNICATION AREA

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents	
				Setting	Meaning (Instructions to Syntax Checker)
				bits 0-3	(where n=0 or 1).
				0nnn	First entry - obtain and initialize work are; if a buffer chain is supplied, the syntax checker will set the relative line number counter to zero.
				1n1n	Last entry - release the work area and return; syntax checking is not performed.
				1000	Normal entry - set relative line number counter to zero; perform syntax checking.
0	0	None	1	110n	Entry after return code of 8 (error - buffer checking incomplete) - continue syntax checking.
				1001	Entry after return code of 12 (complete statements have been checked, but last statement in input buffer is incomplete) - if there is no more input (chain address of last buffer or buffer address is zero), syntax check the incomplete statement and return; if there is a new buffer chain, that is, more input (chain address or buffer address is not zero), resume syntax checking at the incomplete statement.
				bits 4-7	Reserved.
1	1	None	4	xxxx	Address of work area stored by syntax checker on first entry.
4	4	None	4	xxxx	Initial entry - maximum statement size specified at SYSGEN (if 0, checker assumes sufficient storage for largest legal statement is available); entry after return code of 4 (error detected, syntax checking complete, second-level message present), or 8 (error detected, syntax checking incomplete) - address of error message area.
8	8	None	4	xxxx	Initial entry - Temporary work area; subsequent entries - address of second error message, if any.
12	C	None	4	xxxx	Temporary storage area used for GETMAIN.

OPTION WORD

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents		
				Setting	Meaning	Syntax Checker
0	0	None	1	X'00'	FORTRAN H level	FORTRAN
				X'01'	FORTRAN E level	FORTRAN
				X'02'	FORTRAN G level	FORTRAN
				X'03'	GOFORT	FORTRAN
				X'04'	FORTRAN G1	FORTRAN
				X'00'	IPLI level	IPLI
				X'01'	BASIC level	BASIC
				xxxx	Value of left source margin	PL1F
1	1	None	1	bits 0-5	Reserved	FORTRAN
				bit 6=1	FORTRAN G/G1/H Code and Go definition to be loaded on initial entry	FORTRAN
				bit 6=0	FORTRAN G/G1/H Code and Go definition not to be loaded on initial entry	FORTRAN
				bit 7=1	FORTRAN E definition to be loaded on initial entry	FORTRAN
				bit 7=0	FORTRAN E definition not to be loaded on initial entry	FORTRAN
				bit 0=1	Entry from INPUT, Insert, linenum, *, CHANGE	IPLI or BASIC
				bit 1=1	Entry from DELETE	IPLI or BASIC
				bit 2=1	Entry from MERGE or RENUM	IPLI or BASIC
				bit 3=1	Translation already complete	IPLI or BASIC
				bit 4=1	Entry from RUN	IPLI or BASIC
				bit 5-7	Reserved	IPLI or BASIC
				xxxx	Value of right source margin	PL1F
2	2	None	1	xxxx	Record length of fixed-length records; binary zero, if variable-length records.	All

(Part 1 of 2)

Disp Dec.	Disp Hex.	Field Name	Field Size	Contents		
				Setting	Meaning	Syntax Checker
3	3	None	1	bit 0=0	CHAR 60	PL1 or IPLI
				bit 0=1	Char 48	PLI or IPLI
				bit 1=0	Line-numbered data set	All
				bit 1=1	Data set not line-numbered	All
				bit 2	Reserved	All
				bit 3=0	Diagnose an incomplete statement	All
				bit 3=1	Delayed scan - return with code of 12 if last statement in input buffer is incomplete; immediate scan - possible incomplete statement in buffer.	All
				bit 4=0	Fixed-length records	All
				bit 4=1	Variable-length records	All
				bit 5=0	Standard form source input	All
				bit 5=1	Free form source input	All
				bit 6=0		
				bit 6=1		
				bit 7=0	SCAN or SCAN ON specified	All
				bit 7=1	NOSCAN or SCAN OFF specified	All

(Part 2 of 2)

Section 6: Diagnostic Aids

This section contains EDIT program messages, information about the use of TEST, and tables; the tables depict message, module and CSECT cross-reference information, error and exceptional conditions, and register usage. Given a diagnostic message, you can identify a particular module as the probable source of error by referring to the information in this section in the following manner:

1. Obtain the diagnostic message.
2. Determine the message number. Refer to "EDIT MESSAGES" if the message number has been suppressed from the diagnostic message.
3. When the message number has been determined, refer to the MESSAGE CROSS-REFERENCE TABLE to obtain a list of module names.
4. An error in one of these modules has resulted in the display of the diagnostic message. If, from knowing the operation that EDIT was performing, you know which modules were executing, refer to the ERROR AND EXCEPTIONAL CONDITIONS TABLE that relate to the modules. If you cannot determine which modules may have been executing at the time the error occurred, refer to the "module cross-reference table" or the "CSECT cross-reference table".

Message Cross-Reference Table

Message Number	Module Names					
IKJ52301I	IKJEBEDA	IKJEBEIN	IKJEBESA			
IKJ52302I	IKJEBEIN	IKJEBESA				
IKJ52303I	IKJEBEIN	IKJEBESA				
IKJ52304I	IKJEBEDA	IKJEBEIN	IKJEBESA	IKJEBEUI		
IKJ52305I	IKJEBEDA	IKJEBEUI				
IKJ52306I	IKJEBEIN	IKJEBESA				
IKJ52307I	IKJEBEIN					
IKJ52309I	IKJEBEFC	IKJEBEIN	IKJEBEUI	IKJEBEUT		
IKJ52310I	IKJEBEIN	IKJEBESA				
IKJ52311I	IKJEBEIN	IKJEBESA				
IKJ52312I	IKJEBECG	IKJEBECH	IKJEBECI	IKJEBECN	IKJEBEDE	IKJEBEDO
	IKJEBEEN	IKJEBEFI	IKJEBEFO	IKJEBEIM	IKJEBEIN	IKJEBEIP
	IKJEBEIS	IKJEBELI	IKJEBELT	IKJEBEMA	IKJEBEME	IKJEBEMR
	IKJEBERE	IKJEBERN	IKJEBERU	IKJEBESA	IKJEBESC	IKJEBESN
	IKJEBETA	IKJEBEUI	IKJEBEUP	IKJEBEVE		

(Continued)

Message Number	Module Names
IKJ52313I	IKJEBECG IKJEBECH IKJEBECI IKJEBECN IKJEBEDA IKJEBEDE IKJEBEDO IKJEBEEN IKJEBEEX IKJEBEFI IKJEBEFO IKJEBEIM IKJEBEIN IKJEBEIP IKJEBEIS IKJEBELI IKJEBELT IKJEBEMA IKJEBEME IKJEBEMR IKJEBERE IKJEBERU IKJEBESA IKJEBESC IKJEBESN IKJEBETA IKJEBEUP IKJEBEVE
IKJ52314I	IKJEBEIN IKJEBESA
IKJ52315I	IKJEBEDA IKJEBEIN IKJEBESA
IKJ52316I	IKJEBEIN IKJEBESA
IKJ52317I	IKJEBEIN IKJEBESA
IKJ52318I	IKJEBEDA IKJEBEIN IKJEBESA IKJEBEUI
IKJ52330I	IKJEBEIN IKJEBESA
IKJ52331I	IKJEBEIN IKJEBESA
IKJ52332I	IKJEBEIN
IKJ52333I	IKJEBEIN
IKJ52334I	IKJEBEIN
IKJ52335I	IKJEBEIN
IKJ52336I	IKJEBEIN
IKJ52337I	IKJEBECO
IKJ52338I	IKJEBECO
IKJ52339I	IKJEBESA
IKJ52340I	IKJEBESA
IKJ52341I	IKJEBESA
IKJ52342I	IKJEBEIN
IKJ52343I	IKJEBEIN
IKJ52360I	IKJEBEIN
IKJ52361I	IKJEBEIN
IKJ52362I	IKJEBEIN
IKJ52363I	IKJEBEIN IKJEBESC IKJEBESN
IKJ52364I	IKJEBEIN IKJEBESC IKJEBESN
IKJ52365I	IKJEBEIN
IKJ52366I	IKJEBEMA
IKJ52367I	IKJEBESC IKJEBESN

(Continued)

Message Number	Module Names
IKJ52400I	IKJEBEIM
IKJ52402I	IKJEBEUT
IKJ52421I	IKJEBESA
IKJ52422I	IKJEBEAE IKJEBECI Note: See the topic "Using the TEST command to diagnose errors in the EDIT program" for more information.
IKJ52423I	IKJEBERU
IKJ52424I	IKJEBEEX IKJEBEIN
IKJ52425I	IKJEBECI
IKJ52500I	IKJEBECG IKJEBECH IKJEBECN IKJEBELT
IKJ52501I	IKJEBEBO IKJEBEDE IKJEBEDO IKJEBEFI IKJEBEFO IKJEBELT IKJEBERE IKJEBESC IKJEBESN IKJEBETO IKJEBEUP
IKJ52502I	IKJEBEDE IKJEBELT IKJEBESC IKJEBESN
IKJ52503I	IKJEBEDE IKJEBELT IKJEBERN IKJEBESC IKJEBESN
IKJ52504I	IKJEBECG IKJEBECH IKJEBECN IKJEBEDE IKJEBELT IKJEBERE IKJEBERN
IKJ52505I	IKJEBEDE IKJEBELI IKJEBEUP
IKJ52506I	IKJEBECG IKJEBECH IKJEBECN IKJEBEFI
IKJ52507I	IKJEBECG IKJEBECH IKJEBECN IKJEBEIM IKJEBEIP IKJEBEIS IKJEBELT
IKJ52550I	IKJEBEME
IKJ52552I	IKJEBESA
IKJ52553I	IKJEBEME IKJEBESA
IKJ52554I	IKJEBESA
IKJ52555I	IKJEBEEN
IKJ52556I	IKJEBETA
IKJ52558I	IKJEBECG IKJEBECH IKJEBECN
IKJ52559I	IKJEBECG IKJEBECH IKJEBECN
IKJ52560I	IKJEBERE
IKJ52561I	IKJEBEIS
IKJ52562I	IKJEBETA
IKJ52565I	IKJEBERN

Using the Test Command to Diagnose Errors in the EDIT Program

If an EDIT subcommand or a syntax checker, operating under EDIT, terminates abnormally, the user is informed (via message IKJ52422I) that an ABEND has occurred. Assuming that the problem is reproducible and that the terminating program is not an EDIT subtask, the user can recreate the problem while EDIT is running under the TEST command. When the error occurs again, the user can then use TEST subcommands to obtain more information about the ABEND. The following sequence of commands will allow the user to run the EDIT program and to test it with the TEST command.

```
READY

test 'sys1.cmdlib(edit)' cp              User identifies the command
                                         processor.

ENTER COMMAND FOR CP

edit 'datasetname'                       User identifies the Edit
                                         data set.

TEST

load 'sys1.cmdlib (ikjebema)'            User identifies the EDIT
                                         program controller routine.

LOADED AT ........

TEST

at ikjebema.ikjebeae (list lr* 1(104))   User places a breakpoint
                                         at the entry point of
                                         IKJEBEAE.

TEST

go                                       User starts program
                                         execution.

EDIT

subcommands of EDIT                      User recreates problem.
```

With this sequence of commands, the abnormal exit routine (IKJEBEAE) will receive control when the abnormal termination occurs. The breakpoint specified by the user will cause the TEST command to receive control. The LIST subcommand of TEST will display a 104-byte work area. In this area the user will find:

- The PSW at the time of abnormal termination.
- The last problem program PSW before abnormal termination.
- The contents of registers 0 through 15 at the time of abnormal termination.
- The name of the terminating program.

Note: A complete description of this work area can be found in the publication: IBM System/360 Operating System: MVT Guide, GC28-6720; see the topic "STAE Macro".

After this work area is displayed, the user can then employ the TEST subcommands to obtain more information about the program error.

EDIT Messages

The following list contains the EDIT program messages. First-level messages are arranged in message number order. Second-level messages immediately follow the appropriate first-level messages; they can be identified by the asterisks which precede the message numbers. EDIT mode messages are included at the end of the list.

```
IKJ52301I   UTILITY DATA SET  NOT ALLOCATED, TOO MANY DATA SETS+
            DATA SET dsname

  *IKJ52301I  USE FREE COMMAND TO FREE UNUSED DATA SETS

IKJ52302I  DATA SET dsname NOT ALLOCATED, DATA SET NOT ON VOLUME+

  *IKJ52302I  CATALOG INFORMATION INCORRECT

IKJ52303I  DATA SET dsname NOT ALLOCATED, REQUIRED VOLUME NOT
           MOUNTED+

  *IKJ52303I  VOLUME NOT ON SYSTEM AND CANNOT BE ACCESSED

IKJ52304I   UTILITY DATA SET
            DATA SET dsname   NOT ALLOCATED, SYSTEM OR
                              INSTALLATION ERROR+

  *IKJ52304I  DYNAMIC ALLOCATION ERROR CODE code
  *IKJ52304I  CATALOG ERROR CODE code
  *IKJ52304I  CATALOG I/O ERROR

IKJ52305I   UTILITY DATA SET
            DATA SET dsname   NOT ALLOCATED, NOT ENOUGH
                              SPACE ON VOLUMES+

  *IKJ52305I  USE DELETE COMMAND TO DELETE UNUSED DATA SETS

IKJ52306I  DATA SET dsname ALREADY IN USE, TRY LATER+

  *IKJ2306I   DATA SET IS ALLOCATED TO ANOTHER USER OR JOB

IKJ52307I  DATA SET dsname NOT IN CATALOG

IKJ52308I  MEMBER member NOT IN DATA SET dsname

IKJ52309I   UTILITY DATA SET  NOT USABLE+
            DATA SET dsname

  *IKJ52309I  OPEN ERROR CODE code
  *IKJ52309I  I/O ERROR synadinfo
  *IKJ52309I  CANNOT OPEN DATA SET

IKJ52310I  INVALID DATA SET NAME, dsname EXCEEDS 44 CHARACTERS

IKJ52311I  MEMBERmember SPECIFIED BUT dsname NOT A PARTITIONED
           DATA SET

IKJ52312I  NOT ENOUGH MAIN STORAGE TO EXECUTE  COMMAND
                                               subcmd

IKJ52313I  COMMAND SYSTEM ERROR+

  *IKJ52313I  service rtn ERROR CODE code
```

```
IKJ52314I   DATA SET dsname RESIDES ON MULTIPLE VOLUMES,
            NOT SUPPORTED.

IKJ52315I  DATA SET dsname NOT RESOLVED, SYSTEM ERROR+

  *IKJ52315I  DEFAULT ERROR CODE code

IKJ52316I  DATA SET dsname WILL CREATE INVALID CATALOG STRUCTURE+

  *IKJ52316I  A QUALIFIER CANNOT BE BOTH AN INDEX AND THE LAST
              QUALIFIER OF A DATA SET NAME

IKJ52317I   DATA SET dsname NOT ON A DIRECT ACCESS DEVICE,
            NOT SUPPORTED

IKJ52318I   DATA SET dsname NOT ALLOCATED+

  *IKJ52318I  INVALID UNIT IN USER ATTRIBUTE DATA SET
  *IKJ52318I  NO UNIT AVAILABLE

IKJ52330I   ISAM                ORGANIZATION OF DATA SET dsname
            DIRECT              NOT ACCEPTABLE+

  *IKJ52330I  ORGANIZATION MUST BE PARTITIONED OR SEQUENTIAL

IKJ52331I  RECORD FORMAT recfm NOT ACCEPTABLE

IKJ52332I  RECORD FORMAT V NOT ACCEPTABLE FOR type DATA SETS


IKJ52333I  BLOCK VALUE INVALID, USING value+

  *IKJ52333I  MAXIMUM BLOCK VALUE IS DEVICE TRACK SIZE
  *IKJ52333I  MAXIMUM BLOCK VALUE IS SYSGEN OPTION
  *IKJ52333I  BLOCK VALUE MUST BE A MULTIPLE OF LINE FOR RECORD
              FORMAT F DATA SETS
  *IKJ52333I  BLOCK VALUE MUST BE AT LEAST 4 GREATER THAN LINE FOR
              RECORD FORMAT V DATA SETS

IKJ52334I   BLOCK  IGNORED, VALID FOR NEW DATA SET ONLY
            LINE

IKJ52335I  INVALID LINE VALUE FOR language, USING value+

  *IKJ52335I  language REQUIRES A LINE SIZE OF 80
  *IKJ52335I  LINE SIZE FOR type MAY NOT EXCEED line size

IKJ52336I  value INVALID LINE VALUE FOR type DATA SET

IKJ52337I  INVALID LINE NUMBER ENCOUNTERED+

  *IKJ52337I  USE EDIT WITH NONUM OPERAND

IKJ52338I  DATA SET dsname NOT LINE NUMBERED, USING NONUM

IKJ52339I  dsname ALREADY EXISTS

IKJ52340A  ENTER NEW NAME OR HIT CARRIER RETURN TO REUSE DATA SET-

IKJ62341A  ENTER NEW MEMBER OR HIT CARRIER RETURN TO REUSE MEMBER-
```

```
IKJ52342I  INVALID LINE SIZE FOR NUM+

  *IKJ52342I  LINE TOO SHORT FOR LINE NUMBERS

IKJ52343I   RECORD FORMAT F NOT ACCEPTABLE FOR GOFORT (FREE)+

  *IKJ52343I  RECORD FORMAT V IS REQUIRED
  *IKJ52343A  USE EDIT WITHOUT SPECIFYING LINE OPERAND

IKJ52360I  INVALID LINE NUMBER FIELD SPECIFIED
           (start col, end col)+

  *IKJ52360I  STARTING COLUMN MUST BE WITHIN THE RANGE 73-80
  *IKJ52360I  LINE NUMBER FALLS OUTSIDE RECORD SIZE
  *IKJ52360I  LINE NUMBER FIELD OPERANDS ARE VALID FOR ASM
              DATA SETS ONLY

IKJ52361I  ASIS INVALID FOR type DATA SET, USING CAPS

IKJ52362I  NONUM INVALID WITH type DATA SET, USING NUM OPTION

IKJ52363I  function INVALID FOR language

IKJ52364I  function NOT AVAILABLE FOR language+

  *IKJ52364I  THE REQUIRED PROGRAM IS NOT AVAILABLE
  *IKJ52364I  PROGRAM IS NO LONGER USABLE

IKJ52365I  INVALID SOURCE MARGIN (start margin, end margin)+

  *IKJ52365I  SOURCE MARGIN SPECIFIED FALLS OUTSIDE LINE LIMITS

IKJ52366I  INVALID SUBCOMMAND subcmd

IKJ52367I   WARNING, SCAN MAY BE INCOMPLETE+

  *IKJ52367I  dstype SYNTAX CHECKER FAILED BUT
              SUCCESSFULLY RECOVERED

IKJ52400I  INPUT TERMINATED, NEXT LINE NUMBER IS number

IKJ52402I  LINE NUMBER LIMIT limit EXCEEDED

IKJ52421I  NOT ENOUGH DIRECT ACCESS SPACE TO SAVE DATA SET

IKJ52422I   cmd    ENDED DUE TO ERROR+
            subcmd

  *IKJ52422I  SYSTEM COMPLETION CODE IS code
  *IKJ52422I  USER COMPLETION CODE IS code

IKJ52423I  type DATA SET CANNOT BE RUN

IKJ52424I  SYSTEM ERROR+

  *IKJ52424I   UTILITY DATA SET  NOT UNALLOCATED, DYNAMIC
               DATA SET dsname   ALLOCATION ERROR CODE code
  *IKJ52424I   UTILITY DATA SET  NOT UNALLOCATED, CATALOG ERROR
               DATA SET dsname   CODE code

  *IKJ52424I   FILE ddname NOT FOUND

IKJ52425I  COMMAND NOT FOUND - cmd name

IKJ52500I  END OF DATA
```

```
      IKJ52501I  NO LINES IN DATA SET
      IKJ52502I  DATA SET NOT LINE NUMBERED
      IKJ52503I  INVALID LINE NUMBER RANGE
      IKJ52504I  LINE NUMBER number NOT FOUND
      IKJ52505I  TOP OF DATA SET
      IKJ52506I  TEXT NOT FOUND
      IKJ52507I  LINE number TRUNCATED+
        *IKJ52507I  LINE LENGTH IS length
      IKJ52550I  NO OPERAND, SUBCOMMAND IGNORED
      IKJ52552I  EDIT DATA SET dsname AND SAVE DATA SET dsname ARE NOT
                 COMPATIBLE
        *IKJ52552I  EDIT DATA SET HAS RECORD FORMAT recfm, SAVE DATA SET
                    HAS RECORD FORMAT recfm, USE ANOTHER DATA SET
        *IKJ52552I  EDIT DATA SET HAS LINE SIZE line size, SAVE DATA SET
                    HAS LINE SIZE line size, USE ANOTHER DATA SET
        *IKJ52552I  EDIT DATA SET HAS BLOCK LENGTH blksize, SAVE DATA SET
                    HAS BLOCK LENGTH blksize, USE ANOTHER DATA SET
      IKJ52553I  SAVED, DATA SET IS EMPTY
      IKJ52554I  BLOCK VALUE TOO LARGE FOR OUTPUT DEVICE, NOTHING SAVED
      IKJ52555I  NOTHING SAVED
      IKJ52556I  TABSET OF tabset EXCEEDS LINE SIZE OF line size
      IKJ52557I  ONLY FIRST 10 VALID TABS USED
      IKJ52558I  LINE OVERFLOW, NEW LINE CREATED
      IKJ52559I  COUNT OR STRING MISSING
      IKJ52560I  LINE NUMBER LENGTH EXCEEDS LINE SIZE
      IKJ52561I  INSERT TERMINATED, NEXT LINE NUMBER IS number
      IKJ52562I  TABSET OF 0 IGNORED
      IKJ52565I  INCOMPLETE SCAN FOR LINE NUMBER REFERENCES IN LINE
                 (linenum)+
        *IKJ52565I  SYNTAX ERROR OR NON-EXISTENT LINE NUMBER REFERENCED.
      IKJ52567A  ENTER DATA SET TYPE
      IKJ52568I  MISSING DATA SET TYPE
MODE MESSAGES
      EDIT or E
      INPUT
      SAVED
      MERGED
      ENTER SAVE OR END-
```

Module Cross-Reference Table

Module Name	Common Name	MO Diag	Flowchart ID
IKJEBEAA	Load module for EDIT Access Method -- (See next table for information about the csects of IKJEBEAA.)		
IKJEBEAE	Abnormal end exit	--	AB
IKJEBEAT	EDIT attention exit	04	AD
IKJEBEBO	BOTTOM subcommand processor	05	AE
IKJEBECG	CHANGE subcommand processor (2nd)	06	AF-AK
IKJEBECH	CHANGE subcommand processor (1st)	06	AL-AN
IKJEBECI	Command invoker	--	AO,AP
IKJEBECN	CHANGE subcommand processor (3rd)	06	AQ-AT
IKJEBECO	Initial copy routine	--	AU-AW
IKJEBEDA	Data Set ALLOCATION-FREE routine	--	AX
IKJEBEDE	DELETE subcommand processor	07	AY-BA
IKJEBEDO	DOWN subcommand processor	08	BE,BF
IKJEBEEX	EDIT Access Method final processing routine	26	BQ
IKJEBEFC	Final copy routine	--	BR,BS
IKJEBEFI	FIND subcommand processor	09	BT
IKJEBEFO	FORMAT subcommand processing	10	BU,BV
IKJEBEHE	HELP/PROFILE subcommand processor	11	BW
IKJEBEIM	INPUT subcommand processor (2nd)	12	BX-CB
IKJEBEIN	EDIT initialization routine	01	CC,CJ
IKJEBEIP	INPUT subcommand processor (1st)	12	CK,CL
IKJEBEIS	INSERT subcommand processor	13	CM,CN
IKJEBELE	Line editing routine	--	co
IKJEBELI	Line insert/replace subcommand processor	14	CP,CQ
IKJEBELT	LIST subcommand processor	15	CT-CV
IKJEBEMA	EDIT controller routine	02	CW-CY
IKJEBEME	MERGE subcommand processor	16	CZ,DA
IKJEBEMR	Translation routine	--	DB

(Part 1 of 2)

Module Name	Common Name	MO Diag	Flowchart ID
IKJEBEMS	Message selection routine	--	DC
IKJEBEPS	Processor data table search routine	--	DH
IKJEBERE	RENUM subcommand processor	18	DJ-DM
IKJEBERN	BASIC renum service routine	--	DN
IKJEBERU	RUN subcommand processor	19	DP,DQ
IKJEBESA	SAVE subcommand processor	20	DR-EB
IKJEBESC	SCAN subcommand processor (1st)	21	EC-EG
IKJEBESE	String search routine	--	EH
IKJEBESN	SCAN subcommand processor (2nd)	21	EI-EN
IKJEBETA	TABSET subcommand processor	22	EO
IKJEBETO	TOP subcommand processor	23	EP
IKJEBEUI	EDIT Access Method initialization routine	26	EQ
IKJEBEUP	UP subcommand processor	24	ER,ES
IKJEBEUT	EDIT Access Method interface routine	27	ET
IKJEBEVE	VERIFY subcommand processor	25	EU

(Part 2 of 2)

CSECT Cross-Reference Table (Module IKJEBEAA)

Csect Name	Common Name	MO Diag	Flowchart ID
IKJEBEAD	TTR assignment routine	33	AA
IKJEBEAS	Buffer assignment routine	36	AC
IKJEBEDL	Record delete routine	32	BB-BD
IKJEBERE	Delete operation	30	BG
IKJEBEDS	Directory search routine	37	BH-BJ
IKJEBEDU	Directory update routine	34	BK-BO
IKJEBELO	Record locate routine	31	CR,CS
IKJEBEMV	Move routine	--	DD
IKJEBERE	Read block routine	38	DI
IKJEBERR	Read operation	29	DO
IKJEBEWA	Wait routine	39	EV
IKJEBEWB	Write block routine	35	EW
IKJEBEWR	Write operation	28	EX-FA

Error and Exeptional Conditions Tables

These tables provide information about module processing that is a result of error or exceptional conditions within the module or modules invoked. The same information is available at the CSECT level for the EDIT Access Method. The following is an index to these tables:

Module Name	Table Number	Module Name	Table Number
IKJEBEAD	77	IKJEBELI	58
IKJEBEAE	40	IKJEBELO	75
IKJEBEAT	41	IKJEBELT	59
IKJEBEBO	49	IKJEBEMA	39
IKJEBECG	50	IKJEBEME	60
IKJEBECH	50	IKJEBEMR	46
IKJEBECI	42	IKJEBERB	81
IKJEBECN	50	IKJEBERE	61
IKJEBECO	43	IKJEBERN	47
IKJEBEDA	44	IKJEBERR	73
IKJEBEDE	51	IKJEBERU	62
IKJEBEDL	76	IKJEBESA	63
IKJEBEDO	52	IKJEBESC	64
IKJEBEDR	74	IKJEBESE	48
IKJEBEDS	80	IKJEBESN	64
IKJEBEDU	78	IKJEBETA	65
IKJEBEEN	53	IKJEBETO	66
IKJEBEEX	70	IKJEBEUI	69
IKJEBEFC	45	IKJEBEUP	67
IKJEBEFI	54	IKJEBEUT	71
IKJEBEFO	55	IKJEBEVE	68
IKJEBEIM	56	IKJEBEWA	82
IKJEBEIN	38	IKJEBEWB	79
IKJEBEIP	56	IKJEBEWR	72
IKJEBEIS	57		

Table 38. IKJEBEIN Error and Exceptional Conditions (Part 1 of 5)

Indication	Condition	Action
PARSE Return code = 4.	PARSE unable to prompt	Return to TMP.
PARSE Return code = 8.	PARSE interrupted by attention.	Return to TMP.
PARSE Return code = 12.	PARSE received invalid parameters.	Issue message IKJ52313I, and return to the TMP.
PARSE Return code = 16.	PARSE unable to get Storage.	Issue message IKJ52312I, and return to the TMP.
IKJEBEPS Return code = 4.	IKJEBEPS unable to validate data set type entered on EDIT command.	Issue message IKJ52313I, and return to the TMP.
IKJEBIN7 Return code = 4.	IKJEBIN7 unable to fully qualify data set name.	Return to TMP.

(Part 1 of 5)

Table 38. IKJEBEIN Error And Exceptional Conditions (Part 2 of 5)

Indication	Condition	Action
IKJEBIN5 Return code = 4.	IKJEBIN5 unable to prompt for data set type.	Return to TMP.
DAIR Return code = 4.	IKJDAIR received invalid parameters.	Issue message IKJ52313I, and return to the TMP.
DAIR Return code = 16.	IKJDAIR found no available entries in TIOT.	Issue message IKJ52301I, and return to the TMP.
DAIR Return code =20.	IKJDAIR found DDNAME currently unavailable.	Issue message IKJ52313I, and return to the TMP.
DAIR Return code = 8, DARC = X'0000' and CTRC = X'0004'	IKJDAIR encountered a catalog management error.	Issue message IKJ52303I, and return to the TMP.
DAIR Return code = 8, DARC = X'0000' and CRTC = X'0008'	IKJDAIR encountered a catalog management error.	Issue message IKJ52302I, and return to the TMP.
DAIR Return code = 8, DARC = X'0000'and CTRC is other than X'0004' or X'0008'	IKJDAIR encountered a catalog management error.	Issue message IKJ52304I, and return to the TMP.
DAIR Return code = 8 or 12 and DARC = X'1710'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52316I, and return to the TMP.
DAIR Return code = 8 or 12 and DARC = X'1704'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52303I and return to the TMP.
DAIR Return code = 8 or 12 and DARC = X'17XX' Where XX is other than '10', '04', or '08'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52304I, and return to the TMP.
DAIR Return code 12 and DARC = X'1708'	IKJDAIR encountered a dynamic allocation error.	Prompt for 'NEW' or 'OLD'. If user enters 'NEW', continue processing. If user enters 'OLD', issue message IKJ52307I, Return.
DAIR Return code 12 and DARC = X'0218'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52303I, and return to TMP.

(Part 2 of 5)

Table 38. IKJEBEIN Error and Exceptional Conditions (Part 3 of 5)

Indication	Condition	Action
DAIR Return code 12 and DARC = X'020C' or X'0210'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52306I, and return to TMP.
DAIR Return code 12 and DARC = X'0214' or X'21C'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52318I and return to TMP.
DAIR Return code 12 and DARC = X'0404'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52317I and return to TMP.
DAIR Return code 12 and DARC = X'041C'	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52314I and return to TMP.
DAIR Return code 12 and a DARC other than the above.	IKJDAIR encountered a dynamic allocation error.	Issue message IKJ52304I, and return to TMP.
DSIDSORG is not X'02', X'40', or X'00'	Data set allocated does not have a valid organization.	Issue message IKJ52330I, and return to TMP.
User entered a member name, but DS1DSORG = X'40'	Data set is not a partitioned data set.	Issue message IKJ52311I and continue.
DCBOFLGS = B'XXX0XXXX'	Open failed for Edit data set when DSORG is partitioned.	Issue message IKJ52309I, and return to TMP.
BLDL Return code = 8.	BLDL failed for member name.	Issue message IKJ52313I, and return to TMP.
BLDL Return coee = 4.	BLDL could not locate member name.	Prompt for 'NEW' or 'OLD'. If user enters 'NEW', continue processing. If user enters 'OLD', issue message IKJ52307I and return to TMP.
End of TIOT reached without finding the DDNAME returned by IKJDAIR.	DDNAME not found in TIOT.	Issue message IKJ52424I, and return to TMP.
UCBTBYT3 = B'XX0XXXXX'	Edit data set resides on a non-direct access device.	Issue message IKJ52317I, and return to TMP.
Obtain returned a non-zero return code.	DSCB could not be read into core.	Issue message IKJ52313I, and return to TMP.

(Part 3 of 5)

Table 38. IKJEBEIN Error and Exceptional Conditions (Part 4 of 5)

Indication	Condition	Action
DS1RECFM is not X'40', X'50', X'80', or X'90'	Data set has a record format other than F, FB, V, or VB.	Issue message IKJ52331I, and return to TMP.
DAIR Return code = 4, 16, 20, 24, and 28.	Unallocation failed for data set.	Issue message IKJ52313I, and return to TMP.
DAIR Return code = 8 or 12.	Unallocation failed for data set.	Issue message IKJ52424I, and return to TMP.
IKJEBIN8 Return code = 4.	IKJEBIN8 failed to validate command operands.	Return to TMP.
IKJEBECO Return code = 8.	IKJEBECO encountered an I/O error on the utility data set or the edit data set.	Return to TMP.
IKJDFLT Return code = 24.	IKJDFLT interrupted by attention.	Return to TMP.
IKJDFLT Return code = 8.	IKJDFLT unable to fully qualify data set name.	Issue message IKJ52310I, and return to TMP.
IKJDFLT Return code = 16.	IKJDFLT unable to fully qualify data set name.	Issue message IKJ52316I, and return to TMP.
IKJDFLT Return code = 4 or 12.	IKJDFLT unable to fully qualify data set name.	Issue message IKJ52315I, and return to TMP.
IKJDFLT Return code = 28 or 32.	IKJDFLT unable to fully qualify data set name.	Issue message IKJ52313I, and return to TMP.
DS1LRECL does not conform to data set type requirements.	LRECL invalid for data set, data set old.	Issue message IKJ52336I, and return to TMP.
Line value does not Conform to data set type requirements.	Line operand invalid for data set type.	Issue message IKJ52335I, and default accordingly.
DEVTYPE return code not zero.	DEVTYPE failed for old data set.	Issue message IKJ52313I, and return to TMP.

(Part 4 of 5)

Table 38. IKJEBEIN Error and Exceptional Conditions (Part 5 of 5)

Indication	Condition	Action
Block line not an integer or block less than line + 4.	Block operand invalid for new data set.	Issue message IKJ52333I, and default accordingly.
Processor table entry does not permit scan	SCAN specified for non-scanable data set type.	Issue message IKJ52364I, and continue.
BLDL for syntax checker failed.	Syntax Checker not in user's system.	Issue message IKJ52363I, and set CASYNAME to zero
User specified NONUM.	Data set must be line numbered.	Issue message IKJ52362I, and default to 'NUM'.
NUM subfield operands are incorrect for ASM data sets, or were specified for other than ASM data sets.	Start colum and length specified incorrectly.	Issue message IKJ52360, and default accordingly.
DS1LRECL contains a value less than 8.	LRECL less than or equal to 7 for old data sets.	Issue message IKJ52342I, and return to the TMP.
ASIS specified on command.	Data set type requires capital letters.	Issue message IKJ52361I, and default to caps.
PLI or PLIF subfield specified incorrectly.	Source margins invalid.	Issue message IKJ52365I, and default to (2,72).
DS1RECFM is variable (old data set)	RECFM=V data set invalid for type specified.	Issue message IKJ52332I, and return to TMP.
Line or block specified on command.	Line and/or block specified for old data set.	Issue message IKJ52334I, and ignore.
IKJPTGT Return code 4, 16, 20, 24	Command system error.	Issue message IKJ52313I and return to TMP.
IKJPTGT Return code 12	Trying to prompt in a procedure.	Issue message IKJ52568I and return to TMP.
IKJPTGT Return code 28	Not enough storage available.	Issue message IKJ52312I and return to TMP.
BLDL Return code 4	User exit not in the system.	Issue message IKJ52364I and return to TMP.

(Part 5 of 5)

Table 39. IKJEBEMA Error and Exceptional Conditions

Indication	Condition	Action
Non-zero return code from STAX SVC.	STAX (attention EXIT not established.	Message IKJ52313I with return code insertion.
Non-zero return code from STAE SVC.	STAE (abnormal end exit) not established.	Message IKJ52313I with return code insertion.
IKJEBESC Return code 12.	IKJEBESC initialization of LANGPRCR failed.	Return to TMP with return code 12.
IKJEBESC Return code 16.	IKJEBESC initialization of LANGPRCR failed.	Set CASCRC20 = 1 and Call IKJEBESC for recovery attempt.
PUTGET Return code 12.	PUTGET for mode message not successful.	Set ECTMSGF to 1 and invoke PUTGET.
PUTGET Return code 16, 20, or 24.	PUTGET for mode message not successful.	Message IKJ52313I with return code insertion.
PUTGET Return code 28.	PUTGET for mode message not successful.	Message IKJ52312I.
IKJSCAN Return code not 0.	IKJSCAN unsuccessful in scanning buffer.	Message IKJ52313I with return code insertion.
Subcommand not found in IBM or user subcommand table.	Invalid subcommand entered.	Message IKJ52366I with subcommand insertion.
Subcommand Processor Return code 12.	Error return from Subcommand - I/O error in utility data set.	return to TMP with return code 12.
IKJEBEMS return code not 0.	Unsuccessful attempt in putting message to user.	Return to TMP with return code 12.
Register 0 contains a non-zero value for STAE retry routine.	No work area pointer available in STAE retry routine.	Issue SVC 3. (EXIT).

