

DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

```
RRRRRRRR      EEEEEEEEE  AAAAAA      DDDDDDD      RRRRRRR      EEEEEEEEE  CCCCCC      CCCCCC
RRRRRRRR      EEEEEEEEE  AAAAAA      DDDDDDD      RRRRRRR      EEEEEEEEE  CCCCCC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RRRRRRRR      EEEEEEEEE  AA      AA      DD      DD      RRRRRRR      EEEEEEEEE  CC
RRRRRRRR      EEEEEEEEE  AA      AA      DD      DD      RRRRRRR      EEEEEEEEE  CC
RR      RR      EE          AAAAAAAAAA  DD      DD      RR      RR      EE          CC
RR      RR      EE          AAAAAAAAAA  DD      DD      RR      RR      EE          CC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RR      RR      EE          AA      AA      DD      DD      RR      RR      EE          CC
RR      RR      EEEEEEEEE  AA      AA      DDDDDDD      RR      RR      EEEEEEEEE  CCCCCC
RR      RR      EEEEEEEEE  AA      AA      DDCDDDD      RR      RR      EEEEEEEEE  CCCCCC
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      IIIII      SSSSSSS
LL      IIIII      SSSSSSS
LL      II        SS
LL      II        SS
LL      II        SS
LL      II        SS
LL      II        SSSSS
LL      II        SSSSS
LL      II        SS
LL      II        SS
LL      II        SS
LL      II        SS
LLLLLLLLLL  IIIII      SSSSSSS
LLLLLLLLLL  IIIII      SSSSSSS
```

(3)	160	READ NEXT INPUT RECORD
(8)	501	PROCESS RECALL COMMANDS
(11)	688	PROCESS ESCAPE SEQUENCES
(12)	887	RESTORE LOCKED KEYPAD STATE
(13)	919	EXPAND INPUT LINE
(14)	1072	SPECIAL TOKEN LEXICAL PROCESSING
(15)	1099	PROCESS &SYMBOL CONSTRUCT
(16)	1130	PROCESS @FILESPEC CONSTRUCT
(17)	1158	ERROR HANDLER IN CHARACTER INPUT ROUTINES
(18)	1180	RECALL COMMAND

```

0000 1 .TITLE READREC - READ AN INPUT RECORD
0000 2 .IDENT 'V04-000'
0000 3 .DEFAULT DISPLACEMENT,WORD
0000 4
0000 5
0000 6 :*****
0000 7 :*
0000 8 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 :* ALL RIGHTS RESERVED.
0000 11 :*
0000 12 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 :* TRANSFERRED.
0000 18 :*
0000 19 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 :* CORPORATION.
0000 22 :*
0000 23 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 :*
0000 26 :*
0000 27 :*****
0000 28
0000 29 : ABSTRACT:
0000 30
0000 31 : These routines are called by the lexical processing routines to
0000 32 : perform functions which are optional to the basic lexical routines,
0000 33 : but are required by CLI parsing.
0000 34
0000 35 : AUTHOR:
0000 36
0000 37 : Tim Halvorsen, Jan-1982
0000 38
0000 39 : MODIFIED BY:
0000 40
0000 41 : V03-021 HWS0092 Harold Schultz 22-Jul-1984
0000 42 : Add support for execute-only command procedures.
0000 43
0000 44 : V03-020 HWS0077 Harold Schultz 29-Jun-9184
0000 45 : If EOF encountered when reading a command procedure
0000 46 : while processing a line continuation, ignore termination
0000 47 : and return EOL.
0000 48
0000 49 : V03-019 HWS0072 Harold Schultz 08-Jun-1984
0000 50 : Partially undo HWS0006 (setting of the parse position).
0000 51 : Set the parse position prior to processing the @FILESPEC
0000 52 : construct, but save the old parse position set by the
0000 53 : higher level routines and restore it after the '@' processing
0000 54 : has successfully completed.
0000 55
0000 56 : V03-018 HWS0039 Harold Schultz 26-Mar-1984
0000 57 : When using arrow keys, don't duplicate command when changing

```

```
0000 58 : directions in recalling commands and are at the beginning of
0000 59 : the buffer.
0000 60 :
0000 61 : V03-017 HWS0034 Harold Schultz 16-Mar-1984
0000 62 : Always use terminal RAB when outputting escape sequences
0000 63 : and arrow keys.
0000 64 :
0000 65 : V03-016 HWS0006 Harold Schultz 13-Feb-1984
0000 66 : Remove parse location setting (DCL$MARK) when
0000 67 : processing an indirect command file. Let higher
0000 68 : level routines set parse location.
0000 69 : Use PRC_V_CARRCNTL to determine whether or not
0000 70 : a CR/LF is to be inserted before the prompt string.
0000 71 : Fix RECALL_NEXT so it will not skip first command
0000 72 : at bottom of command buffer when recall buffer
0000 73 : not full.
0000 74 : Save RMS STS and STV values in PRC data table
0000 75 :
0000 76 : V03-015 PCG0017 Peter George 03-Jan-1984
0000 77 : Move location of test for PRC_V_FLUSH.
0000 78 :
0000 79 : V03-014 PCG0017 Peter George 03-Jan-1984
0000 80 : Use PRC_V_FLUSH to determine how to handle EOF when
0000 81 : performing a flush.
0000 82 :
0000 83 : V03-013 PCG0016 Peter George 18-Nov-1983
0000 84 : Support up and down arrow recall.
0000 85 : Do not automatically close command procedures when
0000 86 : flushing a record. Add support for erase keypad attribute.
0000 87 :
0000 88 : V03-012 PCG0015 Peter George 27-Sep-1983
0000 89 : Ignore spurious CTRL/Y's.
0000 90 : Only check for CTRL/B's and ESC if input is from terminal.
0000 91 :
0000 92 : V03-011 PCG0014 Peter George 17-Jul-1983
0000 93 : Support 20 recalled commands.
0000 94 :
0000 95 : V03-010 PCG0013 Peter George 01-May-1983
0000 96 : Correctly signal keypad buffer overflows.
0000 97 : Fix CTRL/Y interrupting GOTO bug.
0000 98 :
0000 99 : V03-009 PCG0012 Peter George 20-Apr-1983
0000 100 : Check for CTRL/B with CMPB.
0000 101 :
0000 102 : V03-008 PCG0011 Peter George 06-Apr-1983
0000 103 : Remove GOTO code.
0000 104 : Reformat DCL$INPUT code.
0000 105 : Change test for escape sequence.
0000 106 : Allow the RECALL command to accept letters as arguments.
0000 107 : Update recall buffer when at ctrl/y level.
0000 108 :
0000 109 : V03-007 PCG0010 Peter George 01-Apr-1983
0000 110 : Change STV MOVZWL to MOVZBL.
0000 111 :
0000 112 : V03-006 PCG0009 Peter George 24-Feb-1983
0000 113 : Do not verify lines in EXE-only command procedures.
0000 114 : Lookup terminating escape sequences in the keypad
```

```
0000 115 : symbol table and act on them.
0000 116 : Use new XABTRM and init appropriate fields in its item list.
0000 117 : Add RECALL command and CTRL/B processing.
0000 118 :
0000 119 : V03-005 PCG0008 Peter George 10-Feb-1983
0000 120 : Close SYS$OUTPUT on silent logout.
0000 121 :
0000 122 : V03-004 PCG0007 Peter George 15-Jan-1983
0000 123 : Supply more rigorous test of whether commands
0000 124 : should be verified. Close PPF files before silent logout.
0000 125 :
0000 126 : V03-003 PCG0006 Peter George 28-Dec-1982
0000 127 : If $GET fails because of insufficient quota,
0000 128 : then log the process out.
0000 129 :
0000 130 : V03-002 PCG0005 Peter George 14-Nov-1982
0000 131 : Call DCL$UPCASE instead of DCL$REMBLANKS
0000 132 : Use prompt descriptor instead of WRK_L_PROMPT.
0000 133 :
0000 134 : V03-001 PCG0004 Peter George 28-Oct-1982
0000 135 : Get prompt string from PRC.
0000 136 : Process escape sequences.
0000 137 :---
```

```
0000 139 :  
0000 140 : MACRO LIBRARY CALLS  
0000 141 :  
0000 142 :  
0000 143 PRCDEF ;DEFINE PROCESS WORK AREA  
0000 144 WRKDEF ;DEFINE COMMAND WORK AREA  
0000 145 ITRMDEF ;XABTRM ITEM LIST DEFINITIONS  
0000 146 SYMDEF ;DEFINE SYMBOL TABLE ENTRY FORMAT  
0000 147 PTRDEF ;DEFINE TOKEN DESCRIPTORS  
0000 148 $CLMSGDEF ;DEFINE ERROR/STATUS VALUES  
0000 149 $FABDEF ;DEFINE FAB OFFSETS  
0000 150 $RABDEF ;DEFINE RAB OFFSETS  
0000 151 $XABTRMDEF ;DEFINE TERMINAL XAB  
0000 152 $DVIDEF ;DEFINE $GETDVI ITEM CODES  
0000 153 $TT2DEF ;DEFINE DEVDEPEND2 BITS  
0000 154  
00000000 155 .PSECT DCL$ZLUDE, BYTE, RD, NOWRT  
0000 156  
04 0000 157 ERASE: .BYTE 4  
4B5B1B0D 0001 158 .LONG ^X4B5B1B0D ;<CR>ESC[K
```

```

0005 160 .SBTTL READ NEXT INPUT RECORD
0005 161 :+
0005 162 : DCL$INPUT - READ NEXT INPUT RECORD
0005 163 :
0005 164 : READS THE NEXT INPUT RECORD AND PLACES IT INTO THE INPUT BUFFER.
0005 165 : THE CHARACTER POINTER IS RESET TO THE BEGINNING OF THE RECORD. A
0005 166 : SYMBOL SUBSTITUTION PASS IS PERFORMED IF A SINGLE OCCURRENCE OF A
0005 167 : SINGLE QUOTE IS DETECTED IN THE RECORD.
0005 168 :
0005 169 : INPUTS:
0005 170 :
0005 171 : R11 = ADDRESS OF PRC AREA
0005 172 : R10 = ADDRESS OF WRK AREA
0005 173 :
0005 174 : OUTPUTS:
0005 175 :
0005 176 : WRK_L_CHARPTR POINTS TO BEGINNING OF INPUT RECORD, WHICH
0005 177 : HAS BEEN TERMINATED BY A NULL BYTE.
0005 178 :
0005 179 : RO = FIRST CHARACTER IN INPUT BUFFER
0005 180 :
0005 181 DCL$INPUT:: :INPUT NEXT RECORD
1C BB 0005 182 PUSH  #^M<R2,R3,R4> :SAVE REGISTERS
0007 183 :
0007 184 :
0007 185 : IF AUTOLOGO FLAG SET AND WE ARE AT LEVEL 0 OR CTRL/Y LEVEL,
0007 186 : THEN DELETE THIS PROCESS TO PERFORM AN IMPLIED LOGOUT BUT WITHOUT
0007 187 : ANY LOGOUT MESSAGE. THIS IS FOR THE SPAWN COMMAND.
0007 188 :
12 68 AB 08 E1 0007 189 REINP: BBC #PRC_V_AUTOLOGO,PRC_W_FLAGS(R11),20$ :BRANCH IF FLAG NOT SET
OD FO AA 07 E0 000C 190 BBS #WRK_V_INQUIRE,WRK_W_FLAGS(R10),20$ :BRANCH IF INQUIRE
05 68 AB 08 E0 0011 191 BBS #PRC_V_YLEVEL,PRC_W_FLAGS(R11),10$ :IF SET, AT CONTROL Y/C LEVE
5C AB D5 0016 192 TSTL PRC[_INDEPTH(R11) :INDIRECT LEVEL ZERO?
03 12 0019 193 BNEQ 20$ :BRANCH IF NOT
027D 31 001B 194 10$: BRW SILENT_LOGOUT :PERFORM SILENT LOGOUT
001E 195 :
001E 196 :
001E 197 : GET ADDRESS OF THE RAB ASSOCIATED WITH THIS INDIRECT LEVEL
001E 198 :
OE FO AA 07 E0 001E 199 20$: BBS #WRK_V_INQUIRE,WRK_W_FLAGS(R10),30$ :IF SET, QUERY IN PROGRESS
54 14 AB DO 0023 200 MOVL PRC[_INDINPRAB(R11),R4 :GET ADDRESS OF LEVEL N RAB
09 68 AB 08 E1 0027 201 BBC #PRC_V_YLEVEL,PRC_W_FLAGS(R11),40$ :IF CLR, NOT AT CONTROL Y/C
04 68 AB 04 E0 002C 202 BBS #PRC_V_GOTO,PRC_W_FLAGS(R11),40$ :IF SET, AT GOTO LEVEL
54 08 AB DO 0031 203 30$: MOVL PRC[_INPRAB(R11),R4 :GET ADDRESS OF LEVEL 0 RAB
0035 204 :
0035 205 :
0035 206 : SETUP PROMPT STRING
0035 207 :
OC A0 50 10 AB DO 0035 208 40$: MOVL PRC_L_TRMLIST(R11),RO :GET ADDRESS OF XABTRM ITEM
10 A0 F99E CA BO 0039 209 MOVW WRK_W_PMPTLEN(R10),ITRM_W_PMPTLEN(RO) :SET LENGTH OF PROMPT STRING
10 A0 F9A2 CA DO 003F 210 MOVL WRK_L_PMPTADDR(R10),ITRM[_PMPTADDR(RO) :SET ADDRESS OF PROMPT STRIN
0045 211

```



```

0045 213 :
0045 214 : SETUP INPUT BUFFER AND POINTERS.
0045 215 :
0045 216 GET_INPUT:
20 A4 0100 8F B0 0045 217 MOVW #WRK_C_INPBUFSIZ,RAB$W_USZ(R4) ;SET SIZE OF INPUT BUFFER
52 F894 CA 9E 0048 218 MOVAB WRK_G_INPBUF-2(R10),R2 ;GET ADDRESS OF INPUT BUFFER
24 A4 52 D0 0050 219 MOVL R2,RAB$L_UBF(R4) ;SET ADDRESS OF INPUT BUFFER
F48E CA FF A2 9E 0054 220 MOVAB -1(R2),WRK_L_CHARPTR(R10) ;SET POINTER FOR GET CHARACT
005A 221
005A 222 :
005A 223 : READ THE NEXT INPUT RECORD AND CHECK FOR ERRORS.
005A 224 :
005A 225 :
05 F0 AA 07 E1 005E 226 DISABLE ;DISABLE CONTROL Y/C AST'S
BBC #WRK_V_INQUIPE,WRK_W_FLAGS(R10),10$ ;SKIP IF NOT INQUIRING
0063 227 SETBIT RAB$V_PPF_IND,RAB$Q_ISI(R4) ;SET INDIRECT PPF
0068 228 10$: $GET RAB=(R4) ;GET NEXT RECORD FROM INPUT
0071 229 CLRBIT RAB$V_PPF_IND,RAB$W_ISI(R4) ;CLEAR INDIRECT PPF
23 68 AB 01 E1 0076 230 BBC #PRC_V_CNTRLY,PRC_W_FLAGS(R11),20$ ;BRANCH IF NO CTRL/Y PENDING
1A 18 A4 00000000'8F E1 007B 231 BBC #DEV$V_TRM,RAB$L_CTR(R4),20$ ;SKIP IF NOT TERMINAL
00000000'8F 50 D1 0084 232 CMLP R0,#RMS$_CONTROL_Y ;DOUBLE CHECK FOR WINDOW
OD 12 008B 233 BNEQ 15$ ;SPURIOUS CTRL/Y
51 40 A4 D0 008D 234 MOVL RAB$L_XAB(R4),R1 ;GET ADDRESS OF XABTRM
18 DO 0091 235 MOVL #ITRM_K_MINLEN,- ;SET SHORT LENGTH OF ITEM LI
OC A1 0093 236 XAB$W_ITMLST_LEN(R1)
0519 30 0095 237 BSBW DCL$LOCKED_STATE ;RESTORE LOCKED STATE
04 11 0098 238 BRB 20$
009A 239 15$: CLRBIT PRC_V_CNTRLY,PRC_W_FLAGS(R11) ;IGNORE THIS CTRL/Y
009E 240 20$: ENABLE ;ENABLE CONTROL Y/C AST'S
OE 50 E8 00A0 241 BLBS R0,PROCESS_INPUT ;IF LBC I/O ERROR
0000'8F 50 B1 00A3 242 CMPW R0,#RMS$_SYS&^XFFFF ;ERROR IN QIO?
04 12 00A8 243 BNEQ ERROR1 ;NO, THEN SKIP
50 Or A4 D0 00AA 244 MOVL RAB$L_STV(R4),R0 ;GET PARTICULAR ERROR STATUS
00EA 31 00AE 245 ERROR1: BRW IO_ERROR ;ERROR
00B1 246

```

```

00B1 248 :
00B1 249 : READ COMPLETED SUCCESSFULLY. CLEAN UP AND THEN PROCESS THE INPUT RECORD.
00B1 250 :
00B1 251 PROCESS_INPUT:
53 22 A4 3C 00B1 252 MOVZWL RAB$W_RSZ(R4),R3 ;GET LENGTH OF INPUT RECORD
51 40 A4 D0 00B5 253 MOVL RAB$L_XAB(R4),R1 ;GET ADDRESS OF XABTRM
18 D0 00B9 254 MOVL #ITRM_K_MINLEN,- ;SET SHORT LENGTH OF ITEM LI
0C A1 00BB 255 XAB$W_ITMLST_LEN(R1) ;
50 10 AB D0 00BD 256 MOVL PRC_L_TRMLIST(R11),R0 ;GET ADDRESS OF XABTRM ITEM
06 68 AB 00 E1 00C1 257 BBC #PRC_V_CARRCNTL,PRC_W_FLAGS(R11),2$ ;SKIP IF NO CR/LF INDICATED
10 B0 0000'CF B0 00C6 258 MOVW DCL$CRF,@ITRM_L_PMPTADDR(R0) ;INSERT A CR/LF OTHERWISE
00CC 259 :
00CC 260 : IF IN THE MIDST OF A GOTO SCAN, THEN SKIP THE UNNECESSARY PROCESSING.
00CC 261 :
09 68 AB 04 E1 00CC 262 2$: BBC #PRC_V_GOTO,PRC_W_FLAGS(R11),10$ ;SKIP IF NOT GOTO READ
6243 94 00D1 263 CLRB (R2)[R3] ;SET EOL
00A8 31 00D4 264 BRW RETURN ;RETURN
FF68 31 00D7 265 5$: BRW GET_INPUT ;
00DA 266 :
00DA 267 :
00DA 268 : CHECK FOR CTRL/B AND DEFINED KEYS.
00DA 269 :
16 18 A4 00000000'8F E1 00DA 270 10$: BBC #DEV$V_TRM,RAB$L_CTX(R4),12$ ;SKIP IF NOT TERMINAL
01DF 30 00E3 271 BSBW PROCESS_RECALL ;CHECK FOR RECALL CHAR
03 50 D1 00E4 272 CML RO,#3 ;REPROMPT REQUESTED?
EC 13 00E9 273 BEQL 5$ ;YES, DO IT
0319 30 00EB 274 BSBW PROCESS_ESCAPE ;PROCESS ESCAPE SEQUENCES
03 50 D1 00EE 275 CML ^0,#3 ;REPROMPT REQUESTED?
E4 13 00F1 276 BEQL ;YES, DO IT
6243 94 00F3 277 CLRB (R2)[R3] ;SET EOL INDICATOR
B5 50 E9 00F6 278 BLBC R0,ERROR1 ;BRANCH IF FRROR
00F9 279 :
00F9 280 :
00F9 281 : IF INTERACTIVE AND IF THE RECORD IS NON-NULL, THEN COPY THE COMMAND LINE
00F9 282 : TO THE RECALL BUFFER.
00F9 283 :
6243 94 00F9 284 12$: CLRB (R2)[R3] ;SET EOL INDICATOR
53 D5 00FC 285 TSTL R3 ;NULL COMMAND?
31 13 00FE 286 BEQL 40$ ;YES, THEN SKIP
2C 68 AB 06 E0 0100 287 BBS #PRC_V_MODE,PRC_W_FLAGS(R11),40$ ;IF SET, BATCH JOB
0A 68 AB 0B E0 0105 288 BBS #PRC_V_YLEVEL,PRC_W_FLAGS(R11),15$ ;IF SET, CTRL/Y LEVEL
05 F0 AA 07 E0 010A 289 BBS #WRK_V_INQUIRE,WRK_W_FLAGS(R10),15$ ;IF SET, INQUIRING
5C AB D5 010F 290 TSTL PRC_[_INDEPTH(R11) ;INTERACTIVE LEVEL 0?
1D 12 0112 291 BNEQ 40$ ;IF NO, DON'T SAVE
52 DD 0114 292 15$: PUSHL R2 ;PUSH COMMAND ADDR
53 DD 0116 293 PUSHL R3 ;PUSH COMMAND LEN
5E DD 0118 294 PUSHL SP ;PUSH DESCR ADDR
07 F0 AA 03 E0 011A 295 BBS #WRK_V_CONTIN,WRK_W_FLAGS(R10),20$ ;IS IT CONTIN TYPE PROMPT?
0000'CF 01 FB 011F 296 CALLS #1,DCL$PUT_COMMAND ;SAVE THE COMMAND AWAY
05 11 0124 297 BRB 30$ ;BRANCH
0000'CF 01 FB 0126 298 20$: CALLS #1,DCL$PUT_SEGMENT ;SAVE THE SEGMENT AWAY
53 8ED0 012B 299 30$: POPL R3 ;RESTORE R3
52 8ED0 012E 300 POPL R2 ;RESTORE R2
0131 301 :
0131 302 :
0131 303 :
0131 304 :
SUBSTITUTE ANY SYMBOLS WHICH ARE DELIMITED BY SINGLE QUOTES

```

```

62 53 27 3A 0131 305 40$: LOCC #^A/ /,R3,(R2) ;LINE HAVE POSSIBLE STRING S
      03 13 0135 306 BEQL 50$ ;IF EQL NO
      0490 30 0137 307 BSBW EXPAND ;EXPAND LINE IF APPROPRIATE
      013A 308
      013A 309
      013A 310 : IF VERIFY MODE, WRITE A COPY OF THE COMMAND LINE TO THE LOG FILE
      013A 311
1B 68 AB 07 E1 013A 312 50$: BBC #PRC_V_VERIFY,PRC_W_FLAGS(R11),70$ ;IF CLR, NO LINE VERIFICATIO
16 68 AB 08 E0 013F 313 BBS #PRC_V_YLEVEL,PRC_W_FLAGS(R11),70$ ;IF SET, AT CONTROL Y/C LEVE
0B 68 AB 06 E0 0144 314 BBS #PRC_V_MODE,PRC_W_FLAGS(R11),60$ ;IF SET, BATCH JOB
      5C AB 05 D5 0149 315 TSTL PRC_C_INDEPTH(R11) ;INTERACTIVE LEVEL 0?
      0C 13 014C 316 BEQL 70$ ;IF YES, DON'T ECHO
      012D CB 95 014E 317 TSTB PRC_B_EXONLYL(R11) ;EXE-ONLY PROCEDURE?
      06 12 0152 318 BNEQ 70$ ;IF YES, DON'T ECHO
      51 53 D0 0154 319 60$: MOVL R3,R1 ;THE LENGTH OF THE LINE
      FEA6' 30 0157 320 BSBW DCL$MSGOUT ;OUTPUT INPUT LINE
      015A 321
      015A 322
      015A 323 : IF WE JUST READ A FULL-LINE COMMENT, RE-ISSUE READ NOW AS AN OPTIMIZATION.
      015A 324
20 68 AB 01 E0 015A 325 70$: BBS #PRC_V_CNTRLY,PRC_W_FLAGS(R11),RETURN ;MUST TAKE IMMEDIATE ACTION
      015F 326 ;IF CTRLY HIT - SKIP OPTIMIZ
      54 D4 015F 327 CLRL R4 ;SET NO DOLLAR SIGN SEEN F_LA
96'AF 05 82 3A 0161 328 80$: LOCC (R2)+,#5,B^SPECIAL ;CHECK FOR SPECIAL CHARS
      17 13 0166 329 BEQL RETURN ;IF EQL NO MATCH
      50 03 C2 0168 330 SUBL #3,R0 ;BLANK OR TAB?
      F4 19 016B 331 BLSS 80$ ;IF LSS YES
      10 13 016D 332 BEQL RETURN ;IF EQL END OF LINE
      50 D7 016F 333 DECL R0 ;DOLLAR SIGN OR COMMENT?
      08 1B 0171 334 BLEQU 90$ ;IF LEQU DOLLAR SIGN
07 68 AB 05 E0 0173 335 BBS #PRC_V_IND,PRC_W_FLAGS(R11),RETURN ;BRANCH IF FLUSHING RECORD
      FE8C 31 0178 336 BRW REINP ;GET NEXT RECORD
      E2 54 00 E3 017B 337 90$: BBS #0,R4,.80$ ;IF CLR, FIRST DOLLAR SIGN
      017F 338
      017F 339
      017F 340 : IF THE PREVIOUS RECORD ENDED WITH TRAILING SPACES OR TABS,
      017F 341 : INSERT A SPACE AT THE FRONT OF THE CURRENT INPUT RECORD SO
      017F 342 : THAT PPARAMETERS ARE DELIMITED PROPERLY.
      017F 343
06 FO AA 09 E5 017F 344 RETURN: BBCC #WRK_V_TRAILSPC,WRK_W_FLAGS(R10),10$ ;IF CLR, NO TRAILING SPACE S
      50 20 90 0184 345 MOVB #^A/ /,R0 ;SET SPACE CHARACTER
      FE76' 30 0187 346 BSBW DCL$BACKUPCHAR ;APPEND TO FRONT OF INPUT BU
      1C BA 018A 347 10$: POPR #^M<R2,R3,R4> ;RESTORE REGISTERS
00F3 CB 5F 8F 90 018C 348 MOVB #^A/ /,PRC_B_CONTINUE(R11) ;SET FOR CONTINUATION PROMPT
      FE6B' 30 0192 349 BSBW DCL$GETCHAR ;GET FIRST CHARACTER OF NEW
      05 0195 350 RSB
      0196 351
      0196 352 SPECIAL:
09 20 00 24 21 0196 353 .ASCII /!$/<0>/ / ;SPECIAL CHARACTERS

```

```

019B 355 :
019B 356 : AN INPUT I/O ERROR HAS OCCURRED. IF WE GOT 'RECORD STREAM ACTIVE',
019B 357 : TRY REPEATING THE READ 1000 TIMES. IF THAT FAILS, TRY CANCELING ALL
019B 358 : I/O ON THE INPUT CHANNEL AND THEN REISSUING THE READ.
019B 359 :
019B 360 I/O_ERROR:
0000'8F 50 B1 019B 361 CMPW RO,#RMS$_RSAB^XFFFF ;ERROR RECORD STREAM ACTIVE
48 12 01A0 362 BNEQ 30$ ;IF NO CHECK FOR END_OF_FILE
01A2 363
01A2 364
01A2 365 : WE DON'T WANT TO HAVE TO CANCEL AN RMS I/O UNLESS NECESSARY.
01A2 366 : RETRY UP SEVERAL TIMES TO ALLOW CURRENT WRITE I/O TO COMPLETE.
01A2 367 : THIS AVOIDS SPURIOUS WRITE ABORT MESSAGES FROM USER PROGRAMS,
01A2 368 : CAUSED BY CANCELING AN RMS I/O OPERATION.
01A2 369
52 03E8 8F 3C 01A2 370 MOVZWL #1000,R2 ;SETUP RETRY COUNT
05 FO AA 07 E1 01A7 371 BBC #WRK_V INQUIRE,WRK_W_FLAGS(R10),10$ ;SKIP IF NOT INQUIRING
01AC 372 SETBIT RAB$V_PPF_IND,RAB$W_ISI(R4) ;SET INDIRECT PPF
01B1 373 10$: $GET RAB=(R4) ;SEE IF THE WAITING GAVE RMS
01BA 374 CLRBIT RAB$V_PPF_IND,RAB$W_ISI(R4) ;CLEAR INDIRECT PPF
0000'8F 20 5J E8 01BF 375 BLBS RO,20$ ;THE WAIT DID THE TRICK, GO
50 50 B1 01C2 376 CMPW RO,#RMS$_RSAB^XFFFF ;RECORD STREAM STILL ACTIVE?
21 12 01C7 377 BNEQ 30$ ;BRANCH IF NOT
E5 52 F5 01C9 378 SOBGTR R2,10$ ;RETRY UNTIL STREAM BECOMES
01CC 379
01CC 380
01CC 381 : CONSTANT RETRYING DIDN'T HELP. CANCEL ANY I/O AND TRY AGAIN.
01CC 382
01CC 383 $CANCEL_S_PRC_W_INPCHAN(R11) ;IF NO LUCK, STOP THE I/O ON
01D7 384 $WAIT RAB=(R4) ;WAIT FOR I/O TO COMPLETE
37 11 01E0 385 BRB 40$ ;TRY TO READ AGAIN
01E2 386
01E2 387 :
01E2 388 : READ FINALLY SUCCEEDED. PROCESS THE INPUT RECORD.
01E2 389
52 F894 CA 9E 01E2 390 20$: MOVAB WRK_G_INPBUF-2(R10),R2 ;GET ADDRESS OF INPUT BUFFER
FEC7 31 01E7 391 BRW PROCESS_INPUT ;PROCESS THE INPUT LINE
01EA 392
01EA 393 :
01EA 394 : WE HAVE ENCOUNTERED AN ERROR OTHER THAN RECORD STREAM ACTIVE. RESTORE THE
01EA 395 : DEFAULT READ FORMAT OF NO INITIAL STRING, NO OFFSET AND RESTORE THE DEFAULT
01EA 396 : KEYPAD STATE.
01EA 397
51 40 A4 D0 01EA 398 30$: MOVL RAB$L_XAB(R4),R1 ;GET ADDRESS OF XABTRM
OC A1 18 D0 01EE 399 MOVL #ITRM_K_MINLEN,XAB$W_ITMLST_LEN(R1) ;SET SHORT LENGTH OF ITEM LI
03BC 30 01F2 400 BSBW DCL$LOCKED_STATE ;RESTORE LOCKED KEYPAD STATE
01F5 401
01F5 402 :
01F5 403 : IF WE ARE AT END OF FILE, TERMINATE THE CURRENT PROCEDURE LEVEL AND
01F5 404 : READ THE NEXT RECORD FROM THE PREVIOUS PROCEDURE LEVEL.
01F5 405 :
0000'8F 50 B1 01F5 406 CMPW RO,#RMS$_EOF&^XFFFF ;END OF FILE?
58 12 01FA 407 BNEQ 110$ ;IF NEQ NO
62 94 01FC 408 CLRB (R2) ;SET END OF LINE INDICATOR
28 68 AB 08 E0 01FE 409 BBS #PRC_V_YLEVEL,PRC_W_FLAGS(R11),50$ ;IF SET, AT CONTROL Y/C LEVE
23 FO AA 07 E0 0203 410 BBS #WRK_V_INQUIRE,WRK_W_FLAGS(R10),50$ ;BR IF DOING AN INQUIRE
5C AB D5 0208 411 TSTL PRC[_INDEPTH(R11)] ;INDIRECT LEVEL ZERO?