Table 40. IKJEBEAE Error and Exceptional Conditions

Indication	Condition	Action
SPAUTOPT (SVC 94) Return code = 4	Unable to stop automatic line number prompting.	Set return code = 0 and return.
IKJEBMA2 Non-zero return code	Unable to clean-up stack.	Set return code = 0 and return.
IKJEBEMS Non-zero return code	Unable to put ABEND message to user.	Set return code = 0 and return.

Table 41. IKJEBEAT Error and Exceptional Conditions

Indication	Condition	Action
GETLINE Non-zero return code	Input line not returned from terminal.	Return without posting attention ECB; no return code set.
SCAN Non-zero return code	Command scan error.	Same as above.

Table 42. IKJEBECI Error and Exceptional Conditions (Part 1 of 3)

Indication	Condition	Action
STAE Return code = 4.	No core available.	Issue message IK152312I; set return code to 8, return.
STAE Return code = 8	Invalid cancel or overlay request.	Issue message IKJ52313I; set return code to 8, return.
STAE Return code = 12	STAE routine or parmlist address invalid.	Same as return code 8 for STAE.
STAE Return code = 16	STAE not connected with users RB.	Same as return code 8 for STAE.
IKJSCAN Return code = 4	Invalid parameters.	Issue message IKJ52313I; flush STACK; set return code to 8; cancel STAE, return.
IKJSCAN Return code = 8	No core available.	Issue message IKJ52312I flush STACK; set return code to 8; cancel STAE, return.

(Part 1 of 3)

Table 42. IKJEBECI Error and Exceptional Conditions (Part 2 of 3)

Indication	Condition	Action
BLDL Return code = 4	Command not found in system.	Issue message IKJ52425I; flush STACK; set return code to 8; cancel STAE, return.
BLDL Return code = 8	I/O error.	Issue message IKJ52313I: flush STACK, set return code to 8; cancel STAE, return.
GETMAIN Return code = 4	No core available.	Issue message IKJ52312I; flush STACK; set return code to 8; cancel STAE, return.
IKJSTCK Return code = 4	Invalid parameters.	Issue message IKJ52313I; FREEMAIN the LSD area; set return code to 8, cancel STAE; return.
IKJPTGT Return code = 0	Stack flushed.	FREEMAIN EDIT buffer if needed, cancel STAE. set return code to 0; return.
IKJPTGT Return code = 8	Attention issued.	Cancel STAE; return.
IKJPTGT Return code = 12	Second level informational message present.	Set ECTMSGF switch on; Re-issue PUTGET.
IKJPTGT Return code =12	Second level diagnostic message present.	Flush STACK; cancel STAE; return.
IKJPTGT Return code = 16	NOWAIT for TPUT and no line found.	Issue message IKJ52313I; flush STACK; set return code to 8; cancel STAE, return.
IKJPTGT Return code = 20	NOWAIT for TGET and no line found.	Same as return code 16 for PTGT.
IKJPTGT Return code = 24	Invalid parameters.	Same as return code 16 for PTGT.

(Part 2 of 3)

Table 42. IKJEBECI Error and Exceptional Conditions (Part 3 of 3)

Indication	Condition	Action
IKJPTGT Return code = 28	No core available.	Issue message IKJ52312I; flush STACK; set return code to 8; cancel STAE, return.
TCLEARQ Return code = 4	TCLEARQ unsuccessful.	Issue message IKJ52313I; flush STACK if caller was RUN; set return code to 8; cancel STAE, return.
IKJDAIR Non-zero return code	IKJDAIR unsuccessful.	Issue message IKJ52313I; set return code to 8; return.

(Part 3 of 3)

Table 43. IKJEBECO Error and Exceptional Conditions

Indication	Condition	Action
DCB not opened, DCBOFLGS ≠ X'10'	Open error.	Set return code of 8, issue IKJ52309I, return to invoking routine.
Return code from IKJEBEUI not 0	IKJEBEUI does not complete successfully.	Existing data set is closed, control is returned to the invoking module with a return code of 8.
Line numbers out of sequence	Invalid line number.	Issue message IKJ52337I, set return code to 8 and return to invoking routine.
Invalid line number	Data set not line numbered but data set type being used must be.	Issue message IKJ52337I, return code of 8, return to the invoking module.
EOF with first read flag on	Date set contains no logical records.	Set return code to 4, return to the invoking module.

Table 44. IKJEBEDA Error and Exceptional Conditions

Indication	Condition	Action
IKJDAIR Return code 4	Invalid parameters.	Put out IKJ52313I; set return code to 8; return to caller.
IKJDAIR Return code = 8	Catalog management error.	Put out IK152304I; set return code to 8; return to caller.
IKJDAIR Return code = 12	Dynamic allocation error.	Put out IKJ52304I, IKJ52303I, IKJ52305I, or IKJ52318I depending on value in DARC field; set return code to 8; return to caller.
IKJDAIR Return code = 16	TIOT full.	Put out IKJ52301I; set return code to 8; return to caller.
IKJDAIR Return code = 20	DDNAME unavailable.	Put out IKJ52313I; set return code to 8; return to caller.
IKJDAIR Return code = 24	DSNAME in concatenated group.	Same as return code 20.
IKJDAIR Return code = 28	DSNAME or DDNAME unavailable.	Same as return code 20.
IKJDAIR Return code = 32	DSNAME in DSE more than once.	Same as return code 20.
IKJDAIR Return code = 36	Error in catalog information routine.	Same as return code 20.
IKJDAIR Return code = 40	Return area for qualifiers filled.	Same as return code 20.

Table 45. IKJEBEFC Error and Exceptional Conditions

Indication	Condition	Action
DCB not opened, DCBOFLGS ≠ X'10'	OPEN error.	Set return code of 8, issue msg. IKJ52309I, return to invoking routine.
IKJEBEUT Return code = 4	Empty utility data set.	Set return code of 4, return to invoking routine.
IKJEBEUT Return code = 8	I/O error in utility data set.	Set return code of 12, return to invoking routine.
SYNAD error exit gains control	I/O error in QSAM data set.	Set return code of 8, issue msg. IKJ52309I, return to invoking routine.

Table 46. IKJEBEMR Error and Exceptional Conditions

Indication	Condition	Action
Syntax Checker Return code = 4	Error found, buffer checking complete.	Continue processing.
Syntax Checker Return code = 8	Error found, buffer checking incomplete.	Put out message IKJ52313I; set return code of 4; return to caller.
Syntax Checker Return code = 12	Last statement in buffer is incomplete.	Same as return code of 8.
Syntax Checker Return code = 16	No core available.	Put out message IKJ52312I; set return code to 4; return to caller.
Syntax Checker Return code =20	Syntax checking not available.	Set return code to 8 and return to caller.
IKJEBEUT Return code = 4	Record not found.	Set return code to 0; return to caller.
IKJEBEUT Return code = 8	I/O error in reading utility data set.	Delete new utility data set, set return code to 12; return to caller.

Table 47. IKJEBERN Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEUT Return code of 4	Line not found.	Issue message IKJ52504I and return to caller with return code of 4.
IKJEBEUT Return code of 8	I/O error.	Return to caller with return code of 8.
GETMAIN Return code of 4	GETMAIN for table space failed.	Issue message IKJ52312I and return to caller with return code of 4.
Line number reference unobtainable	Invalid line reference or syntax error.	Issue message IKJ52565I and return to caller with return code of 4.
Line previous to first renumbered line is greater than or equal to first new line number.	Invalid line number range.	Issue message IKJ52503I and return to caller with return code of 4.

Table 48. IKJEBESE Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEUT Return code 4	End of data set on read.	Set return code 8, return.
IKJEBEUT Return code 8	I/O error on utility data set.	Set return code 12, return.

Table 49. IKJEBEBO Error and Exceptional Conditions

Indication	Condition	Action
Return code of 8 from IKJEBEUT	I/O error in reading utility data set	Set return code of 12; return to IKJEBEMA.
Return code of 4 from IKJEBEUT	Utility data set is empty	Set return code of 0; set * to zero; put out message IKJ52501I; return to IKJEBEMA.

Table 50. IKJEBECH, IKJEBECG, IKJEBECN Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJEBEUT Return code = 4	Data set empty.	Issue message IKJ52501I and terminate with return code of 8.
IKJEBEUT Return code = 8	I/O error in Edit data set.	Set return code of 12 and terminate.
IKJEBEUT Return code = 12	Line number exceeds sequence field length.	Terminate with return code of 8.
IKJPUTL Return code = 4	Attention issued while in IKJPUTL.	Terminate.
IKJPUTL Return code = 16	Not enough core available while in IKJPUTL	Issue message IKJ52312I and terminate with return code of 8.
IKJPUTL Return code not = 0,4,16	PUTLINE not successful.	Issue message IKJ52313I and terminate with return code of 8.
IKJGETL Return code = 8	Attention issued while in IKJGETL.	Terminate.
IKJGETL Return code = 16 return code of 8.	End of data reached while IKJGETL reading.	Issue message IKJ52313I and terminate with
IKJGETL Return code = 24	Not enough core available while in IKJGETL.	Issue message IKJ52312I and terminate with return code of 8.
IKJGETL Return code not = 0,4,8,16,24	Getline not successful.	Issue message IKJ52313I and terminate. return code of 8.
IKJPARS Return code = 4	Required prompting could not be done by IKJPARS.	Terminate with return code of 8.
IKJPARS Return code = 8	Attention issued while in IKJPARS.	Terminate with return code of 0.
IKJPARS Return code = 16	Not enough core available while in IKJPARS	Issue message IKJ52312I and terminate with return code of 8.
IKJPARS Return code not = 0,4,8,16	Parse not successful.	Issue message IKJ52313I and terminate with return code od 8.
No operands and UPTNPRM = 1	NOPROMPT user and required operands not specified.	Issue message IKJ52559I and terminate with return code of 8.
IKJEBEUT Return code = 4	Specified line number not in Edit data set.	Issue message IKJ52504I and terminate with return code of 8.

(Part 1 of 2)

Table 50. IKJEBECH, IKJEBECG, IKJEBECN Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
IKJEBEUT Return code = 4	No line numbers in specified range in Edit data set.	Issue messages IKJ52504I and IKJ52500I and terminate.
IKJEBESE Return code of X'08'	Text not found.	Issue message IKJ52506I and terminate.
IKJEBESE Return code of X'0C'	I/O error in Edit data set.	Set return code of 12 and terminate.
IKJEBELE Return code = 4 and CANONUM = 0 or CADSCODE not = CATEXT	Changed line exceeds maximum record size and not NONUM TEXT.	Issue message IKJ52507I and continue processing.
IKJEBELE Return code = 4 CADSCODE = CATEXT	Changed lines exceeds maximum record size and NONUM text.	Issue message IKJ52558I and continue processing.
IKJEBELE Return code = 0 after return code = 4	All overflow lines created.	End line processing.
LANGPRCR Return code = 16	Not enough core while in LANGPRCR.	Issue message IKJ52312I and terminate with return code of 8.
LANGPRCR Return code = 8, 12	Syntax error in changed line with return code of 8.	Issue message IKJ52313I and terminate.
LANGPRCR Return code = 20	Internal error in LANGPRCR.	Set return code of 16 and terminate.
IKJEBEUT Return code = 4 after return code = 0	Some lines changed, but not more lines in range specified.	Issue message IKJ52500I and terminate.

(Part 2 of 2)

All terminations are with a return code of 0 except where indicated differently.

Table 51. IKJEBEDE Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS Return code = 4	Parse unable to prompt.	Return to IKJEBEMA, with return code of 8.
IKJPARS Return code = 8	PARSE interrupted by attention.	Return to IKJEBEMA, with return code 0.
IKJPARS Return code = 12	PARSE received invalid parameters.	Issue message IKJ52313I, return with return code of 8.
IKJPARS Return code = 16	PARSE unable to get Storage.	Issue message IKJ52312I, return with return code of 8.
IKJEBEUT Return code = 8	Utility data set I/O error.	Return to IKJEBEMA with return code = 12.
2nd number less than 1st number	Invalid line number range.	Issue message IKJ52503I, Return with return code of 8.
Line number entered on subcommand	Data set non-line numbered	Issue message IKJ52502I, Return with return code of 8.
IKJEBEUT Return code = 4 (attempting to delete first specified line)	Line not found.	Issue message IKJ52504I, return with return code of 8.
IKJEBEUT Return code = 4 on delete current operation (after successful read)	Unable to delete line after line read successfully.	Issue message IKJ52313I, return with return code of 8.
IKJEBEUT Return code = 4 on read first when first line of data set was deleted	Data set is empty.	Issue message IKJ52501I, set current line pointer to zero.
IKJEBEUT Return code = 0 on read first when first line of data set was deleted	Deleted line, or lines, were at top of data set.	Issue IKJ52505I, set current line pointer to 0.
LANGPRCR Return code = 8 or 12	Call "LANGPRCR" to delete lines from reverse Polish-notated data set.	Issue message IKJ52313I, return with return code of 8.

(Part 1 of 2)

Table 51. IKJEBEDE Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
LANGPRCR Return code = 16	Not enough storage.	Issue message IKJ52312I, return with return code of 8.
LANGPRCR Return code = 4, CASCANSW=1,	LANGPRCR failed to put out message.	Issue message IKJ52313I and return to IKJEBEMA with return code of 8.
LANGPRCR Return code = 20	Call "LANGPRCR" to delete lines from reverse Polish-notated data set.	Return to IKJEBEMA with return code of 16.

(Part 2 of 2)

Table 52. IKJEBEDO Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Command incomplete, unable to prompt.	Set return code 8; return to IKJEBEMA.
IKJPARS Return code = 8	Processing interrupted by attention.	Set return code 0; return to IKJEBEMA.
IKJPARS Return code = 12	Invalid parameters.	Issue message IKJ52313I; set return code of 8; return to IKJEBEMA.
IKJPARS Return code = 16	No space available.	Issue message IKJ52312I; set return code of 8; release PARSE PDL core if required, and return to IKJEBEMA.
IKJEBEUT Return code = 4 when reading last record	Utility data set is empty.	Put out message IKJ52501I; set return code to 0; set * to 0; release PARSE PDL storage, if required, and return to IKJEBEMA.
IKJEBEUT Return code = 8 after reading last record	I/O error in reading utility data set.	Set return code to 12; release PARSE PDL storage, if required, and return to IKJEBEMA.
IKJEBEUT Return code = 4 after reading next record	End of data set.	Issue message IKJ52500I; set * to key of last line referenced; turn 'the line to be verified switch' in comm. area on.

Table 53. IKJEBEEN Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEMS Return code 8 or 12 processing.	Message not put by PUTLINE.	Issue message IKJ52313I, continue exit
IKJEBEMS Return code 16	Message not put by PUTLINE.	Issue message IKJ52312I, continue exit processing.
PUTGET Return code 16, 20 or 24	Prompt not processed by PUTGET.	Issue message IKJ52313I, continue exit processing.
PUTGET Return code 28	Prompt not processed by PUTGET.	Issue message IKJ52312I, continue exit processing.
IKJEBESA Return code 12	Return from IKJEBESA (SAVE subcommand).	Continue exit processing.
IKJSCAN Return code not 0	Return from IKJSCAN.	Issue message IKJ52313I, continue exit processing.
IKJEBESA Return code 8	Data set not saved.	Re-issue prompt message and continue processing based on user response.

Table 54. IKJEBEFI Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt for missing operands on command.	Exit, return code 8.
IKJPARS Return code = 8	Attention issued during PARSE.	Exit, return code 0.
IKJPARS Return code = 12	Invalid parms to PARSE	Issue message IKJ52313I, exit, record code 0.
IKJPARS Return code = 16	Not enough core for PARSE.	Issue message IKJ52312I, exit return code 8.
IKJEBEUT Return code = 4 on first read	Empty utility data set.	Issue message IKJ52501I, exit return code 0.
IKJEBEUT Return code = 4 subsequent reads	End of data set.	Issue message IKJ52506I.
IKJEBEUT Return code = 8 on any read	I/O error in reading utility data set.	Exit, return code 12.
IKJEBESE Return code = 8	String not found by IKJEBESE	Issue message IKJ52506I, exit, return code 0.
IKJEBESE Return code = 16	I/O error on utility data set.	Exit, return code 12.

Table 55. IKJEBEFO Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt.	Return to IKJEBEMA with return code 8.
IKJPARS Return code = 8	Attention ECB posted.	Return to IKJEBEMA with return code 0.
IKJPARS Return code = 12	Invalid PARSE parameters.	Issue message IKJ52313I; return with return code of 8.
IKJPARS Return code = 16	Conditional GETMAIN failure.	Issue message IKJ52312I; set return code of 8, and return to IKJEBEMA.
IKJEBEDA Return code ≠ 8 in allocating a QSAM data set	DA fails to allocate a QSAM data set.	Set return code of 8; release PARSE PDL core if required and return to IKJEBEMA.
IKJEBEFC Return code = 4	Utility data set is empty.	Issue message IKJ52501I; set return code of 0; release PARSE PDL core if required; return to IKJEBEMA.
IKJEBEFC Return code = 8	I/O error in QSAM data set.	Set return code of 8; release PARSE PDL core if required; return to IKJEBEMA.
IKJEBEFC Return code = 12	I/O error in reading utility data set.	Set return code of 12; release PARSE PDL core if required; return to IKJEBEMA.
IKJEBEDA Return code ≠ 8 in unallocating a QSAM data set	DA fails to unallocate a QSAM data set.	Set return code of 8; release PARSE PDL core if required and return to IKJEBEMA.
IKJEBEUT Return code = 4 when reading 1st record.	Utility data set is empty.	Issue message IKJ52501I set return code of zero; release PARSE PDL core if required; return to IKJEBEMA.

(Part 1 of 2)

Table 55. IKJEBEFO Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
IKJEBEUT Return code = 4 when reading current line or following lines.	Record not found	Issue message IKJ52504I; set return code of 8; release PARSE PDL core if required; return to IKJEBEMA.
IKJEBEUT Return code = 8	I/O error in reading utility data set.	Set return code of 12; release PARSE PDL core if required; return to IKJEBEMA.
IKJEBECI Return code 8	Error in IKJEBECI or in FORMAT command.	Set return code of 8; release PARSE PDL storage, if required; return to IKJEBEMA.

(Part 2 of 2)

Table 56. IKJEBEIP and IKJEBEIM Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS Return code = 16	No core for PARSE.	Put out msg. IKJ52312I return to IKJEBEMA with return code of 8.
IKJPARS Return code = 4	No-prompt user.	Return to IKJEBEMA with return code of 8.
IKJPARS Return code = 8	Attention issued in PARSE.	Return to IKJEBEMA with return code 0.
IKJPARS Return code = 12	Invalid PARSE parameters.	Put out msg. IKJ52313I return to IKJEBEMA with return code of 8.
IKJPUTL Return code =16	No core for PUTLINE.	Put out msg IKJ52312I return to IKJEBEMA with return code of 8.
IKJPUTL Return code = 12	Invalid parameters for PUTLINE.	Put out msg. IKJ52313I return to IKJEBEMA with return code of 8.
IKJPUTL Return code = 4	Attention received during PUTLINE.	Return to IKJEBEMA.
IKJGETL Return code = 24	No core available for GETLINE.	Put out msg. IKJ52312I return to IKJEBEMA with return code of 8.
IKJGETL Return code = 8	Attention received during GETLINE.	Return to IKJEBEMA.
IKJGETL Return code not 0,24, 4,8	Internal error in GETLINE.	Put out msg. IKJ52313I return to IKJEBEMA with return code of 8.

(Part 1 of 2)

Table 56. IKJEBEIP and IKJEBEIM Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
IKJEBELE Return code = 4	Input line from terminal too long for data set.	Truncate line and put out msg. IKJ52507I.
IKJEBEUT Return code = 8	I/O error on utility data set.	Return to IKJEBEMA. with return code of 12.
GETMAIN Return code not equal 0	No core for syntax checker buffers.	Put out msg. IKJ52312I and turn off syntax checking.
LANGPRCR Return code = 4	Syntax error in line and statement completely scanned.	If msg. present put to terminal, except for IPLI or BASIC; return to IKJEBEMA.
LANGPRCR Return code = 8	Syntax error and statement not completely scanned.	If msg. present, put to terminal and re-call checker to complete SCAN; if IPLI or BASIC, issue msg. IKJ52313I; return to IKJEBEMA with return code of 8.
LANGPRCR Return code = 20	Internal error in syntax checker.	Set return code of 16 for IKJEBEMA and return to IKJEBEMA.
LANGPRCR Return code = 16	No core for syntax checker.	Put out msg. IKJ52312I and return to IKJEBEMA with return code of 8.

(Part 2 of 2)

Table 57. IKJEBEIS Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code =16	Not enough core for PARSE.	Put out msg. IKJ52312I return to IKJEBEMA with return code of 8.
IKJPARS Return code = 4	Unable to prompt.	Return to IKJEBEMA with return code of 8.
IKJPARS Return code = 8	Attention entered in PARSE.	Return to IKJEBEMA, with return code 0.
IKJPARS Return code =12	Internal error from PARSE.	Put out msg. IKJ52313I return to IKJEBEMA with return code of 8.
IKJEBEUT Return code =0 Key of record = Key of record to be inserted.	No room in data set to insert line.	(a) If data set is numbered, put out IKJ52561I and return to IKJEBEMA with return code of 8. (b) If data set is not numbered, make room and insert the line, return to IKJEBEMA.
IKJEBELE Return code =4	Line from terminal too long to fit in data set.	Truncate line and put out msg. IKJ52507I return to IKJEBEMA.
LANGPRCR Return code =4	Line contains syntax error and is completely scanned.	Return to IKJEBEMA. (Set return code of 8, if CASCANSW=B'1'.)
LANGPRCR Return code =8	Line contains syntax error and is not completely scanned.	Issue message IKJ52313I and return to IKJEBEMA with return code of 8.
LANGPRCR Return code =16	Not enough core for syntax checker.	Put out msg. IKJ52312I return to IKJEBEMA with return code of 8.
LANGPRCR Return code =20	Internal error in syntax checker.	Set return code of 16 for IKJEBEMA and return control.
IKJEBEUT Return code =8	I/O error on utility data set.	Return to IKJEBEMA with return code of 12.

Table 58. IKJEBELI Error and Exceptional Conditions

Indication	Condition	Action
PARSE Return code = 4	No prompt user.	Return to IKJEBEMA with return code of 8.
PARSE Return code = 8	Attention entered.	Return to IKJEBEMA.
PARSE Return code = 12	Invalid parameters for IKJPARS.	Issue IKJ52313I return to IKJEBEMA with return code of 8.
PARSE Return code = 16	Not enough main storage.	Issue IKJ52312I return to IKJEBEMA with return code of 8.
IKJEBELE Return code = 4	Input text length greater than LRECL.	Issue IKJ52507I, truncate line and continue.
IKJEBEUT Return code = 8	I/O error on utility data set.	Post return code 12 and return to IKJEBEMA.
IKJEBEUT Return code = 12	Line number too large to fit in record.	Return to IKJEBEMA.
LANGPRCR Return code = 4	Line contains a syntax error and line is completely scanned.	Return to IKJEBEMA.
LANGPRCR Return code = 8	Line contains a syntax error and is not completely scanned.	Issue message IKJ52313I and return to IKJEBEMA with return code of 8.
IKJEBEUT Return code = 4	Line to be deleted does not exist.	Issue msg. IKJ52504I, return to IKJEBEMA.
LANGPRCR Return code = 20	Internal error.	Set return code 16 and return to IKJEBEMA.
IKJEBEUT Return code = 4 on read first	Data set is empty.	Issue msg. IKJ52501I, set current line pointer to zero and return to IKJEBEMA.
IKJEBEUT Return code = 0 on read first	Deleted line was at top of data set.	Issue msg. IKJ52505I, set current line pointer to zero and return to IKJEBEMA.
LANGPRCR Return code = 16	Not enough main storage for syntax checker.	Issue msg. IKJ52312I, return to IKJEBEMA with return code of 8.

Table 59. IKJEBELT Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS Return code = 4	PARSE unable to prompt.	Set return code 8, return to IKJEBEMA.
IKJPARS Return code = 8	ATTN issued.	Set return code 0, return to IKJEBEMA.
IKJPARS Return code = 12	INVALID parameters sent to PARSE.	Issue msg. IKJ52313I, set return code 8, return to IKJEBEMA.
IKJPARS Return code = 16	Not enough core for PARSING.	Issue msg. IKJ52312I, set return code 8, return to IKJEBEMA.
IKJEBEUT Return code = 8	I/O error on utility data set.	Release PDL core if PARSE was called, return to IKJEBEMA with return code 12.
IKJEBEUT Return code = 4 after entry with option code X'04'	Data set is empty.	Issue msg. IKJ52501I, Free PDL if PARSE was called, return to IKJEBEMA with return code 0.
CANONUM bit in communication area is on	Line no. specified, data set not line-numbered.	Issue msg. IKJ52502I, free PDL, return to IKJEBEMA with return code 8.
linenum1 > linenum2	Invalid line number range.	Issue msg.IKJ52503I, free PDL, return to IKJEBEMA with return code 8.
IKJEBEUT Return code = 4 after entry with option code X'02'.	End of data set encountered.	Issue msg. IKJ52500I free PDL if PARSE was called, return to IKJEBEMA with return code 0.
IKJEBEUT Return code = 4 after entry with option code X'00'	Specified line not found, or current line not found and * ≠ 0.	Issue msg. IKJ52504I, free PDL, return to IKJEBEMA with return code 8.
PUTLINE Return code = 4	ATTN issued.	Free PDL if PARSE WAS called, return to IKJEBEMA with return code 0.
IKJPUTL Return code = 16	Insufficient core for PUTLINE.	Issue msg. IKJ52312I, free PDL if PARSE was called, return to IKJEMEMA with return code 8.

(Part 1 of 2)

Table 59. IKJEBELT Error and Exceptional Condition (Part 2 of 2)

Indication	Condition	Action
IKJPUTL Return code = 12	Invalid parameters sent to PUTLINE.	Issue msg. IKJ52313I, free PDL if PARSE was called, return to IKJEBEMA with return code 8.
IKJPUTL Return code = 8	Output not sent by PUTLINE.	Issues msg. IKJ52313I and returns to IKJEBEMA with return code of 8.

(Part 2 of 2)

Table 60. IKJEBEME Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS return code = 4	Unable to prompt.	Set return code to 8; return to IKJEBEMA.
IKJPARS return code = 8	Attention issued.	Return to IKJEBEMA with return code of 0.
IKJPARS return code = 12	Invalid parameters.	Issue message IKJ52313I; set return code to 8; return to IKJEBEMA.
IKJPARS return code = 16	No core available.	Issue message IKJ52312I; set return code to 8; return to IKJEBEMA
IKJEBEFC return code = 4	Utility data set is empty.	Issue message IKJ52553I; free data set; release PDL core; set return code to 0; return to IKJEBEMA.
IKJEBEFC return code = 8	QSAM error.	Free data set; release PDL core; set return code to 8; return to IKJEBEMA.
IKJEBEFC return code = 12	I/O error on utility data set.	Free data set; release PDL core; set return code to 12; return to IKJEBEMA.
IKJEBECI return code = 8	Merge unsuccessful.	Release PDL core; set return code to 8; free data set; return to IKJEBEMA.
IKJEBECO return code = 4	QSAM data set empty.	Issue message IKJ52553I, delete new utility data set; free data set; release PDL core, set return code to 0; return to IKJEBEMA.

(Part 1 of 2)

Table 60. IKJEBEME Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
IKJEBECO return code = 8	Unsuccessful copy.	Free data set; release PDL core; set return code to 8; return to IKJEBEMA.
IKJEBEMR return code = 12	I/O error in new utility data set.	Free imtermediate QSAM data set; release PDL storage; set return code of 8; return to IKJEBEMA.
IKJEBEMR return code = 8	Syntax checker not operational.	Release PDL core; return code to 16 return to IKJEBEMA.
IKJEBEMR return code = 4	Unsuccessful retranslation.	Release PDL core; RC to 0; return to IKJEBEMA.
IKJPTGT return code = 8	Attention issued.	Release PDL core; set return code to 0; return to IKJEBEMA.
IKJPTGT return code = 12	Second level message present.	Set ECTMSGF switch on; re-issue PUTGET.
IKJPTGT return code = 16	NOWAIT for TPUT and no line found.	Issue message IKJ52313I; release PDL core; set return code to 8; return to IKJEBEMA.
IKJPTGT return code = 20	NOWAIT for TGET and no line found.	Same as return code 16 from PUTGET.
IKJPTGT return code = 24	Invalid parameters.	Same as return code 16 from PUTGET.
IKJPTGT return code = 28	No core available.	Issue message IKJ52312I; release PDL core; set return code to 8; return to IKJEBEMA.
IKJEBEUT return code = 4	Record not found.	Convert relative record number to decimal; continue processing.
IKJEBEUT return code = 8	I/O error on utility data set.	Release PDL core; set return code to 12; return to IKJEBEMA.
IKJEBEDA return code = 8	IKJDAIR not successful.	Release PDL core; set return code to 8; return to IKJEBEMA.

(Part 2 of 2)

Table 61. IKJEBERE Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt for missing operand.	Exit, return code 8.
IKJPARS return code = 8	Attention issued.	Exit, return code 0.
IKJPARS return code = 12	Invalid parms to PARSE.	Issue msg. IKJ52313I exit, return code 8.
IKJPARS return code = 16	Not enough core for PARSE.	Issue msg. IKJ52312I exit, return code 8.
IKJEBEUI return code ≠ 0	Unable to allocate new utility data set.	Release PARSE PDL if present and exit, return code 8.
IKJEBEUT return code = 4 read first rcd	Empty data set.	Issue msg. IKJ52501I exit, return code 0
IKJEBEUT return code = 8	I/O error on utility data set.	Unallocate new utility data set, exit, return code 12.
IKJEBEUT return code = 12	Invalid line number.	Unallocate new utility data set, exit, return code 8.
IKJEBERN return code = 4	BASIC RENUM unsuccessful.	Unallocate new utility data set, exit, return code 8.
IKJEBERN return code = 8	I/O error in new utility data set.	Unallocate new utility data set; exit with return code of 12.
CANONUM = 1; user specified "old line number".	Data set is not line-numbered.	Issue message IKJ52502I; exit with return code of 8.
IKJEBEMR return code = 4	BASIC/IPLI RENUM unsuccessful in reconstructing Polish data set.	Unallocate new utility DCB; set return code of 8 and return to IKJEBEMA.
IKJEBEMR return code = 8	BASIC/IPLI syntax checker not operational.	Set return code of 16 and return to IKJEBEMA.
IKJEBEUT return code = 4 on reading "old line number"	Line not found on read.	Msg. IKJ52504I; exit, after unallocating new utility data set, set return code of 8.

Table 62. IKJEBERU Error and Exceptional Conditions (Part 1 of 2)

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt for missing operand.	Exit, return code 8.
IKJPARS Return code = 8	Attention issued.	Exit, return code 0.
IKJPARS Return code = 12	Invalid parms to PARSE.	Issue msg. IKJ52313I exit, return code 8.
IKJPARS Return code = 16	Not enough core for PARSE.	Issue, msg. IKJ52312I exit, return code 8.
IKJEBEUT Return code = 4 on first read	Empty GOFORT dataset.	Issue msg. IKJ52501I exit, return code = 0.
IKJEBEUT Return code = 8 on any read	I/O error on utility ds for GOFORT.	Exit, return code 12 freemain core for dataset.
IKJEBEDA Return code = 8	Unable to allocate RUN ds.	Exit, return code 8.
IKJEBEFC Return code = 4	Empty dataset.	Exit, return code = 0 after msg. IKJ52501I
IKJEBEFC Return code = 8	I/O error on utility ds.	Exit, return code 12 after link to IKJEBEDA to unallocate RUN dataset.
IKJEBEDA Return code = 0	Unable to unallocate RUN dataset.	Exit, return code 8 (note: if return code 8 was returned from IKJEBEFC return code from IKJEBERU is 12 in any case, no matter what return code comes from IKJEBEDA.)

(Part 1 of 2)

Table 62. IKJEBERU Error and Exceptional Conditions (Part 2 of 2)

Indication	Condition	Action
STACK Return code = 4	Invalid parms to PARSE.	Issue msg. IKJ52313I and exit, return code 8.
IKJEBECI Return code = 0 (GOFORT data set)	Error in IKJEBECI or in attached command.	Delete data set from STACK and exit, return code 8.
LANGPRCR Return code = 4 BASIC or IPLI	Error in LANGPRCR.	Return to IKJEBEMA with return code 8.
LANGPRCR Return code = 8 or 12	Error in LANGPRCR.	Issue message IKJ52313I; Return to IKJEBEMA with return code 8.
LANGPRCR Return code = 16	Insufficient storage.	Issue message IKJ52312I; Return to IKJEBEMA with return code 8.
IKJEBECI Return code = 8 (Data set is not GOFORT)	Error in IKJEBECI or in an attached command.	Return with return code of 8.

(Part 2 of 2)

Table 63. IKJEBESA Error and Exceptional Conditions (Part 1 of 5)

Indication	Condition	Action
IKJPARS Return code = 4	IKJPARS unable to prompt.	Set return code 8, return to caller.
IKJPARS Return code = 8	ATTN issued during PARSING.	Set return code 0, return to caller.
IKJPARS Return code = 12	Invalid parameter sent to IKJPARS.	Issue message IKJ52313I, Set return code 8, return to caller.
IKJPARS Return code = 16	Not enough core for PARSING.	Issue msg. IKJ52312I set return code 8, return to caller.
IKJDFLT Return code = 8	Length of dsname becomes > 44 characters when it is fully-qualified.	Issue msg. IKJ52310I, set return code 8, return to caller.
IKJDFLT Return code = 4	Prompting error in IKJDFLT.	Issue msg. IKJ52313I, set return code 8, return to caller.
IKJDFLT Return code = 28	Invalid parameters sent to IKJDFLT.	Issue msg. IKJ52313I, set return code 8, return to caller.
IKJDFLT Return code = 32	IKJDFLT required prompting.	Issue msg. IKJ52313I, set return code 8, return to caller.
IKJDFLT Return code = 12	Catalog I/O error, dsname syntax error, or unavailable catalog data set, in IKJDFLT.	Msg. issued by IKJDFLT; IKJEBESA sets return code 0 and returns to caller.
IKJDFLT Return code = 16	Data set exists at another index level.	Issue msg. IKJ52316I, set return code 8, return to caller.
IKJDFLT Return code = 24	ATTN issued during defaulting of dsname.	Set return code 0; return to caller.
SAVE dsname = EDIT dsname. member name specified on SAVE but not on EDIT	Extraneous member name specified.	Issue msg. IKJ52311I, proceed as if no operands were entered on subcommand.
IKJDAIR Return code = 4	Invalid parameters sent to IKJDAIR.	Issue msg. IKJ52313I, unallocate data set if necessary, set return code 8; return to caller.
IKJDAIR Return code = 16	No TIOT entries available for allocation.	Issue msg. IKJ52301I, set return code 8, return to caller.

(Part 1 of 5)

Table 63. IKJEBESA Error and Exceptional Conditions (Part 2 of 5)

Indication	Condition	Action
IKJDAIR Return code = 8 Catalog return code (CTRC) = X'04'	Catalog management error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52302I, set return code 8, return to caller.
IKJDAIR Return code = 8 Catalog return code CTRC = X'14'	Catalog management error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52306I, set return code 8, return to caller.
IKJDAIR Return code = 8 Catalog return code CTRC = none of above. (IKJDFLT was not called)	Catalog management error after IKJDAIR is called to allocate a data set as new.	Issue message IKJM339I, use PUTGET to issue msg. IKJ52340A and get user input.
IKJDAIR Return code = 8 CTRC = none of above	Data set not allocated as new; data set name not in catalog.	Issue msg. IKJ52304I; set return code of 8; return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code (DARC) = X'6708'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52302I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'0204'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52312I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'0214'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52318I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'021C'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52318I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'0210'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52306I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'0218'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52303I, set return code 8, return to caller.

(Part 2 of 5)

Table 63. IKJEBESA Error and Exceptional Conditions (Part 3 of 5)

Indication	Conditions	Action
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'0404'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52317I, set return code 8, return to caller.
IKJDAIR Return code = 12 Dynamic allocation Return code DARC = X'041C'	Dynamic allocation error after IKJDAIR is called to allocate a data set as new.	Issue msg. IKJ52314I, set return code 8, return to caller.
IKJDAIR Return code = 12, no error message exists which corresponds to the setting of the DARC field (IKJDFLT was not called.)	Data set not allocated as new, no member name specified.	Issue message IKJM339I, use PUTGET to issue msg. IKJ52340A and get user input.
IKJDAIR Return code = 12 DARC = none of above (IKJDFLT was called.)	Data set not allocated as new; data set name not in catalog	Issue msg. IKJ52304I; set return code of 8; return to caller.
IKJPTGT Return code = 8	ATTN issued during PUTGET.	Set return code 0, return to caller.
IKJPTGT Return code = 12	PUTGET unable to prompt.	Unallocate data set if necessary, set return code 0, return to caller.
IKJPTGT Return code = 16	PUTGET unable to send message.	Issue msg. IKJ52313I, unallocate data set if necessary, set return code 0, return to caller.
IKJPTGT Return code = 20	PUTGET unable to obtain user input.	Issue msg. IKJ52313I, unallocate data set if necessary, set return code 0, return to caller.
IKJPTGT Return code = 24	Invalid parameters sent to PUTGET.	Issue msg. IKJ52313I, unallocate data set if necessary set return code 0, return to caller.