```

```

21 13 020B 412      BEQL 70$      ;IF EQL, YES
FDFO' 30 020D 413      BSBW DCL$UNSTACK ;UNSTACK INDIRECT FILE
      0210 414
      0210 415 :
      0210 416 : IF WE JUST RETURNED BACK TO LEVEL 0, FORCE THE PROMPT STRING BACK TO NORMAL
      0210 417 : BEFORE RE-ISSUING A READ FOR THE NEXT COMMAND.
      0210 418 :
      5C AB 05 0210 419      TSTL PRC_L_INDEPTH(R11) ;INDIRECT LEVEL ZERO?
      04 12 0213 420      BNEQ 40$ ;BRANCH IF NOT
OOF3 CB 94 0215 421      CLR B PRC_B_CONTINUE(R11) ;SET FOR NORMAL PROMPT
      0219 422
      0219 423 :
      0219 424 : IF CURRENTLY PROCESSING A LINE CONTINUATION, RETURN AN EOL CHARACTER.
      0219 425 :
      0219 426 : IF INDIRECT FILE RECOGNITION IS DISABLED, THIS IS A FLUSH OF A COMMAND WITH
      0219 427 : '-' AS THE LAST CHARACTER. RETURN AN EOL CHARACTER. OTHERWISE, PROCESS
      0219 428 : THE LINE JUST READ
      0219 429 :
OD 68 AB 06 E0 0219 430 40$: BBS #PRC_V_FLUSH,PRC_W_FLAGS(R11),50$ ;IF BIT IS SET,THEN FLUSH
O8 68 AB 05 E0 021E 431      BBS #PRC_V_IND,PRC_W_FLAGS(R11),50$ ;IF BIT IS SET,THEN FLUSH
O3 FO AA 03 E0 0223 432      BBS #WRK_V_CONTIN,WRK_W_FLAGS(R10),50$ ;IF BIT IS SET,THEN FLUSH
      FDDC 31 0228 433      BRW REINP ;READ THE NEXT LINE
      FF51 31 022B 434 50$: BRW RETURN ;FLUSH THE RECORD
      022E 435
      022E 436 :
      022E 437 : IF WE GOT AN END OF FILE WHILE READING THE LEVEL 0 PROCEDURE IN A
      022E 438 : NON-INTERACTIVE JOB, TERMINATE THE JOB STEP. IGNORE EOF'S (CTRL/Z)
      022E 439 : IN AN INTERACTIVE JOB.
      022E 440 :
14 68 AB 04 E5 022E 441 70$: BBCC #PRC_V_GOTO,PRC_W_FLAGS(R11),90$ ;BR IF NOT IN A GOTO
      FDCA' 30 0233 442      BSBW DCL$DEALGOTO ;DEALLOCATE GOTO SYMBOL
      0236 443      STATUS USGOTO ;SET FINAL STATUS OF UNSTATI
      023D 444      SETBIT WRK_V_COMMAND,WRK_W_FLAGS(R10) ;MARK COMMAND EXECUTION ERRO
      0241 445      ERRMSG ;PRINT THE ERROR
      FDB9' 30 0244 446 80$: BSBW DCL$SET STATUS ;GIVE ERROR HANDLER A CHANCE
      68 AB 0E E0 0247 447 90$: BBS #PRC_V_EOFLOGO,PRC_W_FLAGS(R11),- ;IF SILENT LOGOUT REQUESTED
      4F 024B 448      SILENT_LOGOUT ;
      C8 68 AB 06 E1 024C 449      BBC #PRC_V_MODE,PRC_W_FLAGS(R11),40$ ;IF NOT BATCH JOB, IGNORE EO
      FDAC' 31 0251 450      BRW DCL$ABORT ;LOG OUT BATCH JOB
      0254 451
      0254 452 :
      0254 453 : SOME OTHER TYPE OF I/O ERROR HAS OCCURRED. ISSUE AN ERROR MESSAGE,
      0254 454 : THEN TERMINATE THE CURRENT PROCEDURE LEVEL AND PARSE THE NEXT COMMAND.
      0254 455 : IF THE ERROR WAS INSUFFICIENT QUOTA, THEN CANCEL THE CTRL/Y AST, OUTPUT
      0254 456 : THE ERROR MESSAGE, AND THEN LOG THE PROCESS OUT.
      0254 457 :
00000000'8F 50 D1 0254 458 110$: SETBIT WRK_V_COMMAND,WRK_W_FLAGS(R10) ;MARK COMMAND EXECUTION ERRO
      2C 13 0258 459      CMLP RO,#SS$_EXQUOTA ;IS ERROR DUE TO EXCEEDED QU
      025F 460      BEQL ABORT ;YES, THEN BRANCH
      0261 461      ASSUME PRC_L_STV EQ PRC_L_STS+4
      0261 462      ASSUME RAB$L_STV EQ RAB$L_STS+4
      0084 CB 08 A4 7D 0261 463      MOVQ RAB$L_STS(R4),PRC_L_STS(R11) ;STORE STS AND STV VALUES
      0267 464 120$: ERRMSG ;OUTPUT ERROR MESSAGE
      18 68 AB 08 E0 026A 465      BBS #PRC_V_YLEVEL,PRC_W_FLAGS(R11),STATUS ;IF SET, CNTL Y/C LEVEL
      13 FO AA 07 E0 026F 466      BBS #WRK_V_INQUIRE,WRK_W_FLAGS(R10),STATUS ;SKIP IF IN INQUIRE
      5C AB 05 0274 467      TSTL PRC_L_INDEPTH(R11) ;INDIRECT LEVEL ZERO?
      07 12 0277 468      BNEQ 130$ ;IF NEQ NO

```

READREC
V04-000

- READ AN INPUT RECORD
READ NEXT INPUT RECORD

F 8

16-SEP-1984 00:11:48 VAX/VMS Macro V04-00
4-SEP-1984 23:42:34 [DCL.SRC]READREC.MAR;1

Page 11
(6)

C6 68 AB	06	E0	0279	469	BBS	#PRC V_MODE,PRC_W_FLAGS(R11),80\$:BR IF BATCH
	07	11	027E	470	BRB	STATUS-	:
	50	DD	0280	471	PUSHL	RO	:SAVE ERROR/STATUS VALUE
FD7B'	30	0282	472	130\$:	BSBW	DCL\$UNSTACK	:UNSTACK INDIRECT FILE
01	BA	0285	473		POPR	#*M<RO>	:RESTORE ERROR/STATUS VALUE
FD76'	30	0287	474	STATUS:	BSBW	DCL\$SET STATUS	:SET COMPLETION STATUS
FD73'	31	028A	475		BRW	DCL\$RESTART	:
		028D	476				:

```

028D 478 :
028D 479 : OUTPUT AN ERROR MESSAGE AND LOG THE PROCESS OUT
028D 480 :
50 DD 028D 481 ABORT:  PUSHL  RO           :SAVE THE STATUS
FD6E' 30 028F 482      BSBW   DCL$DSBCONTRLY :CANCEL THE CTRL/Y AST
50 8ED0 0292 483      POPL   RO           :RESTART THE STATUS
FD65' 31 0295 484      ERRMSG :OUTPUT THE ERROR MESSAGE
0298 485      BRW    DCL$ABORT :LOG THE PROCESS OUT
0298 486 :
0298 487 :
0298 488 : PERFORM A SILENT LOGOUT BY CANCELING THE SUPERVISOR MODE EXIT HANDLERS
0298 489 : (SO THAT THE PROCESS IS DELETED), AND INVOKING $EXIT WITH THE LATEST
0298 490 : STATUS FOR THIS PROCESS.
0298 491 :
0298 492 $ILENT_LOGOUT::
50 FD62' 30 0298 493      BSBW   DCL$CLOSE PPFS      :CLOSE ALL PPF FILES STILL O
1C AB  D0 029E 494      MOVL   PRC_L_INDFAB(R11),RO :GET ADDR OF INDIRECT FAB
0114 CB  B0 02A2 495      MOVW   PRC_W_OUTIFI(R11),- :GET INTERNAL FILE INDEX OF
02 A0    02A6 496      FABSW_IFI(RO) :
02A8 497      $CLOSE FAB=(RO) :CLOSE INDIRECT OUTPUT FILE
02B1 498      $CANEXH_S :CANCEL SUPERVISOR MODE EXIT
02BA 499      $EXIT_S_PRC_L_LSTSTATUS(R11) :EXIT PROCESS WITH FINAL STA

```

```

02C5 501 .SBTTL PROCESS RECALL COMMANDS
02C5 502 :---
02C5 503 : PROCESS_RECALL - PROCESS RECALL COMMANDS
02C5 504 :
02C5 505 : SUBROUTINE TO CHECK FOR AND PROCESS CTRL/B AND ARROW RECALL COMMANDS.
02C5 506 :
02C5 507 : INPUTS:
02C5 508 :
02C5 509 :     R2 = ADDRESS OF NEW INPUT RECORD
02C5 510 :     R3 = LENGTH OF NEW INPUT RECORD, EXCLUDING NULL AT END OF LINE
02C5 511 :     R4 = ADDRESS OF RAB
02C5 512 :
02C5 513 : OUTPUTS:
02C5 514 :
02C5 515 :     INPUT BUFFER IS INITIALIZED
02C5 516 :     RO IS SET
02C5 517 :---
02C5 518 PROCESS_RECALL:                                ;PROCESS RECALL COMMAND
02C5 519 :
02C5 520 :
02C5 521 : CHECK FOR CTRL/B.
02C5 522 :
02C5 523 :     CMPB    #^X02,RAB$W_STV0(R4)                ;TERMINATOR CTRL/B?
02C5 524 :     BEQL    RECALL_PREV                          ;YES, THEN PROCESS IT
02C5 525 :     MOVL    #1,RO                                ;RETURN
02C5 526 :     RSB
02C5 527 :
02C5 528 :
02C5 529 : REPROMPT WITH THE PREVIOUS COMMAND.
02C5 530 :
02C5 531 RECALL_PREV:
02C5 532 :     BSBW    DCL$LOCKED STATE                    ;RESTORE LOCKED STATE
02C5 533 :     TSTB    WRK_B_RECALLCNT(R10)                 ;FIRST TIME?
02C5 534 :     BEQL    10$                                  ;YES, THEN ALWAYS GO AHEAD
02C5 535 :     TSTB    @WRK_L_RECALLPTR(R10)               ;POINTER ADJUST. BY RECALL_NEXT?
02C5 536 :     BEQL    5$                                    ;NO, OK AS IS
02C5 537 :     DECL    WRK_L_RECALLPTR(R10)                 ;YES, SET IT BACK TO NORMAL
02C5 538 :
02C5 539 :     5$:    CMPL    PRC_L_RECALLPTR(R11),-        ;RETURNED TO THE ORIGIN?
02C5 540 :           WRK_L_RECALLPTR(R10)
02C5 541 :           BEQL    END_OF_LIST
02C5 542 :           CMPB    WRK_B_RECALLCNT(R10),-        ;MAX # OF COMMANDS DISPLAYED?
02C5 543 :           #WRK_C_RECALLMAX+1
02C5 544 :           BEQL    END_OF_LIST
02C5 545 :           10$:  CLRQ    -(SP)
02C5 546 :           PUSHL   SP
02C5 547 :           CALLS   #1,DCL$GET_PREV_COMMAND
02C5 548 :           BLBC    RO,20$
02C5 549 :           INCB    WRK_B_RECALLCNT(R10)
02C5 550 :           MOVAB   PRC_G_COMMANDS(R11),RO
02C5 551 :           CMPL    RO,WRK_L_RECALLPTR(R10)
02C5 552 :           BNEQ    15$
02C5 553 :           INCL    WRK_L_RECALLPTR(R10)
02C5 554 :           15$:  CMPB    WRK_B_RECALLCNT(R10),-        ;YES, FIX POINTER TO INDICATE
02C5 555 :           #WRK_C_RECALLMAX+1                    ;THIS TO RECALL NEXT.
02C5 556 :           RECALL_CURR                             ;MAX # OF COMMANDS DISPLAYED?
02C5 557 :           BNEQ

```



```

SE 08 C0 0310 558 20$: ADDL #8,SP ;RESTORE THE STACK
OD 11 0313 559 BRB END_OF_LIST ;OUTPUT A BLANK LINE
    0315 560
    0315 561
    0315 562 : REPROMPT WITH THE COMMAND THAT IS ON THE STACK.
    0315 563
    0315 564 RECALL_CURR:
51 0067 30 0315 565 BSBW ERASE_LINE ;ERASE THE PREVIOUS LINE
    8E 7D 0318 566 MOVQ (SP)+,R1 ;GET COMMAND DESCRIPTOR
50 00D1 30 0318 567 BSBW INSERT_COMMAND ;MODIFY THE XABTRM
    03  D0 031E 568 MOVL #3,R0 ;SET REPROMPT RETURN STATUS
    05  0321 569 RSB ;RETURN
    0322 570
    0322 571
    0322 572 : REPROMPT WITH A BLANK LINE.
    0322 573
    0322 574 END_OF_LIST:
005A 30 0322 575 BSBW ERASE_LINE ;ERASE THE PREVIOUS LINE
    51 7C 0325 576 CLRQ R1 ;INIT COMMAND DESCRIPTOR
00C5 30 0327 577 BSBW INSERT_COMMAND ;REPROMPT WITH THIS COMMAND
50 03  D0 032A 578 MOVL #3,R0 ;SET REPROMPT RETURN STATUS
    05  032D 579 RSB ;RETURN
    032E 580
    032E 581 : REPROMPT WITH THE NEXT COMMAND.
    032E 582
    032E 583
    032E 584 : NOTE: WHEN AT BEGINNING OF BUFFER (NULL PRECEDING CURRENT RECORD),
    032E 585 : SPECIAL PROCESSING IS DOWN. THE DCL$GET_CURR_COMMAND IS USED
    032E 586 : INSTEAD OF DCL$GET_NEXT_COMMAND. THE WRK_RECALL_POINTER IS USED
    032E 587 : AS A FLAG TO DETERMINE WHETHER OR NOT THE FIRST COMMAND IN THE
    032E 588 : BUFFER HAS BEEN ALREADY PROCESSED. NORMALLY, THE POINTER IS POINTING
    032E 589 : TO THE BEGINNING OF A COMMAND (FIRST BYTE ALWAYS NULL). IF THE
    032E 590 : POINTER IS POINTING TO A NON-NULL BYTE, IT MEANS THAT THE FIRST
    032E 591 : COMMAND IN THE BUFFER HAS ALREADY BEEN PROCESSED (THE POINTER WILL
    032E 592 : BE POINTING TO THE LENGTH BYTE FOLLOWING THE NULL ).
    032E 593 : IN THIS CASE, THE POINTER IS CORRECTED AND THE COMMAND IS
    032E 594 : RETRIEVED WITH THE DCL$GET_NEXT_COMMAND INSTEAD OF DCL$GET_CURR_COMMAND.
    032E 595
    032E 596 RECALL_NEXT:
0280 30 032E 597 BSBW DCL$LOCKED STATE ;RESTORE LOCKED STATE
C5 AA 95 0331 598 TSTB WRK_B_RECALLCNT(R10) ;ANY NEXT COMMANDS TO RECALL?
    EC 13 0334 599 BEQL END_OF_LIST ;NO, THEN REPEAT CURRENT COMMAND
    0336 600 :
EA BA 95 0336 601 TSTB @WRK_L_RECALLPTR(R10) ;POINTING TO BEGINNING OF COMMAND?
    05 13 0339 602 BEQL 25$ ;YES, HANDLE NORMALLY
    033B 603 :
EA AA D7 033B 604 DECL WRK_L_RECALLPTR(R10) ;NO, ADJUST POINTER
    2C 11 033E 605 BRB 30$ ;AND GET NEXT COMMAND.
    0340 606 :
50 EA AA 01 C3 0340 607 25$: SUBL3 #1,WRK_L_RECALLPTR(R10),R0 ;POINT R0 TO CURR. COMMAND-1
    51 0133 CB 9E 0345 608 MOVAB PRC_G_COMMANDS(R11),R1 ;R1 = ADDR. OF BEGINNING OF BUF.
    51 50 D1 034A 609 CMPL R0,R1 ;BUFFER UNDERFLOW?
    05 1E 034D 610 BGEQU 28$ ;NO, R0 OK AS IS
50 0401 C0 9E 034F 611 MOVAB PRC_C_CMDRUF SIZ(R0),R0 ;YES, POINT R0 TO TOP OF BUFFER
    60 95 0354 612 28$: TSTB (R0) ;PREVIOUS COMMAND EXIST?
    14 12 0356 613 BNEQ 30$ ;YES, PROCESS NORMALLY
    0358 614 ;

```

```

0358 615 : SPECIAL CASE: THERE ARE EXACTLY 21 COMMANDS IN THE COMMAND BUFFER
0358 616 : AND RECALL_PREV HAS RETRIEVED THE LAST VALID COMMAND (IT LEAVES
0358 617 : POINTER TO FIRST COMMAND.) IN THIS CASE, DO A GET NEXT INSTEAD OF
0358 618 : GET CURRENT.
0358 619 :
CS AA 91 0358 620 CMPB WRK_B_RECALLCNT(R10),- ;HAVE MAX. COMMANDS BEEN DISPL?
15 035B 621 #WRK_C_RECALLMAX+1
OE 13 035C 622 BEQL 30$ ;YES, GET NEXT COMMAND.
035E 623 ;
7E 7C 035E 624 CLRQ -(SP) ;NO, ALLOCATE A DESCRIPTOR
5E DD 0360 625 PUSHL SP ;PUSH DESCRIPTOR ADDR.
0000'CF 01 FB 0362 626 CALLS #1,DCL$GET_CURR_COMMAND ;GET CURRENT COMMAND
EA AA D6 0367 627 INCL WRK_L_RECALLPTR(R10) ;INDICATE 1ST COMMAND PROCESSED.
A9 11 036A 628 BRB RECALL_CURR ;OUTPUT CURRENT COMMAND
036C 629 ;
7E 7C 036C 630 30$: CLRQ -(SP) ;ALLOCATE A DESCRIPTOR
5E DD 036E 631 PUSHL SP ;PUSH DESCR ADDRESS
0000'CF 01 FB 0370 632 CALLS #1,DCL$GET_NEXT_COMMAND ;RECALL THE SPECIFIED COMMAND
CS AA 97 0375 633 DECB WRK_B_RECALLCNT(R10) ;DECREMENT THE RECALL COUNT
9B 12 0378 634 BNEQ RECALL_CURR ;BRANCH IF NOT NOW ZERO
5E 08 C0 037A 635 ADDL #8,SP ;RESTORE THE STACK
A3 11 037D 636 BRB END_OF_LIST ;OUTPUT A BLANK LINE

```

```

037F 638 :
037F 639 : ERASE THE CURRENT LINE IF THE TERMINAL IS ANSI CRT.
037F 640 :
037F 641 : RO,R5 ARE DESTROYED
037F 642 :
037F 643 ERASE_LINE:
      7E 7C 037F 644 CLRQ -(SP) ;CREATE ITEM LIST
      F8 AE 9F 0381 645 PUSHAB -8(SP) ;SET BUFFER ADDRESS
001C0004 8F DD 0384 646 PUSHL #DVI$_DEVDEPEND2@16+4 ;SET ITEM CODE
      7E 7C 038A 647 CLRQ -(SP) ;ALLOCATE AN IOSB
      50 5E DO 038C 648 MOVL SP,R0 ;GET ADDRESS OF IOSB
      038F 649 $GETDVIW S,EFN=#EXESC,SYSEFN,- ;GET DEVDEPEND2
      038F 650 CHAN=PRC W INPCHAN(R11),-
      038F 651 ITMLST=8(R0),-
      038F 652 IOSB=(R0)
      3D 50 E9 03AB 653 BLBC R0,90$ ;BRANCH IF ERROR
      50 6E 3C 03AE 654 MOVZWL (SP),R0 ;GET IOSB STATUS
      37 50 E9 03B1 655 BLBC R0,90$ ;BRANCH IF ERROR
      03B4 656
      32 08 AE 18 F1 03B4 657 BBC #TT2$_ANSICRT,8(SP),90$ ;BRANCH IF ANSI CRT BIT CLEAR
      55 0C AB DO 03B9 658 MOVL PRC L OUTRAB(R11),R5 ;SET ADDRESS OF OUTPUT FILE RAB
28 A5 FC40 CF 9E 03BD 659 10$: MOVAB ERASE+1,RAB$_RBF(R5) ;SET ADDRESS OF OUTPUT RECORD
22 A5 FC39 CF 9B 03C3 660 MOVZBW ERASE,RAB$_RSZ(R5) ;SET SIZE OF OUTPUT RECORD
      03C9 661 DISABLE ;DISABLE CONTROL Y/C AST'S
      06 00 F0 03CD 662 INSV #0,#RAB$_PPF_RAT,- ;DISABLE CR FORMAT WRITES
      02 A5 08 03D0 663 #RAB$_PPF_RAT,RAB$_ISI(R5)
      03D3 664 $PUT RAB=(R5) ;OUTPUT RECORD
      06 02 F0 03DC 665 INSV #FAB$_CR,#RAB$_PPF_RAT,- ;ENABLE CR FORMAT WRITES
      02 A5 08 03DF 666 #RAB$_PPF_RAT,RAB$_ISI(R5)
      03E2 667 ENABLE ;ENABLE CONTROL Y/C AST'S
      50 10 AB DO 03E4 668
      10 B0 B4 03E8 669 MOVL PRC L TRMLIST(R11),R0 ;GET ADDRESS OF XABTRM ITEM LIST
      03EB 670 CLRW @ITRM_L_PMPTADDR(R0) ;REMOVE CR/LF FROM PROMPT STRING
      5E 18 C0 03EB 671
      05 03EE 672 90$: ADDL #6*4,SP ;RESTORE STACK
      03EE 673 RSB

```

```
03EF 675 :  
03EF 676 : REPROMPT WITH RECALLED COMMAND.  
03EF 677 :  
03EF 678 INSERT_COMMAND:  
50 10 AB D0 03EF 679 MOVL PRC_L TRMLIST(R11),R0 ;GET ADDRESS OF XABTRM ITEM LIST  
28 A0 D4 03F3 680 CLRL ITRM_C OFFSET(R0) ;REQUEST A FRESH READ  
18 A0 51 B0 03F6 681 MOVW R1,ITRM_W_INILEN(R0) ;SET LENGTH OF INITIAL STRING  
1C A0 52 D0 03FA 682 MOVL R2,ITRM_L_INIADDR(R0) ;SET ADDRESS OF INITIAL STRING  
50 40 A4 D0 03FE 683 MOVL RAB$L_XABTR4),R0 ;GET ADDRESS OF XABTRM  
30 D0 0402 684 MOVL #ITRM_K_LENGTH,- ;SET EXPANDED LENGTH OF ITEM LIST  
0C A0 0404 685 MOVL XAB$W_ITMLST_LEN(R0) ;  
05 0406 686 RSB ;RETURN
```

```

0407 688 .SBTTL PROCESS ESCAPE SEQUENCES
0407 689 :---
0407 690 : PROCESS_ESCAPE - PROCESS ESCAPE SEQUENCES
0407 691 :
0407 692 : SUBROUTINE TO CHECK FOR AND PROCESS ESCAPE SEQUENCES.
0407 693 :
0407 694 : INPUTS:
0407 695 :
0407 696 : R2 = ADDRESS OF NEW INPUT RECORD
0407 697 : R3 = LENGTH OF NEW INPUT RECORD, EXCLUDING NULL AT END OF LINE
0407 698 : R4 = ADDRESS OF RAB
0407 699 :
0407 700 : OUTPUTS:
0407 701 :
0407 702 : R2 = ADDRESS OF NEW INPUT RECORD
0407 703 : R3 = LENGTH OF NEW INPUT RECORD, EXCLUDING NULL AT END OF LINE
0407 704 :---
0407 705 :
0407 706 PROCESS_ESCAPE: ;PROCESS ESCAPE SEQUENCES
0407 707 :
0407 708 :
0407 709 : CHECK FOR ESCAPE SEQUENCES.
0407 710 :
0407 711 : CMPB #1,RAB$W_STV2(R4) ;ESCAPE SEQUENCE?
0407 712 : BLSSU 5$ ;YES, THEN PROCESS
0407 713 : BSBW DCL$LOCKED_STATE ;RESTORE LOCKED KEYPAD STATE
0407 714 : MOVL #1,R0 ;SET REPROMPT STATUS
0407 715 : RSB ;RETURN
0414 716 :
0414 717 :
0414 718 : REPROMPT IF NO KEY OR SYMBOL FOUND.
0414 719 :
0414 720 70$: MOVQ (SP)+,R4 ;RESTORE R4/R5
0417 721 80$: MOVQ (SP)+,R2 ;RESTORE R2/R3
041A 722 : MOVL R3,R1 ;CREATE R1/R2 DESCRIPTOR
041D 723 : BSBW INSERT_COMMAND ;REPROMPT WITH THIS COMMAND
041F 724 : BSBW DCL$LOCKED_STATE ;RESTORE LOCKED KEYPAD STATE
0422 725 : MOVL #3,R0 ;SET REPROMPT STATUS
0425 726 : RSB ;RETURN
0426 727 :
0426 728 :
0426 729 : FIND ASSOCIATED META-KEY NAME.
0426 730 :
0426 731 5$: MOVQ R2,-(SP) ;SAVE R2/R3
0429 732 : CLRL -(SP) ;ALLOCATE SPACE FOR RETURN ADDRESS
042B 733 : PUSHL SP ;POINT AT IT
042D 734 : MOVZBL RAB$W_STV2(R4),-(SP) ;GET TERMINATOR LENGTH
0431 735 : PUSHAB (R2)[R3] ;GET TERMINATOR ADDRESS
0434 736 : CALLS #3,G^GET_KEY_NAME ;LOOK UP THE ESCAPE SEQUENCE
0438 737 : MOVL (SP)+,R2 ;GET ADDRESS OF ASCII META-KEY NAME
043E 738 : BLBC R0,80$ ;SKIP IF NONE
0441 739 :
0441 740 :
0441 741 : SPECIAL CASE UP AND DOWN ARROW KEYS.
0441 742 :
0441 743 :
0441 744 : CMPB (R2),#4 ;IS THE STRING SHORT ENOUGH?
0444 : BGTR 10$ ;NO, THEN SKIP

```

```

OE A4 01 91
   19 1F
01A1 30
50 01 D0
   05
54 8E 7D
52 8E 7D
51 53 D0
   D0 10
   018F 30
50 03 D0
   05
7E 52 7D
   7E D4
   5E DD
7E OE A4 9A
   6243 9F
00000000 GF 03 FB
   52 8E D0
   D6 50 E9

```

```

5055 8F 01 A2 B1 0446 745      CMPW 1(R2),#^A'UP'      ;CHECK FOR UP ARROW
                                BEQL 8$                          ;BRANCH IF MATCH
4E574F44 8F 01 A2 D1 044E 747      CMPL 1(R2),#^A'DOWN'    ;CHECK FOR DOWN ARROW
                                BNEQ 10$                         ;BRANCH IF NO MATCH
                                52 8E 7D 0458 749 7$: MOVQ (SP)+,R2      ;RESTORE R2/R3
                                FED0 31 045B 750      BRW RECALL_NEXT        ;DO THE RECALL
                                52 8E 7D 045E 751 8$: MOVQ (SP)+,R2      ;RESTORE R2/R3
                                FE6B 31 0461 752      BRW RECALL_PREV        ;DO THE RECALL
                                0464 753
                                0464 754
                                0464 755 : FIND ASSOCIATED SYMBOL VALUE.
                                0464 756
                                0464 757 : (SP) = INITIAL OFFSET
                                0464 758 : 4(SP) = NEW COMMAND LENGTH
                                0464 759 : 8(SP) = RAB ADDRESS
                                0464 760 : 12(SP) = R5
                                0464 761 : 16(SP) = INITIAL COMMAND ADDRESS
                                0464 762 : 20(SP) = INITIAL (LATER NEW) COMMAND LENGTH
                                0464 763
                                7E 54 7D 0464 764 10$: MOVQ R4,-(SP)      ;SAVE R4/R5
                                51 82 9A 0467 765      MOVZBL (R2)+,R1          ;GET DESCRIPTOR OF SYMBOL NAME
                                FB93' 30 046A 766      BSBW DCL$SEARCH_KEYPAD  ;SEARCH SYMBOL TABLE FOR MATCH
                                A4 50 E9 046D 767      BLBC R0,70$            ;SKIP IF NOT FOUND
                                0470 768
                                0470 769
                                0470 770 : /ERASE - COPY STRING INTO INPUT BUFFER.
                                0470 771
                                2A 54 04 E1 0470 772      BBC #SYM_V_ERASE,R4,20$ ;IF /NOERASE, THEN SKIP
                                51 DD 0474 773      PUSHL R1                ;SAVE NEW LENGTH
                                7E D4 0476 774      CLRL -(SP)              ;SET INITIAL OFFSET
                                55 10 AE D0 0478 775      MOVL 16(SP),R5          ;GET INITIAL ADDRESS
                                54 DD 047C 776      PUSHL R4                ;SAVE FLAGS
                                65 62 51 28 047E 777      MOVCL3 R1,(R2),(R5)     ;COPY VALUE TO INPUT BUFFER
                                54 8ED0 0482 778      POPL R4                 ;RESTORE FLAGS
                                4D 11 0485 779      BRB 30$                 ;
                                0487 780
                                0487 781 95$: STATUS BUFOVF          ;SET OVERFLOW STATUS
                                0120 30 048E 782      BSBW DCL$LOCKED_STATE  ;RESTORE LOCKED KEYPAD STATE
                                00E0 31 0491 783      BRW 67$                 ;RETURN ERROR
                                0494 784
                                0494 785 96$: STATUS SYMOVF          ;SET OVERFLOW STATUS
                                00D4 31 049B 786      BRW 66$                 ;RETURN
                                049E 787
                                049E 788
                                049E 789 : /NOERASE - COPY STRING INTO INPUT BUFFER.
                                049E 790
                                7E 0C AE 51 C1 049E 791 20$: ADDL3 R1,12(SP),-(SP) ;SAVE NEW LENGTH
                                00000100 8F 6E D1 04A3 792      CMPL (SP),#WRK_C_INPBUFSIZ ;WILL STRING FIT IN BUFFER?
                                DB 1A 04AA 793      BGTRU 95$               ;NO, THEN RETURN ERROR
                                55 0C AE 10 AE C1 04AC 794      ADDL3 16(SP),12(SP),R5 ;FIND CURRENT EOL
                                50 04 AE D0 04B2 795      MOVL 4(SP),R0           ;GET ADDRESS OF RAB
                                53 OF A0 9A 04B6 796      MOVZBL RAB$W_STV2+1(R0),R3 ;GET CURSOR OFFSET FROM EOL
                                55 53 C2 04BA 797      SUBL R3,R5              ;BACK UP TO INSERTION POINT
                                7E 10 AE 53 C3 04BD 798      SUBL3 R3,16(SP),-(SP)  ;SAVE INITIAL OFFSET
                                3E BB 04C2 799      PUSHR #^M<R1,R2,R3,R4,R5> ;SAVE REGISTERS
                                6541 65 53 28 04C4 800      MOVCL3 R3,(R5),(R5)[R1] ;MOVE THE TEXT
                                3E BA 04C9 801      POPR #^M<R1,R2,R3,R4,R5> ;RESTORE REGISTERS