(Part 3 of 5)

Table 63. IKJEBESA Error and Exceptional Conditions (Part 4 of 5)

Indication	Conditions	Action
IKJPTGT Return code = 28	Insufficient core for PUTGET.	Issue msg. IKJ52312I, unallocate data set if necessary, set return code 0, return to caller.
IKJDAIR Return code 8 or 12	Catalog management or dynamic allocation error after calling IKJDAIR to allocate an old data set.	Issue message as described above for error during allocation as new. If no message corresponds to the CTRC or DARC setting, issue message IKJ52304I. Set return code 8, return to caller.
DA08DSO field in DAIR parameter block indicates partitioned organization	IKJDAIR was called to allocate an old sequential data set; allocation successful, but DSORG = PO.	Unallocate data set, continue processing as a partitioned data set, using 'TEMPNAME' as member name.
DA08DSO field in DAIR parameter block is non-zero, but does not indicate partitioned or sequential organization	An old data set is allocated, and DSORG is set but is not PS and not PO.	Issue message IKJ52330I, unallocate data set, set return code 8, return to caller.
IKJDAIR Return code 8 or 12 entry code X'18'	Catalog management or dynamic allocation error during unallocation of data set.	Issue message IKJ52424I, Issue 'SAVED' mode message if IKJEBEFC copied records into the data set, set return code 8, return to caller.
DA08DSO field in DAIR parameter block indicates sequential organization	IKJDAIR was called to allocate an old partitioned data set; allocation successful, but DSORG = PS.	Issue message IKJ52311I, unallocate data set, continue processing data set as sequential.
DEVTYPE svc Non-zero return code	Unable to obtain track length of output device.	Issue message IKJ52313I, unallocate data set, set return code 8, return to caller.
Blocksize greater than device track length	EDIT blocksize too large.	Issue message IKJ52554I, unallocate data set, set return code 8, return to caller.
OBTAIN svc Non-zero return code	Unable to read DSCB into core.	Issue message IKJ52313, unallocate data set, return code to 8, return to caller.

(Part 4 of 5)

Table 63. IKJEBESA Error and Exceptional Conditions (Part 5 of 5)

Indication	Conditions	Action
In the DS1RECFM field of the DSCB, the "S" -bit is on.	A PDS has standard blocks, or spanned records.	Issue message IKJ52331I, unallocate data set, set return code 8, return to caller.
SAVE data set is a PDS, and its record format, blocksize, or logical record length is not compatible with that of the utility data set.	SAVE data set is incompatible with utility data set.	Issue msg. IKJ52552I, unallocate data set, set return code 8, return to caller.
BLDL svc return code = 8	I/O error on SAVE data set.	Issue msg. IKJ52313I, unallocate data set, set return code 8, return to caller.
BLDL Return code = 0	The specified member in the SAVE data set exists, and the SAVE dsname and member are not both the same as the EDIT dsname and member.	Issue message IKJ52339I, use PUTGET to issue msg. IKJ52341A and get user input.
BLDL Return code = 0	The specified member in the SAVE data set exists, the SAVE and EDIT dsnames & member names are identical, and NEW was specified on the EDIT command.	Issue message IKJ52339I, use PUTGET to issue msg. IKJ52341A and get user input.
IKJEBEFC Return code = 4	Utility data set is empty.	Issue msg. IKJ52553I, unallocate data set if necessary, set return code 0, return to caller.
IKJEBEFC Return code = 8	QSAM error in copying records from utility to SAVE data set.	Unallocate data set if necessary, set return code 8, return to caller.
IKJEBEFC Return code = 12	I/O error on utility data set.	Unallocate data set if necessary, set return code 12, return to caller.

Table 64. IKJEBESC, IKJEBESN Error and Exceptional Conditions (Part 1 of 3)

Indication	Condition	Action
CASCAN switch in processor data table is off.	Syntax checking is invalid for data set type.	Issue msg. IKJ52363I, set return code 0, return to IKJEBEMA.
Syntax checker name (CASYNAME) in processor data table is set to X'0000000000000000'or X'FFFFFFFFFFFFFFFF'.	Syntax checker is not in system or is not operational.	Issue msg.IKJ52364I set return code 12, if input source is an in-storage procedure; otherwise, clear input queue, set return code of 0, and return to IKJEBEMA.
IKJPARS Return code = 12.	Invalid parameters sent to IKJPARS.	Issue msg.IKJ52313I, set return code 8, return to caller.
IKJPARS return code = 16	Insufficient core for parsing.	Issue msg.IKJ52312I, set return code 8, return to caller.
IKJPARS return code 8.	Attention issued.	Return to caller with return code 0.
IKJPARS Return code = 4.	Unable to prompt.	Return to caller with code 8.
Syntax checker return code = 20 to IKJEBESN	Syntax checker has become inoperative.	Set return code 16 and return to IKJEBEMA.
Syntax checker Non-zero return code	Syntax checker unsuccessful on an initial entry.	Call syntax checker for final entry, delete syntax checker, set pointer to checker (CAPTCHK) to 0; set syntax checker name to X'FFFFFFFFFFFFFFFF' to indicate checker not operative, turn off scan & prompt switches (CASCANSW & CAPROMPT), set return code 16, turn on syntax checker recovery bit (CASCRC20) so that controller will turn it off, and return to IKJEBEMA.
IKJEBEUT Return code = 8 to IKJEBESC	I/O error in utility data set.	Turn off input bit (INPUTCD) in options word, release PDL core if necessary, return to IKJEBEMA with return code 12.

(Part 1 of 3)

Table 64. IKJEBESC, IKJEBESN Error and Exceptional Conditions (Part 2 of 3)

Indication	Condition	Action
CANONUM switch in communication area is on.	Line number specified, but data set not line-numbered.	Issue msg.IKJ52502I, release PDL core if necessary, return to IKJEBEMA with return code 0
CASCRC20 bit in communication area is on, and checker is again operational.	Syntax checker failed during line-scanning, but was successfully deleted, re-loaded, and re-called for initial entry (and for line-translation, if a run-time data set is required).	Turn off CASCRC20 bit, issue msg.IKJ52367I, release PDL core if necessary, and return to IKJEBEMA with return code 0.
GETMAIN SVC Non-zero return code	Unable to obtain buffer to send lines for syntax-checking.	Issue msg.IKJ52312I, delete syntax checker if a run-time data set is not required, and if input scanning is not in effect, free PDL return to IKJEBEMA with return code 0.
IKJEBEUT Return code = 4 with entry code X'04'.	Empty data set.	Issue msg.IKJ52501I, free buffer reset 'SCAN/NOSCAN' bit in options word, free PDL if necessary, delete syntax checker if it doesn't keep a run-time data set, and if input scanning is not in effect, return to IKJEBEMA with return code 0.
linenum1>linenum2	Invalid line no. range specified.	Issue msg.IKJ52503I, continue as when data set is empty, return with code of 8.
IKJEBEUT Return code = 8 to IKJEBESN	I/O error on utility data set.	Free buffer, reset 'SCAN/NOSCAN' bit in options word, free PDL if necessary, delete syntax checker unless it keeps a run-time data set or if input scanning is not in effect. return to IKJEBEMA with return code 12.

(Part 2 of 3)

Table 64. IKJEBESC, IKJEBESN Error and Exceptional Conditions (Part 3 of 3)

Indication	Condition	Action
Syntax checker Return code = 8	Syntax error found, syntax checker has not completed scanning the whole buffer.	Call IKJEBEMS to send error msg. unless syntax checker has sent it (msg pointer=0). Recall syntax checker to continue scanning, unless ATTN has been issued.
Syntax checker Return code = 4	Syntax error found, syntax checker has completed scanning buffer.	Proceed as for syntax checker return code 8, except re-call syntax checker only if more lines remain to be syntax-checked.
Syntax checker return code = 12	Last statement in buffer may be incomplete.	If no lines remain to be syntax-checked, recall syntax-checker to diagnose the last statement. Otherwise chain a new buffer to the last one, or send a new buffer chain, and recall syntax-checker to continue scanning (turn on 'INCOMPST' bit in syntax checker work area to indicate that this entry follows a return code 12).

Table 65. IKJEBETA Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt for missing operands.	exit, return code 8.
IKJPARS Return code = 8	Attention issued during PARSE.	Exit, return code 0.
IKJPARS Return code = 12	Invalid parms to PARSE.	Issue msg. IKJ52313I, exit, return code 8.
IKJPARS Return code = 16	Not enough core for PARSE.	Issue msg. IKJ52312I, Exit, return code 8.
GETLINE Return code = 8	Attention issued.	Exit, return code 0.
GETLINE Return code = 12	NOWAIT spec, no input returned.	Issue msg. IKJ52313I, exit, return code 8.
GETLINE Return code = 16	End of data or procedure input.	Re-issue GETLINE.
GETLINE Return code = 20	Invalid parms to GETLINE	Issue msg. IKJ52313I, exit, return code 8.
GETLINE Return code = 24	Not enough core for GETLINE.	Issue msg. IKJ52312I, exit, return code 8.

Table 66. IKJEBETO Error and Exceptional Conditions

Indication	Condition	Action
Return code of 8 from IKJEBEUT	I/O error in reading utility data set	Set return code of 12; return to IKJEBEMA.
Return code of 4 from IKJEBEUT	Utility data set is empty	Set return code of 0; set * to zero; issue message IKJ52501I, return to IKJEBEMA.

Table 67. IKJEBEUP Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Command incomplete unable to prompt.	Set return code of 8; return to IKJEBEMA.
IKJPARS Return code = 8	Processing interrupted by attention; communications ECB posted.	Set return code of 0; and return to IKJEBEMA.
IKJPARS Return code = 12	Invalid parameters passed to IKJPARS.	Issue message IKJ52313I, set return code of 8; return to IKJEBEMA.
IKJPARS Return code = 16	No space available.	Issue message IKJ52312I set return code of 8; return to IKJEBEMA.
IKJEBEUT Return code = 4 after reading 1st record	Utility data set is empty.	Issue message IKJ52501I, set return code of 0; set * to 8; release PARSE PDL core if required and return to IKJEBEMA.
IKJEBEUT Return code = 8	I/O error in reading utility data set.	Set return code fo 12; release PARSE PDL core if required and return to IKJEBEMA.
IKJEBEUT Return code = 4 When reading records previous to current record.	The number of records current record is less than "count".	Issue message IKJ52505I; set * to key of last line referenced; turn CALNTOVF to 1.

Table 68. IKJEBEVE Error and Exceptional Conditions

Indication	Condition	Action
IKJPARS Return code = 4	Unable to prompt.	Set return code to 8; return to IKJEBEMA.
IKJPARS Return code = 8	Attention issued.	Set return code to 0; return to IKJEBEMA.
IKJPARS Return code = 12	Invalid parameters.	Put out message IKJ52313I; set return code to 8; return to IKJEBEMA.
IKJPARS Return code = 16	No core available.	Put out message IKJ52312I; set return code to 8; return to IKJEBEMA.

Table 69. IKJEBEUI Error and Exceptional Conditions

Indication	Condition	Action
GETMAIN Nonzero return code	Getmain for workarea and buffers unsuccessful.	Issue message IKJ52312I and terminate, return code 4.
Test DCBOFLGS after OPEN	OPEN of utility data set not successful.	Issue message IKJ52309I, unallocate utility data set, free dynamic core, and terminate, return code 12.
DAIR Nonzero return code	Utility data set not allocated.	Issue message as described below. Free dynamic core and terminate, return code 8
DAIR Return code = 8 CTRC field = X'0028'	Catalog I/O error.	Issue IKJ52304I.
DAIR Return code = 8 CTRC field not X'0028'	Catalog management error	Issue IKJ52304I with CTRC field.
DAIR Return code = 12 DARC field = X'0220'	Insufficient direct-access space available.	Issue IKJ52305I.
DAIR Return code = 12 DARC field = X'021C'	No unit available for allocation	Issue IKJ52318I.
DAIR Return code = 12 DARC field = X'0214'	Invalid unit in user attributes data set.	Issue IKJ52318I.
DAIR Return code = 12 and anything else in DARC	Other dynamic allocation error.	Issue IKJ52304I with DARC field.

Table 70. IKJEBEEX Error and Exceptional Conditions

Indication	Condition	Action
DAIR Nonzero return code	Utility data set unallocation not successful.	If return code is 12 decimal, issue message IKJ52424I. For any other nonzero return code issue message IKJ52313I, continue processing in either case.

Table 71. IKJEBEUT Error and Exceptional Conditions

Indication	Condition	Action
access method Return code = 4	Record not found.	Return with return code 4.
access method Return code greater than 4	I/O error.	Issue message IKJ52309I and return with return code 8.
Line number length obtained from field of communication area. Line number of record compared to table entry of same length.	Line number limit exceeded in WRITE operation with numbered data set.	Issue message IKJ52402I and return, return code 12.

Table 72. IKJEBEWR Error and Exceptional Conditions

Indication	Condition	Action
Error flag on in DCBSTATS field in utility workarea upon return from IKJEBELO	I/O error in searching for record having same key as record to be written.	Return with same return code as received from IKJEBELO.
Error flag on in DCBSTATS field in utility workarea upon return from IKJEBEDL	I/O error in deleting old record having same key as record to be written.	Return with same return code as received from IKJEBEDL.
IKJEBEAD Nonzero return code	Error in adding new block.	Return with same return code as received from IKJEBEAD.
IKJEBEWB Nonzero return code	Error in writing block.	Return with same return code as received from IKJEBEWB.
IKJEBEDU Nonzero return code	Error in updating directory block entry.	Return with same return code as received from IKJEBEWD.

Table 73. IKJEBERR Error and Exceptional Conditions

Indication	Condition	Action
IKJEBELO Nonzero return and error flag on in DCBSTATS field in utility workarea	I/O error in locating record.	Return with same return code as received from IKJEBELO.

Table 74. IKJEBEDR Error and Exceptional Conditions

Indication	Condition	Action
IKJEBELO Nonzero return code and error flag not set in DCBSTATS in utility workarea.	Record to be deleted does not exist in data set.	Return with return code 4.
IKJEBELO Nonzero return code error flag set on in DCBSTATS field	I/O error in locating record.	Return with same return code received from IKJEBELO.
IKJEBEDL Return code other than 0 or 8 error flag on in DCBSTATS field.	Error in deleting record, I/O error.	Return with same return code as received from IKJEBEDL.
IKJEBEDL Return code other than 0 or 8 error flag not set in DCBSTATS	Error other than I/O error in deleting record.	Return with return code 4.

Note: Return code 8 from IKJEBEDL indicates data set is empty and is not considered an exceptional condition in IKJEBEDR, which returns with return code 0.

Table 75. IKJEBELO Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEDS Return code not less than 8	I/O error in directory search or empty data set.	Return with same return code as received from IKJEBEDS.
IKJEBEDS Return code = 4	*Directory search indicates record not in data set.	Data block indicated by IKJEBEDS read in, then returns with return code 4.
IKJEBERB Nonzero return code	Error in reading data block.	Return with same return code as received from IKJEBERB.
Current reference key compared to key of each record in block without success.	*Record not found in data block.	Return with return code 4.

*Not an error condition.

Table 76. IKJEBEDL Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEWB Nonzero return code	I/O error in writing empty data block.	Return with same return code as received from IKJEBEWB.
IKJEBEDS Nonzero return code	Directory search not successful (occurs twice).	Return with same return code as received from IKJEBEDS.
Last level directory block has zero entries	*Data set empty after record removal.	Return with return code 8.
IKJEBERB Nonzero return code	I/O error in reading directory block.	Return with same return code as received from IKJEBERB.

*Not an error condition.

Table 77. IKJEBEAD Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEAS Nonzero return code	I/O error from buffer assignment.	Return with same return code as received from IKJEBEAS.
IKJEBEWB or IKJEBEWA Nonzero return code	I/O error in BSAM write of new block.	Return with same return code as received from IKJEBEWB or IKJEBEWA.
IKJEBERB Nonzero return code	Error in reading empty block into core.	Return with same return code as received from IKJEBERB.

Table 78. IKJEBEDU Error and Exceptional Conditions

Indication	Condition	Action
Error flag on in DCBSTATS field in utility workarea upon return from IKJEBEDS	I/O error in directory search.	Return with same return code as received from IKJEBEDS.
IKJEBERB Nonzero return code	Error in reading lower level directory block.	Return with same return code as received from IKJEBEDS.
IKJEBEAD Nonzero return code	Error in getting new block for block split.	Return with same return code as received from IKJEBEAD.
IKJEBEWB Nonzero return code	Error in writting new block after split.	Return with same return code as received from IKJEBEWB.

Table 79. IKJEBEWB Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEWA Nonzero return code	I/O error in previously initiated I/O.	Return with same return code as received from IKJEBEWA.

Table 80. IKJEBEDS Error and Exceptional Conditions

Indication	Condition	Action
DSDB NUM.INDEX field zero	Data set empty.	Return with return code 8, second word of input parameter list is address of DSDB.
IKJEBERB Nonzero return code	Error in reading directory block.	Return with same return code as received from IKJEBERB.
Record key greater than key in last entry in DSDB	Record key larger than limits of data set.	Return with return code 4, second word of parameter list contains the address of the directory block containing entry with highest key. BUFRER field of this directory block points to this entry.

Table 81. IKJEBERB Error and Exceptional Conditions

Indication	Condition	Action
IKJEBEWA Nonzero return code	Error in previously initiated I/O.	Return with same return code as received from IKJEBEWA.
IKJEBEAS Nonzero return code	Error in obtaining empty buffer.	Return with same return code as received from IKJEBEAS.
IKJEBEWA Nonzero return	Error in reading in block.	Return with same return code as received from IKJEBEWA.

Table 82. IKJEBEWA Error and Exceptional Conditions

Indication	Condition	Action
Test EBC flags after I/O complete	I/O error in XDAP read or write.	Synad routine stores error information in utility workarea and turns on error flag in DCBSTATS field in utility workarea. Return with return code 12.
Test error flag in DCBSTATS field in utility workarea	I/O error in BSAM write.	CHECK will have entered synad routine, which stores error information in utility workarea and turns on error flag in DCBSTATS field before returning control to CHECK. IKJEBEWA returns with return code 12.

Register Usage Table

Module Name	Register Contents at Entry
IKJEBEAA	1 -- address of parameter list (See next table for information about the csects of IKJEBEAA.)
IKJEBEAE	0 -- entry code 1 -- address of STAE work area 2 -- used with entry code 12
IKJEBEAT	1 -- address of parameter list 13 -- address of save area 14 -- return address
IKJEBEBO	1 -- address of EDIT communication area (IKJEBECA)
IKJEBECG	1 -- address of EDIT Communication area (IKJEBECA)
IKJEBECH	1 -- address of EDIT communication area
IKJEBECI	1 -- address of parameter list
IKJEBECN	1 -- address of EDIT communication area
IKJEBECO	1 -- address of EDIT communication area
IKJEBEDA	1 -- address of EDIT communication area
IKJEBEDE	1 -- address of EDIT communication area
IKJEBEDO	1 -- address of EDIT communication area
IKJEBEEX	0 -- address of EDIT Access Method work area (UTILWORK) 1 -- address of EDIT communication area
IKJEBEFC	1 -- address of EDIT communication area
IKJEBEFI	1 -- address of EDIT communication area
IKJEBEFO	1 -- address of EDIT communication area
IKJEBEHE	1 -- address of EDIT communication area
IKJEBEIM	1 -- address of EDIT communication area
IKJEBEIN	1 -- address of command processor parameter list (CPPL)
IKJEBEIP	1 -- address of EDIT communication area
IKJEBEIS	1 -- address of EDIT communication area
IKJEBELE	1 -- address of parameter list
IKJEBELI	1 -- address of EDIT communication area
IKJEBELT	1 -- address of EDIT communication area
IKJEBEMA	1 -- address of EDIT communication area
IKJEBEME	1 -- address of EDIT communication area
IKJEBEMR	1 -- address of EDIT communication area

(Continued)

Module Name	Register Contents at Entry
IKJEBEMS	0 -- address of EDIT communication area 1 -- address of parameter list
IKJEBEPS	1 -- address (or complimented address) of 8-byte field
IKJEBERE	1 -- address of EDIT communication area
IKJEBERN	1 -- address of parameter list
IKJEBERU	1 -- address of EDIT communication area
IKJEBESA	1 -- address of EDIT communication area
IKJEBESC	1 -- address of EDIT communication area
IKJEBESE	0 -- address of EDIT communication area 1 -- address of parameter list
IKJEBESN	1 -- address of EDIT communication area
IKJEBETA	1 -- address of EDIT communication area
IKJEBETO	1 -- address of EDIT communication area
IKJEBEUI	1 -- address of EDIT communication area
IKJEBEUP	1 -- address of EDIT communication area
IKJEBEUT	0 -- address of EDIT communication area 1 -- address of 3-word parameter list
IKJEBEVE	1 -- address of EDIT communication area

Csect Name (Module IKJEBEAA)	Register Contents at Entry
IKJEBEAD	1 -- address of 2-word parameter list
IKJEBEAS	1 -- address of 2-word parameter list
IKJEBEDL	1 -- address of 2-word parameter list
IKJEBEDR	0 -- address of a key value or zero 1 -- address of UTILWORK
IKJEBEDS	1 -- address of 2-word parameter list
IKJEBEDU	1 -- address of 3-word parameter list
IKJEBELO	1 -- address of 3-word parameter list
IKJEBEMV	0 -- length of data to be moved 1 -- address of 2-word parameter list
IKJEBERB	1 -- address of 2-word parameter list
IKJEBERR	1 -- address of 1- to 3-word parameter list
IKJEBEWA	1 -- address of 2-word parameter list
IKJEBEWB	1 -- address of 2-word parameter list
IKJEBEWR	1 -- address of 3-word parameter list

Section 7: Appendix

This appendix describes the formats of the EDIT command and subcommands.

Formats of the EDIT Command and Subcommands

The notation which defines the syntax used in this section is as follows:

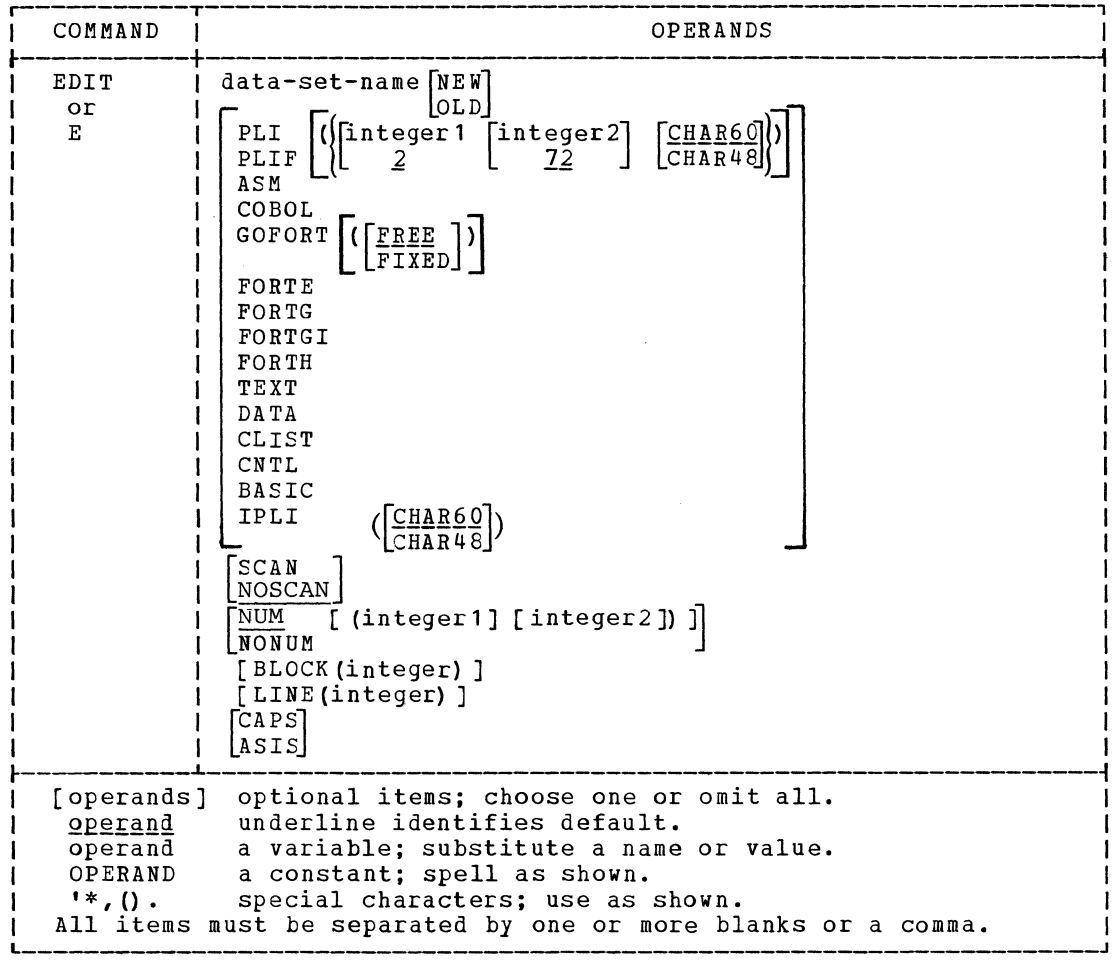
- A The underscore indicates a default value.
  B

- A Braces group related, alternative items; for proper operation,
  B one of the items within braces must be specified.

- C Brackets group related, alternative items. Items within
  D brackets are optional; none need be specified for proper operation.

For a more detailed description of these formats see the publication, IBM System/360 Operating System: Time Sharing Option, Command Language Reference, GC28-6732.

EDIT COMMAND



BOTTOM SUBCOMMAND

SUBCOMMAND	OPERANDS
{BOTTOM or B}	None

CHANGE SUBCOMMAND

SUBCOMMAND	OPERANDS
{CHANGE or C}	[line-number1[line-number2]] [* [count]] [special delimiter string1[special delimiter[string2] [special delimiter[ALL]]]] [count2]

DELETE SUBCOMMAND

SUBCOMMAND	OPERANDS
{DELETE or D}	[line-number-1 [line-number-2]] [* [count]]

DOWN SUBCOMMAND

SUBCOMMAND	OPERANDS
DOWN	[count]

END SUBCOMMAND

SUBCOMMAND	OPERANDS
END	None

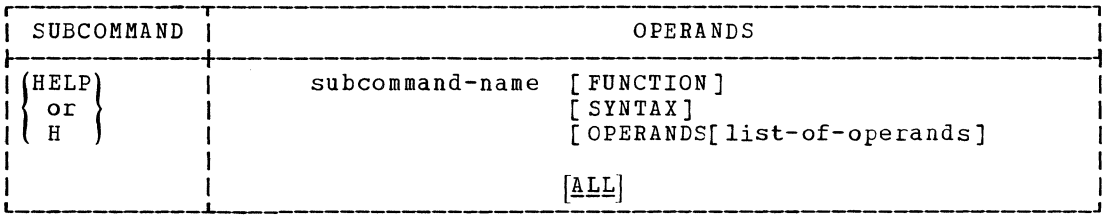
FIND SUBCOMMAND

SUBCOMMAND	OPERANDS
FIND	special delimiter string[special delimiter[count]]

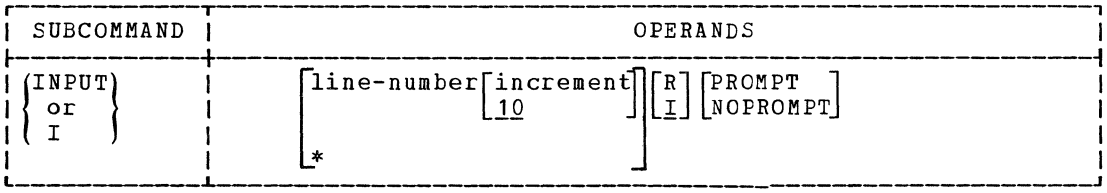
FORMAT SUBCOMMAND

A description of the format of this subcommand can be found in the Program Product publication, IBM System/360 Operating System: Time Sharing Option, TSO Data Utilities: COPY, FORMAT, LIST, MERGE, User's Guide and Reference, GC28-6765.

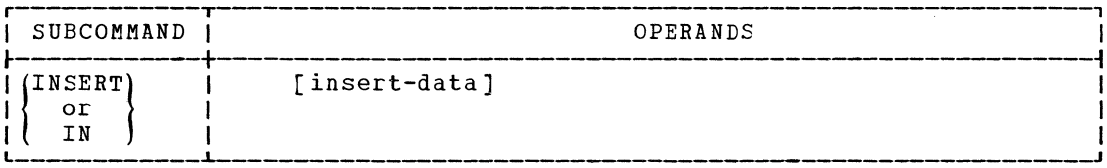
HELP SUBCOMMAND



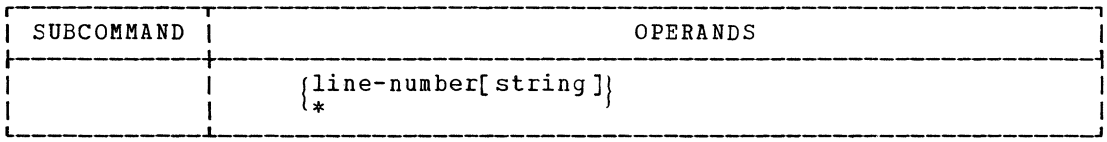
INPUT SUBCOMMAND



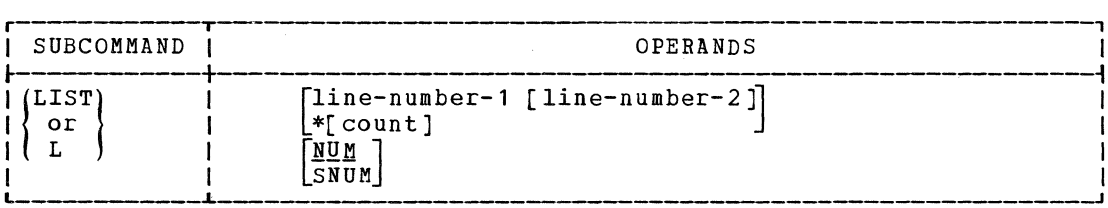
INSERT SUBCOMMAND



INSERT/REPLACE/DELETE SUBCOMMAND



LIST SUBCOMMAND



MERGE SUBCOMMAND

A description of the format of this subcommand can be found in the Program Product publication, IBM System/360 Operating System: Time Sharing Option, TSO Data Utilities: COPY, FORMAT, LIST, MERGE, User's Guide and Reference, GC28-6765.

PROFILE SUBCOMMAND

COMMAND	OPERANDS
{PROFILE / PROF}	CHAR({character / BS}) LINE({character / ATTN})
	NOCHAR NOLINE
	[PROMPT / NOPROMPT] [INTERCOM / NOINTERCOM]
	[PAUSE / NOPAUSE] [MSGID / NOMSGIC]

RENUM SUBCOMMAND

SUBCOMMAND	OPERANDS
{RENUM / or REN}	[new-line-number[increment[old-line-number]]]

RUN SUBCOMMAND

SUBCOMMAND	OPERANDS
RUN or R	['parameters']
	[TEST / NOTEST]
	[LMSG / SMSG]
	[LPREC / SPREC]
	[CHECK / OPT]

SAVE SUBCOMMAND

SUBCOMMAND	OPERANDS
{SAVE or S}	[data-set-name]

SCAN SUBCOMMAND

SUBCOMMAND	OPERANDS
{SCAN or SC}	{line-number-1[line-number-2] / *[count]} [ON / OFF]

TABSET SUBCOMMAND

SUBCOMMAND	OPERANDS
{TABSET or TAB}	[ON[(integer-list)] / OFF / IMAGE]

TOP SUBCOMMAND

SUBCOMMAND	OPERANDS
TOP	

UP SUBCOMMAND

SUBCOMMAND	OPERANDS
UP	[count]

VERIFY SUBCOMMAND

SUBCOMMAND	OPERANDS
{VERIFY or V}	ON OFF

Glossary

abnormal end of task (ABEND): Termination of a task prior to normal completion because of an error condition.

allocate: To assign a resource for use in performing a specific task.

attention exit routine: A routine that receives control when an attention interruption is received by the system.

attention interruption: An interruption of instruction execution caused by a terminal user hitting the attention key. See also "simulated attention".

attention key: A function key on terminals that causes an interruption of execution by the CPU.

attributes: See user attributes.

BASIC: An algebra-like language used for problem solving by engineers, scientists, and others who may not be professional programmers.

block: One record or several records grouped together in an unbroken sequence for transfer in or out of main storage as a unit.

catalog:

1. noun: In the System/360 Operating System, a collection of data set indexes that are used by the control program to locate a volume containing a specific data set.
2. verb: To include the volume identification of a data set in the catalog.

character-deletion character: A character within a line of terminal input specifying that it and the immediately preceding character are to be removed from the line.

character string: Any sequence of characters.

command: In TSO, a request from a terminal for the execution of a particular program, called a command processor. The command processor is in a command library under the command name. Any subsequent commands processed directly by that command processor are called subcommands.

command language: The set of commands, subcommands, and operands recognized by TSO.

command processor (CP): In TSO, a problem program executed as the result of entering a command at the terminal. Any problem program can be defined as a command processor by assigning a command name to the program and including the program in a command library.

command scan: A service routine used by command processors to check the syntax of TSO commands and subcommands.

CP: See "command processor".

current line: See "current record".

current line pointer: A pointer that indicates the line of a line data set with which a user is currently working. A terminal user can refer to the value of the current line pointer by entering an asterisk (*) with EDIT subcommands.

current record: The record in the utility data set pointed to by the current line pointer.

DAIR: See "dynamic allocation interface routine".

data block: A control block in the EDIT Access Method which contains user records and associated keys and line numbers.

data set: A collection of data that is accessible by the system. The data set usually resides on an auxiliary storage device.

data set organization: The arrangement by data management of information in a data set. For example, sequential organization, or partitioned organization.

data set name: The term or phrase used to identify a data set (see qualified name).

directory block: A control block in the EDIT Access Method which contains pointers to data blocks or to lower-level directory blocks.

dynamic allocation interface routine (DAIR): A TSO service routine that performs various data management functions.

ECT: See "environment control table".

EDIT access method: The portion of the EDIT Command Processor which reads, writes and deletes records in the utility data set.

edit data set: The QSAM-formatted data set built by the EDIT Command Processor.

edit mode: The operating condition of EDIT in which terminal input is considered to be EDIT subcommands.

EDIT session: The time elapsed from the entry of the EDIT command to the termination of EDIT Command processing.

environment control table (ECT): A control block which contains information about the user's environment in the foreground region.

first-level message: Diagnostic message which identifies a general condition; more specific information is available in a second-level message if the text is followed by a "+".

format data set: The data set built by the EDIT Command Processor for operation of the FORMAT Program Product Command.

GETLINE: A service routine used by command processors to obtain input. GETLINE passes successive lines from the source indicated by the current input stack element: the terminal, or an in-storage list.

input mode: The operating condition of EDIT in which terminal input is considered to be data.

input stack: A push-down list of sources of input for GETLINE. Possible sources are the terminal or an in-storage list.

insertion text: Portion added to a message which gives specific detail to the general condition identified by the message.

in-storage data set: See "reverse Polish-notated data set".

in-storage list: A chain of input lines in main storage, such as commands in an EXEC procedure, that are used in place of terminal input.

interruption: A transfer of CPU control to the control program of the Operating System. The transfer is initiated automatically by the computing system or by a problem state program through the execution of a supervisor call (SVC) instruction. The transfer of control occurs in such a way that control can later be restored to the interrupted program, or, in systems that perform more than one task at a time, to a different program.

ITF: BASIC: A conversational subset of BASIC designed for ease of use at a terminal.

ITF: PL/I: A conversational subset of PL/I designed for ease of use at the terminal.

key: The mechanism used by the EDIT Access Method to locate records within the utility data set; a binary number equivalent to the line number of the record.

keyword: A command operand that consists of a specific character string (such as FORTLIB or PRINT) and optionally a parenthesized value.

LANGPRCR: The BASIC/IPLI syntax checker.

line:

1. A single string of one or more characters, ending with a carriage return, typed at a terminal and entered into the system.
2. In the EDIT Access Method - a record and its associated key and line number.

line deletion character: A terminal character that specifies that it and all preceding characters are to be deleted from a line of terminal input.

line number: In the EDIT Access Method - a number associated with a record in the utility data set; the printable equivalent of the record's key.

load module: The output of the linkage editor; a program in a form suitable for loading into main storage for execution.

logical record: A record that is defined in terms of the information it contains rather than by its physical qualities.

lower-level directory block: A control block in the EDIT Access Method which contains pointers to data blocks.

member: A partition of a partitioned data set.

merge data set: The data set built by the EDIT Command Processor for operation of the MERGE Program Product Command.

name: A one to eight character alphameric term that identifies a data set, a command or control statement, a program, or a cataloged procedure. The first character of the name must be alphabetic.

operand: In the TSO command language, information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor. Some operands are positional, identified by their sequence in the command input line, others are identified by keywords.