```

```

65 62 54 DD 04CB 802          PUSHL R4          ;SAVE FLAGS
      51 28 04CD 803          MOVCL3 R1,(R2),(R5) ;COPY VALUE TO INPUT BUFFER
      54 8ED0 04D1 804          POPL R4          ;RESTORE FLAGS
      04D4 805
      04D4 806
      04D4 807 ; PROCESS /SET_STATE AND /LOCK
      04D4 808
      1C 54 00DA 30 04D4 809 30$: BSBW DCL$LOCKED_STATE ;RESTORE LOCKED KEYPAD STATE
      54 02 E1 04D7 810          BBC #SYM_V_STATE,R4,50$ ;BRANCH IF NO SET STATE SPECIFIED
      52 51 D0 04DB 811          MOVL R1,R2 ;GET LENGTH/ADDR OF STATE STRING
      51 82 9A 04DE 812          MOVZBL (R2)+,R1
      FB1C 30 04E1 813          BSBW DCL$ALLOC_STATE ;ALLOCATE IT
      AD 50 E9 04E4 814          BLBC R0,96$ ;BRANCH IF NO ROOM FOR SYMBOL
      54 03 E1 04E7 815          BBC #SYM_V_LOCK,R ;BRANCH IF NOT /LOCK
      50 4C AB D0 04EB 816          MOVL PRC [LASTKEY(R11),R0 ;GET OLD LOCKED STATE
      FB0E 30 04EF 817          BSBW DCL$DEALLOC_STATE ;DEALLOCATE IT
      48 AB D0 04F2 818          MOVL PRC_L_CURRKEY(R11),- ;LOCK NEW KEY STATE
      4C AB 04F5 819          PRC_L_LASTKEY(R11)
      04F7 820
      04F7 821 ; PROCESS /ERASE.
      04F7 822
      04F7 823
      34 54 04 E1 04F7 824 50$: BBC #SYM_V_ERASE,R4,52$ ;IF /NOERASE, THEN SKIP
      FE81 30 04FB 825          BSBW ERASE_LINE ;ERASE THE LINE
      2D 54 01 E1 04FE 826          BBC #SYM_V_TERMINATE,R4,52$ ;IF /NOTERMINATE, THEN SKIP
      55 0C AB D0 0502 827          MOVL PRC [OUTRAB(R11),R5 ;SET ADDRESS OF OUTPUT FILE RAB
      50 10 AB D0 0506 828 51$: MOVL PRC_L_TRMLIST(R11),R0 ;GET ADDRESS OF XABTRM ITEM LIST
      10 A0 D0 050A 829          MOVL ITRM [PMPTADDR(R0),- ;WRITE THE PROMPT STRING
      28 A5 050D 830          RAB$[RBF(R5)
      0C A0 B0 050F 831          MOVW ITRM [PMPTLEN(R0),-
      22 A5 0512 832          RAB$[RSZ(R5)
      0514 833          DISABLE
      06 00 F0 0518 834          INSV #0,#RAB$V_PPF_RAT,- ;DISABLE CONTROL Y/C AST'S
      02 A5 08 051B 835          #RAB$$PPF_RAT,RAB$W_ISI(R5) ;DISABLE CR FORMAT WRITES
      051E 836          $PUT RAB=(R5) ;OUTPUT RECORD
      06 02 F0 0527 837          INSV #FAB$M_CR,#RAB$V_PPF_RAT,- ;ENABLE CR FORMAT WRITES
      02 A5 08 052A 838          #RAB$$PPF_RAT,RAB$W_ISI(R5) ;ENABLE CONTROL Y/C AST'S
      052D 839          ENABLE
      052F 840
      052F 841 ; PROCESS /ECHO.
      052F 842
      052F 843
      4C 54 01 E1 052F 844 52$: BBC #SYM_V_TERMINATE,R4,60$ ;IF /NOTERMINATE, THEN SKIP
      38 54 00 E1 0533 845          BBC #SYM_V_ECHO,R4,65$ ;IF /NOECHO, THEN SKIP
      55 0C AB D0 0537 846          MOVL PRC [OUTRAB(R11),R5 ;SET ADDRESS OF OUTPUT FILE RAB
      50 04 AE D0 053B 847 55$: MOVL 4(SP),R0 ;GET SIZE OF OUTPUT RECORD
      10 AE 50 C1 053F 848          ADDL3 R0,16(SP),R1 ;FIND END OF OUTPUT RECORD
      61 0D 90 0544 849          MOVW #^XOD,(R1) ;INSERT CR AT END
      50 6E C2 0547 850          SUBL (SP),R0 ;SUBTRACT OUT INITIAL OFFSET
      28 A5 51 50 C3 054A 851          SUBL3 R0,R1,RAB$L_RBF(R5) ;SET ADDRESS OF OUTPUT RECORD
      22 A5 50 01 A1 054F 852          ADDW3 #1,R0,RAB$W_RSZ(R5) ;SET SIZE OF OUTPUT RECORD
      0554 853          DISABLE ;DISABLE CONTROL Y/C AST'S
      06 00 F0 0558 854          INSV #0,#RAB$V_PPF_RAT,- ;DISABLE CR FORMAT WRITES
      02 A5 08 055B 855          #RAB$$PPF_RAT,RAB$W_ISI(R5)
      055E 856          $PUT RAB=(R5) ;OUTPUT RECORD
      06 02 F0 0567 857          INSV #FAB$M_CR,#RAB$V_PPF_RAT,- ;ENABLE CR FORMAT WRITES
      02 A5 08 056A 858          #RAB$$PPF_RAT,RAB$W_ISI(R5)

```

```

056D 859      ENABLE
056F 860
50 01 D0 056F 861 65$:  MOVL  #1,R0      ;ENABLE CONTROL Y/C AST'S
OC AE 8E D5 0572 862 66$:  TSTL  (SP)+      ;SET NORMAL RETURN STATUS
54 8E D0 0574 863 67$:  MOVL  (SP)+,12(SP) ;POP INITIAL OFFSET
52 8E 7D 0578 864      MOVQ  (SP)+,R4    ;REPLACE OLD LENGTH WITH NEW
7D 057B 865      MOVQ  (SP)+,R2    ;RESTORE R4/R5
05 057E 866      RSB                    ;RESTORE R2/R3
057F 86      ;RETURN
057F 868      ;
057F 869      ; PROCESS /NOTERMINATE.
057F 870      ;
50 10 AB D0 057F 871 60$:  MOVL  PRC_L_TRMLIST(R11),R0 ;GET ADDRESS OF XABTRM ITEM LIST
28 A0 8E D0 0583 872      MOVL  (SP)+,ITRM_L_OFFSET(R0) ;SET CHARACTER TO START ECHOING AT
03 54 28 A0 D6 0587 873      INCL  ITRM_L_OFFSET(R0)
OC AE 28 A0 D4 058A 874      BBC   #SYM_V_ERASE,R4,61$ ;IF /NOERASE, THEN SKIP
18 A0 0C AE D0 0591 875 61$:  MOVL  (SP)+,T2(SP) ;START WITH FIRST CHARACTER
1C A0 08 AE D0 0595 876      MOVW  12(SP),ITRM_W_INILEN(R0) ;REPLACE OLD LENGTH WITH NEW
50 54 8E 7D 059A 877      MOVL  8(SP),ITRM_C_INIADDR(R0) ;SET LENGTH OF INITIAL STRING
OC A0 0C AE D0 059F 878      MOVQ  (SP)+,R4    ;SET ADDRESS OF INITIAL STRING
50 40 A4 D0 05A2 879      MOVQ  (SP)+,R4    ;RESTORE R4/R5
OC A0 30 D0 05A6 880      MOVL  RAB$L_XAB(R4),R0 ;GET ADDRESS OF XABTRM
52 8E 7D 05A8 881      MOVL  #ITRM_K_LENGTH,- ;SET EXPANDED LENGTH OF ITEM LIST
50 03 D0 05AB 882      MOVL  XAB$W_ITMLST_LEN(R0)
7D 05AA 883      MOVQ  (SP)+,R2    ;RESTORE R2/R3
05 05AD 884      MOVL  #3,R0 ;SET REPROMPT RETURN STATUS
05 05B0 885      RSB                    ;RETURN

```



```

05B1 887 .SBTTL RESTORE LOCKED KEYPAD STATE
05B1 888 :---
05B1 889 : DCL$LOCKED_STATE - RESTORE LOCKED KEYPAD STATE
05B1 890 :
05B1 891 : DELETE ANY TEMPORARY KEYPAD STATE THAT MAY HAVE BEEN SET AND RESTORE THE
05B1 892 : LOCKED KEYPAD STATE.
05B1 893 :
05B1 894 : INPUTS:
05B1 895 :
05B1 896 : R11 = ADDRESS OF PRC DATA STRUCTURE
05B1 897 : PRC_L_CURRKEY = CURRENT, POSSIBLY TEMPORARY, KEYPAD STATE
05B1 898 : PRC_L_LASTKEY = DEFAULT LOCKED KEYPAD STATE
05B1 899 :
05B1 900 : OUTPUTS:
05B1 901 :
05B1 902 : PRC_L_CURRKEY IS UPDATED.
05B1 903 :
05B1 904 :---
05B1 905 :
05B1 906 DCL$LOCKED_STATE:: ;RESTORE LOCKED KEYPAD STATE
50 50 DD 05B1 907 PUSHL RO ;SAVE RO
48 AB D1 05B3 908 CMPL PRC_L_CURRKEY(R11),- ;IS TEMPORARY STATE IN EFFECT?
4C AB 05B6 909 CMPL PRC_L_LASTKEY(R11)
OC 13 05B8 910 BEQL 90$ ;NO, THEN DONE
50 48 AB D0 05BA 911 MOVL PRC_L_CURRKEY(R11),RO ;GET ADDRESS OF ASCII STATE
FA3F' 30 05BE 912 BSBW DCL$DEALLOC_STATE ;DEALLOCATE TEMPORARY STATE
4C AB D0 05C1 913 MOVL PRC_L_LASTKEY(R11),- ;RESTORE THE LOCKED STATE
48 AB 05C4 914 MOVL PRC_L_CURRKEY(R11)
50 8ED0 05C6 915 90$: POPL RO ;RESTORE RO
05 05C9 916 RSB ;RETURN
05CA 917

```

```

05CA 919 .SBTTL EXPAND INPUT LINE
05CA 920 :---
05CA 921 : EXPAND - EXPAND INPUT LINE WITH SYMBOL SUBSTITUTIONS
05CA 922 :
05CA 923 : SUBROUTINE TO EXPAND INPUT LINE BY EXECUTING ALL STRING SUBSTITUTION
05CA 924 : COMMANDS. THE UNUSED AREA IN THE EXPANSION BUFFER IS USED TEMPORARILY
05CA 925 : TO HOLD THE EXPANDED COPY OF THE COMMAND LINE. AFTER ALL SUBSTITUTIONS
05CA 926 : ARE PERFORMED, THE EXPANDED COPY IS COPIED BACK INTO THE INPUT RECORD
05CA 927 : BUFFER SO THAT SEMANTIC PARSING CAN CONTINUE.
05CA 928 :
05CA 929 : INPUTS:
05CA 930 :
05CA 931 : WRK_L_CHARPTR = POINTER TO NEW INPUT RECORD
05CA 932 :
05CA 933 : OUTPUTS:
05CA 934 :
05CA 935 : R2 = ADDRESS OF NEW INPUT RECORD
05CA 936 : R3 = LENGTH OF NEW INPUT RECORD, EXCLUDING NULL AT END OF LINE
05CA 937 : WRK_L_CHARPTR = POINTER TO EXPANDED INPUT RECORD
05CA 938 :---
05CA 939 :
05CA 940 EXPAND:
05CA 941 PUSHR #^M<R4,R5,R6,R7,R8,R9> ;EXPAND INPUT LINE
05CA 942 PUSHL WRK_L_MARKPTR(R10) ;SAVE REGISTERS
05CA 943 PUSHL WRK_L_EXPANDBPTR(R10) ;SAVE MARKER POINTER
05CA 944 MOVW WRK_W_FLAGS(R10),-(SP) ;SAVE EXPANSION BUFFER POINTER
05CA 945 BISW #WRK_M_INPSUBST!WRK_M_NOUPCASE!WRK_M_STAR,WRK_W_FLAGS(R10) ;SAVE CURRENT PARSING FLAGS
05CA 946 O5E0 946 ;PREVENT PROCESSING OF !,-,@,ETC.
05CA 947 O5E0 947 ;AND PREVENT UPCASING OF INPUT CHARS
05CA 948 O5E0 948 ;AND ACCEPT '*' AS A TERMINATOR
05CA 949 O5E0 949 CLRW -(SP) ;INITIALIZE ITERATION COUNTER
05CA 950 O5E2 950 :
05CA 951 O5E2 951 : GET NEXT CHARACTER FROM INPUT RECORD
05CA 952 O5E2 952 :
05CA 953 10$: BSBW DCL$GETCHAR ;GET NEXT CHARACTER FROM INPUT LINE
05CA 954 CMPB RO,#^A/'/ ;STRING SUBSTITUTION COMMAND?
05CA 955 BNEQ 70$ ;IF NEQ NO
05CA 956 BBC #WRK_V_QUOTE,WRK_W_FLAGS(R10),20$ ;IF LBC NOT IN QUOTE
05CA 957 BSBW DCL$SETCHAR ;CHECK NEXT CHARACTER
05CA 958 CMPB RO,#^A/'/ ;NEXT CHARACTER ALSO SINGLE QUOTE?
05CA 959 BNEQ 60$ ;IF NEQ NO
05CA 960 BSBW DCL$GETCHAR ;GOBBLE SECOND SINGLE QUOTE
05CA 961 BRB 20$ ;TRY SYMBOL SUBSTITUTION
05CA 962 O5FC 962 :
05CA 963 O5FC 963 : ONE SINGLE QUOTE WAS DETECTED IN A DOUBLE QUOTED STRING - TREAT LITERALLY
05CA 964 O5FC 964 :
05CA 965 60$: MOVZBL #^A/'/,RO ;INSERT SINGLE QUOTE WITHIN STRING
05CA 966 O5FF 966 :
05CA 967 O5FF 967 : IF COMMENT IS DETECTED, THEN DO SUBSTITUTIONS BUT IF AN ERROR IS
05CA 968 O5FF 968 : DETECTED, THEN NO ERROR IS ISSUED AND A NULL STRING IS SUBSTITUTED.
05CA 969 O5FF 969 :
05CA 970 70$: BBS #WRK_V_QUOTE,WRK_W_FLAGS(R10),80$ ;BRANCH IF IN QUOTED STRING
05CA 971 CMPB RO,#^A'^!' ;START OF COMMENT STRING?
05CA 972 BNEQ 80$ ;BRANCH IF NOT A COMMENT
05CA 973 SETBIT WRK_V_COMMENT,WRK_W_FLAGS(R10) ;MARK WE ARE IN A COMMENT
05CA 974 O60E 974 ;SO THAT ERRORS ARE NOT REPORTED
05CA 975 O60E 975 :

```

```

03FO 8F BB
F48A CA DD
F486 CA DD
7E FO AA BO
FO AA OC20 BF AB
7E B4
FA1B' 30
27 50 91
15 12
27 FO AA 04 E1
FA0E' 30
27 50 91
05 12
FA06' 30
1A 11
O5FC 962
O5FC 963
50 27 9A
O5FF 966
O5FF 967
O5FF 968
O5FF 969
OA FO AA 04 E0
21 50 91
05 12
0609 973
060E 974
060E 975

```

```

060E 976 : WRITE THE CURRENT CHARACTER INTO THE EXPANSION BUFFER AND LOOP
060E 977 :
F9EF' 30 060E 978 80$: BSBW DCL$PUTCHAR ;PUT CHARACTER IN EXPANSION BUFFER
CF 12 0611 979 BNEQ 10$ ;IF NOT EOL, KEEP SCANNING
00B9 31 0613 980 BRW 90$ ;END OF LINE - TERMINATE SCAN
0616 981 :
0616 982 : SYMBOL SUBSTITUTION REQUESTED. GET THE SYMBOL AND SEARCH THE SYMBOL
0616 983 : TABLE, AND IF NOT FOUND THERE, TRY AS A LEXICAL FUNCTION.
0616 984 :
F486 CA DD 0616 985 20$: PUSHL WRK_L_EXPANDPTR(R10) ;SAVE PLACE IN EXPANSION BUFFER
7E FO AA BO 061A 986 MOVW WRK_W_FLAGS(R10),-(SP) ;SAVE LEXICAL FLAGS
FO AA 10 AA 061E 987 BICW #WRK_M_QUOTE,WRK_W_FLAGS(R10)
OC EO 0622 988 BBS #WRK_V_COMMENT- ;ARE WE IN A COMMENT?
06 FO AA 0624 989 WRK_W_FLAGS(R10),22$
FO AA 0800 BF AA 0627 990 BICW #WRK_M_NOUPCASE,WRK_W_FLAGS(R10) ;NO, THEN UPCASE THE SYMBOL
062D 991 ;PRETEND WE'RE NOT IN A STRING
062D 992 ;SO GETOKEN STOPS BEFORE END-OF-STRING
50 27 9A 062D 993 22$: MOVZBL #^A/'/,R0 ;INSERT SINGLE QUOTE WITHIN STRING
F9CD' 30 0630 994 BSBW DCL$PUTCHAR ;IN CASE SUBSTITUTION NOT ALLOWED
F9CA' 30 0633 995 BSBW DCL$GETOKEN ;GET/COPY NEXT TOKEN
0636 996 :
0636 997 : IF IN COMMENT, ONLY ALLOW F$VERIFY TO BE SUBSTITUTED
0636 998 :
29 FO AA OC E1 0636 999 BBC #WRK_V_COMMENT,WRK_W_FLAGS(R10),28$ ;BRANCH IF NOT IN COMMENT
04 51 D1 0638 1000 CMPL P1,#4 ;AT LEAST 4 CHARACTER TOKEN?
1C 19 063E 1001 BLSS 25$ ;IF NOT, SKIP SYMBOL SUBSTITUTION
7E 62 D0 0640 1002 MOVL (R2),-(SP) ;PUSH FIRST FOUR CHARACTERS
7E 51 7D 0643 1003 MOVQ R1, -(SP) ;SAVE DESCRIPTOR
51 04 9A 0646 1004 MOVZBL #4,R1 ;CREATE TEMPORARY DESCRIPTOR
52 08 AE 9E 0649 1005 MOVAB 8(SP),R2
F9B0' 30 064D 1006 BSBW DCL$UPCASE ;UPCASE THE SYMBOL
51 8E 7D 0650 1007 MOVQ (SP)+,R1 ;RESTORE THE REGISTERS
45562446 8F 8E D1 0653 1008 CMPL (SP)+,#^A'F$VE' ;IS IT F$VERIFY WITHIN A COMMENT?
08 13 065A 1009 BEQL 28$ ;IF SO, ALLOW SUBSTITUTION
02 AE F486 CA D0 065C 1010 25$: MOVL WRK_L_EXPANDPTR(R10),2(SP) ;COPY 'SYMBOL TO EXPANSION BUFFER
45 11 0662 1011 BRB 50$
51 D5 0664 1012 28$: TSTL R1 ;ZERO LENGTH SYMBOL?
1F 13 0666 1013 BEQL 40$ ;!IF EQL YES
56 51 7D 0668 1014 MOVQ R1,R6 ;SAVE STRING PARAMETERS
F992' 30 066B 1015 BSBW DCL$SEARCH ;SEARCH FOR SYMBOL
16 50 E8 066E 1016 BLBS R0,40$ ;IF LBS SYMBOL DEFINITION FOUND
0671 1017 CLRBIT WRK_V_INPSUBST,WRK_W_FLAGS(R10) ;DO PROCESSING OF !,-,@,ETC.
0676 1018 ;TO ALLOW CONTINUATIONS IN FUNCT. ARGS
09 FO AA OC E1 0676 1019 BBC #WRK_V_COMMENT- ;ARE WE IN A COMMENT?
51 56 7D 0678 1020 WRK_W_FLAGS(R10),32$
F97F' 30 067B 1021 MOVQ R6,R1 ;YES, THEN UPCASE 'F$VERIFY'
56 51 7D 067E 1022 BSBW DCL$UPCASE
F979' 30 0681 1023 MOVQ R1,R6
F976' 30 0684 1024 32$: BSBW DCL$LEXIF ;EVALUATE LEXICAL FUNCTION
0687 1025 40$: BSBW DCL$CVT STRING ;CONVERT RESULT TO CHARACTER STRING
068A 1026 SETBIT WRK_V_INPSUBST,WRK_W_FLAGS(R10) ;DISABLE !,-,@,ETC.
068F 1027 :
068F 1028 : TRAILING SINGLE QUOTES ARE OPTIONAL AFTER SYMBOL - GOBBLE IT
068F 1029 :
F96E' 30 068F 1030 BSBW DCL$SETCHAR ;PEEK AT NEXT CHARACTER
27 50 91 0692 1031 CMPB R0,#^A/'/ ;SYMBOL END WITH SINGLE QUOTE?
03 12 0695 1032 BNEQ 30$ ;BRANCH IF TRAILING SINGLE QUOTE

```

```

F966' 30 0697 1033      BSBW  DCL$GETCHAR      ;GOBBLE TRAILING QUOTE
      069A 1034 :
      069A 1035 : APPEND THE SYMBOL TRANSLATION TO THE FRONT OF THE INPUT BUFFER
      069A 1036 : AND RESET THE INPUT POINTER TO POINT TO IT. THIS IS DONE IN CASE
      069A 1037 : THERE ARE ANY SINGLE QUOTES IN THE TRANSLATION WHICH REQUIRE SUBSTITUTION.
      069A 1038 :
      069A 1039 30$:  SUBL  R1,WRK_L_CHARPTR(R10) ;CALCULATE ADDRESS TO COPY STRING
      069F 1040      MOVL  WRK_L_CHARPTR(R10),R0 ;POINT TO NEW POSITION IN INBUF
01 AO 62 51 28 06A4 1041      MOVC  R1,(R2),1(R0) ;CONCATENATE STRING TO INPUT BUFFER
      06A9 1042 :
      06A9 1043 : A MAXIMUM OF 1000 SUBSTITUTIONS IS ALLOWED PER LINE, TO PREVENT ANY
      06A9 1044 : INFINITE LOOPS FROM OCCURRING DUE TO RECURSIVE SUBSTITUTIONS.
      06A9 1045 :
      06A9 1046 50$:  MOVW  (SP)+,WRK_W_FLAGS(R10) ;RESTORE FLAGS QUOTE AND NOUPCASE
      06AD 1047      POPL  WRK_L_EXPANDPTR(R10) ;RETRIEVE ADDRESS IN EXPANSION BUFFER
FF28 6E 01 03E8 8F 3D 06B2 1048      ACBW  #1000,#1,(SP),10$ ;CHECK FOR SUBSTITUTION LOOP
      06BA 1049      STATUS EXPSYN ;EXPRESSION SYNTAX ERROR
      F486 CA 57 56 C1 06C1 1050      ADDL3 R6,R7,WRK_L_EXPANDPTR(R10) ;POINT AT END OF SYMBOL
      F48A CA 77 9E 06C7 1051      MOVAB -(R7),WRK_L_MARKPTR(R10) ;SET ADDRESS OF ""
      F931' 31 06CC 1052      BRW  DCL$PARSEERR ;REPORT ERROR
      06CF 1053 :
      06CF 1054 : END OF LINE DETECTED. MOVE EXPANDED LINE BACK INTO THE INPUT BUFFER
      06CF 1055 : FOR THE LEXICAL PROCESSING.
      06CF 1056 :
      06CF 1057 90$:  MOVL  4(SP),R7 ;GET SAVED EXPANSION POINTER
      59 F486 CA 04 AE D0 06D3 1058      SUBL3 R7,WRK_L_EXPANDPTR(R10),R9 ;CALCULATE LENGTH OF EXPANDED LINE
      58 F996 CA 57 C3 06D9 1059      MOVAB WRK_G_INPBUF+WRK_C_INPBUFSIZ(R10),R8 ;FIND END OF INPUT BUFFER
      58 59 C2 06DE 1060      SUBL  R9,R8 ;COMPUTE ADDRESS TO MOVE LINE TO
      68 67 59 28 06E1 1061      MOVC  R9,(R7),(R8) ;MOVE EXPANDED LINE TO END OF INPUT BUFFER
      52 58 7D 06E5 1062      MOVQ  R8,R2 ;SET INPUT LINE PARAMETERS
      F48E CA FF A2 9E 06E8 1063      DECL  R3 ;DECREMENT LENGTH TO EXCLUDE EOL CHAR
      F48E CA FF A2 9E 06EA 1064      MOVAB -1(R2),WRK_L_CHARPTR(R10) ;SET ADDRESS OF EXPANDED INPUT LINE
      FO AA 8E B5 06F0 1065      TSTW  (SP)+ ;REMOVE ITERATION COUNTER FROM STACK
      FO AA 8E B0 06F2 1066      MOVW  (SP)+,WRK_W_FLAGS(R10) ;RESTORE CURRENT PARSING FLAGS
      F486 CA 8ED0 06F6 1067      POPL  WRK_L_EXPANDPTR(R10) ;RESTORE EXPANSION BUFFER POINTER
      F48A CA 8ED0 06FB 1068      POPL  WRK_L_MARKPTR(R10) ;RESTORE MARKER POINTER
      03F0 8F BA 0700 1069      POPR  #^M<R4,R5,R6,R7,R8,R9> ;RESTORE REGISTERS
      05 0704 1070      RSB

```

```

0705 1072 .SBTTL SPECIAL TOKEN LEXICAL PROCESSING
0705 1073 :---
0705 1074 :
0705 1075 : THIS ROUTINE IS CALLED TO PROCESS LEXICAL TOKENS WITH SPECIAL CHARACTERS.
0705 1076 : THE LIST OF CHARACTERS ARE:
0705 1077 :
0705 1078 :     &SYMBOL      - TOKEN IS SUBSTITUTED WITH SYMBOL VALUE
0705 1079 :     @FILESPEC   - TOKEN IS SUBSTITUTED WITH FIRST RECORD CONTAINED
0705 1080 :                   IN THE ASSOCIATED PROCEDURE FILE.
0705 1081 :
0705 1082 : INPUTS:
0705 1083 :
0705 1084 :     R11 = ADDRESS OF PRC AREA
0705 1085 :     R10 = ADDRESS OF WRK AREA
0705 1086 :     R0  = FIRST CHARACTER IN TOKEN
0705 1087 :
0705 1088 : ADDITIONAL INPUT AND OUTPUT SPECIFICATIONS ARE GIVEN WITH EACH ROUTINE.
0705 1089 :
0705 1090 :---
0705 1091 :
0705 1092 DCL$SPECIAL::
26 50 91 0705 1093      CMPB      RO,#^A'&'          ;SUBSTITUTION?
40 8F 50 91 0708 1094      BEQL      AMPERSAND        ;BRANCH IF SO
22 13 91 070A 1095      CMPB      RO,#^A'@'          ;INDIRECTION?
05 0710 1096      BEQL      INDIRECT          ;BRANCH IF SO
0705 1097      RSB              ;IF NOT KNOWN, IGNORE IT

```

```

0711 1099 .SBTTL PROCESS &SYMBOL CONSTRUCT
0711 1100 :---
0711 1101 :
0711 1102 : HANDLE &SYMBOL CONSTRUCT.
0711 1103 :
0711 1104 : INPUTS:
0711 1105 :
0711 1106 : R0 = CHARACTER IN INPUT BUFFER (&)
0711 1107 : R1/R2 = DESCRIPTOR OF TOKEN, INCLUDING '&' CHARACTER
0711 1108 :
0711 1109 : OUTPUTS:
0711 1110 :
0711 1111 : R0 = NEXT CHARACTER IN INPUT BUFFER
0711 1112 : R1/R2 = DESCRIPTOR OF UPDATED TOKEN
0711 1113 :
0711 1114 : THE EXPANSION BUFFER WILL BE OVERWRITTEN WITH SYMBOL VALUE
0711 1115 :
0711 1116 :---
0711 1117 AMPERSAND:
      3C  BB 0711 1118 PUSH  #*M<R2,R3,R4,R5> ;SAVE REGISTERS
      52  D6 0713 1119 INCL  R2 ;POINT TO SYMBOL NAME
      51  D7 0715 1120 DECL  R1 ;ADJUST LENGTH OF SYMBOL NAME
      03  13 0717 1121 BEQL  40$ ;BRANCH IF NULL SYMBOL NAME
      F8E4' 30 0719 1122 BSBW  DCL$SYMBOL_STRING ;SEARCH LOCAL/GLOBAL SYMBOL TABLES
00 BE 62 51 28 071C 1123 40$: MOVC  R1,(R2),@ (SP) ;OVERWRITE '&SYMBOL' WITH ITS VALUE
      F486 CA 53 D0 0721 1124 MOVL  R3,WRK_L_EXPANDPTR(R10) ;SET NEW EXPANSION BUFFER POINTER
      3C  BA 0726 1125 POPR  #*M<R2,R3,R4,R5> ;RESTORE REGISTERS
      F8D5' 30 0728 1126 BSBW  DCL$SETCHAR ;PEEK AT NEXT CHARACTER IN INPUT BUFFER
51 F486 CA 52 C3 0728 1127 SUBL3 R2,WRK_L_EXPANDPTR(R10),R1 ;CALCULATE LENGTH OF NEW TOKEN
      05 0731 1128 RSB