PARSE: A service routine used by command processors to check the syntax of parameters and operands.

partitioned data set: A data set that is stored in direct access storage and can be cataloged like any other data set. A partitioned data set is often called a program library. It is divided into independent partitions called members, each of which normally contains a program or part of a program, in the form of one or more sequential blocks. Each program library contains a built-in directory (or index) that the control program can use to locate a program in the library. Each member has a unique name listed in a directory at the beginning of the data set. Members can be added or deleted as needed. Records within members are organized sequentially.

physical record: A record that is defined in terms of physical qualities rather than by the information it contains. (See record.)

PL/I: A high-level programming language that has features of both COBOL and FORTRAN, plus additional features.

profile (user): The set of characteristics that describe the user to the system.

prompting: A system function that helps a terminal user by requesting him to supply operands necessary to continue processing.

PUTGET: A service routine used by command processors to put messages to the terminal and to receive the user's responses to the messages.

PUTLINE: A service routine that sends output to the terminal. Putline selectively puts out messages according to whether or not a user has suppressed prompting or is executing a command procedure.

qualified name: A data set name that is composed of two or more names separated by periods. (For example, MOORE.SALES.JUNE.)

read current: Instruction to the EDIT Access Method to locate the current record.

read last: Instruction to the EDIT Access Method to locate the last record (the highest key) of the utility data set.

read next: Instruction to the EDIT Access Method to locate the record following (with the next higher key) the current record.

read top: Instruction to the EDIT Access Method to locate the first record (the lowest key) of the utility data set.

record:

1. One or more data fields that represent an organized body of related data, such as all of the basic accounting information concerning a single sales transaction. (See also logical record and physical record.)
2. In the EDIT Access Method - a line entered by the user into the system during an EDIT session; each record is identified by a unique key.

region control task (RCT): The control program routine handling quiesce/restore and LOGON/LOGOFF. There is one RCT for each active foreground region.

renum data set: The utility data set built by the EDIT Command Processor for operation of the Renum subcommand of EDIT; becomes the Utility data set when renumbering is complete.

retry routine: A routine which is used to attempt recovery from an error.

reverse Polish-notated data set: A data set built and maintained in main storage by LANGPRCR.

run data set: The data set built by the EDIT Command Processor for operation of the RUN TSO Command.

second-level message: Diagnostic message associated with a first-level message; contains specific information about the general condition identified in the first-level message.

STACK: A service routine that manipulates the input stack.

STAE (specify task asynchronous exit): A macro instruction specifying a routine to receive control in the event of the issuing task's abnormal termination (ABEND).

STAI (subtask ABEND intercept): A keyword of the ATTACH macro instruction specifying a routine to receive control after the abnormal termination of a daughter task.

subcommand: For TSO, a subcommand is a request for a particular operation to be performed, the particular operation falling within the scope of work requested by the command to which the subcommand applies.

swap: To write an image of a foreground job's main storage region to auxiliary storage, and to read in another job's main storage image into the region.

swap data set: A data set dedicated to the swapping operation.

syntax checker: A program that tests source statements in a programming language for violations of that language's syntax.

task: A unit of work for the central processing unit defined by the control program.

TCAM: See "telecommunications access method".

telecommunications access method (TCAM): A generalized terminal I/O support package, providing application program independence of terminal characteristics.

terminal: A device resembling a typewriter that is used to communicate with the system.

terminal monitor program (TMP): A program that accepts and interprets commands from the terminal, and causes the appropriate command processors to be scheduled and executed.

terminal user: See "user".

TGET: An I/O macro instruction used by problem programs to obtain a line of input from the terminal.

time sharing control task (TSC): A TSO system task that handles system initialization, allocation of time-shared regions, swapping, and general control of the time-sharing operation.

time sharing option (TSO): An option of the operating system providing conversational time sharing from remote terminals.

TMP: See "terminal monitor program".

TSC: See "time sharing control task".

user: Under TSO, anyone with an entry in the User Attribute Data Set; anyone eligible to log on.

user attributes: A set of parameters in the User Attribute Data Set (UADS). The parameters describe the user to the system: whether he is authorized to use the ACCOUNT command, what size main storage region he is to be assigned, etc.

USERID: See "user identification".

user identification (USERID): A one to eight character symbol identifying each system user.

user profile table (UPT): A table of user attributes kept for each active user, built by Logon from information in the LOGON command, the UADS, and the Logon procedure.

utility data set: The data set built and maintained by the EDIT Access Method.

verification: A use of the EDIT command in which all subcommands are acknowledged and any textual changes are displayed as they are made.

verify message: Message containing the current record and key, displayed by the EDIT command processor.

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Order No. GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

* - see current line pointer

abnormal end exit routine
   error and exceptional conditions 414
abnormal end of task (ABEND)
   definition 463
allocate
   definition 463
attention exit routine
   definition 463
   error and exceptional conditions 414
   method of operation diagram 487
attention interruption
   definition 463
   EDIT handling of 52
attention key
   definition 463
attributes - See user attributes

B - see BOTTOM subcommand
BASIC
   data set requirement 29
   definition 463
BASIC renum service routine 61
block
   definition 463
BOTTOM subcommand
   see also subcommand processor
   summary of operations 63
   syntax 459

C - see CHANGE subcommand
catalog
   definition 463
CHANGE subcommand
   see also subcommand processor
   summary of operations 67
   syntax 459
character-deletion character
   definition 463
   user specification of 106
character string
   definition 463
command
   definition 463
command invoker service routine 55
command language
   definition 463
command processor (CP)
   definition 463
command scan
   definition 463
   EDIT program's use of 51
   function 17
controller
   error and exceptional conditions 413
   method of operation diagram 483
CP - see command processor
current line
   definition 463
current line pointer
   definition 463
   updating of 25
   use of 18
current record
   definition 463

D - see DELETE subcommand
DAIR - see dynamic allocation interface routine
data block
   definition 463
data set
   created by EDIT program 26
   definition 463
data set allocation service routine 56
data set name
   definition 463
   operand processing 33
data set organization
   definition 463
   of EDIT data set 26
   of Utility data set 26
delete operation routine
   error and exceptional conditions 449
   method of operation diagram 539
DELETE subcommand
   see also subcommand processor
   summary of operations 71
   syntax 459
directory block
   definition 463
   use of 138
directory search routine
   error and exceptional conditions 451
   method of operation diagram 543
directory update routine
   error and exceptional conditions 450
   method of operation diagram 547
DOWN subcommand
   see also subcommand processor
   summary of operations 73
   syntax 459
dynamic allocation interface routine (DAIR)
   definition 463
   function 17

ECT - see environment control table
EDIT access method
  definition 464
  operation of 135
EDIT command
  proper syntax of 458
  validation of operands 33
edit data set
  allocation of 26,35
  definition 464
  use of 26
edit mode
  change of mode to 22
  definition of 464
  initial selection of 33
EDIT program
  access method 135-143
  controller 49-54
  error and exceptional conditions 413
  processing 49
    abnormal end and attention exits 49
    establishing mode of operation 50
    handling attention interrupt 52
    invoking EDIT subcommand processor 51
    obtaining EDIT subcommand 50
    SCAN processing 49
    termination processing 53
    validating an EDIT subcommand 51
    verify message handling 52
  error and attention exit routines 53,54
  initialization 33-47
    data set type, prompting for 43
    error and exceptional conditions 408-412
    operand processing 33
      BLOCK keyword 38
      CAPS/ASIS keyword 39
      data set name 29
      GOFORT data set type 40
      LINE keyword and keyword value 36
      NEW/OLD keyword 30
      NUM/NONUM keyword 39
      PLI or PLI/F source margins 36
      SCAN/NOSCAN keyword 36
      processor dependent information, obtaining 43
      summary of operations 41
      termination processing 43
  service routines 54
  subcommand processors 62-133
    BOTTOM 62
    CHANGE 64
    DELETE 69
    DOWN 72
    END 74
    FIND 76
    FORMAT 80
    HELP 83
    INPUT 85
    INSERT 93
    Line Insert/Replace/Delete 96
    LIST 98
    MERGE 102
    PROFILE 106
    RENUM 109
    RUN 112
    SAVE 116
EDIT program, subcommand processors (continued)
    SCAN 119
    TABSET 125
    TOP 128
    UP 130
    VERIFY 132
EDIT service routines 54-62
  BASIC renum 61
  command invoker 55
  data set allocation 56
  final copy 57
  initial copy 56
  line edit 57
  message selection 58
  processor data set search 60
  string search 61
  translation 58
EDIT session
  definition 464
EDIT subcommands
  invoking processors for 51
  obtaining of 50
  processors - see EDIT program
  syntax 459-462
  validation of 51
END subcommand
  see also subcommand processor
  summary of operations 75
  syntax 459
environment control table (ECT)
  definition 464
externals - see syntax

final copy service routine 57
final processing routine 142
  error and exceptional conditions 447
  method of operation diagram 531
FIND subcommand
  see also subcommand processor
  summary of operations 78
  syntax 459
first-level message
  definition 464
  handling 58-60
format data set
  definition 464
  use of 26,80
FORMAT subcommand
  see also subcommand processor
  summary of operations 82
  syntax 460

GETLINE
  definition 464
  function 17

H - see HELP subcommand
HELP subcommand
  see also subcommand processor
  summary of operations 15
  syntax 460

I - see INPUT subcommand
IKJEBEAA
summary of 154
IKJEBEAD
error and exceptional conditions 450
flowchart 225
summary of 155
IKJEBEAE
error and exceptional conditions 414
flowchart 226
summary of 156
IKJEBEAS
flowchart 227
summary of 157
IKJEBEAT
error and exceptional conditions 414
flowchart 228
summary of 158
IKJEBEBO
error and exceptional conditions 419
flowchart 229
summary of 159
IKJEBECA
data layout 403
IKJEBECG
error and exceptional conditions 420-421
flowchart 230-235
summary of 160
IKJEBECH
error and exceptional conditions 420-421
flowchart 236-238
summary of 161
IKJEBECI
error and exceptional conditions 414-416
flowchart 239-240
summary of 162
IKJEBECN
error and exceptional conditions 420-421
flowchart 241-244
summary of 164
IKJEBECO
error and exceptional conditions 416
flowchart 245-247
summary of 165
IKJEBEDA
error and exceptional conditions 417
flowchart 248
summary of 166
IKJEBEDE
error and exceptional conditions 422
flowchart 249-251
summary of 167
IKJEBEDL
error and exceptional conditions 450
flowchart 252-254
summary of 168
IKJEBEDO
error and exceptional conditions 423
flowchart 255,256
summary of 169
IKJEBEDR
error and exceptional conditions 449
flowchart 257
summary of 170
IKJEBEDS
error and exceptional conditions 451
flowchart 258-260
summary of 171
IKJEBEDU
error and exceptional conditions 450
flowchart 261-265
summary of 172
IKJEBEEN
error and exceptional conditions 424
flowchart 266
summary of 173
IKJEBEEX
error and exceptional conditions 447
flowchart 267
summary of 174
IKJEBEFC
error and exceptional conditions 418
flowchart 269,269
summary of 175
IKJEBEFI
error and exceptional conditions 425
flowchart 270
summary of 176
IKJEBEFO
error and exceptional conditions 426-427
flowchart 271,272
summary of 177
IKJEBEHE
flowchart 273
summary of 178
IKJEBEIA
summary of 179
IKJEBEIM
error and exceptional conditions 427-428
flowchart 274-278
summary of 180
IKJEBEIN
error and exceptional conditions 408-412
flowchart 279-286
summary of 181
IKJEBEIP
error and exceptional conditions 427-428
flowchart 287,288
summary of 182
IKJEBEIS
error and exceptional conditions 429
flowchart 289,290
summary of 184
IKJEBELE
flowchart 291
summary of 185
IKJEBELI
error and exceptional conditions 430
flowchart 292,293
summary of 186
IKJEBELO
error and exceptional conditions 449
flowchart 294,295
summary of 187
IKJEBELT
error and exceptional conditions 431-432
flowchart 296-298
summary of 188
IKJEBEMA
error and exceptional conditions 413
flowchart 299-301
summary of 189
IKJEBEMA (Csect of IKJEBEMA)
summary of 190

IKJEBEME
error and exceptional conditions 432-433
flowchart 302,303
summary of 192
IKJEBEMR
error and exceptional conditions 418
flowchart 304
summary of 193
IKJEBEMS
flowchart 305
summary of 194
IKJEBEMV
flowchart 306
summary of 195
IKJEBEPD
data layout 417
summary of 196
IKJEBEPR
error and exceptional conditions 131
flowchart 349-351
summary of 239
IKJEBEPS
flowchart 307
summary of 197
IKJEBEPS (Csect of IKJEBEPS)
summary of 198
IKJEBERB
error and exceptional conditions 452
flowchart 308
summary of 199
IKJEBERE
error and exceptional conditions 434
flowchart 309-312
summary of 200
IKJEBERN
error and exceptional conditions 419
flowchart 313
summary of 202
IKJEBERR
error and exceptional conditions 448
flowchart 314
summary of 203
IKJEBERU
error and exceptional conditions 435-436
flowchart 315,316
summary of 204
IKJEBESA
error and exceptional conditions 437-441
flowchart 317-327
summary of 205
IKJEBESA (Csect of IKJEBESA)
summary of 206
IKJEBESC
error and exceptional conditions 442-444
flowchart 328-332
summary of 207
IKJEBESE
error and exceptional conditions 419
flowchart 333
summary of 208
IKJEBESN
error and exceptional conditions 442-444
flowchart 334-339
summary of 209
IKJEBETA
error and exceptional conditions 445
flowchart 340
summary of 210
IKJEBETO
error and exceptional conditions 445
flowchart 341
summary of 211
IKJEBEUI
error and exceptional conditions 447
flowchart 342
summary of 212
IKJEBEUP
error and exceptional conditions 446
flowchart 343,344
summary of 213
IKJEBEUT
error and exceptional conditions 448
flowchart 345
summary of 214
IKJEBEVE
error and exceptional conditions 446
flowchart 346
summary of 215
IKJEBEWA
error and exceptional conditions 452
flowchart 347
summary of 216
IKJEBEWB
error and exceptional conditions 451
flowchart 348
summary of 217
IKJEBEWR
error and exceptional conditions 448
flowchart 349-352
summary of 218
IKJEBMA1
summary of 219
IKJEBMA2
summary of 220
IKJEBMA8
summary of 221
IKJEBMA9
summary of 222
IKJEBSA9
summary of 223
implicit subcommand function - see Line Insert/Replace/Delete
IN - see INSERT subcommand
initial copy service routine 56
initialization
EDIT access method 142
error and exceptional conditions 447
method of operation diagram 531
EDIT program 33
error and exceptional conditions 408-412
method of operation diagram 481
summary of operations 41
input mode
change of mode to 21,22
definition of 464
initial selection of 33
input stack
definition 464
deleting of 53
INPUT subcommand
see also subcommand processor
summary of operations 88
syntax 460
INSERT subcommand
see also subcommand processor
summary of operations 95

INSERT subcommand (continued)
syntax 460
Insert/Replace/Delete - see Line Insert/Replace/Delete
insertion text
definition 464
use of 59
in-storage data set - see reverse Polish-notated data set
in-storage list
definition 464
interface routine
error and exceptional conditions 179
method of operation diagram 533
interruption
definition 464
ITF:BASIC
data set requirement 29
definition 464
ITF:PLI
data set requirement 29
definition 464

key
definition 464
subcommand processors' use of 23,24
keyword
definition 464
ASIS 39
BLOCK 38
CAPS 39
FIXED 40
FREE 40
LINE 36
NEW 35
NONUM 39
NOSCAN 36,49
NUM 39
OLD 35
SCAN 36,49

L - see LIST subcommand
LANGPRCR
BASIC/IPLI use of 29
definition 464
line
definition 464
line-deletion character
definition 464
user specification of 106
line edit service routine 57
Line Insert/Replace/Delete subcommand
see also subcommand processor
summary of operations 97
syntax 460
line number
changing of - see RENUM subcommand
definition 464
subcommand processors' use of 23,24
LIST subcommand
see also subcommand processor
summary of operations 100
syntax 460
load module
definition 464
in EDIT program
load module, in EDIT program (continued)
list of 145-149
naming conventions 145
logical record
definition 464
lower-level directory block
definition 464
use of 140

member
definition 464
merge data set
definition 464
use of 26,102
MERGE subcommand
see also subcommand processor
summary of operations 104
syntax 461
message selection routine 58
mode of EDIT operation
Edit 21,22
establishment of 33,50
Input 21,22

name
definition 464

operand
data set name 33
definition 465
source margins 36

PARSE
definition 465
function 17
partitioned data set
definition 465
processing 35
physical record
definition 465
PLI
definition 465
validation of source margins 36
processor data table search service routine 60
PROF - see PROFILE subcommand
profile, user
definition 465
PROFILE subcommand
see also subcommand processor
summary of operations 107
syntax 461
prompting
definition 465
PUTGET
definition 465
function 17
PUTLINE
definition 465
function 17

qualified name
definition 465

R - see RUN subcommand
RCT - see region control task
read block routine
   error and exceptional conditions 452
   method of operation diagram 555
read current
   definition 465
read last
   definition 465
read next
   definition 465
read operation routine
   error and exceptional conditions 448
   method of operation diagram 537
read top
   definition 465
record
   definition 465
   subcommand processors' use of 23,24
record delete routine
   error and exceptional conditions 450
   method of operation diagram 543
record locate routine
   error and exceptional conditions 449
   method of operation diagram 541
region control task (RCT)
   definition 465
   relationship of EDIT to 14,15
REN - see RENUM subcommand
renum data set
   definition 465
   use of 26,109
RENUM subcommand
   see also subcommand processor
   summary of operations 111
   syntax 461
retry routine
   abnormal end exit use of 53
   definition 465
reverse Polish-notated data set
   definition 465
   updating of 25,62
   use of 138
run data set
   definition 465
   use of 29,112
RUN subcommand
   see also subcommand processor
   summary of operations 114
   syntax 461

S - see SAVE subcommand
SAVE subcommand
   see also subcommand processor
   summary of operations 118
   syntax 462
SC - see SCAN subcommand
SCAN subcommand
   summary of operations 121
   syntax 462
second-level message
   definition 465
   handling 58-60
STACK
   definition 465
   function 17
STAE (specify task asynchronous exit)
   definition 465
   processing 485
STAI (subtask ABEND intercept)
   definition 465
   processing 485
string search serving routine 66
subcommand
   definition 466
   EDIT, functions of 18,19
   validation of 51
subcommand processor
   invoking of 51,52
   BOTTOM
      see also IKJEBEBO
      method of operation diagram 489
      operation of 62
   CHANGE
      see also IKJEBECG,IKJEBECH,IKJEBECN
      method of operation diagram 491
      operation of 64
   DELETE
      see also IKJEBEDE
      method of operation diagram 493
      operation of 69
   DOWN
      see also IKJEBEDO
      method of operation diagram 49
      operation of 72
   END
      see also IKJEBEEN
      operation of 74
   FIND
      see also IKJEBEFI
      method of operation diagram 497
      operation of 76
   FORMAT
      see also IKJEBEFO
      method of operation diagram 499
      operation of 80
   HELP
      see also IKJEBEHE
      method of operation diagram 501
      operation of 83
   INPUT
      see also IKJEBEIM,IKJEBEIP
      method of operation diagram 503
      operation of 85
   INSERT
      see also IKJEBEIS
      method of operation diagram 505
      operation of 93
   Line Insert/Replace/Delete
      see also IKJEBELI
      method of operation diagram 507
      operation of 96
   LIST
      see also IKJEBELT
      method of operation diagram 509
      operation of 98
   MERGE
      see also IKJEBEME
      method of operation diagram 511
      operation of 102
   PROFILE
      see also IKJEBEPR
      method of operation diagram 513
      operation of 106

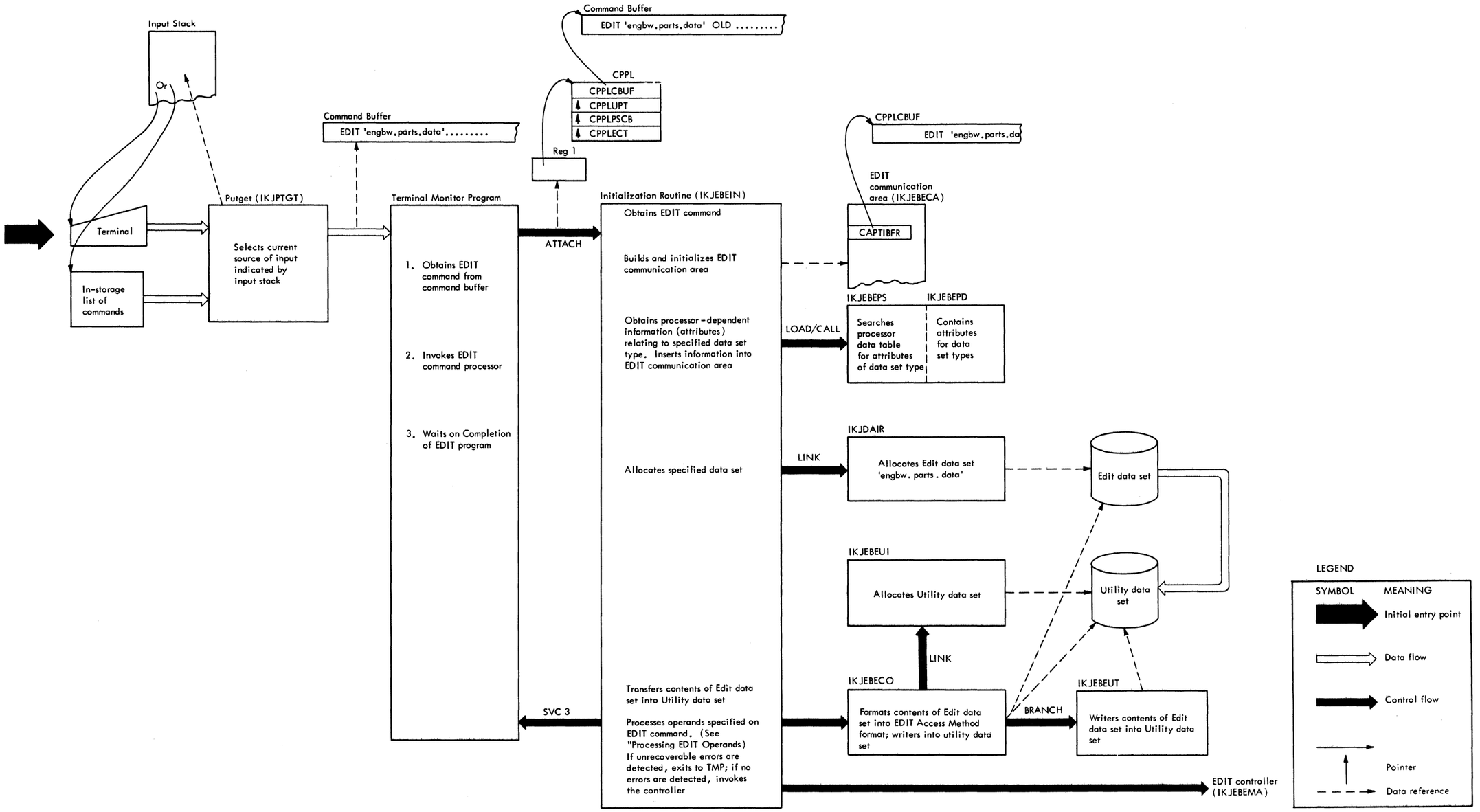
subcommand processor (continued)
RENUM
see also IKJEBERE
method of operation diagram 515
operation of 109
RUN
see also IKJEBERU
method of operation diagram 517
operation of 112
SAVE
see also IKJEBESA
method of operation diagram 519
operation of 116
SCAN
see also IKJEBESC, IKJEBESN
method of operation diagram 521
operation of 119
TABSET
see also IKJEBETA
method of operation diagram 523
operation of 125
TOP
see also IKJEBETO
method of operation diagram 525
operation of 128
UP
see also IKJEBEUP
method of operation diagram 527
operation of 130
VERIFY
see also IKJEBEVE
method of operation diagram 529
operation of 132
swap
definition 466
swap data set
definition 466
syntax
of EDIT command 458
of EDIT subcommands 459-462
syntax checker
definition 466
use of 29,30,49,119

TAB - see TABSET subcommand
TABSET subcommand
see also subcommand processor
summary of operations 127
syntax 462
task
definition 464
TCAM - see telecommunications access method
telecommunications access method (TCAM)
definition 464
terminal
definition 464
terminal monitor program (TMP)
definition 464
operation of 16,17
return of control to 49,53
terminal user
definition 464
TGET
definition 464
time sharing control task (TSC)
definition 464
relationship of EDIT to 14
time sharing option (TSO)
definition 464
relationship of EDIT to 13-17
TMP - see terminal monitor program
TOP subcommand
see also subcommand processor
summary of operations 129
syntax 462
translation service routine 58
TSC - see time sharing control task
TSO - see time sharing option
TTR assignment routine
error and exceptional conditions 450
method of operation diagram 545

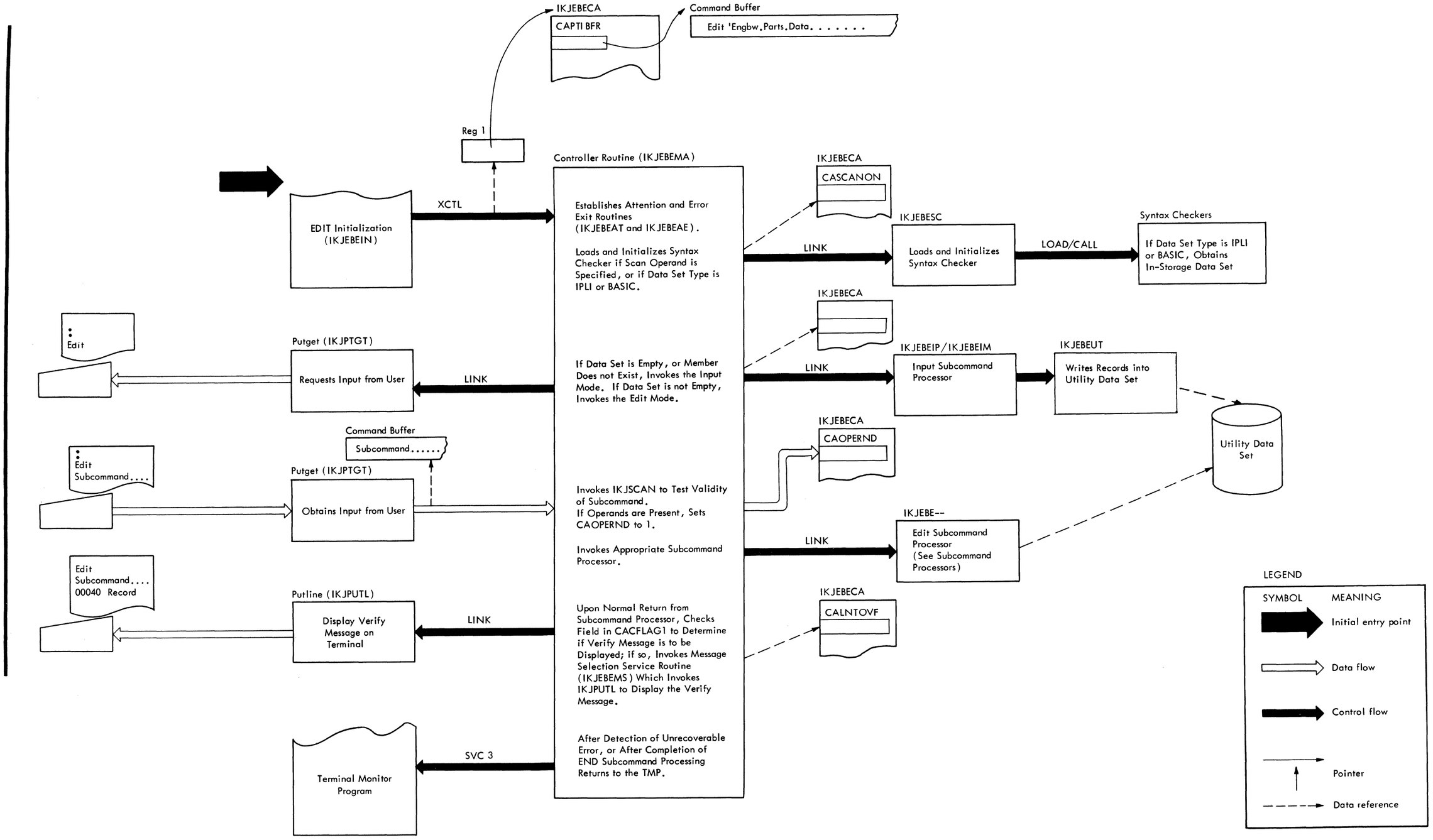
UP subcommand
see also subcommand processor
summary of operations 131
syntax 462
UPT - see user profile table
user
definition 464
user attributes
definition 464
user identification (USERID)
definition 464
user profile table (UPT)
definition 464
updating of 106
USERID - see user identification
utility data set
definition 464

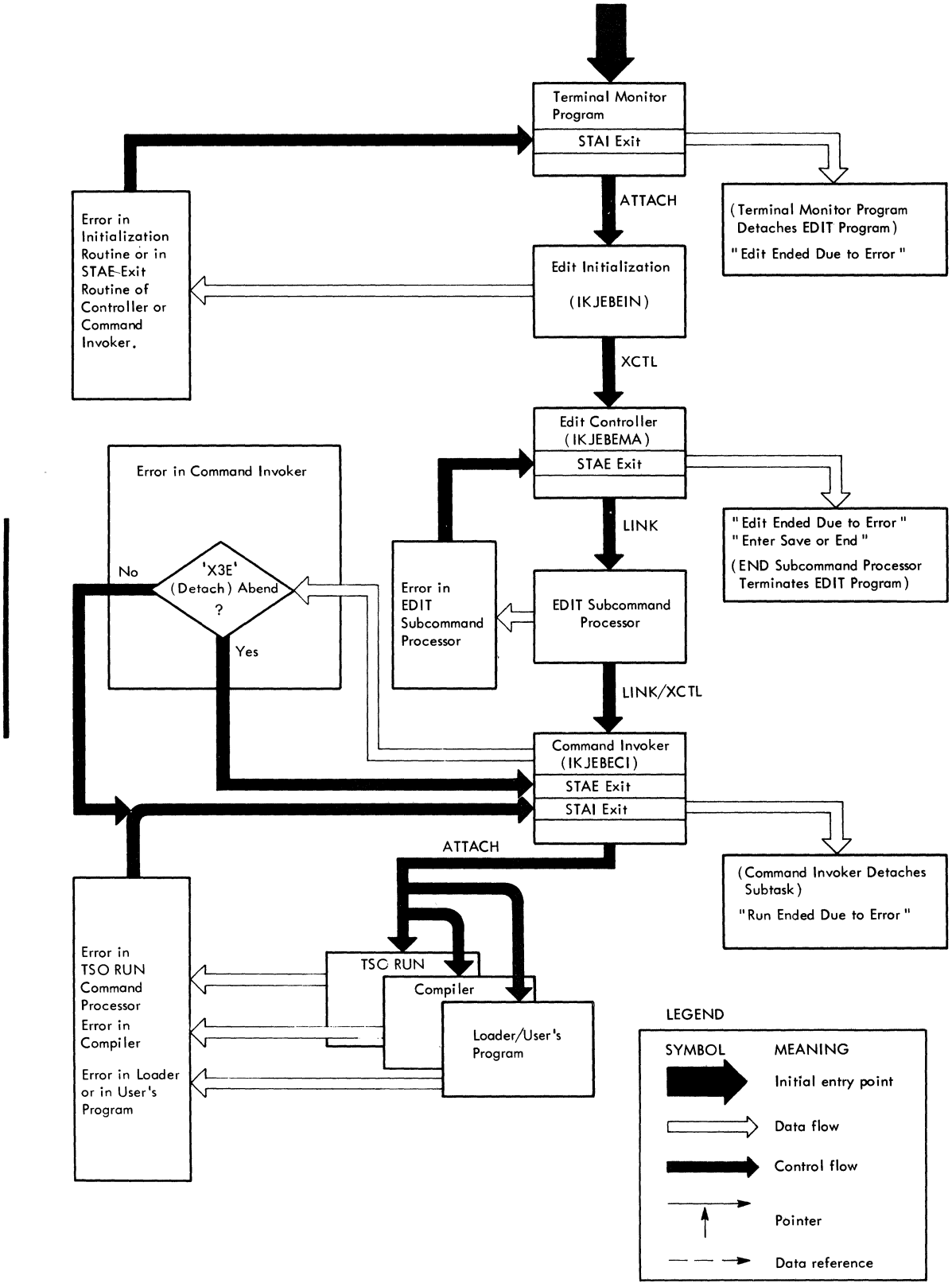
V - see VERIFY subcommand
verification
definition 464
verify message
definition 464
function 26
handling 52,58,60
VERIFY subcommand
see also subcommand processor
summary of operation 133
syntax 462

wait routine
error and exceptional conditions 452
method of operation diagram 557
write block routine
error and exceptional conditions 451
method of operation diagram 549
write operation routine
error and exceptional conditions 448
method of operation diagram 535



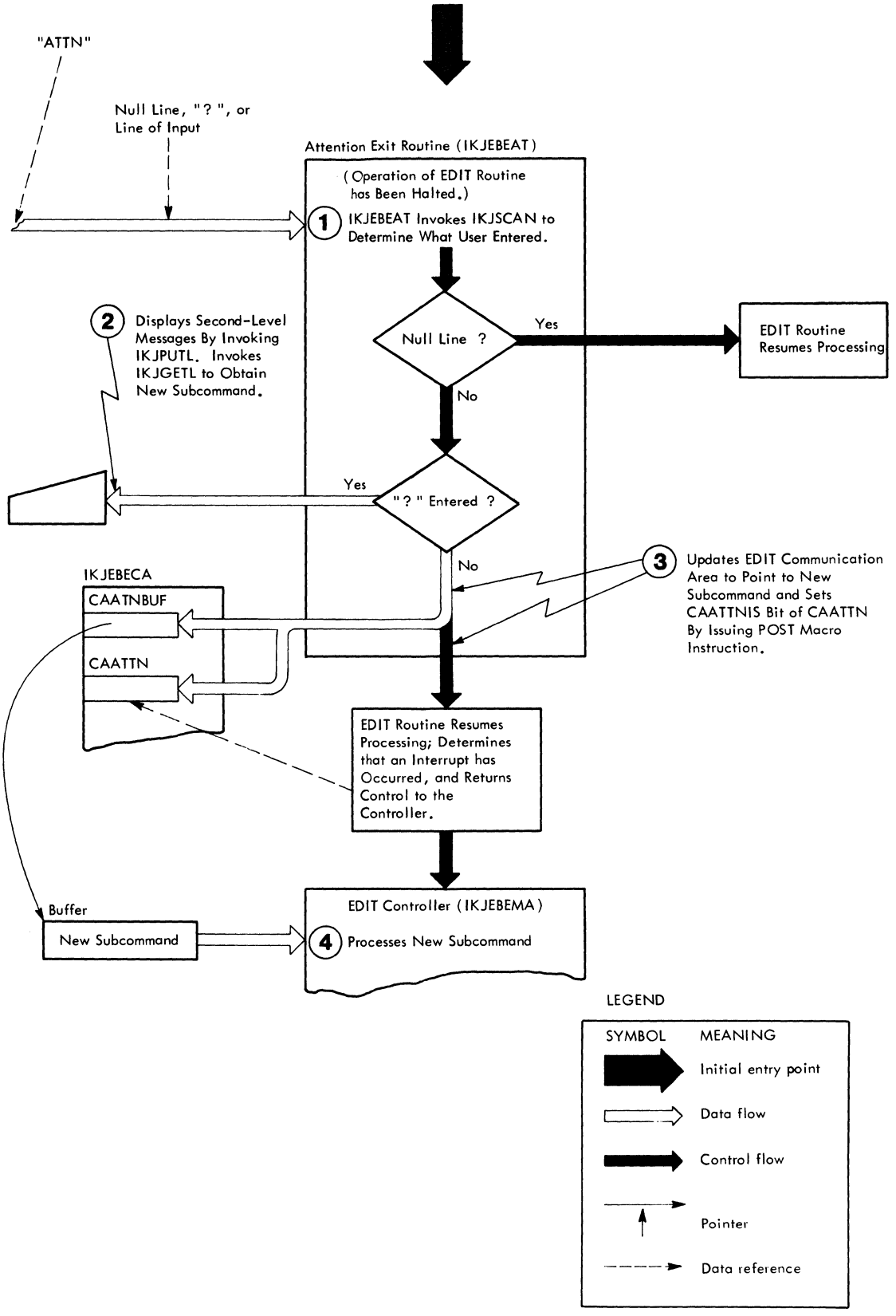
Method of Operation Diagram 1. Initialization