```

```

0732 1130 .SBTTL PROCESS @FILESPEC CONSTRUCT
0732 1131 :---
0732 1132 :
0732 1133 : HANDLE @FILESPEC CONSTRUCT.
0732 1134 :
0732 1135 : INPUTS:
0732 1136 :
0732 1137 : RO = CHARACTER IN INPUT BUFFER (@)
0732 1138 :
0732 1139 : AT THIS POINT, ONLY THE @ HAS BEEN SEEN, AND THE FILESPEC
0732 1140 : HAS NOT YET BEEN PROCESSED.
0732 1141 :
0732 1142 : OUTPUTS:
0732 1143 :
0732 1144 : RO = NEXT CHARACTER IN INPUT BUFFER
0732 1145 :---
0732 1146 INDIRECT:
19 68 AB 05 E2 0732 1147 BBSS #PRC_V_IND,PRC_W_FLAGS(R11),90$ ;SKIP IF INDIRECTION DISABLED
0737 1148 :AND DISABLE WHILE PROCESSING FILESPEC
F48A CA DD 0737 1149 PUSHL WRK_L_MARKPTR(R10) ;SAVE OLD PARSE POSITION
F8C2' 30 073B 1150 BSBW DCL$MARK ;SET PARSE POSITION FOR '@' PROCESSING
F8BF' 30 073E 1151 BSBW DCL$STACKIND ;STACK CURRENT INDIRECT LEVEL
09 50 E9 0741 1152 CLRBIT PRC_V_IND,PRC_W_FLAGS(R11) ;ENABLE INDIRECTION AGAIN
F48A CA 8E D0 0745 1153 BLBC RO,DCL$CHARERROR ;BRANCH IF ERROR DETECTED
F8B5 30 0748 1154 MOVL (SP)+,WRK_L_MARKPTR(R10) ;RESTORE OLD PARSE POSITION
05 074D 1155 BSBW DCL$INPUT ;GET FIRST LINE, CHARACTER OF PROCEDURE
0750 1156 90$: RSB

```

```
0751 1158 .SBTTL ERROR HANDLER IN CHARACTER INPUT ROUTINES
0751 1159 :---
0751 1160 :
0751 1161 : THIS ROUTINE IS CALLED TO PERFORM ANY SPECIAL PROCESSING WHEN AN
0751 1162 : ERROR IS DETECTED BY THE CHARACTER INPUT ROUTINES.
0751 1163 :
0751 1164 : INPUTS:
0751 1165 :
0751 1166 : RO = STATUS CODE
0751 1167 :
0751 1168 : OUTPUTS:
0751 1169 :
0751 1170 : NONE
0751 1171 :---
0751 1172 DCL$CHARERROR::
F8AC' 30 0751 1173 BSBW DCL$ERRORMSG ;REPORT ERROR MESSAGE
      50 DD 0754 1174 PUSHL RO ;SAVE STATUS CODE
F8A7' 30 0756 1175 BSBW DCL$FLUSH ;FLUSH INPUT BUFFER
      50 BEDD 0759 1176 POPL RO ;RESTORE STATUS CODE
F8A1' 30 075C 1177 BSBW DCL$SET_STATUS ;SET COMPLETION STATUS
F89E' 31 075F 1178 BRW DCL$RESTART ;START THE PARSING ALL OVER AGAIN
```



```

0762 1180 .SBTTL RECALL COMMAND
0762 1181 :
0762 1182 :+ DCL$RECALL - RECALL COMMAND
0762 1183 :
0762 1184 : THIS ROUTINE IS CALLED TO EXECUTE THE DCL RECALL COMMAND. THE RECALL
0762 1185 : COMMAND REPROMPTS THE USER WITH A COMMAND THAT HE HAS PREVIOUSLY ENTERED
0762 1186 : OR DISPLAYS FOR THE USER THE LIST OF ALL THE COMMANDS IN THE COMMAND BUFFER.
0762 1187 :
0762 1188 : INPUTS:
0762 1189 :
0762 1190 : R8 = ADDRESS OF SCRATCH BUFFER DESCRIPTOR
0762 1191 : R9 = ADDRESS OF SCRATCH STACK
0762 1192 : R10 = ADDRESS OF COMMAND WORK AREA
0762 1193 : R11 = ADDRESS OF PROCESS WORK AREA
0762 1194 :
0762 1195 : OUTPUTS:
0762 1196 :
0762 1197 : RO = STATUS CODE
0762 1198 :-
0762 1199 :
0762 1200 DCL$RECALL::
0762 1201 :
0762 1202 :
0762 1203 : SKIP IF ENTERED FROM A COMMAND PROCEDURE.
0762 1204 :
0762 1205 : BBS #PRC V MODE,- ; IF SET, NOT INTERACTIVE
0762 1206 : PRC_Q_FLAGS(R11),3$ ;
0762 1207 : BBS #PRC V YLEVEL,- ; IF SET, AT CONTROL Y/C LEVEL
0762 1208 : PRC_Q_FLAGS(R11),5$ ;
0762 1209 : TSTL PRC_L_INDEPTH(R11) ; INDIRECT LEVEL ZERO?
0762 1210 : BEQL 5$ ; BRANCH IF YES
0762 1211 3$: BRW 90$ ; RETURN
0762 1212 :
0762 1213 :
0762 1214 :
0762 1215 : REMOVE THIS COMMAND FROM THE RECALL BUFFER.
0762 1216 :
0762 1217 5$: MOVL PRC_L_RECALLPTR(R11),- ; UPDATE WRK RECALL PTR
0762 1218 : WRK_L_RECALLPTR(R10) ;
0762 1219 : CLRQ -(SP) ; ALLOCATE COMMAND DESCRIPTOR
0762 1220 : PUSHL SP ; POINT TO IT
0762 1221 : CALLS #1,DCL$GET_PREV_COMMAND ; GET THE RECALL COMMAND
0762 1222 : CLRQ (SP)+ ; DISCARD THE DESCRIPTOR
0762 1223 : MOVL WRK_L_RECALLPTR(R10),- ; UPDATE PRC RECALL PTR
0762 1224 : PRC_L_RECALLPTR(R11) ;
0762 1225 :
0762 1226 :
0762 1227 : PARSE THE COMMAND.
0762 1228 :
0762 1229 :
0762 1230 : MOVL #1,R6 ; ASSUME BACKING UP ONE COMMAND
0762 1231 : BSBW DCL$GETDVAL ; GET NEXT DESCRIPTOR
0762 1232 : CMPL #PTR_K_ENDLINE,R5 ; EOL?
0762 1233 : BEQL 30$ ; YES, EXECUTE THE COMMAND
0762 1234 : CMPL #PTR_K_PARAMETR,R5 ; NUMBER?
0762 1235 : BNEQ 20$ ; NO, MUST BE /ALL
0762 1236 : MOVQ R1,R2 ; COPY DESCRIPTOR
0762 1237 : MOVQ R2,-(SP) ; SAVE IT ON THE STACK

```

```

0A 68 AB E0
08 68 AB E0
5C AB D5
C3 13
0113 31
012F CB D0
EA AA
7E 7C
5E DD
0000'CF 01 FB
8E 7C
EA AA D0
012F CB
56 01 D0
F86F' 30
55 04 D1
6C 13
55 03 D1
52 51 7D
7E 52 7D

```

```

51 01 D0 07A1 1237 MOVL #1,R1 ; SET DECIMAL RADIX
      F859' 30 07A4 1238 BSBW DCL$CNVNOEDIT ; CONVERT NUMBER TO BINARY
52 8E 7D 07A7 1239 MOVQ (SP)+,R2 ; RESTORE THE DESCRIPTOR
      50 D5 07AA 1240 TSTL R0 ; WAS IT A NUMBER?
      05 13 07AC 1241 BEQL 10$ ; YES, CHECK BOUNDS
56 52 7D 07AE 1242 MOVQ R2,R6 ; SAVE THE STRING DESCRIPTOR
      19 11 07B1 1243 BRB 35$ ; GET THE COMMAND
14 51 D1 07B3 1244 10$: CMPL R1,#WRK_C_RECALLMAX ; WITHIN BOUNDS?
      0C 1A 07B6 1245 BGTRU 95$ ; NO, SIGNAL ERROR
56 51 D0 07B8 1246 MOVL R1,R6 ; GET BACKUP COUNT
      07 13 07BB 1247 BEQL 95$ ; SIGNAL ERROR IF ZERO
      43 11 07BD 1248 BRB 30$ ; GET THE COMMAND
56 14 D0 07BF 1249 20$: MOVL #WRK_C_RECALLMAX,R6 ; /ALL WAS SPECIFIED
      5F 11 07C2 1250 BRB 50$ ; DISPALY THE COMMANDS
      07C4 1251
      07C4 1252 95$: STATUS IVVALU ; SET INVALID VALUE STATUS
      05 07CB 1253 RSB ; RETURN
      07CC 1254
      07CC 1255 ;
      07CC 1256 ; FETCH SPECIFIED COMMAND BY STRING.
      07CC 1257 ;
59 14 D0 07CC 1258 35$: MOVL #WRK_C_RECALLMAX,R9 ; /ALL WAS SPECIFIED
      7E 7C 07CF 1259 CLRQ -(SP) ; ALLOCATE COMMAND DESCRIPTOR
      5E DD 07D1 1260 36$: PUSHL SP ; PUSH DESCR ADDRESS
04 B8 0000'CF 01 FB 07D3 1261 CALLS #1,DCL$GET_PREV_COMMAND ; GET THE PREVIOUS COMMAND
      04 BE 6E 28 07D8 1262 MOVQ3 (SP),@4(SP),@4(R8) ; COPY COMMAND TO SCRATCH BUFFER
      68 6E D0 07DE 1263 MOVL (SP),(R8) ; COPY INITIAL LENGTH
      51 68 7D 07E1 1264 MOVQ (R8),R1 ; SET COMMAND DESCRIPTOR
      F819' 30 07E4 1265 BSBW DCL$TRIM ; UPCASE AND TRIM THE COMMAND
      51 56 D1 07E7 1266 CMPL R6,R1 ; MAKE SURE WE ARE CHECKING A SUBSTRING
      08 1A 07EA 1267 BGTRU 361$ ; SKIP THIS ONE IF NOT
62 56 00 67 56 2D 07EC 1268 CMPCS R6,(R7),#0,R6,(R2) ; DO THE STRINGS MATCH?
      1A 13 07F2 1269 BEQL 37$ ; YES, THEN REPROMPT
      DA 59 F5 07F4 1270 361$: SOBGTR R9,36$ ; LOOP TILL SEARCHED ALL COMMANDS
      5E 08 C0 07F7 1271 ADDL #8,SP ; RESTORE THE STACK
50 00038238 8F D0 07FA 1272 MOVL #CLIS_CMDNOTFND,R0 ; SET COMMAND NOT FOUND STATUS
      05 0801 1273 RSB ; RETURN
      0802 1274
      0802 1275 ;
      0802 1276 ; FETCH SPECIFIED COMMAND BY NUMBER.
      0802 1277 ;
      7E 7C 0802 1278 30$: CLRQ -(SP) ; ALLOCATE COMMAND DESCRIPTOR
      5E DD 0804 1279 32$: PUSHL SP ; PUSH DESCR ADDRESS
0000'CF 01 FB 0806 1280 CALLS #1,DCL$GET_PREV_COMMAND ; GET THE PREVIOUS COMMAND
      F6 56 F5 080B 1281 SOBGTR R6,32$ ; LOOP TILL BACKED UP FAR ENOUGH
      080E 1282
      080E 1283 ;
      080E 1284 ; SET UP RAB TO REPROMPT WITH THE COMMAND.
      080E 1285 ;
      51 8E 7D 080E 1286 37$: MOVQ (SP)+,R1 ; GET COMMAND DESCRIPTOR
54 14 AB D0 0811 1287 MOVL PRC_L_INDINPRAB(R11),R4 ; ASSUME USING INDIRECT INPUT RAB
      0B E1 0815 1288 BBC #PRC_YLEVEL,- ; BRANCH IF NOT AT CTRL/Y LEVEL
      04 68 AB 0817 1289 PRC_Q_FLAGS(R11),40$
54 08 AB D0 081A 1290 MOVL PRC_L_INPRAB(R11),R4 ; USE LEVEL 0 INPUT RAB
      FBCE 30 081E 1291 40$: BSBW INSERT_COMMAND ; INSERT THE COMMAND
      64 11 0821 1292 BRB 90$
      0823 1293

```

```

0823 1294 :
0823 1295 : DISPLAY ALL COMMANDS
0823 1296 :
20312000 8F D0 0823 1297 50$: MOVL #^X20312000,- : SET INITIAL NUMBER STRING
      F890 CA 0829 1298      WRK_G_INPBUF-6(R10) :
50 EA AA D0 082C 1299 55$: MOVL WRK_L_RECALLPTR(R10),R0 : HAVE WE ALREADY DISPLAYED THEM ALL?
50 FF A0 9E 0830 1300      MOVAB -1(R0),R0 :
51 0133 CB 9E 0834 1301      MOVAB PRC_G_COMMANDS(R11),R1 : WRAP IF NECESSARY
51 50 D1 0839 1302      CMPL R0,R1 :
50 0533 CB 9E 083C 1303      BGEQU 57$ :
      60 95 0843 1304 57$: MOVL PRC_G_COMMANDS+PRC_C_CMDBUFSIZ-1(R11),R0 :
      40 13 0845 1305      TSTB (R0) : BRANCH IF NO COMMANDS LEFT
      7E 7C 0847 1306      BEQL 90$ :
      SE DD 0849 1307      CLRQ -(SP) :
0000 CF 01 FB 084B 1308      PUSHL SP : ALLOCATE COMMAND DESCRIPTOR
51 8E 7D 0850 1310      CALLS #1,DCL$GET_PREV_COMMAND : PUSH DESCR ADDRESS
51 03 C0 0853 1311      MOVQ (SP)+,R1 : GET THE PREVIOUS COMMAND
52 03 C2 0856 1312      ADDL #3,R1 : GET COMMAND DESCRIPTOR
7E 51 7D 0859 1313      SUBL #3,R2 : INSERT THE COMMAND NUMBER
      F7A1 30 085C 1314      MOVQ R1,-(SP) : SAVE COMMAND DESCRIPTOR
39 51 8E 7D 085F 1315      BSBW DCL$MSGOUT : OUTPUT THE COMMAND
      01 A2 91 0862 1316      MOVQ (SP)+,R1 : RESTORE COMMAND DESCRIPTOR
      11 12 0866 1317      CMPB 1(R2),#^X39 : IS ONES DIGIT A 9?
31 62 91 0868 1318      BNEQ 60$ : NO, THEN SKIP
      07 12 086B 1319      CMPB (R2),#^X31 : IS TENS DIGIT A 1?
62 2F32 8F B0 086D 1320      BNEQ 58$ : NO, THEN SKIP
      05 11 0872 1321      MOVW #^X2F32,(R2) : YES, INSERT '2/'
62 2F31 8F B0 0874 1322 58$: BRB 60$ : BRANCH
      01 A2 96 0879 1323 60$: MOVW #^X2F31,(R2) : INSERT '1/'
      087C 1324      INCB 1(R2) : INCREMENT THE COMMAND NUMBER
      087C 1325 :
      087C 1326 : CHECK FOR NO MORE COMMANDS.
      087C 1327 :
012F CB D1 087C 1328      CMPL PRC_L_RECALLPTR(R11),- : HAVE WE RETURNED TO THE ORIGIN?
      EA AA 0880 1329      WRK_L_RECALLPTR(R10) :
      03 13 0882 1330      BEQL 90$ :
      A5 56 F5 0884 1331      SOBGTR R6,55$ : OR DISPLAYED THE MAX # OF COMMANDS?
      0887 1332 :
      0887 1333 90$: STATUS NORMAL : SET SUCCESS
      05 088E 1334      RSB : ALL DONE
      088F 1335 :
      088F 1336      .END