IKJEBECA
CAPTI BFR
Command Buffer
Edit 'Engbw.Parts.Data. . . . . .
Reg 1
Controller Routine (IKJEBEMA)
EDIT Initialization (IKJEBEIN)
XCTL
Establishes Attention and Error Exit Routines (IKJEBEAT and IKJEBEAE).
Loads and Initializes Syntax Checker if Scan Operand is Specified, or if Data Set Type is IPLI or BASIC.
IKJEBECA
CASCANON
IKJEBESC
LINK
Loads and Initializes Syntax Checker
LOAD/CALL
Syntax Checkers
If Data Set Type is IPLI or BASIC, Obtains In-Storage Data Set
Edit
Putget (IKJPTGT)
Requests Input from User
LINK
If Data Set is Empty, or Member Does not Exist, Invokes the Input Mode. If Data Set is not Empty, Invokes the Edit Mode.
IKJEBECA
IKJEBEIP/IKJEBEIM
Input Subcommand Processor
LINK
IKJEBEUT
Writes Records into Utility Data Set
Utility Data Set
Command Buffer
Subcommand.....
Edit
Subcommand....
Putget (IKJPTGT)
Obtains Input from User
Invokes IKJSCAN to Test Validity of Subcommand.
If Operands are Present, Sets CAOPERND to 1.
Invokes Appropriate Subcommand Processor.
IKJEBECA
CAOPERND
IKJEBE--
Edit Subcommand Processor (See Subcommand Processors)
LINK
Edit
Subcommand....
00040 Record
Putline (IKJPUTL)
Display Verify Message on Terminal
LINK
Upon Normal Return from Subcommand Processor, Checks Field in CACFLAG1 to Determine if Verify Message is to be Displayed; if so, Invokes Message Selection Service Routine (IKJEBEMS) Which Invokes IKJPUTL to Display the Verify Message.
IKJEBECA
CALNTOVF
Terminal Monitor Program
SVC 3
After Detection of Unrecoverable Error, or After Completion of END Subcommand Processing Returns to the TMP.
LEGEND
SYMBOL
MEANING
Initial entry point
Data flow
Control flow
Pointer
Data reference

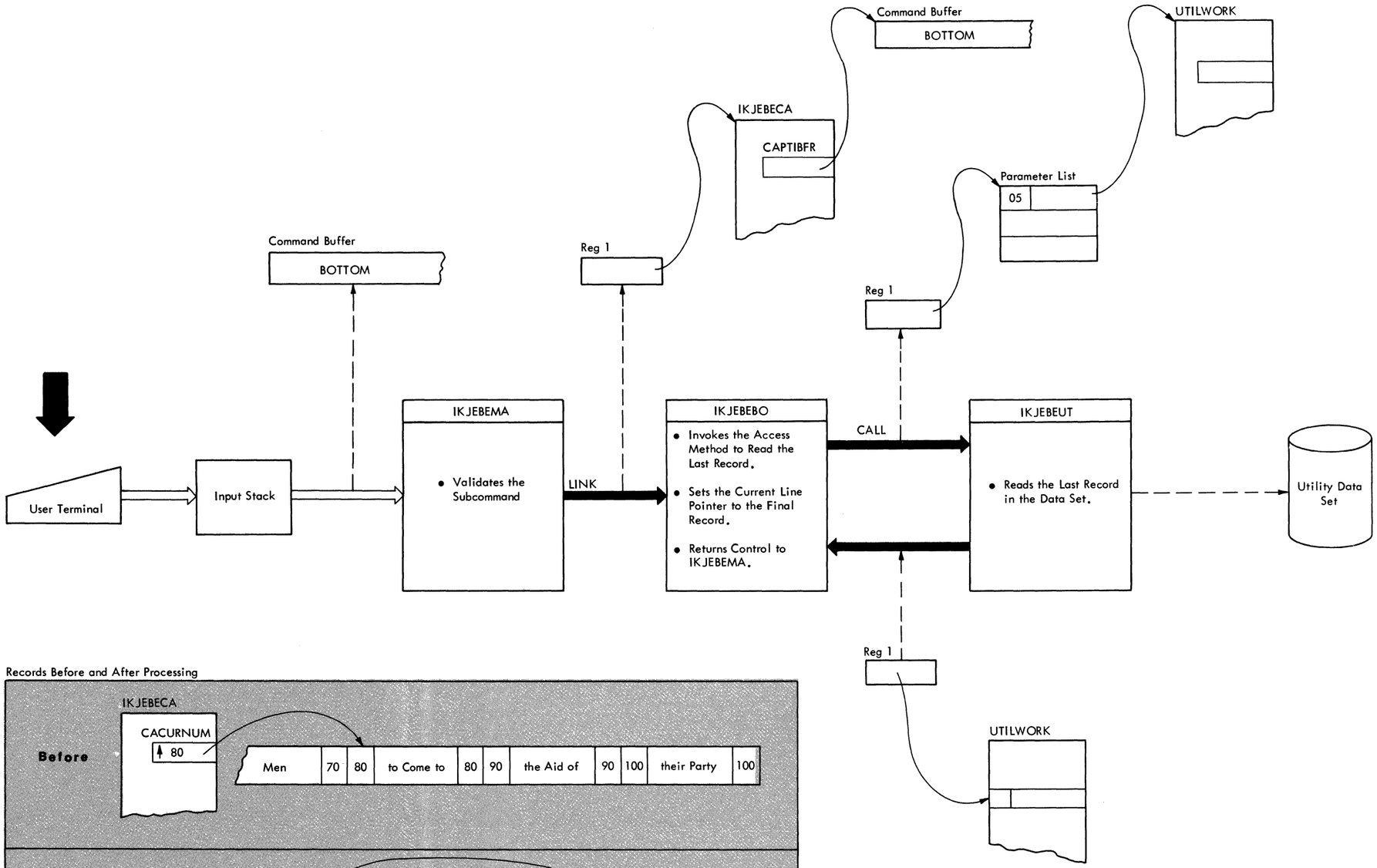


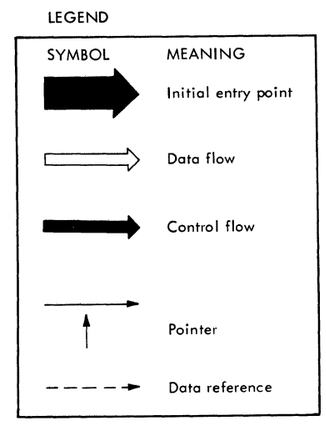
Method of Operation Diagram 3.

EDIT Program STAE/STAI Processing

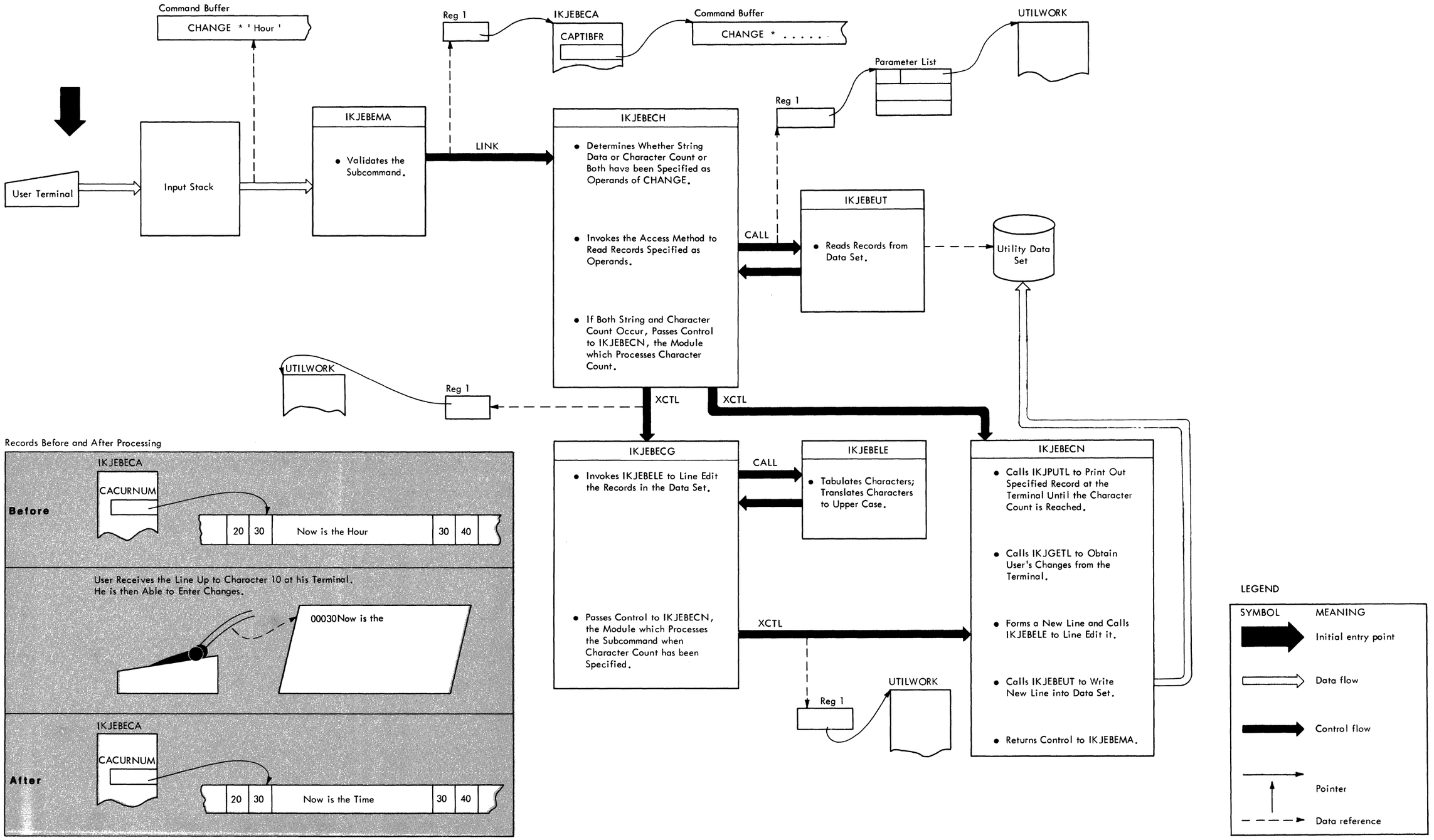


Method of Operation Diagram 4.
EDIT Attention Handling

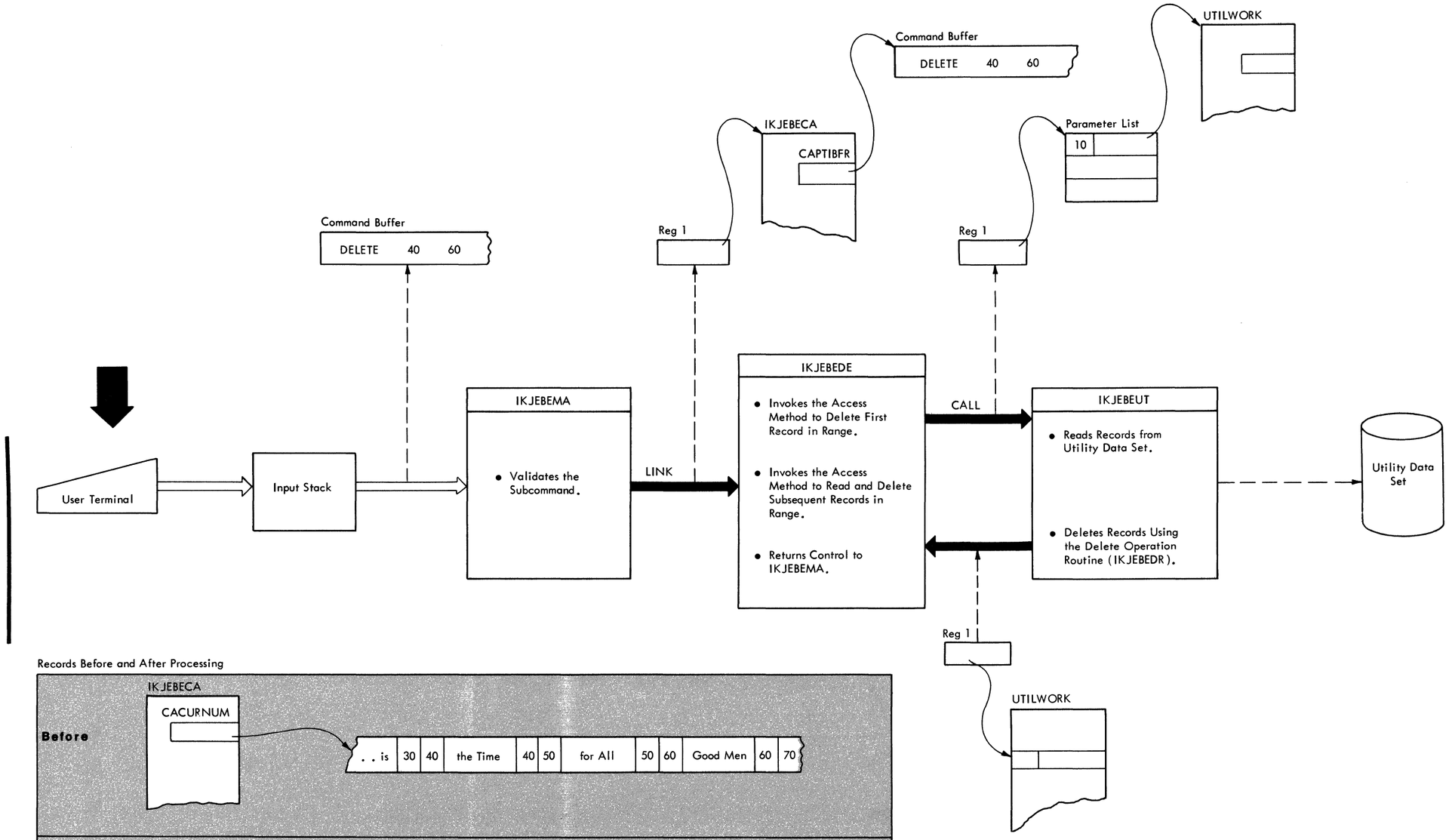


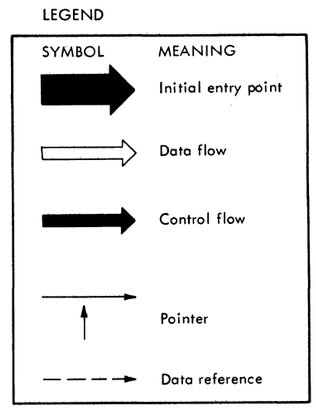


Method of Operation Diagram 5. BOTTOM Subcommand

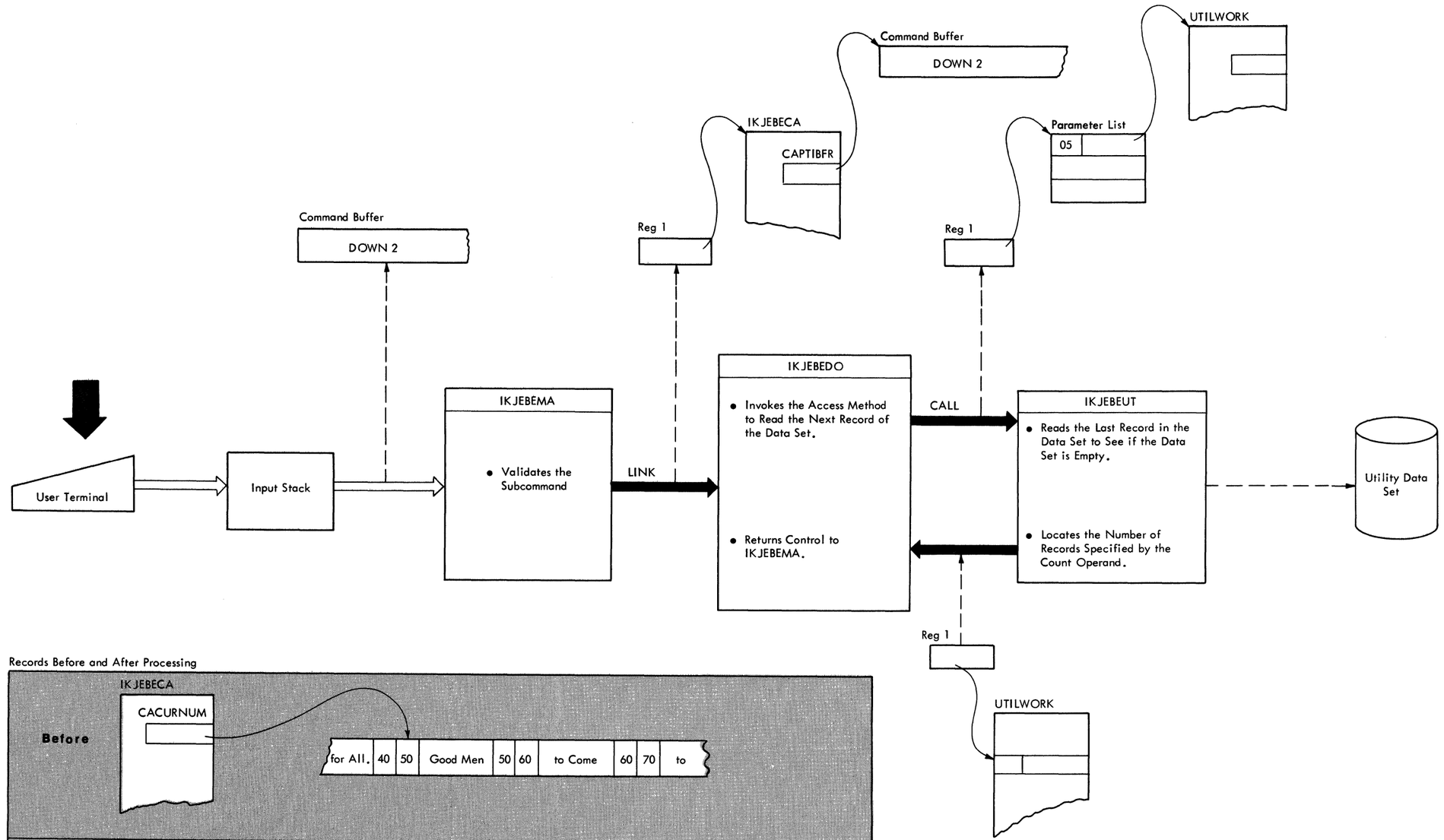


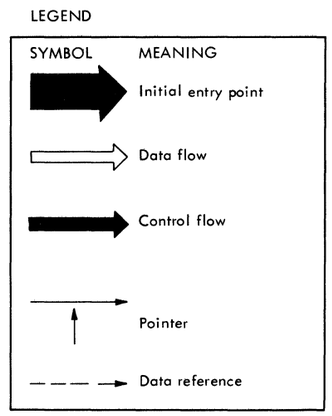
Method of Operation Diagram 6. CHANGE Subcommand



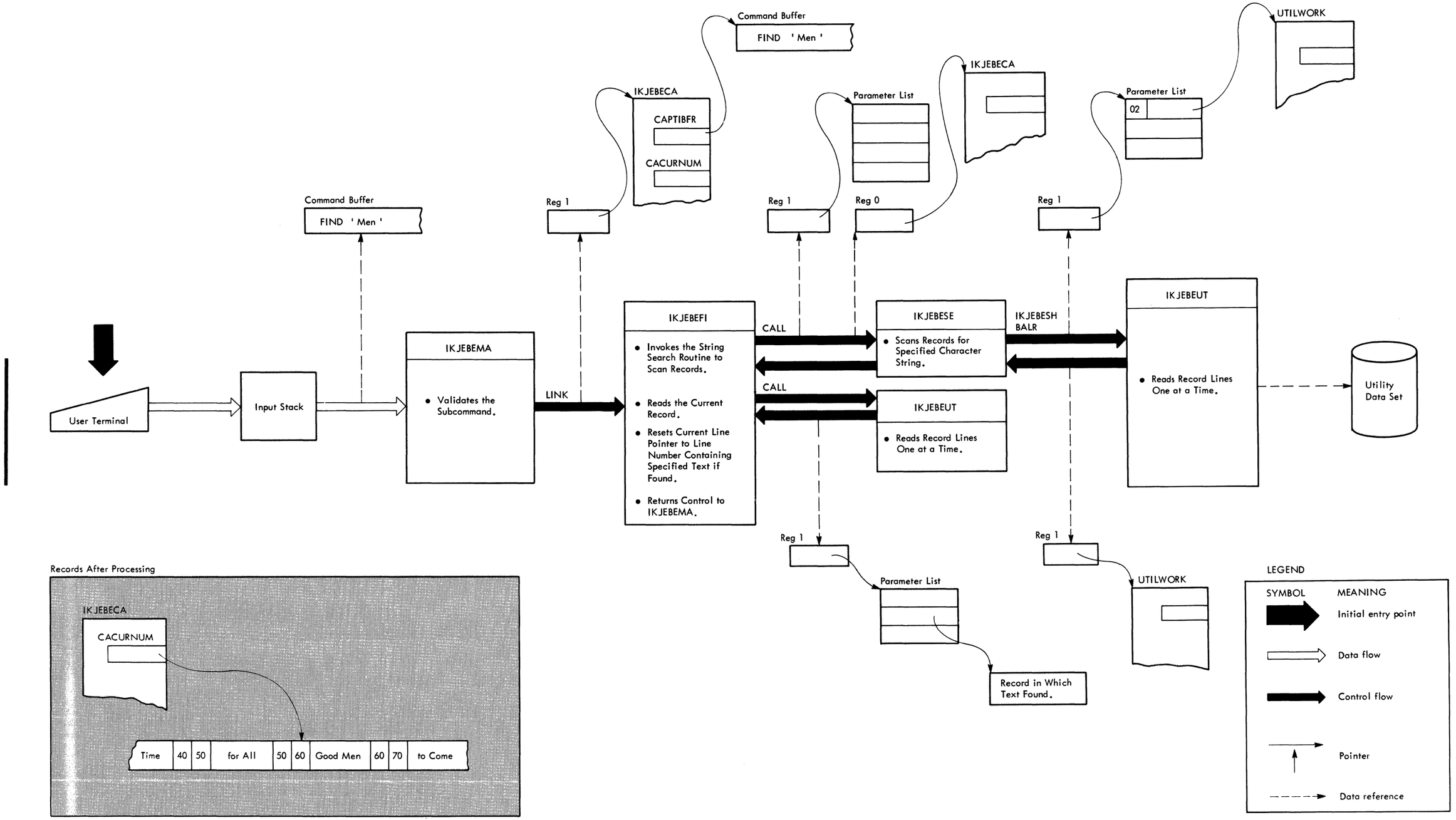


Method of Operation Diagram 7. DELETE Subcommand

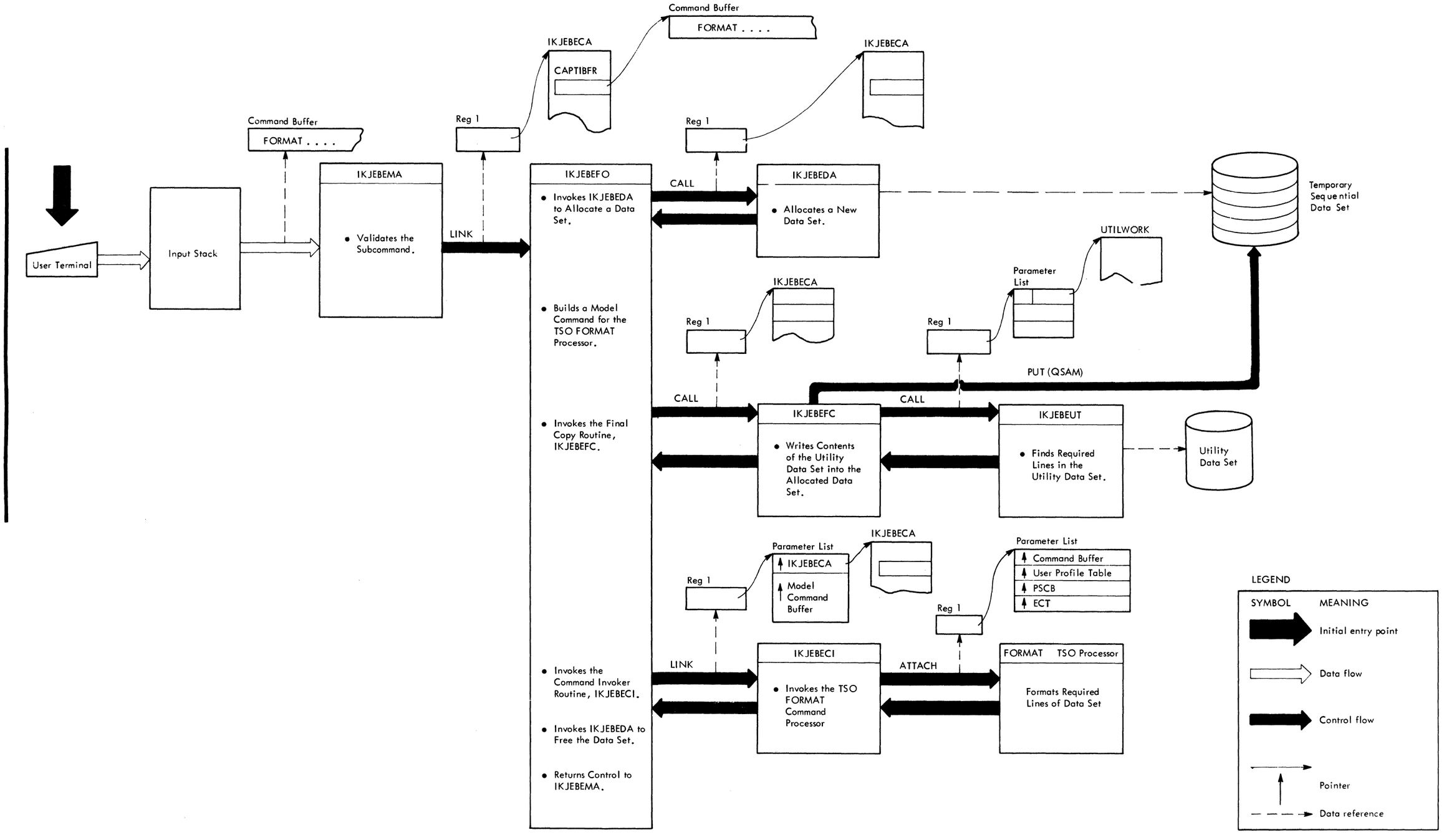




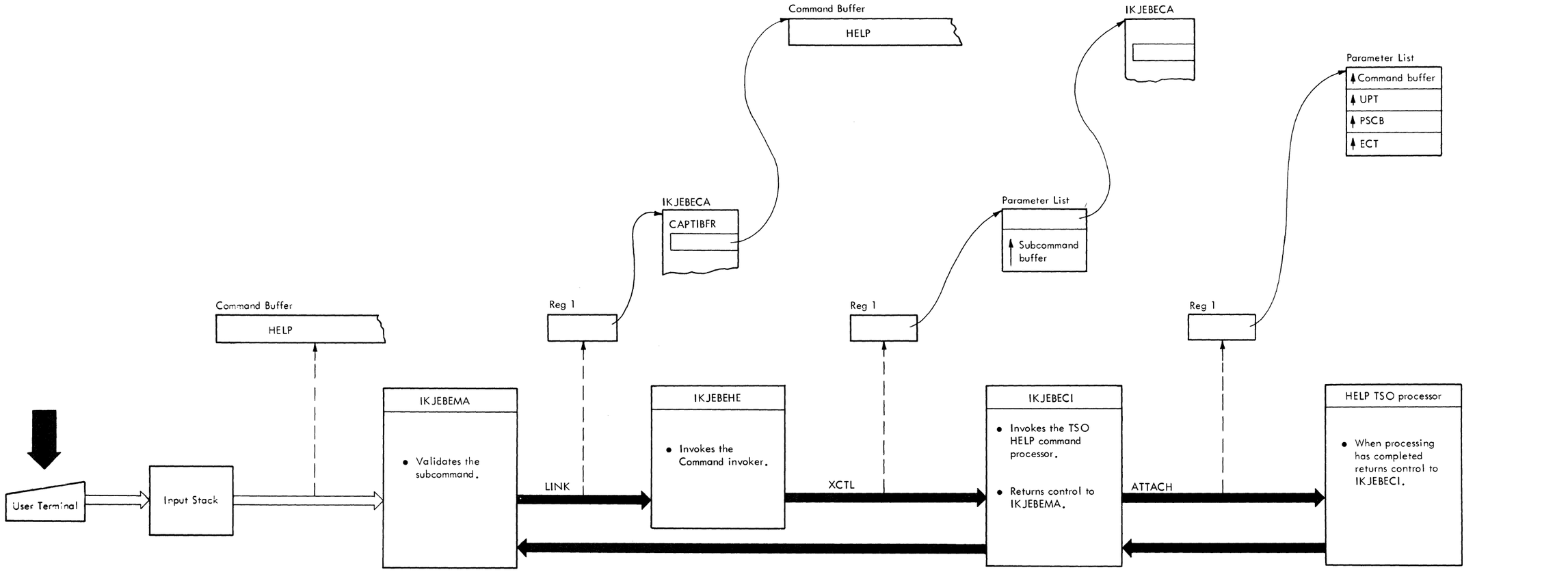
Method of Operation Diagram 8. DOWN Subcommand

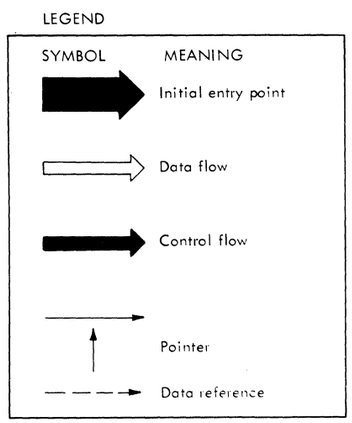


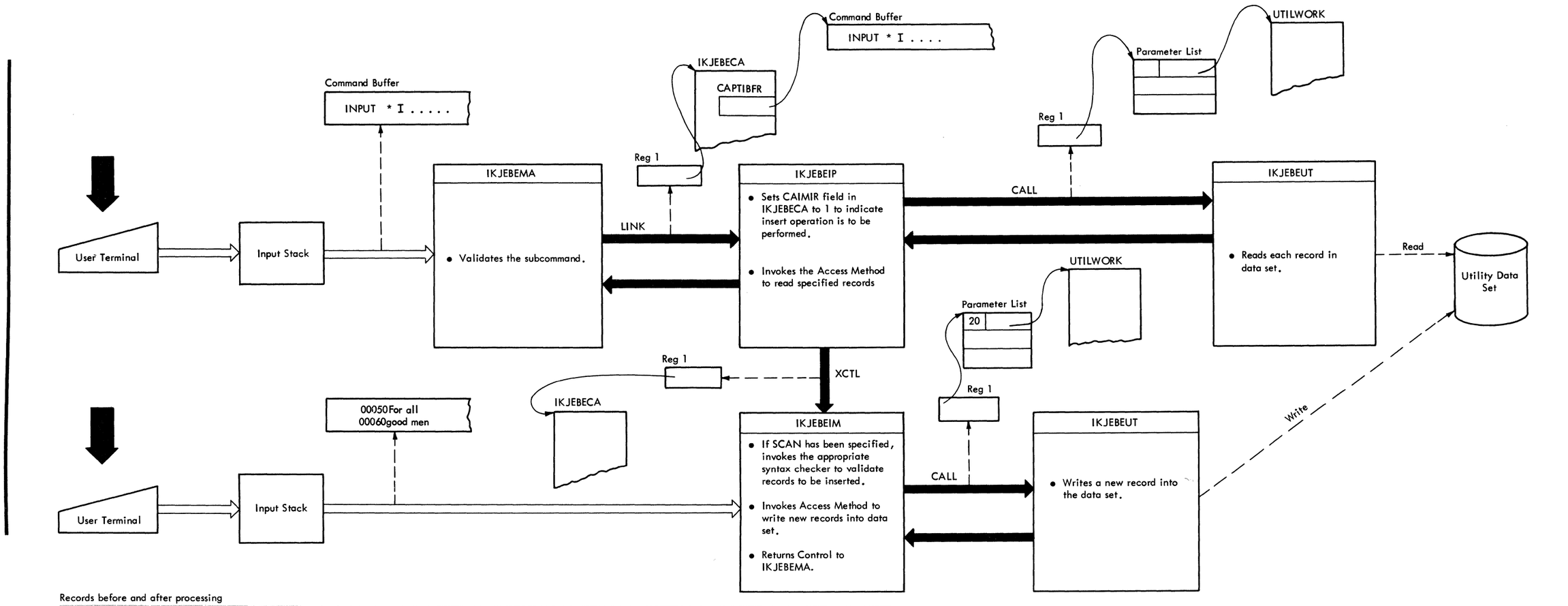
Method of Operation Diagram 9. FIND Subcommand



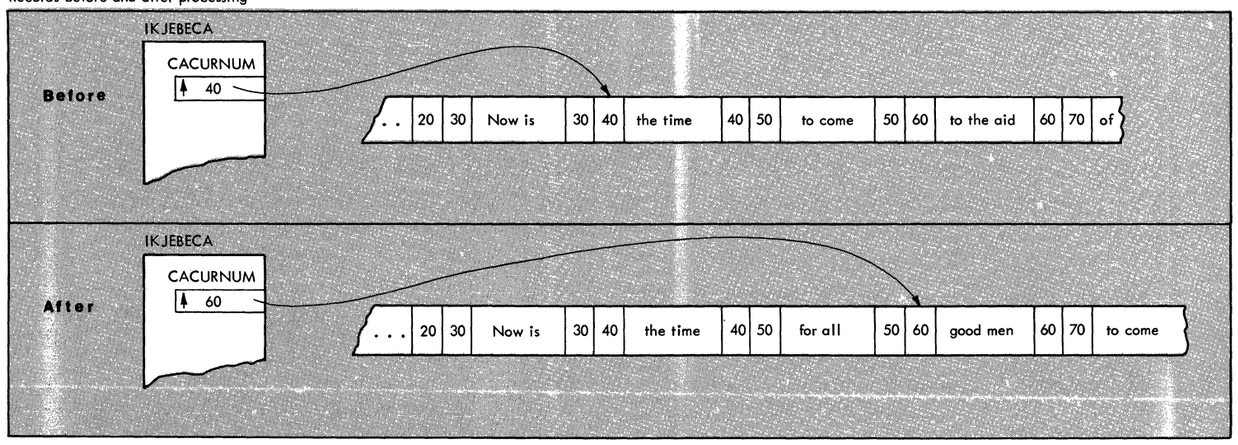
Method of Operation Diagram 10. FORMAT Subcommand

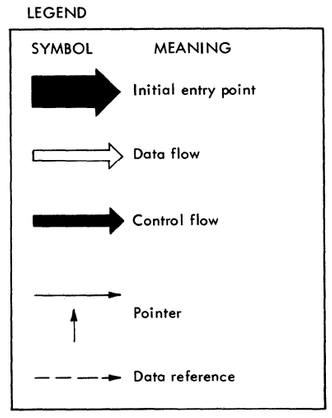

Command Buffer
HELP
IKJEBECA
CAPTIBFR
Parameter List
Subcommand buffer
IKJEBECA
Parameter List
Command buffer
UPT
PSCB
ECT
Command Buffer
HELP
Reg 1
Reg 1
Reg 1
IKJEBEMA
• Validates the subcommand.
LINK
IKJEBEHE
• Invokes the Command invoker.
XCTL
IKJEBECI
• Invokes the TSO HELP command processor.
• Returns control to IKJEBEMA.
ATTACH
HELP TSO processor
• When processing has completed returns control to IKJEBECI.
User Terminal
Input Stack


LEGEND
SYMBOL
MEANING
Initial entry point
Data flow
Control flow
Pointer
Data reference

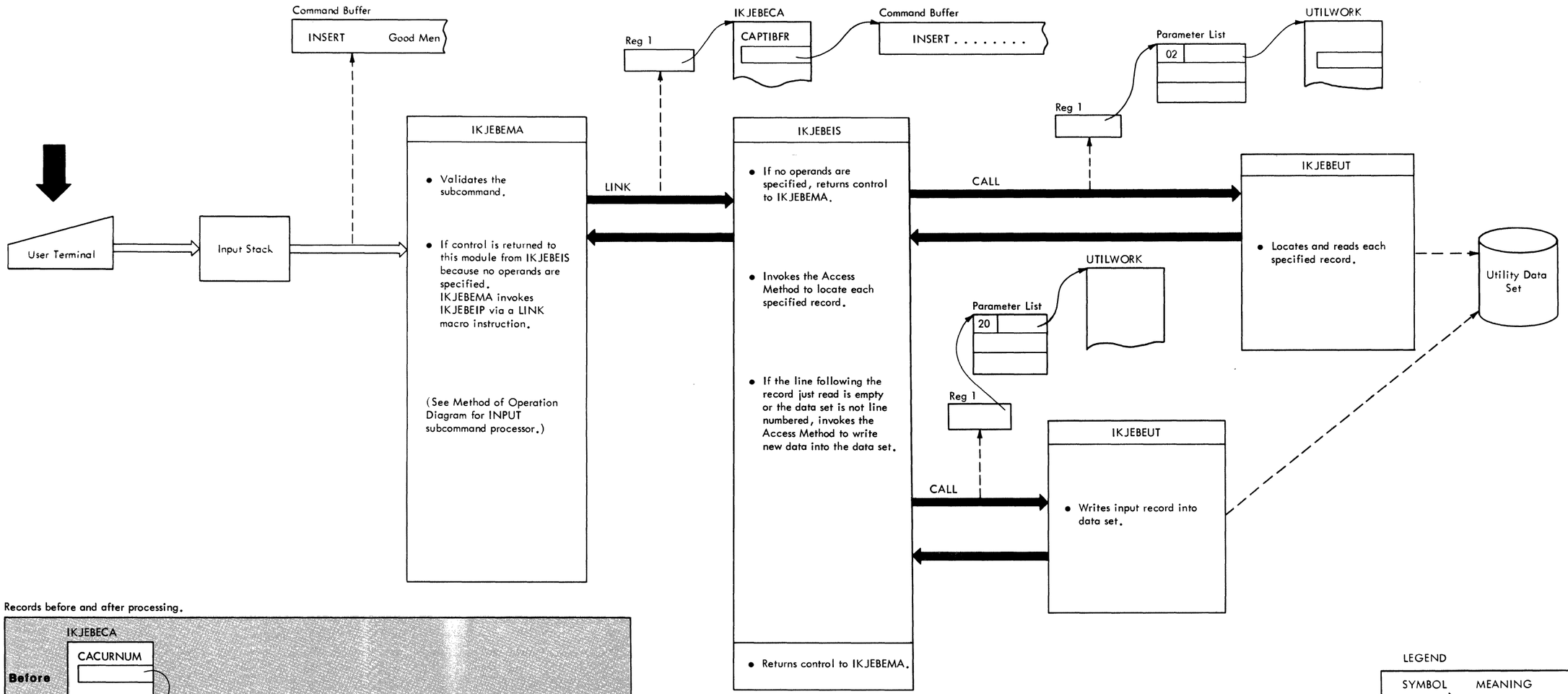


Records before and after processing

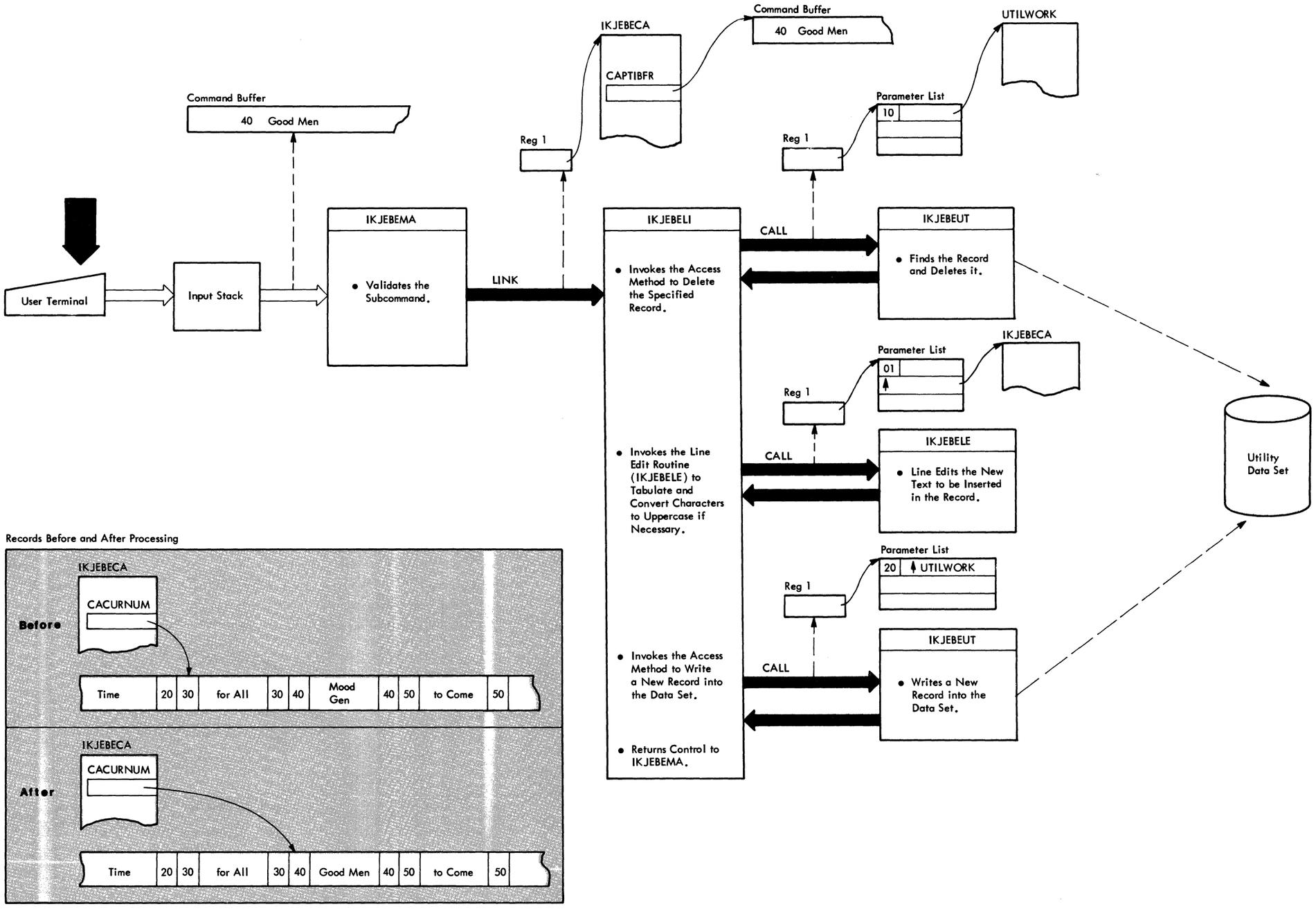




Method of Operation Diagram 12. INPUT Subcommand

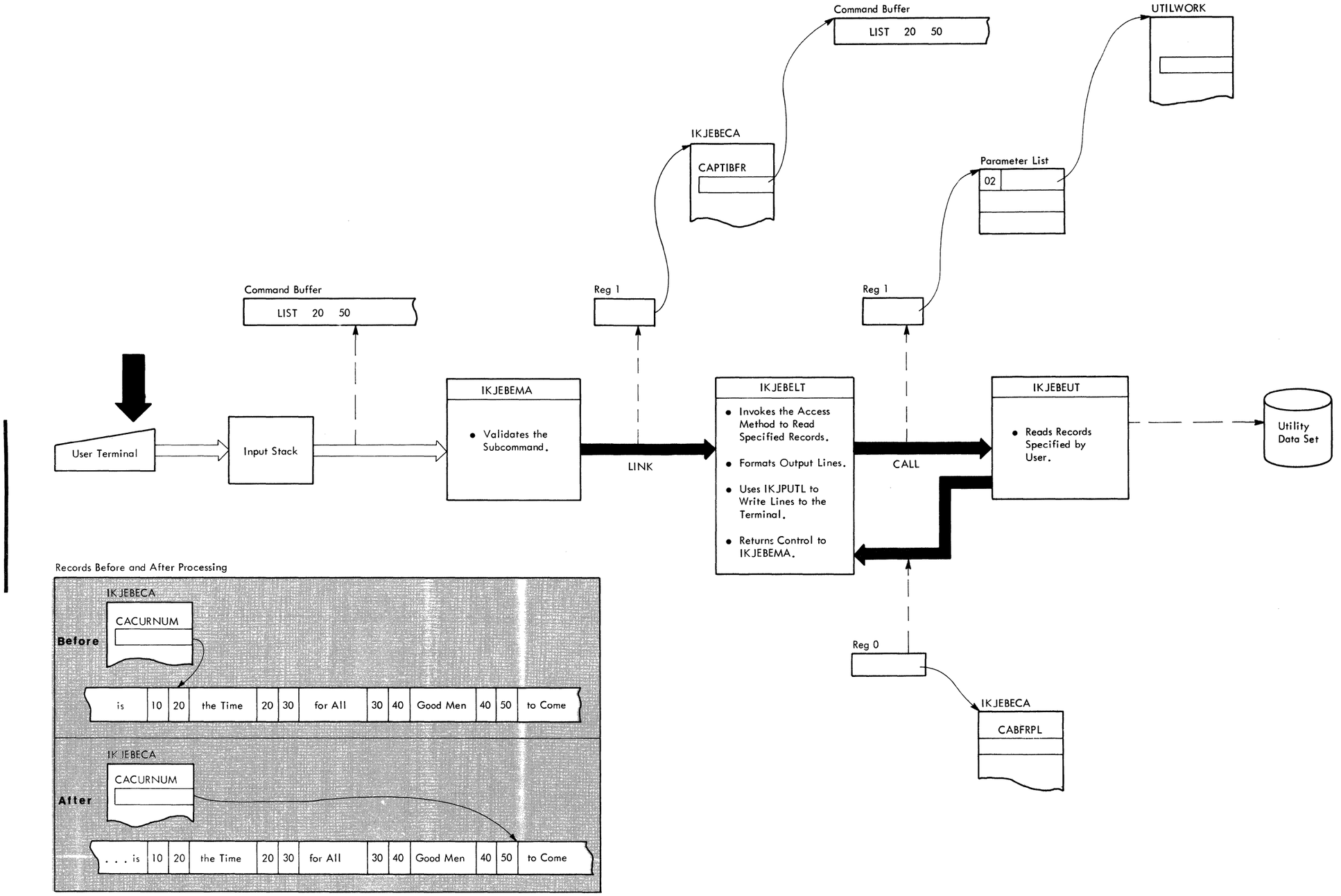


Method of Operation Diagram 13. INSERT Subcommand

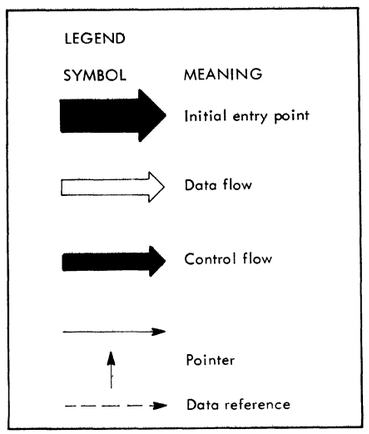


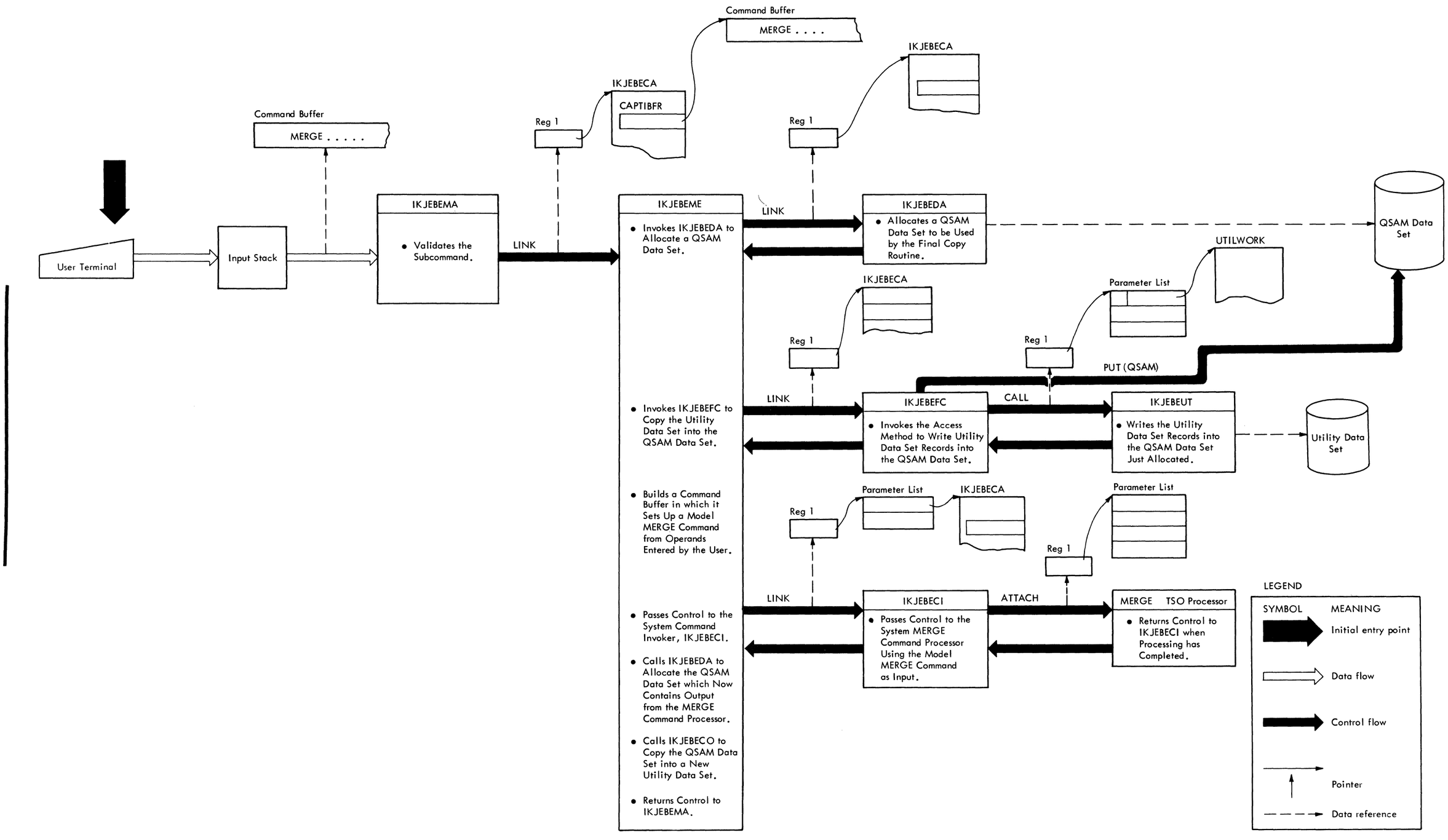
Records Before and After Processing

Method of Operation Diagram 14. Line Insert/Replace/Delete Subcommand

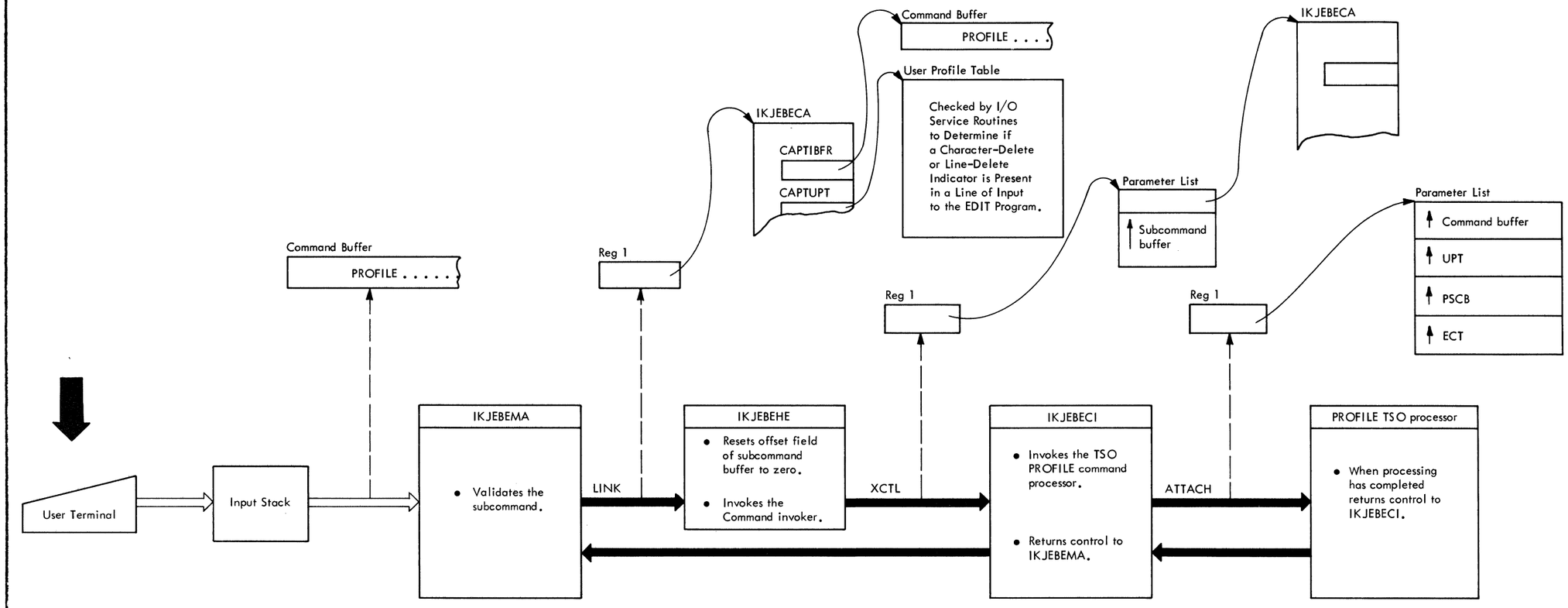


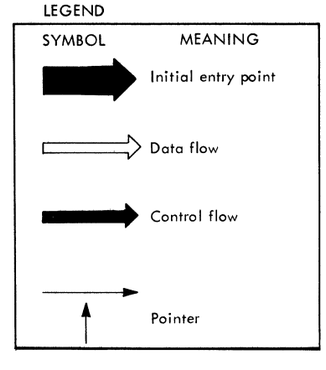
Records Before and After Processing



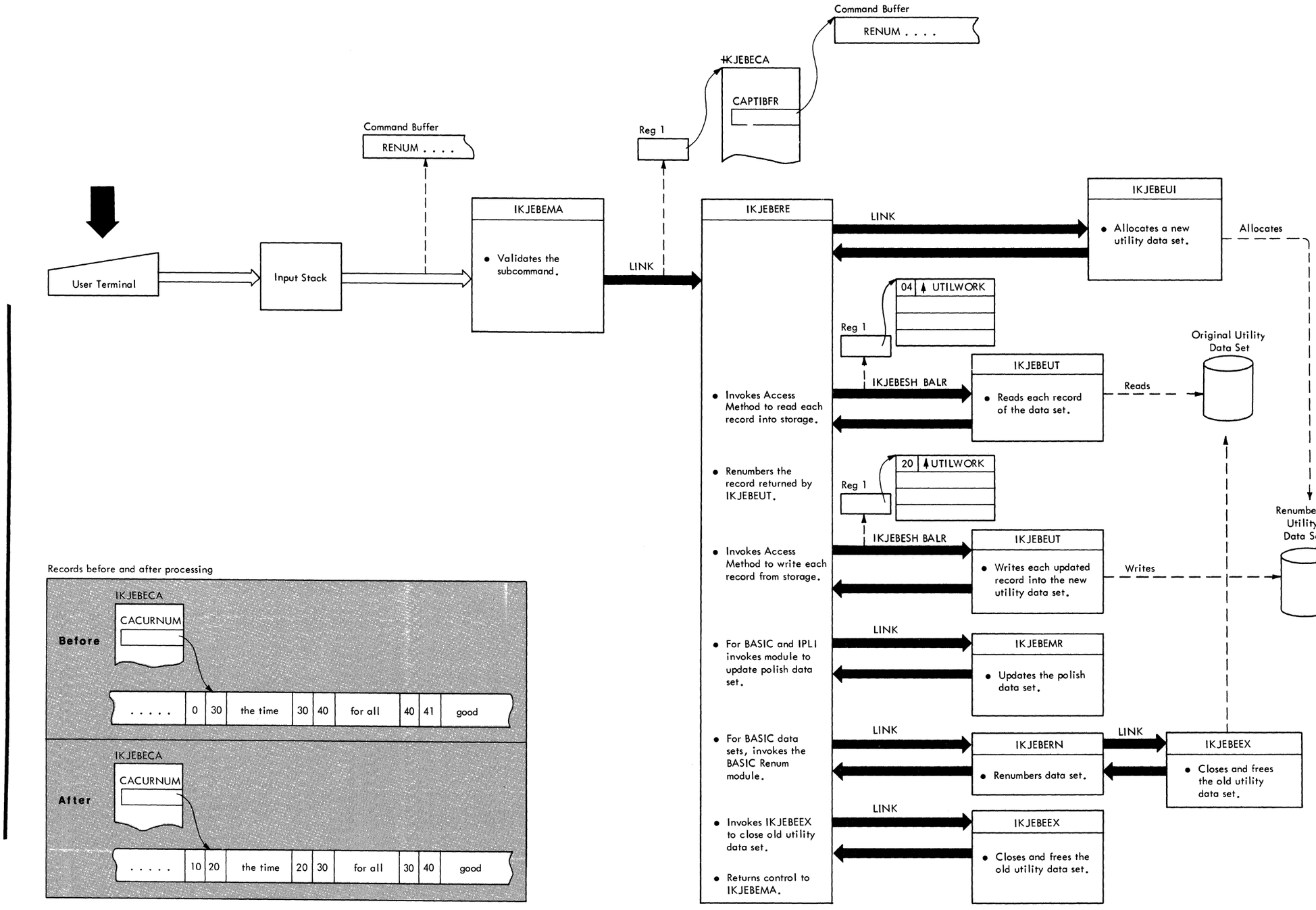


Method of Operation Diagram 16. MERGE Subcommand



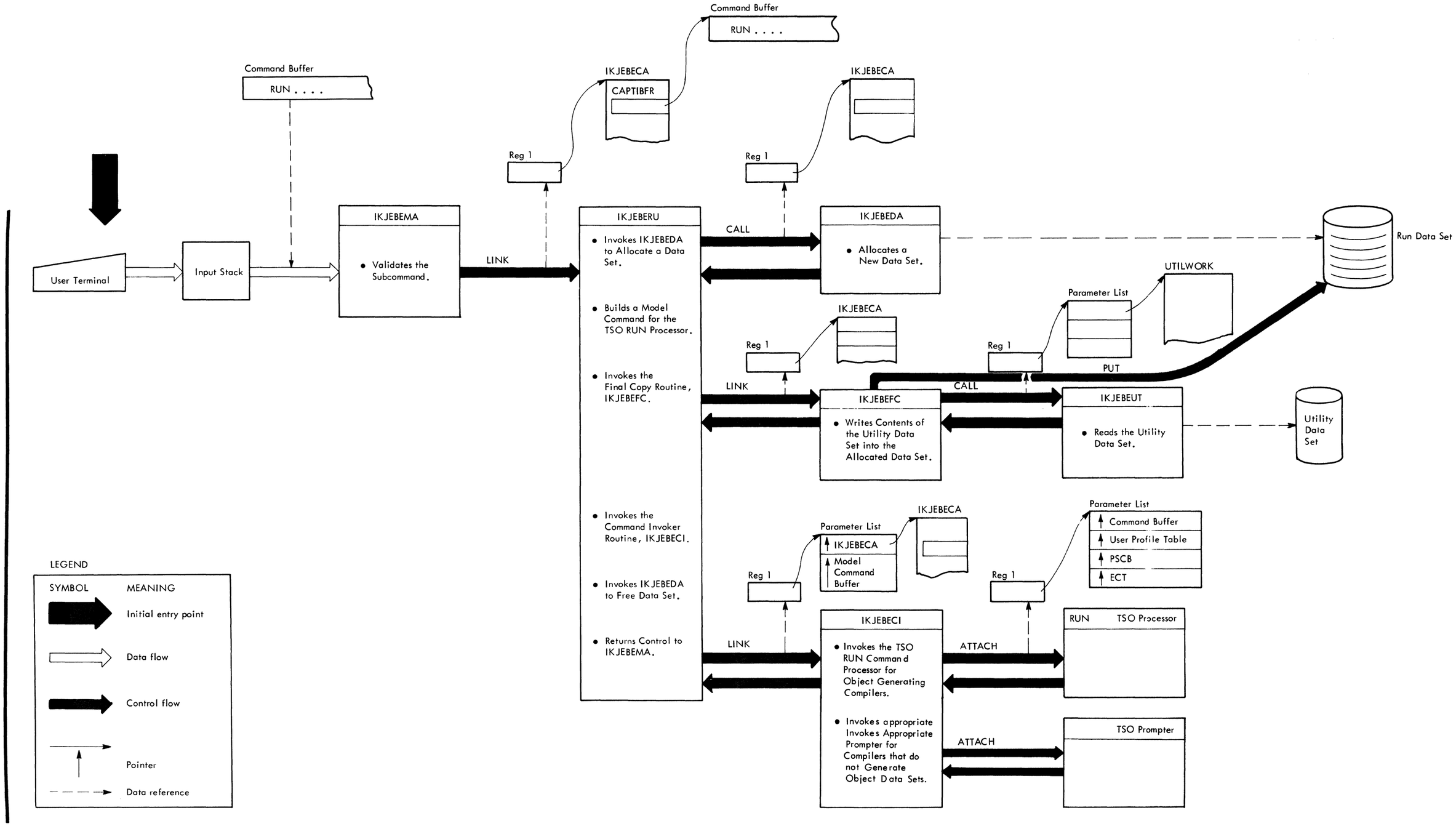


Method of Operation Diagram 17. PROFILE Subcommand

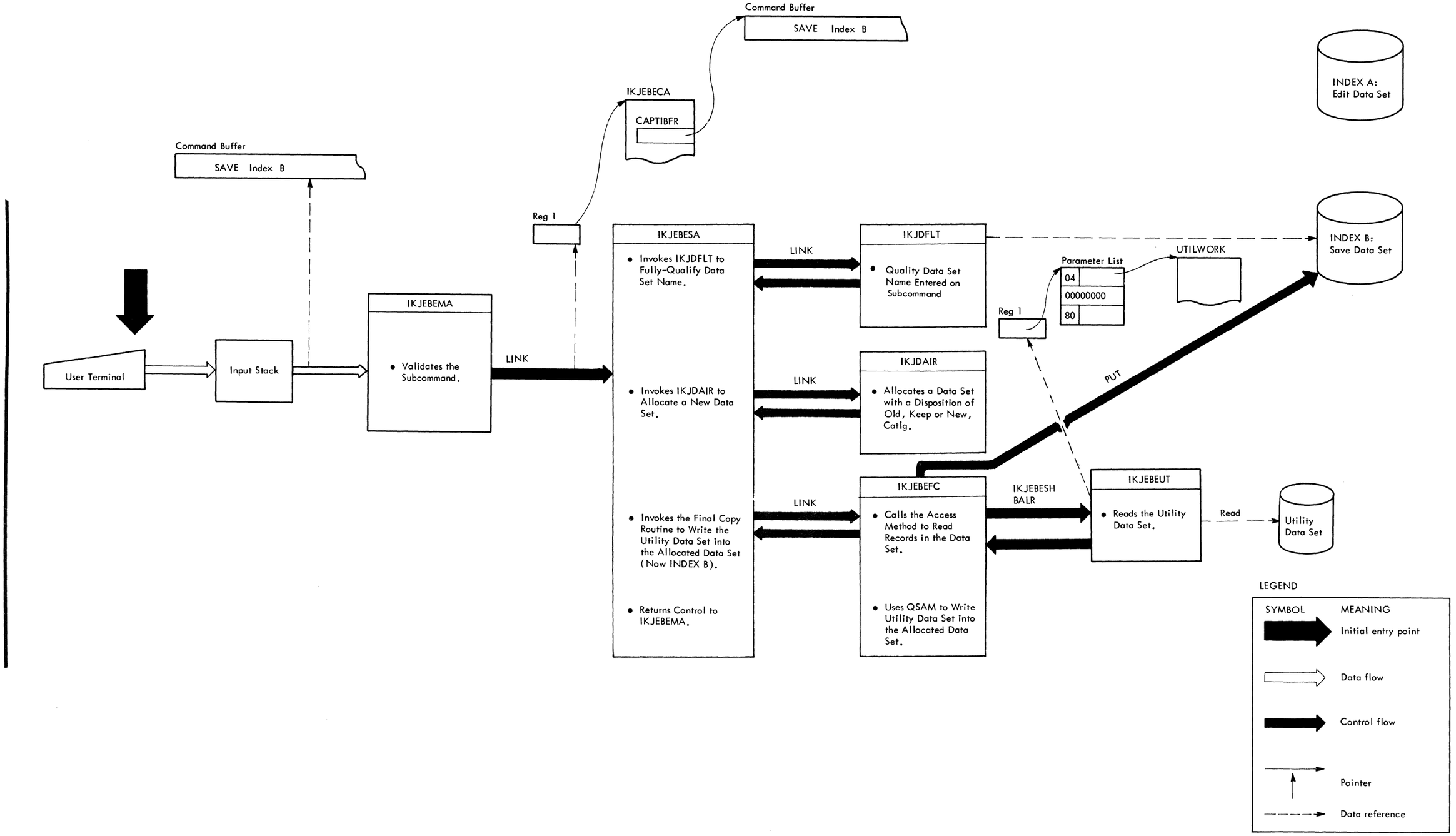


Records before and after processing

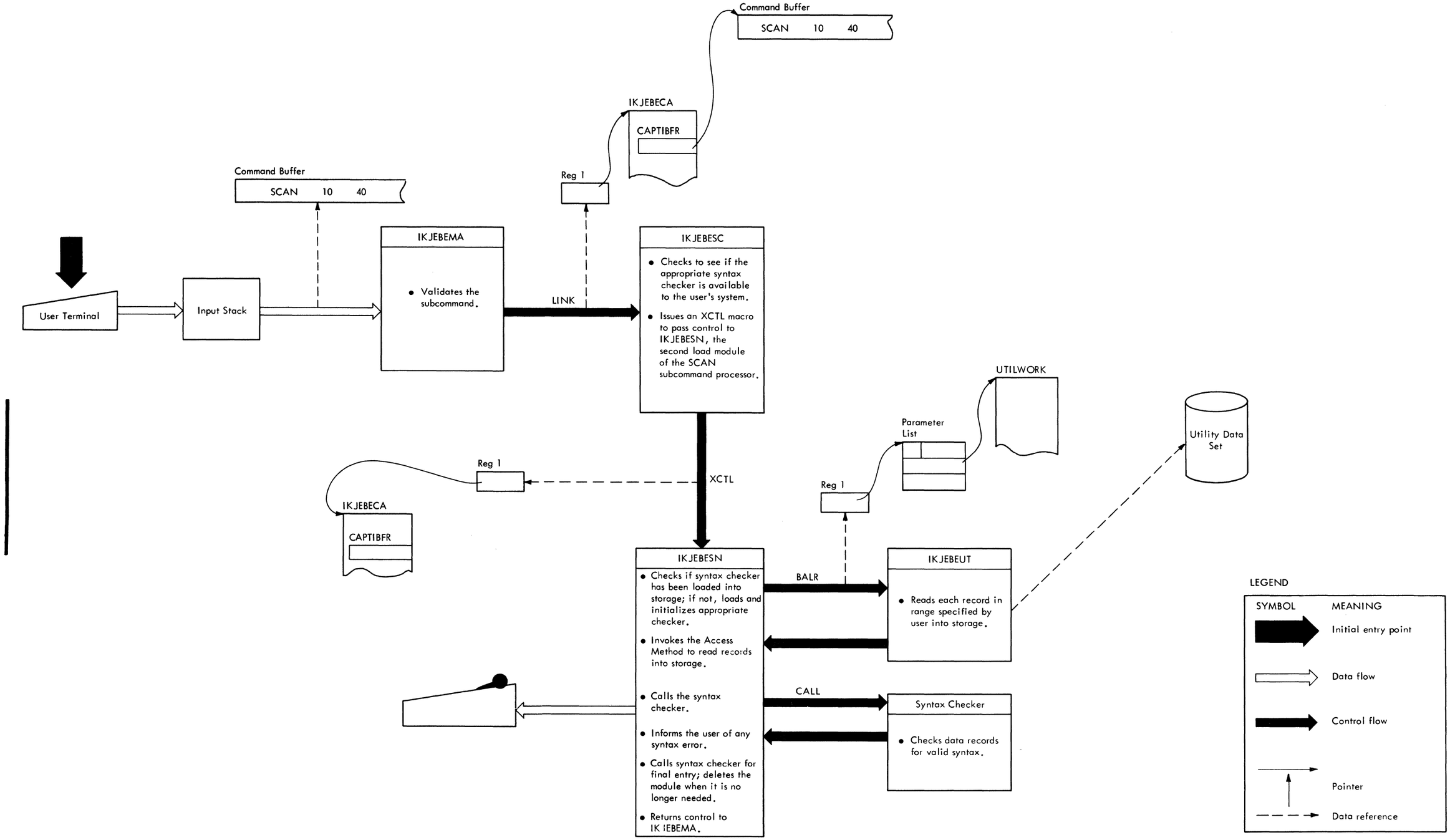
Method of Operation Diagram 18. RENUM Subcommand



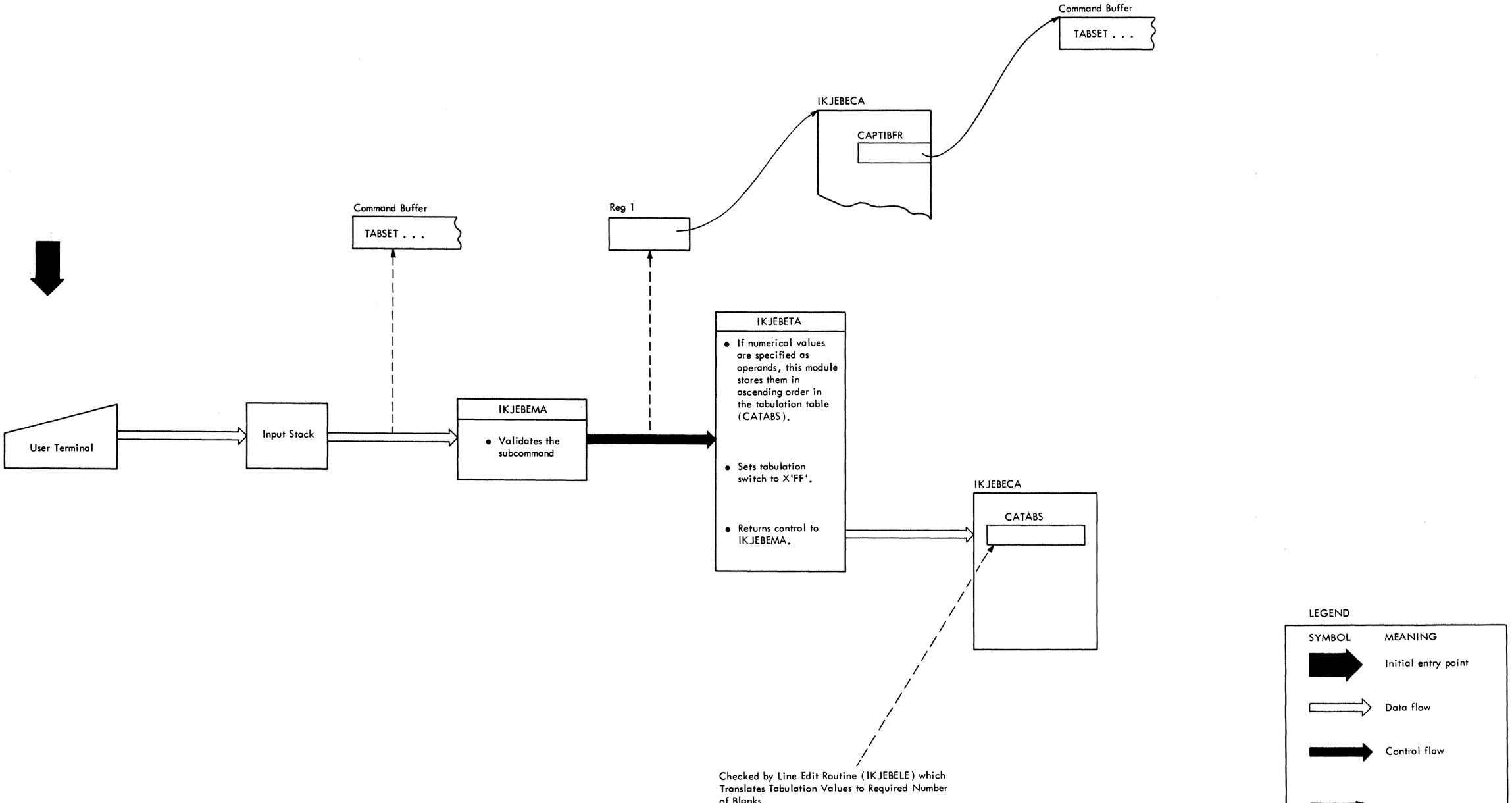
Method of Operation Diagram 19. RUN Subcommand