```

```

SS.TMP1      = 00000001
SS.TMP2      = 00000065
SST1        = 00000001
ABORT        0000028D R    02
AMPERSAND    00000711 R    02
CLIS_BUFOVF  = 00038018
CLIS_CMDNOTFND = 00038238
CLIS_EXPSYN  = 00038038
CLIS_IVVALU  = 00038088
CLIS_NORMAL  = 00030001
CLIS_SYMOVF  = 00038138
CLIS_USGOTO  = 00038148
DCL$ABORT    ***** X    02
DCL$ALLOC_STATE ***** X    02
DCL$BACKUP_CHAR ***** X    02
DCL$CHARERROR 00000751 RG   02
DCL$CLOSE_PPFS ***** X    02
DCL$CNVNOEDIT ***** X    02
DCL$CRLF      ***** X    02
DCL$CVT_STRING ***** X    02
DCL$DEALGOTO  ***** X    02
DCL$DEALLOC_STATE ***** X    02
DCL$DISABLE    ***** X    02
DCL$DSBCONTRLY ***** X    02
DCL$ERRORMSG   ***** X    02
DCL$FLUSH      ***** X    02
DCL$GETCHAR    ***** X    02
DCL$GETDVAL    ***** X    02
DCL$GETTOKEN   ***** X    02
DCL$GET_CURR_COMMAND ***** X    02
DCL$GET_NEXT_COMMAND ***** X    02
DCL$GET_PREV_COMMAND ***** X    02
DCL$INPUT      00000005 RG   02
DCL$LEXIF      ***** X    02
DCL$LOCKED_STATE 000005B1 RG   02
DCL$MARK       ***** X    02
DCL$MSGOUT     ***** X    02
DCL$PARSERR    ***** X    02
DCL$PUTCHAR    ***** X    02
DCL$PUT_COMMAND ***** X    02
DCL$PUT_SEGMENT ***** X    02
DCL$RECALL     00000762 RG   02
DCL$RESTART    ***** X    02
DCL$SEARCH     ***** X    02
DCL$SEARCH_KEYPAD ***** X    02
DCL$SETCHAR    ***** X    02
DCL$SET_STATUS ***** X    02
DCL$SPECIAL    00000705 RG   02
DCL$STACKIND   ***** X    02
DCL$SYM_STRING ***** X    02
DCL$STRIP      ***** X    02
DCL$UNSTACK    ***** X    02
DCL$UPCASE     ***** X    02
DEV$V_TRM      ***** X    02
DVIS_DEVDEPEND2 = 0000001C
END_OF_LIST    00000322 R    02
ERASE          00000000 R    02

```

```

ERASE_LINE    0000037F R    02
ERRORT        000000AE R    02
EXESC_SYSEFN  ***** X    02
EXPAND        000005CA R    02
FAB$M_CR      = 00000002
FAB$W_IFI     = 00000002
GET_INPUT     00000045 R    02
GET_KEY_NAME  ***** X    02
INDIRECT      00000732 R    02
INSERT_COMMAND 000003EF R    02
IO_ERROR      0000019B R    02
ITRM_C_LENGTH 00000030
ITRM_C_MINLEN 00000018
ITRM_K_LENGTH 00000030
ITRM_K_MINLEN 00000018
ITRM_L_INIADDR 0000001C
ITRM_L_INIRET 00000020
ITRM_L_MODIFIERS 00000004
ITRM_L_MODRET 00000008
ITRM_L_OFFRET 0000002C
ITRM_L_OFFSET 00000028
ITRM_L_PMPTADDR 00000010
ITRM_L_PMPTRET 00000014
ITRM_W_INICODE 0000001A
ITRM_W_INILEN 00000018
ITRM_W_MODCODE 00000002
ITRM_W_MODLEN 00000000
ITRM_W_OFFCODE 00000026
ITRM_W_OFFLEN 00000024
ITRM_W_PMPTCODE 0000000E
ITRM_W_PMPTLEN 0000000C
PRC_B_CONTINUE 000000F3
PRC_B_DEFRADIX 000000AE
PRC_B_EXMDEPMOD 000000AD
PRC_B_EXMDEPWID 000000AC
PRC_B_EXONLYL 0000012D
PRC_B_FLAGS2   000000AF
PRC_B_IMGFLAG 00000078
PRC_B_OUTFLAGS 0000012C
PRC_B_PROMPTLEN 000000F0
PRC_C_CMDBUFSIZ = 00000401
PRC_C_LENGTH   00000534
PRC_G_COMMANDS 00000133
PRC_G_PROMPT   000000F4
PRC_K_LENGTH   00000534
PRC_L_CURRKEY 00000048
PRC_L_EXMDEPADR 000000A8
PRC_L_EXTARG   00000094
PRC_L_EXTBLK   0000008C
PRC_L_EXTCOD   0000009C
PRC_L_EXTHND   00000090
PRC_L_EXTPRM   00000098
PRC_L_IDFLNK   0000008C
PRC_L_IMGACTSTS 00000080
PRC_L_INDCLOCK 0000007C
PRC_L_INDEPTH 0000005C
PRC_L_INDFAB   0000001C

```

READREC
Symbol table

- READ AN INPUT RECORD

C 10

16-SEP-1984 00:11:48 VAX/VMS Macro V04-00
4-SEP-1984 23:42:34 [DCL.SRC]READREC.MAR;1

Page 34
(18)

```

PRC_L_INDINPRAB      00000014
PRC_L_INDOUTRAB     00000018
PRC_L_INPRAB        00000008
PRC_L_LASTKEY       0000004C
PRC_L_LSTSTATUS     000000B0
PRC_L_ONCTLY        000000B8
PRC_L_ONERROR       0000006C
PRC_L_OUTOFBAND     000000B4
PRC_L_OUTRAB        0000000C
PRC_L_OUTRABCTX     00000118
PRC_L_PPFLIST       00000070
PRC_L_RECALLPTR     0000012F
PRC_L_RESTART       00000058
PRC_L_SAVAP         00000000
PRC_L_SAVFP         00000004
PRC_L_SEVERITY      00000050
PRC_L_SPWN          000000C0
PRC_L_STACKLM       000000A4
PRC_L_STACKPT       000000A0
PRC_L_STATUS        00000054
PRC_L_STS           00000084
PRC_L_STV           00000088
PRC_L_SYMBOL        00000060
PRC_L_TMBX          00000074
PRC_L_TRMLIST       00000010
PRC_Q_ALLOCREG      00000020
PRC_Q_COMMAND       000000E0
PRC_Q_FLUSHTIME     000000D0
PRC_Q_GLOBAL        00000028
PRC_Q_IMAGENAME     000000D8
PRC_Q_KEYPAD        00000040
PRC_Q_LABEL         00000030
PRC_Q_LOCAL         00000038
PRC_Q_SAVEPRIV     000000E8
PRC_T_OUTDVI        0000C11C
PRC_V_AUTOLOGO      = 00000008
PRC_V_CARRCNTL      = 00000000
PRC_V_CNTRL         = 00000001
PRC_V_EOFLOGO       = 0000000E
PRC_V_FLUSH         = 00000006
PRC_V_GOTO          = 00000004
PRC_V_IND           = 00000005
PRC_V_MODE          = 00000006
PRC_V_VERIFY        = 00000007
PRC_V_YLEVEL        = 0000000B
PRC_W_ASTIOSB       000000C6
PRC_W_ASTRETN       000000C8
PRC_W_ASTSTATUS     000000C4
PRC_W_ATTMBX        0000007A
PRC_W_FLAGS         00000068
PRC_W_INPCHAN       00000064
PRC_W_ONLEVEL       0000006A
PRC_W_OUTIFI        00000114
PRC_W_OUTISI        0G000116
PRC_W_OUTMBXCHN     000000CA
PRC_W_OUTMBXREF     000000CE
PRC_W_OUTMBXSIZ     000000CC

```

```

PRC_W_PMPTCTRL      000000F1
PRC_W_WAITIOSB     00000066
PROCESS_ESCAPE      00000407 R      02
PROCESS_INPUT       000000B1 R      02
PROCESS_RECALL      000002C5 R      02
PTR_B_LEVEL         00000004
PTR_B_NUMBER        00000005
PTR_B_PARMCNT       00000006
PTR_B_VALUE         00000000
PTR_C_LENGTH        0000000C
PTR_K_ENDLINE       = 00000004
PTR_K_LENGTH        0000000C
PTR_K_PARAMETER     = 00000003
PTR_L_DESCR         00000000
PTR_L_ENTITY        00000008
RABSL_CTX           = 00000018
RABSL_RBF           = 00000028
RABSL_STS           = 00000008
RABSL_STV           = 0000000C
RABSL_UBF           = 00000024
RABSL_XAB           = 00000040
RABSS_PPF_RAT       = 00000008
RABSV_PPF_IND       = 0000000E
RABSV_PPF_RAT       = 00000006
RABSW_ISI           = 00000002
RABSW_RSZ           = 00000022
RABSW_STVO          = 0000000C
RABSW_STV2          = 0000000E
RABSW_USZ           = 00000020
RECALL_CURR         00000315 R      02
RECALL_NEXT         0000032E R      02
RECALL_PREV         000002CF R      02
REINP               00000007 R      02
RETURN              0000017F R      02
RMSS_CONTROLY      ***** X      02
RMSS_EOF            ***** X      02
RMSS_RSA            ***** X      02
RMSS_SYS            ***** X      02
SILENT_LOGOUT       0000029B RG     02
SPECIAL             00000196 R      02
SS$ EXQUOTA         ***** X      02
STATUS              00000287 R      02
SYM_B_FLAGS         0000000B
SYM_B_NONUNIQUE     0000000B
SYM_B_TYPE          0000000A
SYM_L_BL            00000004
SYM_L_FL            00000000
SYM_T_SYMBOL        0000000C
SYM_V_ECHO          = 00000000
SYM_V_ERASE         = 00000004
SYM_V_LOCK          = 00000003
SYM_V_STATE         = 00000002
SYM_V_TERMINATE     = 00000001
SYM_W_SIZE          00000008
SYSSCANCEL          ***** GX     02
SYSSCANEXH         ***** GX     02
SYSSCLOSE           ***** GX     02

```

READREC
Symbol table

- READ AN INPUT RECORD

D 10

16-SEP-1984 00:11:48 VAX/VMS Macro V04-00
4-SEP-1984 23:42:34 [DCL.SRC]READREC.MAR;1

SYS\$EXIT	*****	GX	02	WRK_W_PMPTLEN	FFFFF99E
SYS\$GET	*****	GX	02	XABS\$W_ITMLST_LEN	= 0000000C
SYS\$GETDVIW	*****	GX	02	_\$\$_	= 000000EF
SYS\$PUT	*****	GX	02		
SYS\$WAIT	*****	GX	02		
TT2\$V_ANSICRT	= 00000018				
WRK_B_CMDOPT	FFFFFFC3				
WRK_B_MAXPARM	FFFFFFD0				
WRK_B_MINPARM	FFFFFFD1				
WRK_B_PARMCNT	FFFFFFCE				
WRK_B_PARMSUM	FFFFFFCF				
WRK_B_RECALLCNT	FFFFFFC5				
WRK_B_VALLEV	FFFFFFC4				
WRK_B_VERBTYP	FFFFFFC2				
WRK_C_INPBUFSIZ	= 00000100				
WRK_C_LENGTH	FFFFFF486				
WRK_C_RECALLMAX	= 00000014				
WRK_G_BUFFER	FFFFFF492				
WRK_G_INPBUF	FFFFFF896				
WRK_G_RESULT	FFFFFF9B6				
WRK_K_LENGTH	FFFFFF486				
WRK_L_CHARPTR	FFFFFF48E				
WRK_L_DISALLOW	FFFFFFE6				
WRK_L_ERRORRTN	FFFFFF9AE				
WRK_L_EXPANDPTR	FFFFFF486				
WRK_L_IMAGE	FFFFFFE2				
WRK_L_MARKPTR	FFFFFF48A				
WRK_L_PAROUT	FFFFFFD2				
WRK_L_PMPTADDR	FFFFFF9A2				
WRK_L_PROMPTRTN	FFFFFF9A6				
WRK_L_PROPTR	FFFFFFFC6				
WRK_L_QUABLK	FFFFFFCA				
WRK_L_READRTN	FFFFFF9AA				
WRK_L_RECALLPTR	FFFFFFEA				
WRK_L_RSLEND	FFFFFFB6				
WRK_L_RSLNXT	FFFFFFBA				
WRK_L_SAVAP	FFFFFFF8				
WRK_L_SAVFP	FFFFFFFC				
WRK_L_SAVSP	FFFFFFF4				
WRK_L_SIGNALRTN	FFFFFFD6				
WRK_L_SPECRTN	FFFFFF9B2				
WRK_L_TAB_VEC	FFFFFFDE				
WRK_L_VERB	FFFFFFBE				
WRK_M_INPSUBST	= 00000400				
WRK_M_NOUPCASE	= 00000800				
WRK_M_QUOTE	= 00000010				
WRK_M_STAR	= 00000020				
WRK_V_COMMAND	= 00000001				
WRK_V_COMMENT	= 0000000C				
WRK_V_CONTIN	= 00000003				
WRK_V_INPSUBST	= 0000000A				
WRK_V_INQUIRE	= 00000007				
WRK_V_QUOTE	= 00000004				
WRK_V_TRAILSPC	= 00000009				
WRK_W_FLAGS	FFFFFFF0				
WRK_W_FLAGS2	FFFFFFF2				
WRK_W_IMGCHAN	FFFFFFEE				

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	FFFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL\$ZCODE	0000088F (2191.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.80
Command processing	80	00:00:00.70	00:00:06.75
Pass 1	351	00:00:15.31	00:00:43.74
Symbol table sort	0	00:00:01.60	00:00:04.80
Pass 2	235	00:00:03.98	00:00:10.02
Symbol table output	33	00:00:00.23	00:00:00.64
Psect synopsis output	2	00:00:00.03	00:00:00.26
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	710	00:00:21.94	00:01:07.02

The working set limit was 1500 pages.
 79799 bytes (156 pages) of virtual memory were used to buffer the intermediate code.
 There were 60 pages of symbol table space allocated to hold 1072 non-local and 94 local symbols.
 1336 source lines were read in Pass 1, producing 22 object records in Pass 2.
 58 pages of virtual memory were used to define 41 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
_\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	13
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	20
TOTALS (all libraries)	33

1301 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:READREC/OBJ=OBJ\$:READREC MSRC\$:READREC/UPDATE=(ENH\$:READREC)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/LIB

The image displays a grid of 100 small terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different system command and its corresponding output. The commands are as follows:

- Row 1: 10 identical windows showing the command `RECALL` and its output.
- Row 2: 10 identical windows showing the command `RECALL` and its output.
- Row 3: 10 identical windows showing the command `RECALL` and its output.
- Row 4: 10 identical windows showing the command `RECALL` and its output.
- Row 5: 10 identical windows showing the command `RECALL` and its output.
- Row 6: 10 identical windows showing the command `RECALL` and its output.
- Row 7: 10 identical windows showing the command `RECALL` and its output.
- Row 8: 10 identical windows showing the command `RECALL` and its output.
- Row 9: 10 identical windows showing the command `RECALL` and its output.
- Row 10: 10 identical windows showing the command `RECALL` and its output.

Some windows in the grid contain specific command names in large letters, such as `MESSAGE LIS`, `READREC LIS`, `RECALL SUB LIS`, `RPCLINT LIS`, `PARSENT LIS`, and `ON LIS`, which correspond to the labels in the provided text. The rest of the windows show the standard output for the `RECALL` command, including file names and their locations.

MESSAGE
LIS

READREC
LIS

RECALL SUB
LIS

RPCLINT
LIS

PARSENT
LIS

ON
LIS