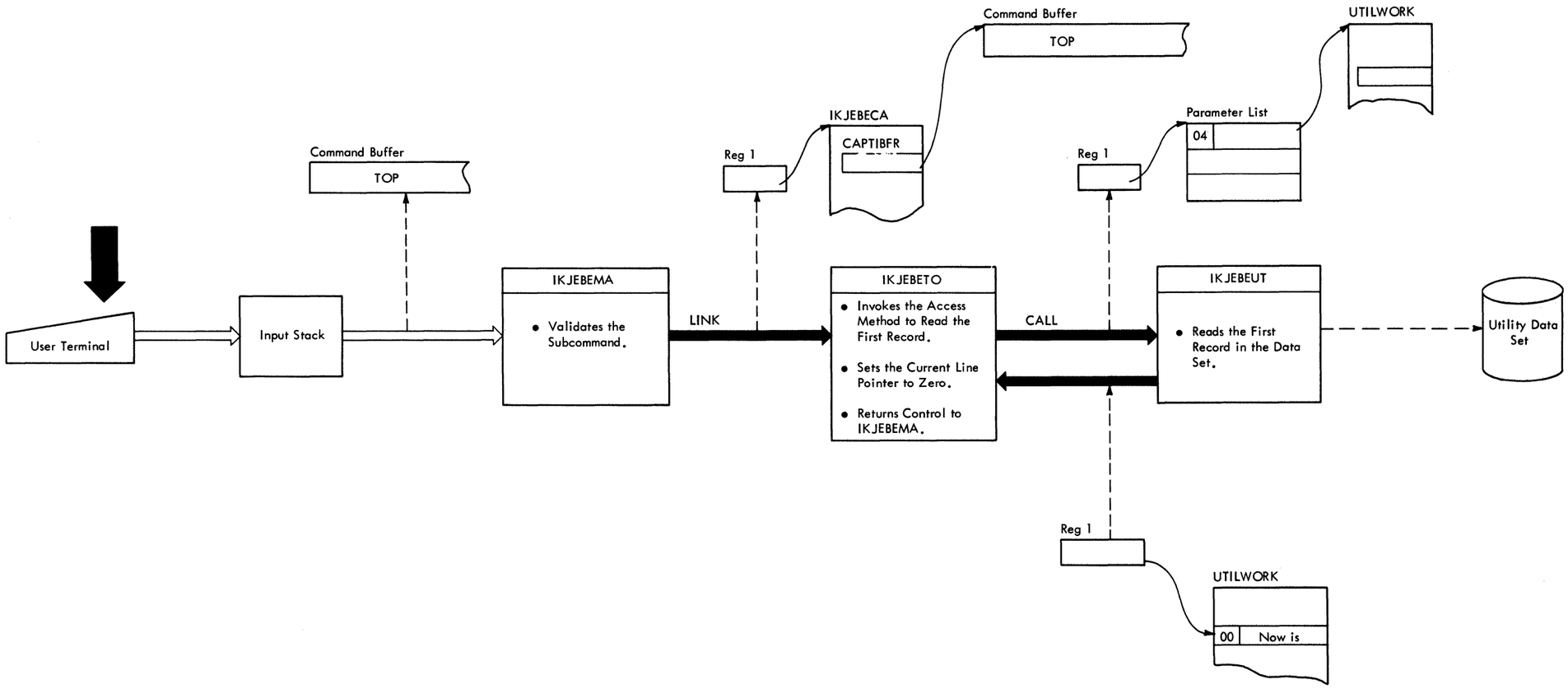
Method of Operation Diagram 20. SAVE Subcommand

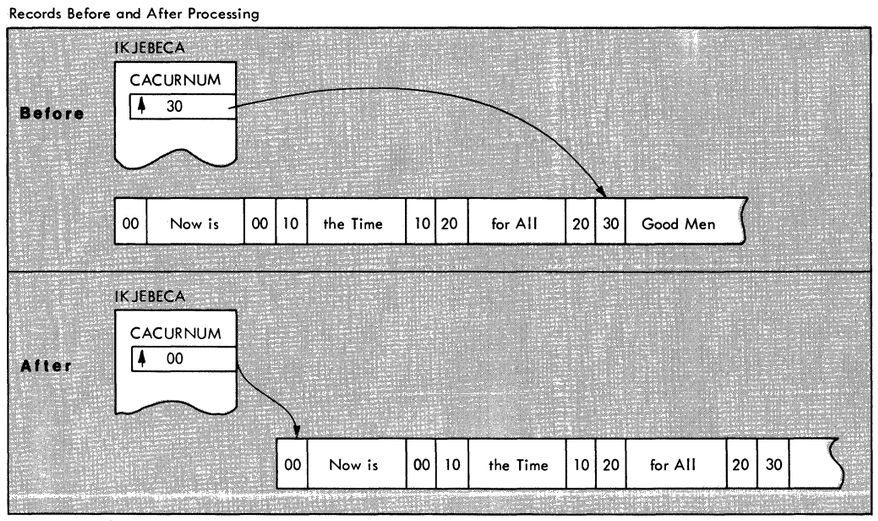


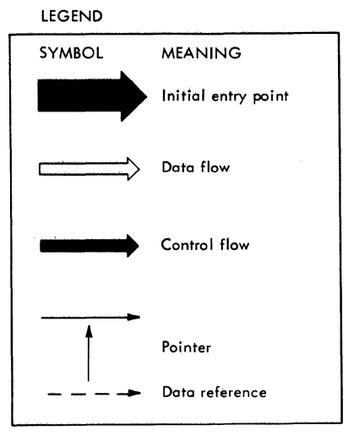
Method of Operation Diagram 21. SCAN Subcommand



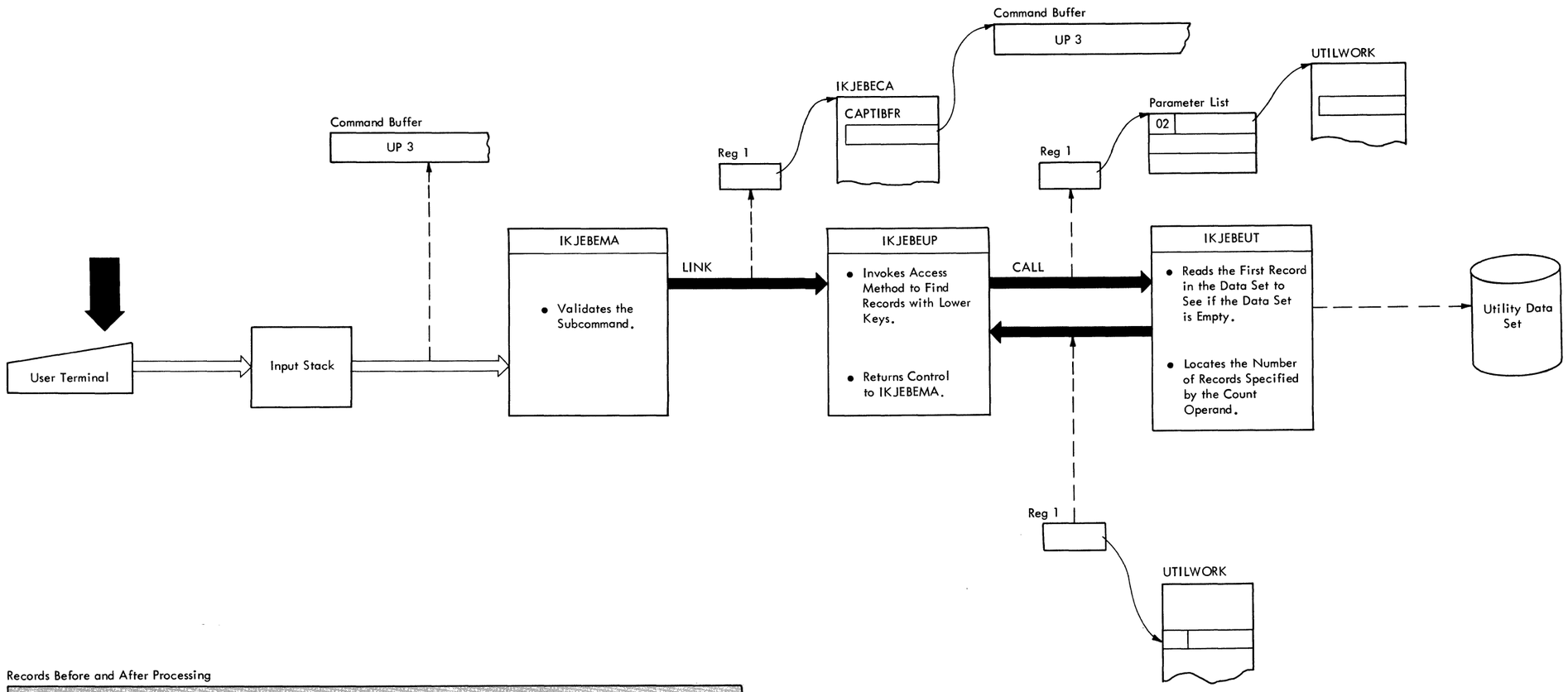
Method of Operation Diagram 22. TABSET Subcommand

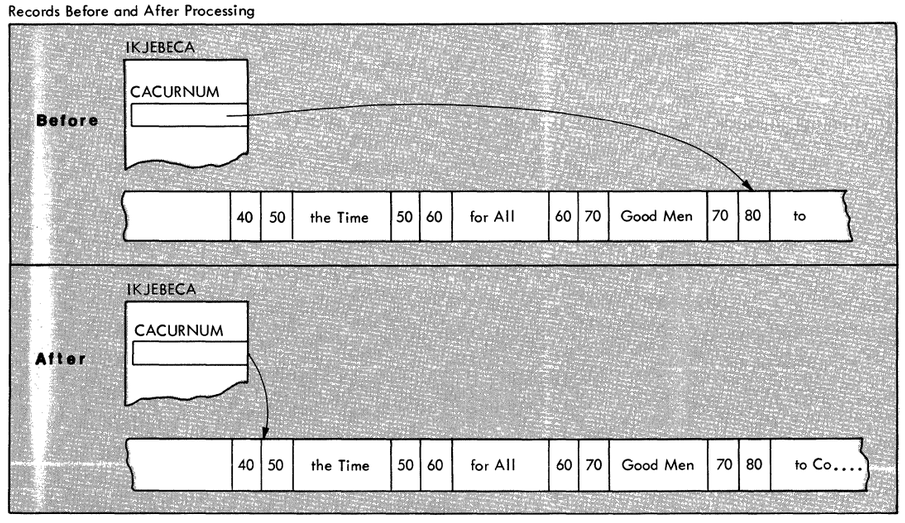


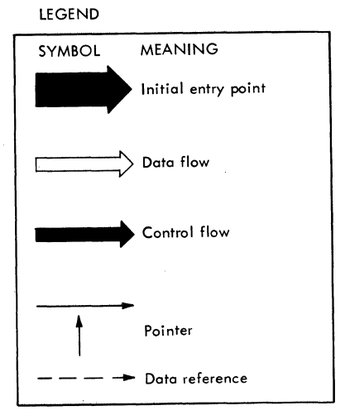




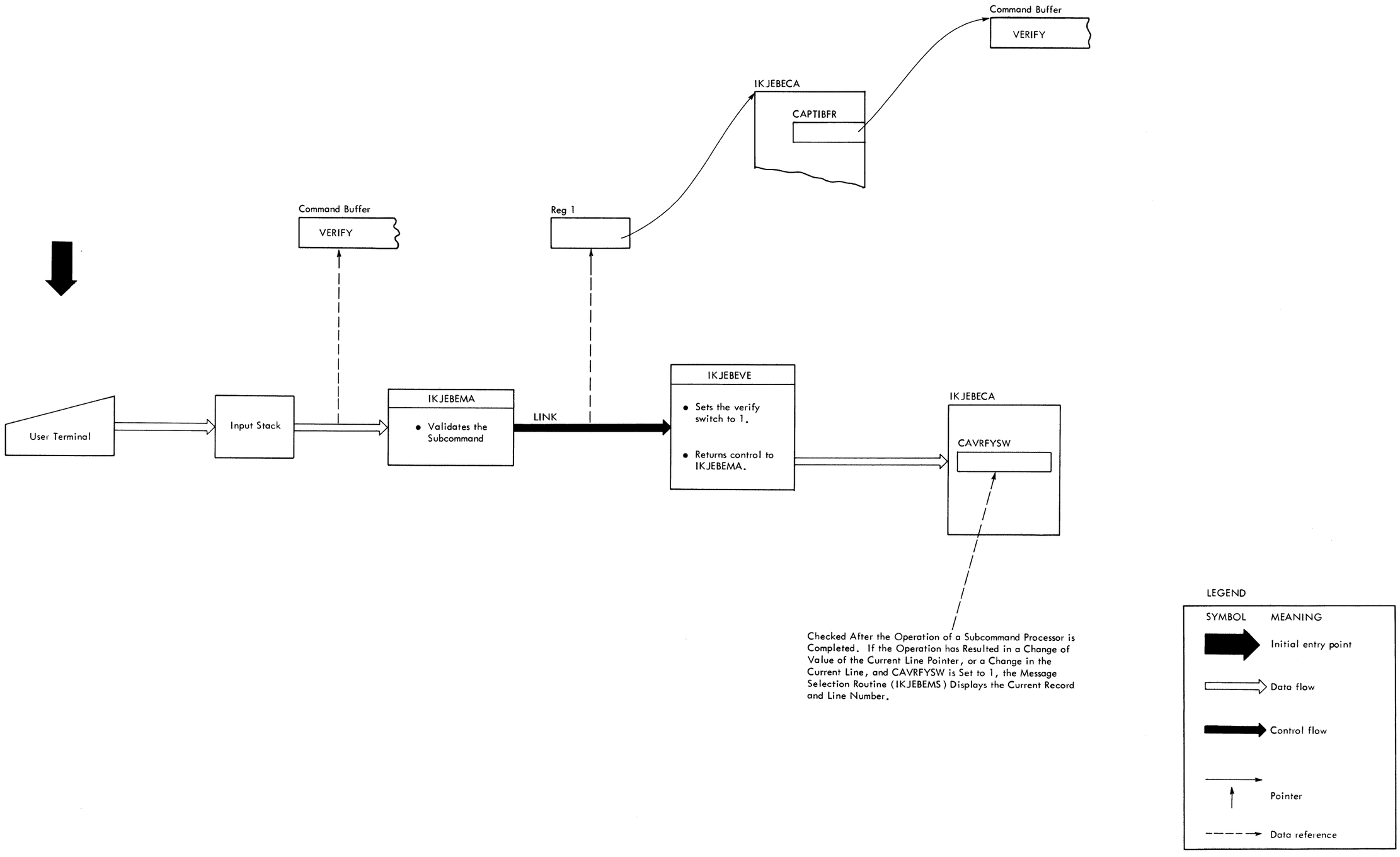
Method of Operation Diagram 23. TOP Subcommand



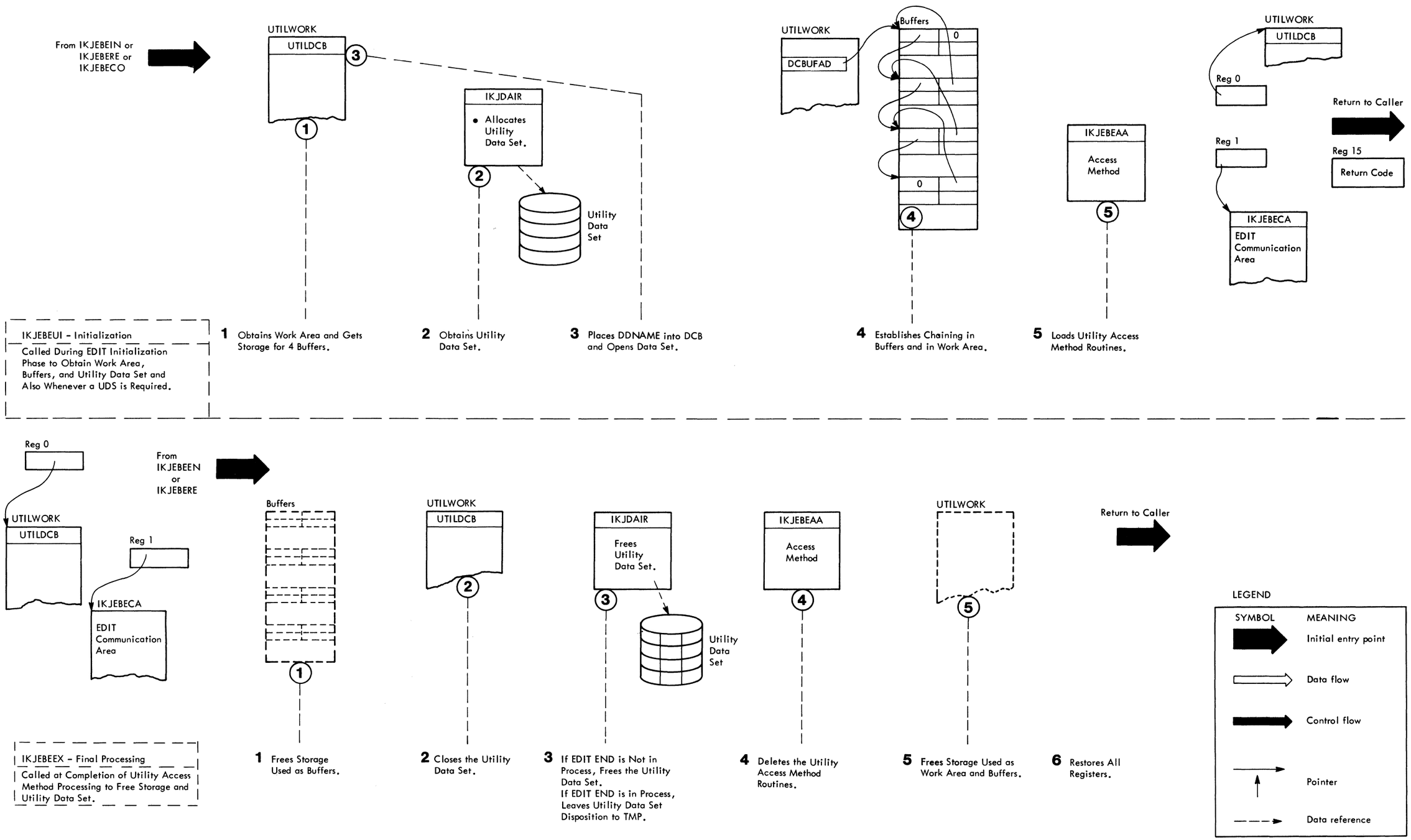




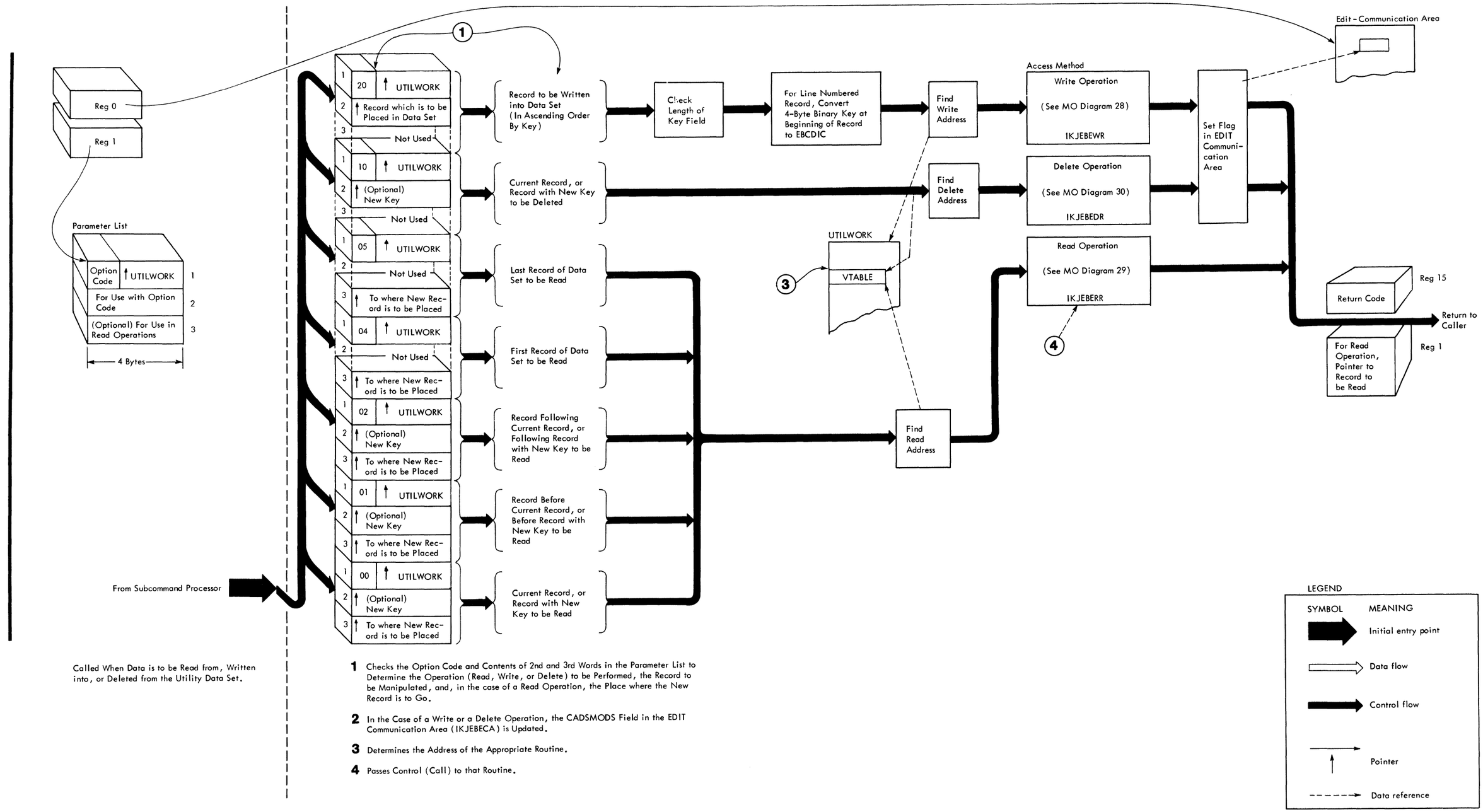
Method of Operation Diagram 24. UP Subcommand



Method of Operation Diagram 25. VERIFY Subcommand



Method of Operation Diagram 26. EDIT Access Method Initialization and Final Processing (IKJEBEUI and IKJEBEEX)



1 Checks the Option Code and Contents of 2nd and 3rd Words in the Parameter List to Determine the Operation (Read, Write, or Delete) to be Performed, the Record to be Manipulated, and, in the case of a Read Operation, the Place where the New Record is to Go.

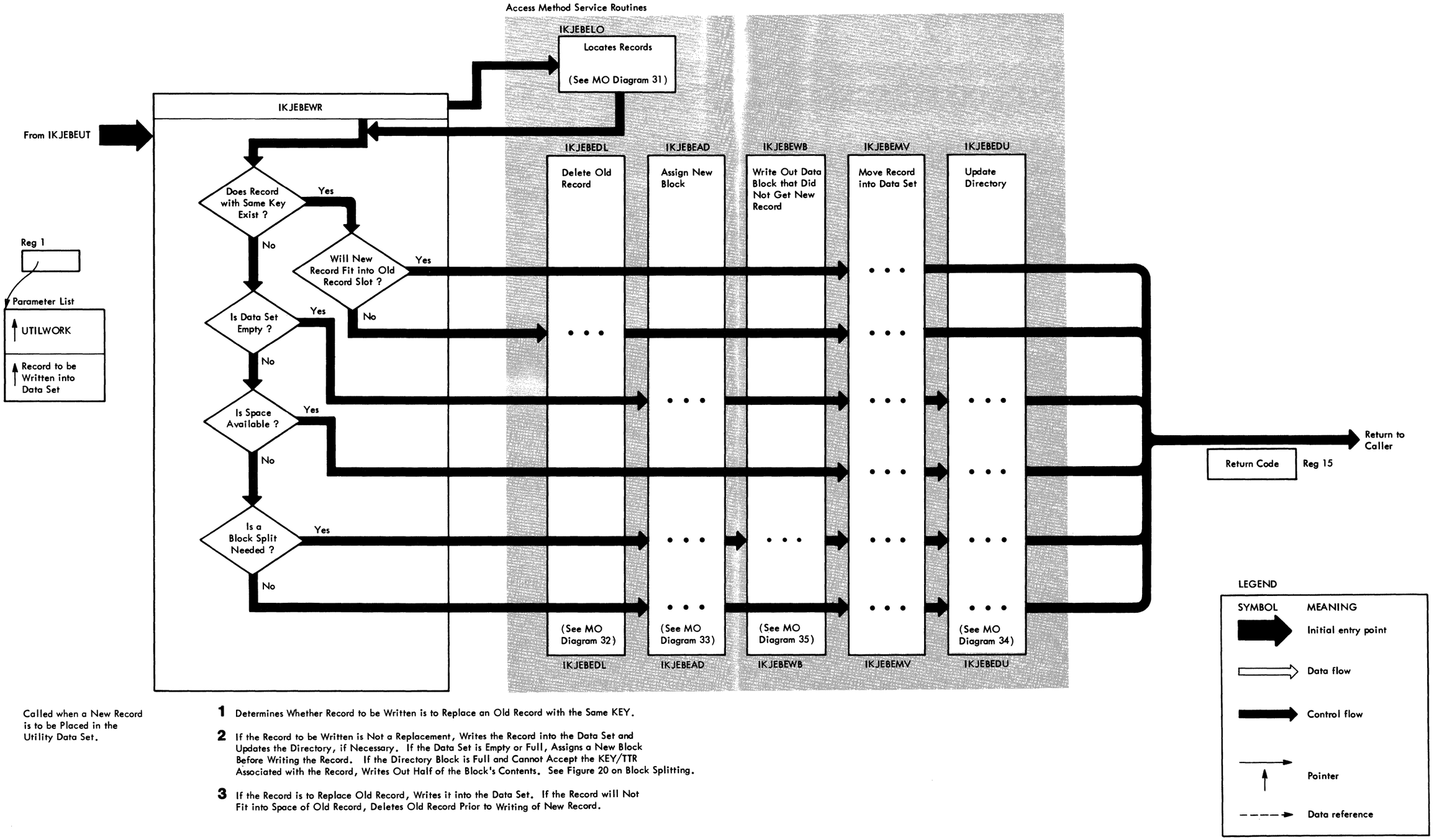
2 In the Case of a Write or a Delete Operation, the CADSMODS Field in the EDIT Communication Area (IKJEBECA) is Updated.

3 Determines the Address of the Appropriate Routine.

4 Passes Control (Call) to that Routine.

LEGEND

SYMBOL	MEANING
➡	Initial entry point
⇨	Data flow
➡	Control flow
→	Pointer
- - - - →	Data reference



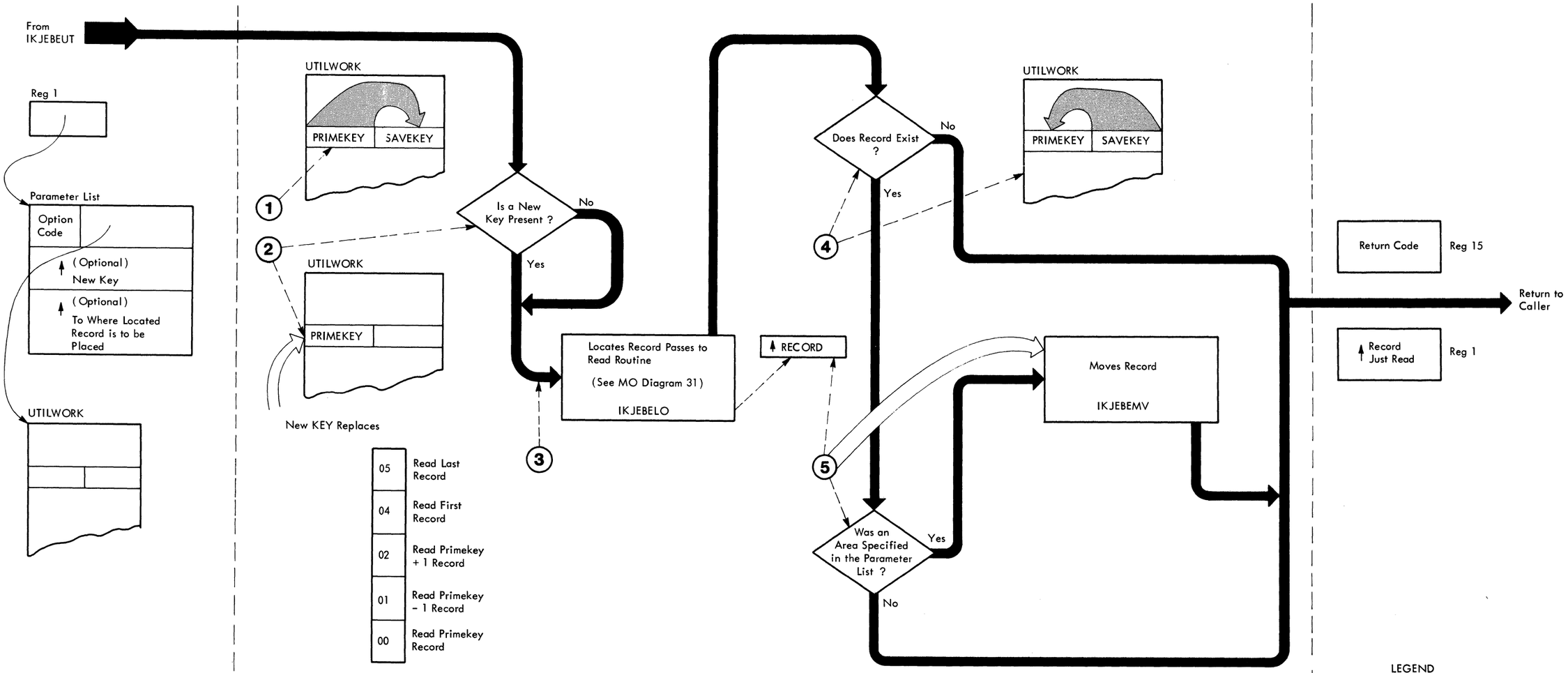
Called when a New Record is to be Placed in the Utility Data Set.

1 Determines Whether Record to be Written is to Replace an Old Record with the Same KEY.

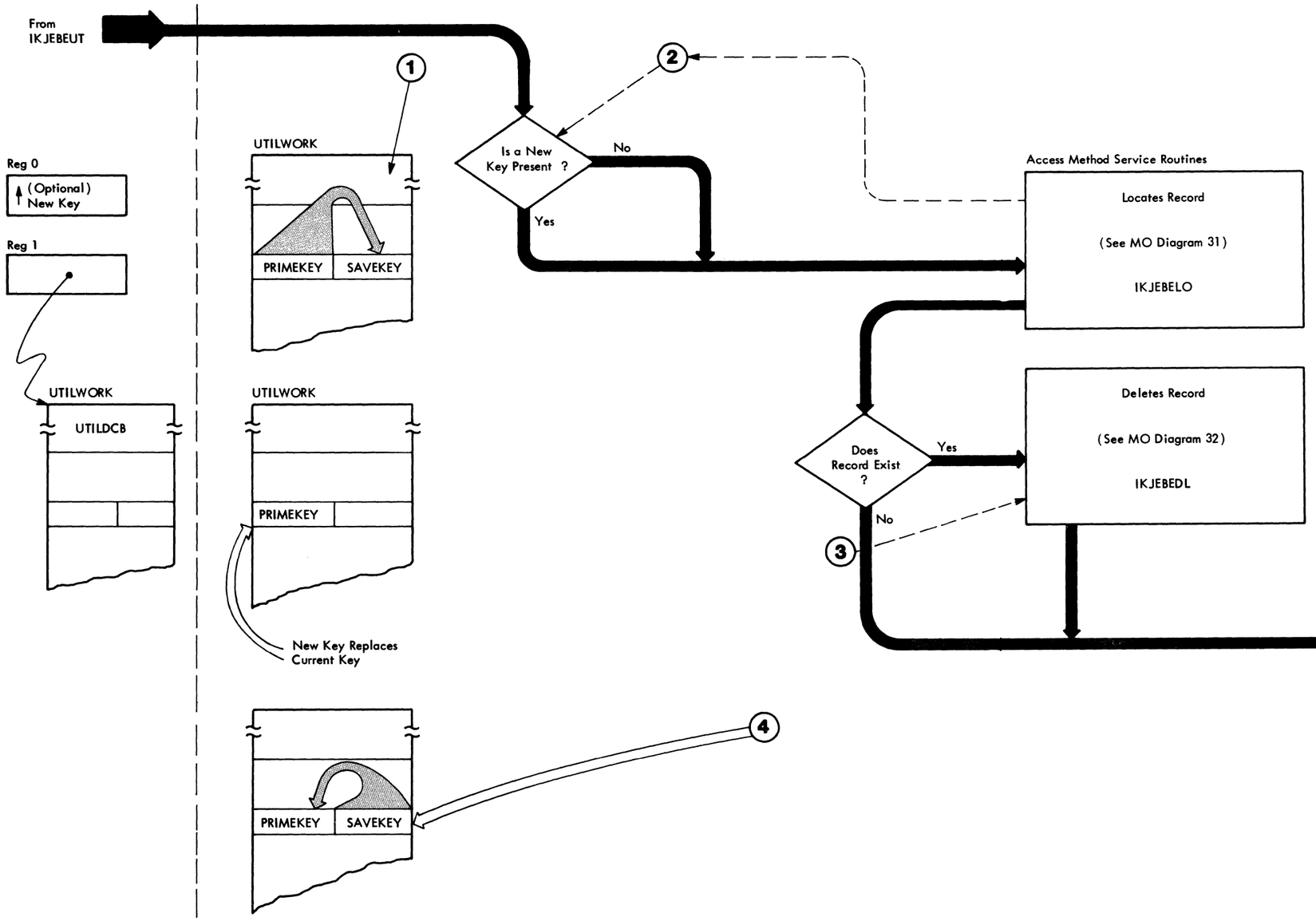
2 If the Record to be Written is Not a Replacement, Writes the Record into the Data Set and Updates the Directory, if Necessary. If the Data Set is Empty or Full, Assigns a New Block Before Writing the Record. If the Directory Block is Full and Cannot Accept the KEY/TTR Associated with the Record, Writes Out Half of the Block's Contents. See Figure 20 on Block Splitting.

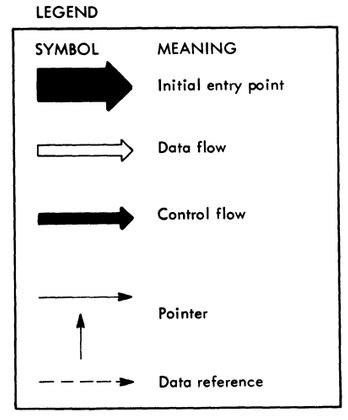
3 If the Record is to Replace Old Record, Writes it into the Data Set. If the Record will Not Fit into Space of Old Record, Deletes Old Record Prior to Writing of New Record.

Method of Operation Diagram 28. EDIT Access Method Write Operation (IKJEBEWR)



Method of Operation Diagram 29. EDIT Access Method Read Operation (IKJEBERR)





Called when a Record is to be Deleted from the Utility Data Set.

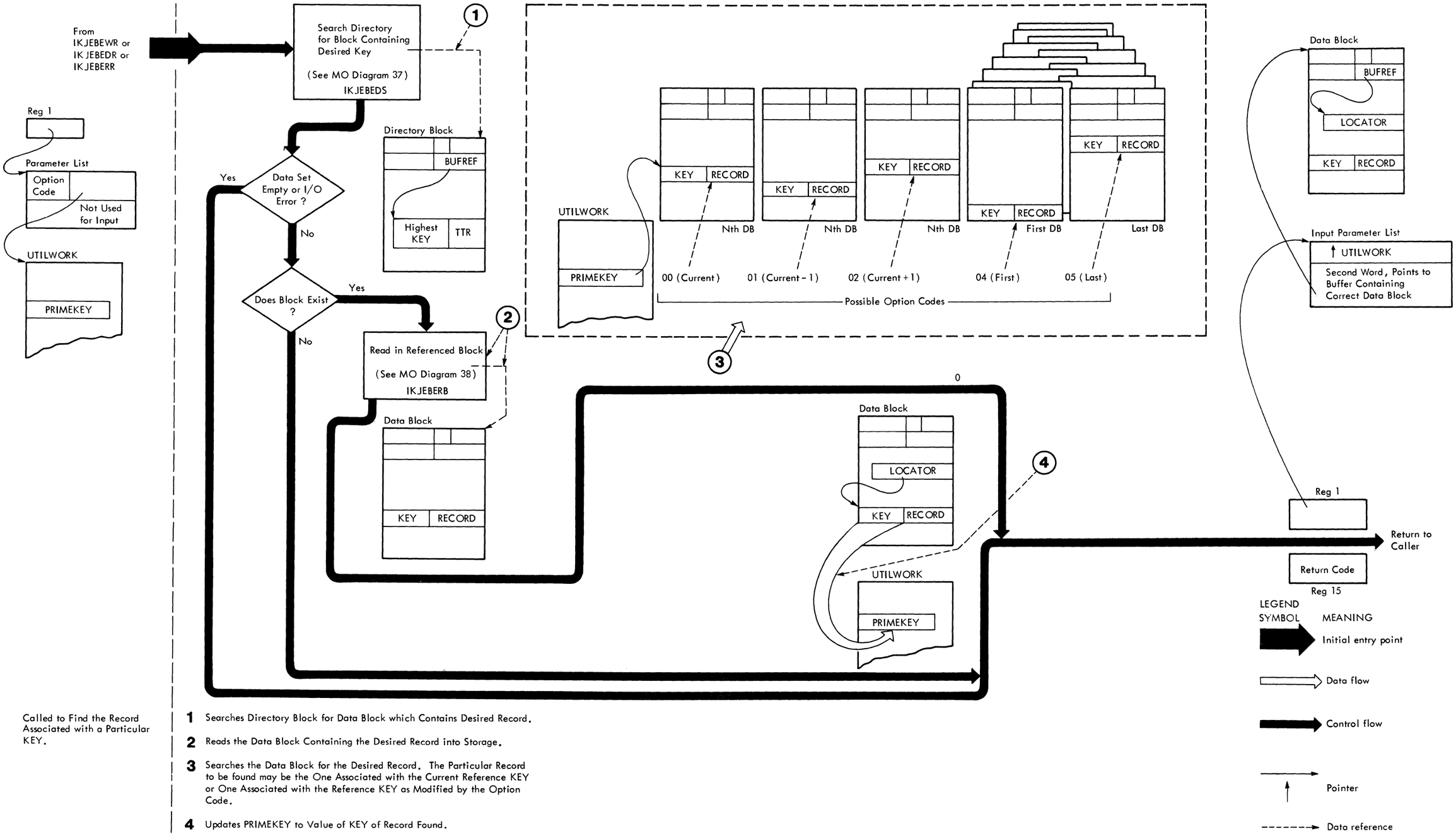
1 Saves the Current Reference Key.

2 Locates the Record Associated with the New KEY, if there is One, or with the Current Reference Key.

3 Deletes Record with Same KEY Value as New KEY or Current Reference KEY.

4 Restores PRIMEKEY to Value of Current Reference KEY if Record Associated with New KEY Cannot be Found.

Method of Operation Diagram 30. EDIT Access Method Delete Operation (IKJEBEDR)



Called to Find the Record Associated with a Particular KEY.

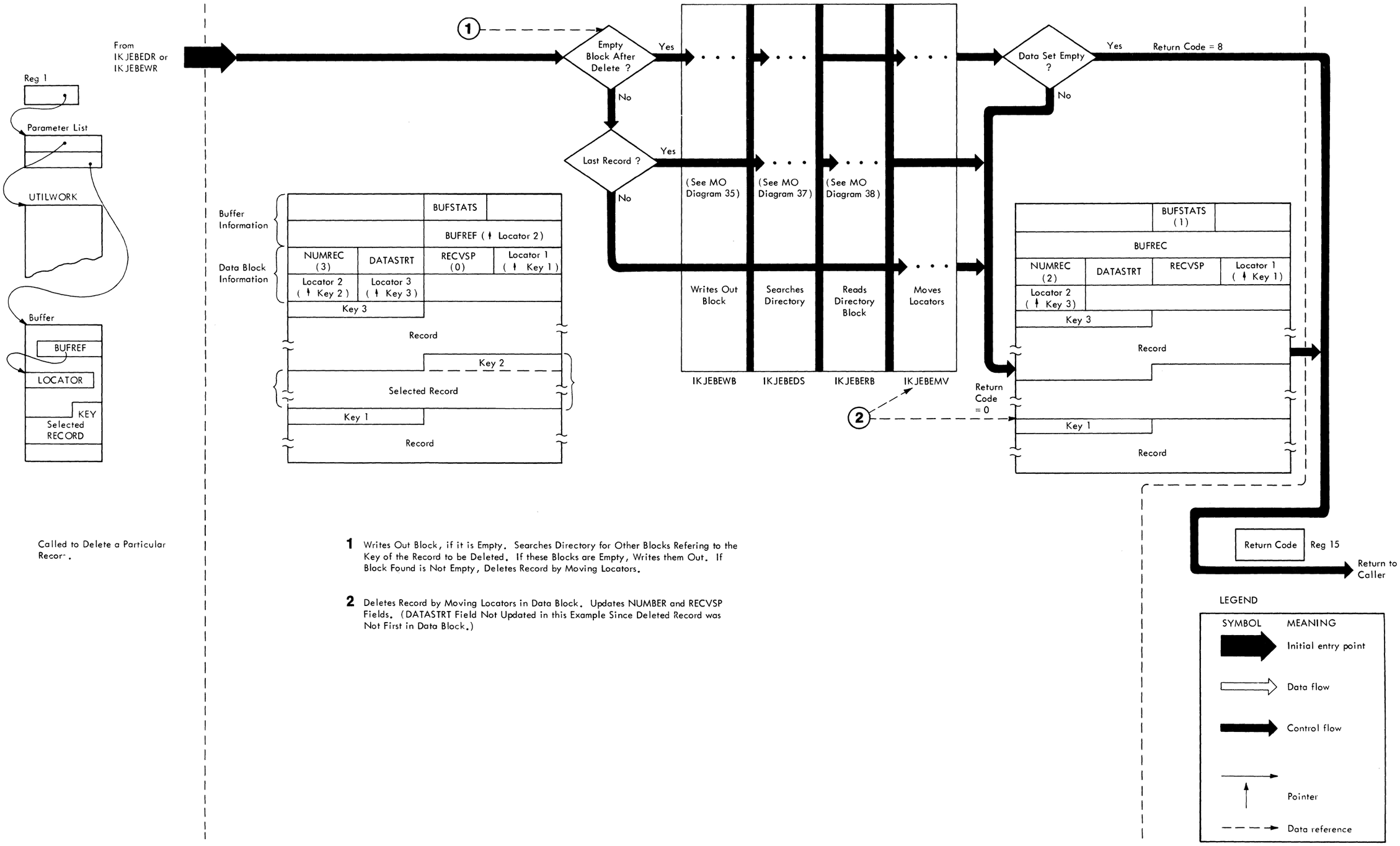
1 Searches Directory Block for Data Block which Contains Desired Record.

2 Reads the Data Block Containing the Desired Record into Storage.

3 Searches the Data Block for the Desired Record. The Particular Record to be found may be the One Associated with the Current Reference KEY or One Associated with the Reference KEY as Modified by the Option Code.

4 Updates PRIMEKEY to Value of KEY of Record Found.

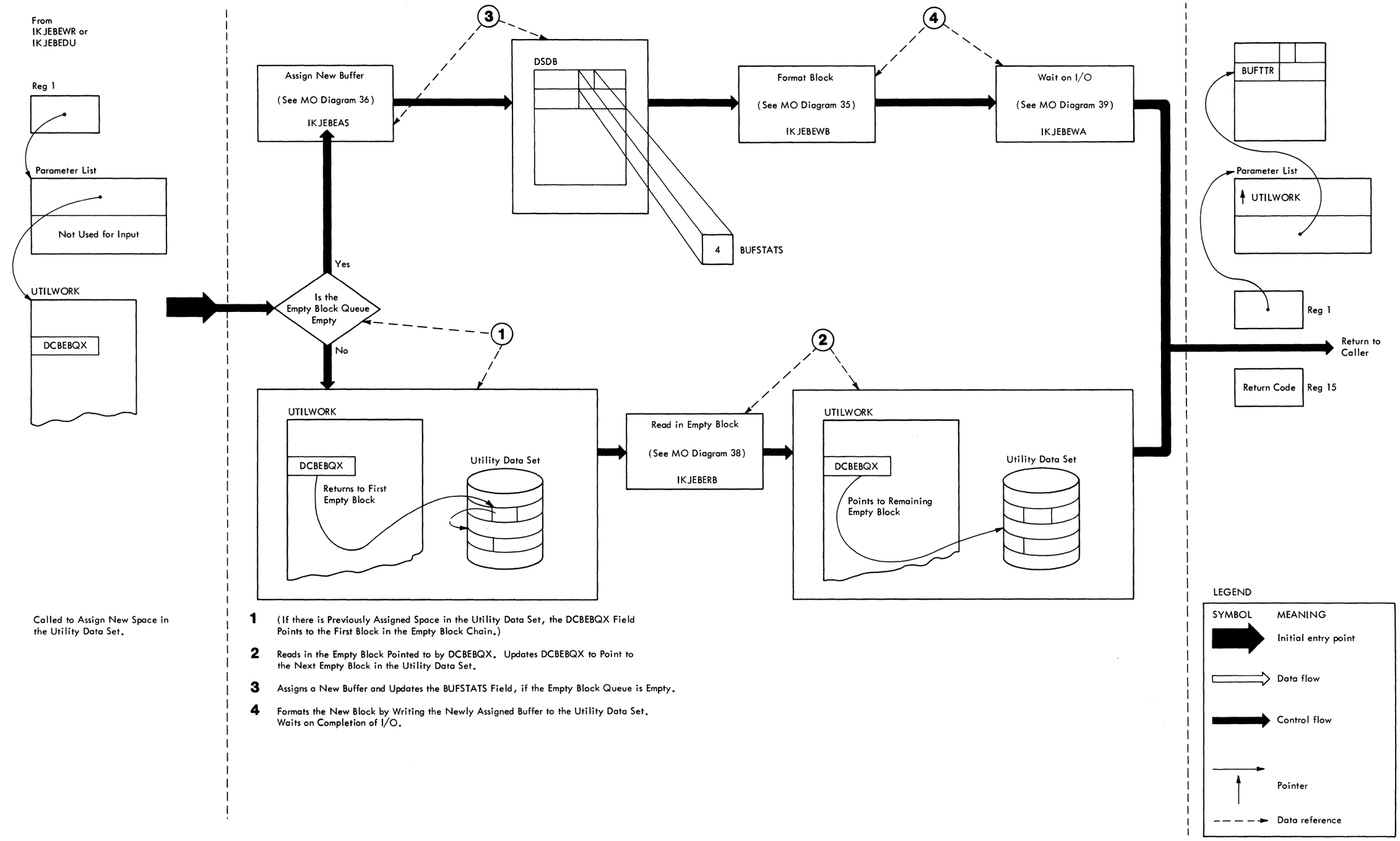
Method of Operation Diagram 31. EDIT Access Method Record Locate (IKJEBELO)



1 Writes Out Block, if it is Empty. Searches Directory for Other Blocks Refering to the Key of the Record to be Deleted. If these Blocks are Empty, Writes them Out. If Block Found is Not Empty, Deletes Record by Moving Locators.

2 Deletes Record by Moving Locators in Data Block. Updates NUMBER and RECVSP Fields. (DATASTRT Field Not Updated in this Example Since Deleted Record was Not First in Data Block.)

Method of Operation Diagram 32. EDIT Access Method Record Delete (IKJEBEDL)



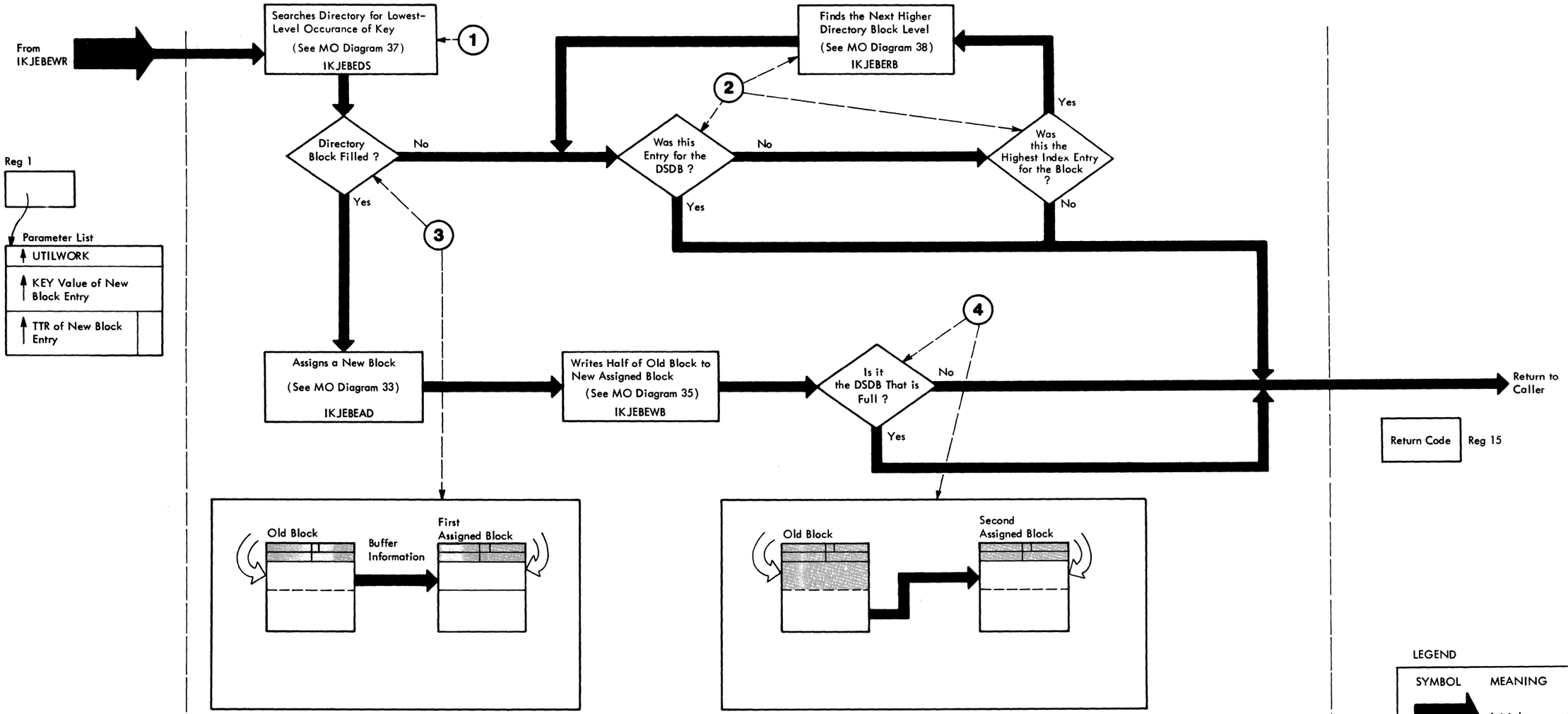
1 (If there is Previously Assigned Space in the Utility Data Set, the DCBEBQX Field Points to the First Block in the Empty Block Chain.)

2 Reads in the Empty Block Pointed to by DCBEBQX. Updates DCBEBQX to Point to the Next Empty Block in the Utility Data Set.

3 Assigns a New Buffer and Updates the BUFSTATS Field, if the Empty Block Queue is Empty.

4 Formats the New Block by Writing the Newly Assigned Buffer to the Utility Data Set. Waits on Completion of I/O.

Method of Operation Diagram 33. EDIT Access Method TTR Assignment (IKJEBEAD)



Called When New Block KEY/TTR Combination Must be Added to Directory Block or When an Existing KEY/TTR Must be Modified.

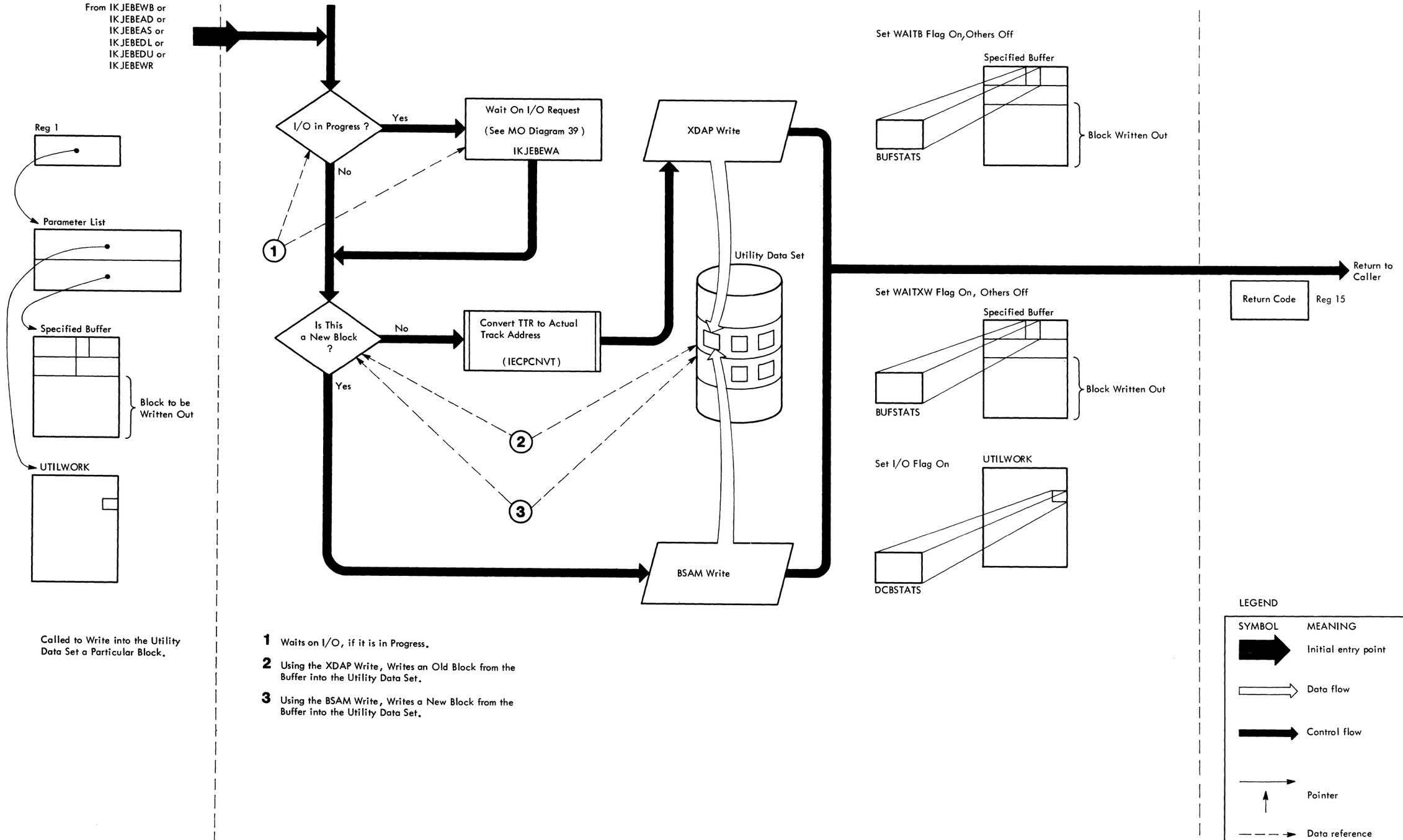
1 Searches Directory Blocks for Lowest Level Occurrance of KEY/TTR Combination. Performs Required Update.

2 If Directory Block is Not Full, Performs Update in Succeeding Higher-Level Directory Blocks, Including DSDB if Necessary.

3 If Directory Block is Full, Assigns a New Block and Writes Half of Old Block Contents into New Block. Inserts New Block KEY/TTR Combination into Proper Block. (See Figure 20 on Block Splitting.)

4 If Directory Block that is Full is the DSDB, Assigns a Second Block and Writes Remaining Half of Old Block into New Block. Inserts One Index Entry for Each New Block (LLDB) into the New DSDB. (See Figure 20 On Block Splitting.)

Method of Operation Diagram 34. EDIT Access Method Directory Update (IKJEBEDU)



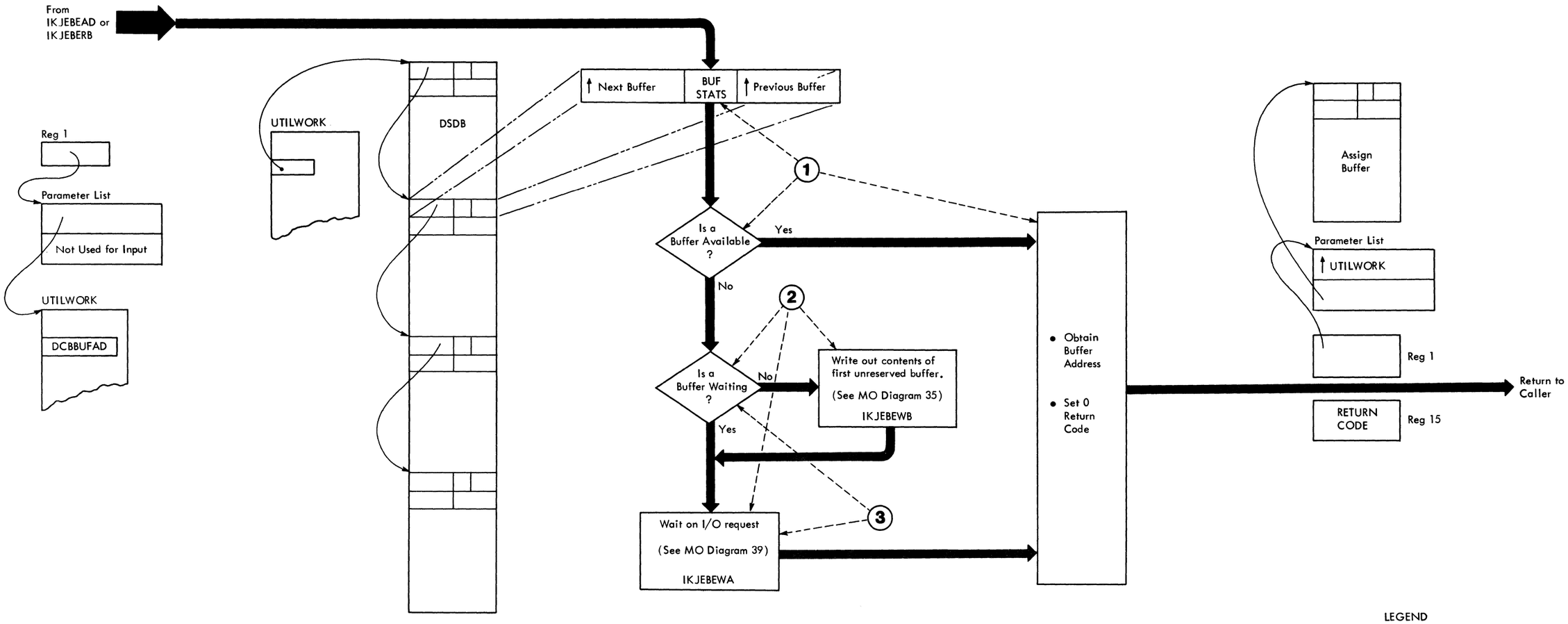
Called to Write into the Utility Data Set a Particular Block.

1 Waits on I/O, if it is in Progress.

2 Using the XDAP Write, Writes an Old Block from the Buffer into the Utility Data Set.

3 Using the BSAM Write, Writes a New Block from the Buffer into the Utility Data Set.

Method of Operation Diagram 35. EDIT Access Method Write Block (IKJEBEWB)



Called to Provide an I/O Buffer when a Buffer is Needed for Manipulation of the Utility Data Set.

1 Refers to the BUFSTATS Field to Find an Available Buffer.

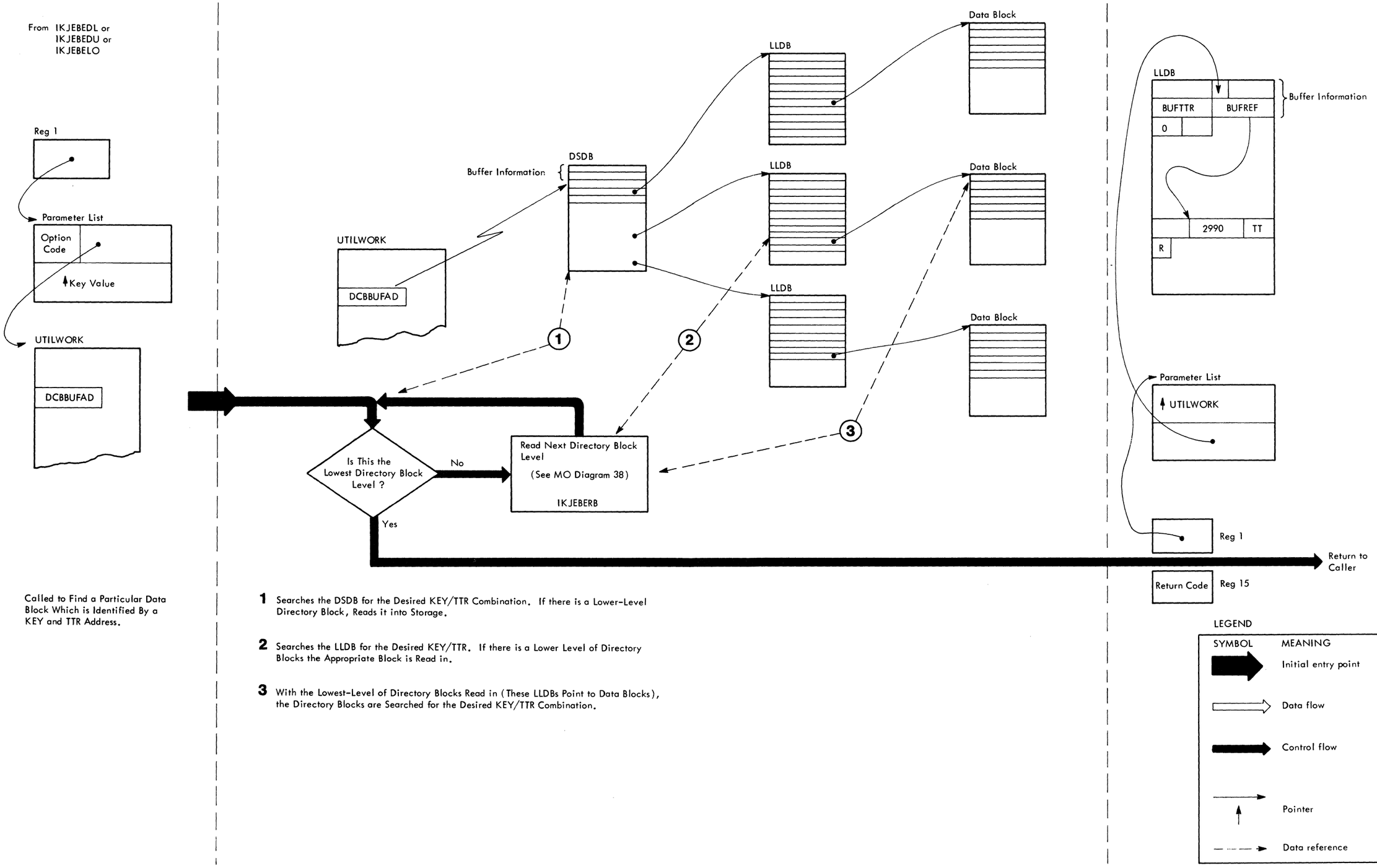
2 Writes Out the First Unreserved Buffer, if No Buffer is Free or Waiting on I/O. Waits for Completion of the Write Operation and Obtains the Buffer Address.

3 If a Buffer is Waiting on I/O, Waits on Completion of the I/O. Obtains the Buffer Address.

LEGEND

SYMBOL	MEANING
	Initial entry point
	Data flow
	Control flow
	Pointer
	Data reference

Method of Operation Diagram 36. EDIT Access Method Buffer Assignment (IKJEBEAS)



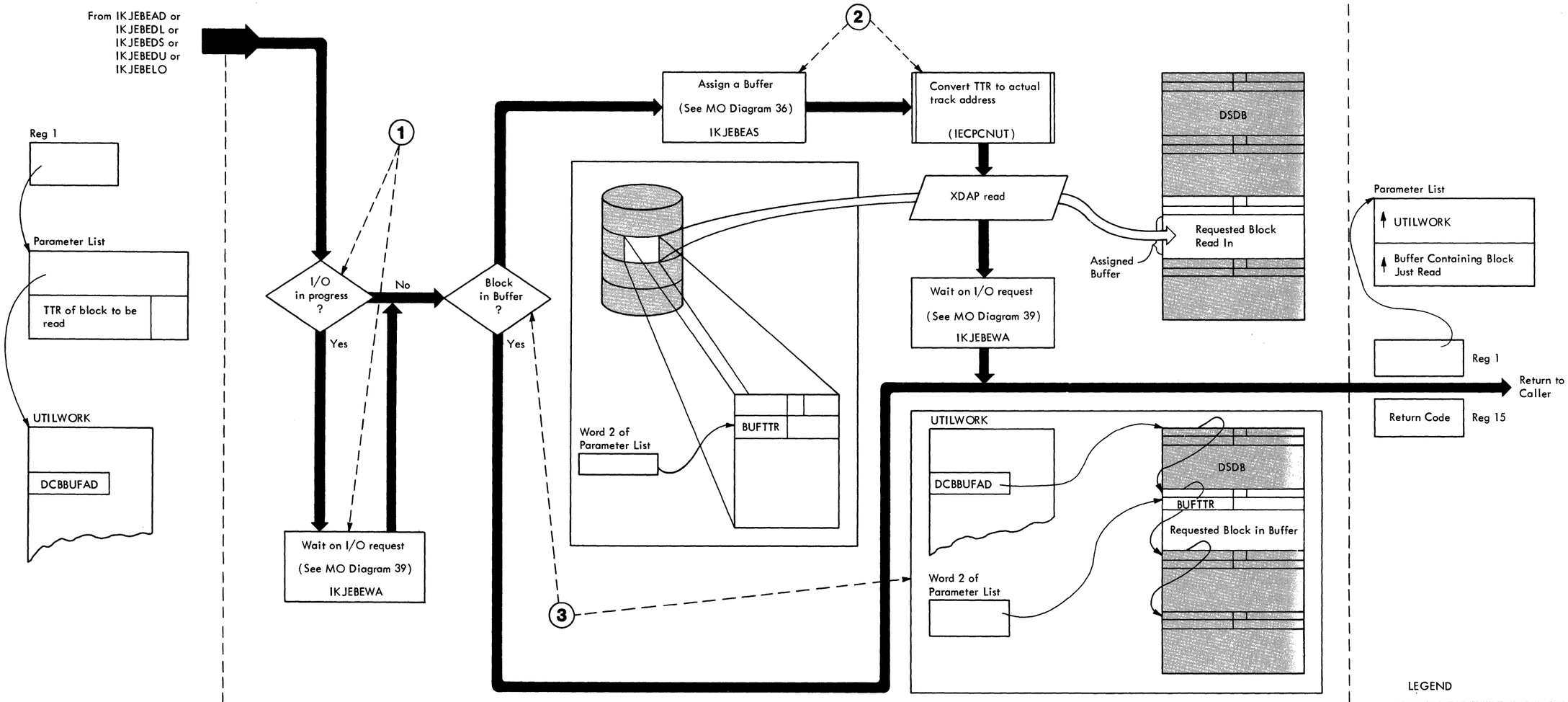
Called to Find a Particular Data Block Which is Identified By a KEY and TTR Address.

1 Searches the DSDB for the Desired KEY/TTR Combination. If there is a Lower-Level Directory Block, Reads it into Storage.

2 Searches the LLDB for the Desired KEY/TTR. If there is a Lower Level of Directory Blocks the Appropriate Block is Read in.

3 With the Lowest-Level of Directory Blocks Read in (These LLDBs Point to Data Blocks), the Directory Blocks are Searched for the Desired KEY/TTR Combination.

Method of Operation Diagram 37. EDIT Access Method Directory Search (IKJEBEDS)

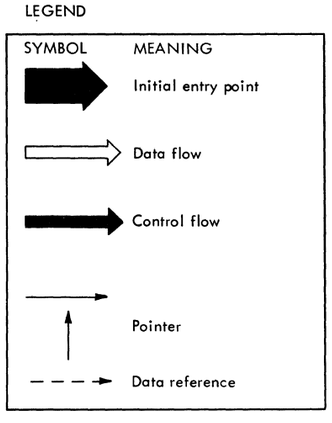


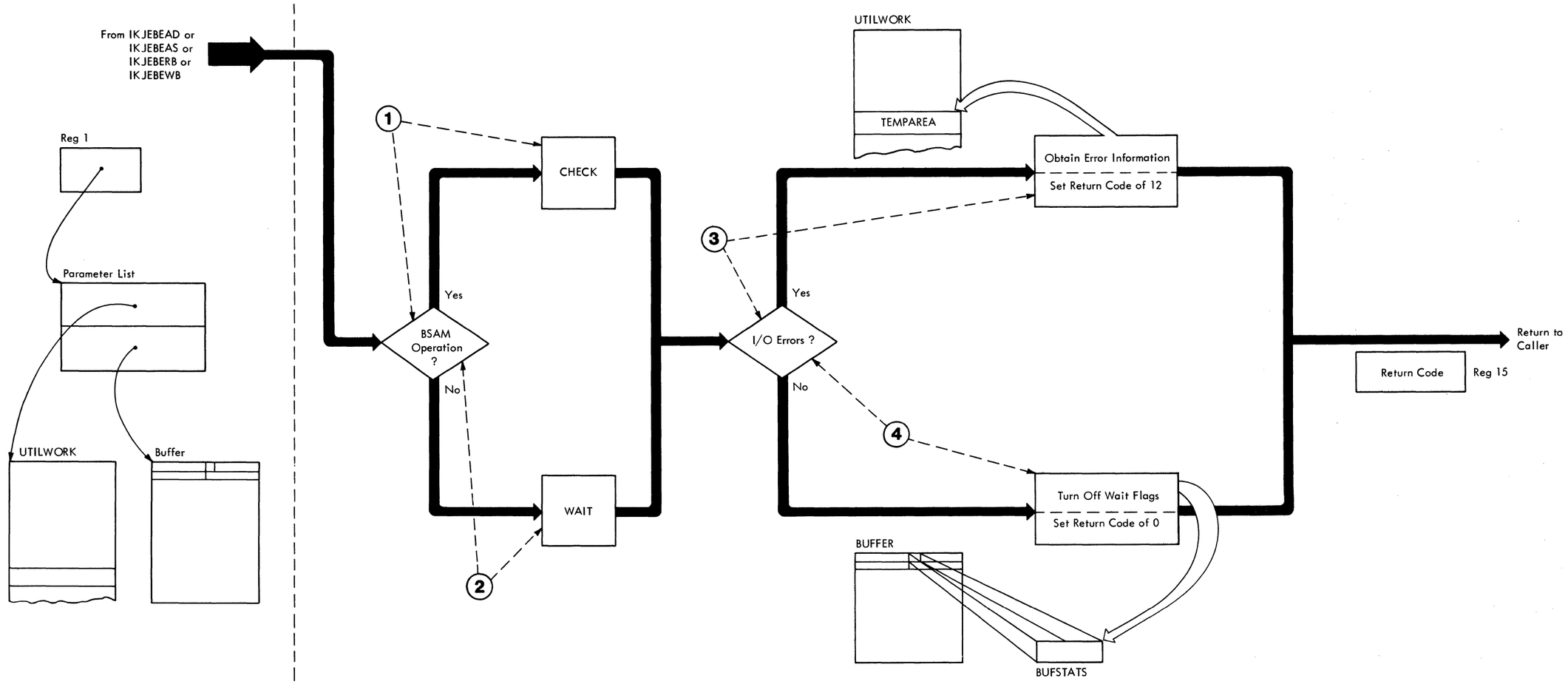
Called to Read into Storage a Particular Block.

1 Waits on I/O, if it is in Progress.

2 Reads the Desired Block into Storage, if the Block is in the Utility Data Set. Inserts a Pointer to the Block in the Second Word of the Output Parameter List.

3 If the Block is Already in a Buffer, a Pointer to the Buffer is Inserted in the Second Word of the Output Parameter List.





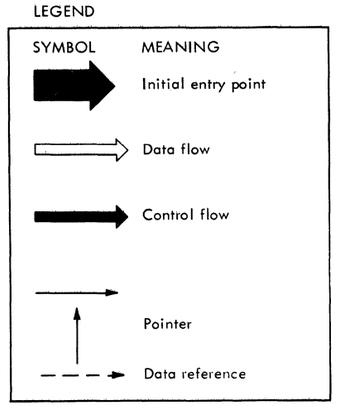
Called to Wait On Completion of an I/O Operation in a Particular Buffer.

1 Uses CHECK to Ensure Completion of a BSAM I/O Operation.

2 If Not a BSAM Operation, Uses WAIT to Ensure Completion of XDAP I/O Operation.

3 Inserts Error Information in TEMPAREA if an I/O Error has Occurred.

4 If No I/O Error has Occurred, Turns Off Wait Flags in BUFSTATS Field of Buffer.



Method of Operation Diagram 39. EDIT Access Method Wait (IKJEBEWA)

IBM System/360 Operating System:
Time Sharing Option
Command Processor Program Logic Manual Volume 3
EDIT

Order No. GY28-6773-1

READER'S COMMENT FORM

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Name
Address

What is your occupation?

How did you use this manual?
As a reference source
As a classroom text
As a self-study text

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N.Y. 10601

Attention: Department 813 U

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

CUT ALONG THIS LINE

System/360 OS TSO Command Processor PLM Vol. 3 (S360-36) Printed in U.S.A. GY28-6773-1

System/360 OS TSO Command Processor PLM Vol. 3 (S360-36) Printed in U.S.A. GY28-6773-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]