```
DDDDDDDDDDD         CCCCCCCCCCC  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 LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLLLLLLLLLLLLLL
```

FILEID**INITIAL

I 16

```
IIIIII    NN      NN    IIIIII    TTTTTTTTTT    IIIIII      AAAAAA    LL
IIIIII    NN      NN    IIIIII    TTTTTTTTTT    IIIIII    AAAAAAAA    LL
  II      NN      NN      II          TT          II      AA      AA  LL
  II      NN      NN      II          TT          II      AA      AA  LL
  II      NNNN    NN      II          TT          II      AA      AA  LL
  II      NNNN    NN      II          TT          II      AA      AA  LL
  II      NN  NN  NN      II          TT          II      AA      AA  LL
  II      NN  NN  NN      II          TT          II      AA      AA  LL
  II      NN    NNNN      II          TT          II      AAAAAAAAAA  LL
  II      NN    NNNN      II          TT          II      AAAAAAAAAA  LL
  II      NN      NN      II          TT          II      AA      AA  LL          ....
  II      NN      NN      II          TT          II      AA      AA  LL          ....
IIIIII    NN      NN    IIIIII        TT        IIIIII    AA      AA  LLLLLLLLLL  ....
IIIIII    NN      NN    IIIIII        TT        IIIIII    AA      AA  LLLLLLLLLL  ....

LL              IIIIII      SSSSSSSS
LL              IIIIII    SSSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  INITIAL - COMMAND INTERPRETER INITIALIZATION
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ; D. N. CUTLER 29-MAR-77
0000    28 ;
0000    29 ; COMMAND LANGUAGE INTERPRETER INITIALIZATION
0000    30 ;
0000    31 ; MODIFICATIONS:
0000    32 ;
0000    33 ;       V03-022 HWS0091         Harold Schultz          22-Jul-1984
0000    34 ;               Initialize PRC_B_EXONLYL, PRC_V_SAVCMDV, and
0000    35 ;               PRC_V_SAVIMGV
0000    36 ;
0000    37 ;       V03-021 HWS0043         Harold Schultz          30-Mar-1984
0000    38 ;               Don't preset CTL$GT_CLINAME.
0000    39 ;
0000    40 ;       V03-020 HWS0033         Harold Schultz          15-Mar-1984
0000    41 ;               When spawning a sub-process, pass on to RMS the verify
0000    42 ;               image flag setting of the parent process.
0000    43 ;
0000    44 ;       V03-019 TMK0001         Todd M. Katz            07-Mar-1984
0000    45 ;               A hash code field, LNMX$W_HASH, has been added to every
0000    46 ;               translation block of every logical name and logical name table.
0000    47 ;               This hash code field will be used in an optimization of logical
0000    48 ;               name table name processing. However, within the context of a
0000    49 ;               spawned process, during processing of the logical name table
0000    50 ;               records, the contents of the hash code field in every
0000    51 ;               translation block should just be ignored.
0000    52 ;
0000    53 ;       V03-018 HWS0009         Harold Schultz  13-Feb-1984
0000    54 ;               Add PRC_V_CARRCNTL flag processing to indicate
0000    55 ;               presence or absence of carriage control in prompt
0000    56 ;               field instead of using contents of prompt field.
0000    57 ;
```

```
0000     58 ;       V03-017 HWS0001         Harold Schultz  03-Feb-1984
0000     59 ;               Don't set protection mask for private logical name
0000     60 ;               tables.
0000     61 ;
0000     62 ;       V03-016 PCG0018         Peter George    22-Sep-1983
0000     63 ;               Fill in IDF_L_FILENAME for initial stack frame.
0000     64 ;               Correctly turn off image verification in subprocesses.
0000     65 ;
0000     66 ;       V03-015 PCG0017         Peter George    12-Sep-1983
0000     67 ;               Expect unwanted logical names to have confine attribute.
0000     68 ;               Propagate CCL bits from PPD data structure.
0000     69 ;               Set CTL$GT_CLINAME.
0000     70 ;
0000     71 ;       V03-014 PCG0016         Peter George    16-Aug-1983
0000     72 ;               Correctly supply address of IOSB to IO$M_TT_PROCESS QIO.
0000     73 ;               Do not perfrom IO$M_TT_PROCESS QIO in nowait subprocesses.
0000     74 ;               Remove code that deassigns transmitted SYS$INPUT and SYS$OUTPUT.
0000     75 ;
0000     76 ;       V03-013 PCG0015         Peter George    29-Jun-1983
0000     77 ;               Use event flags more intelligently.
0000     78 ;               Do an IO$M_TT_PROCESS set mode QIO when creating an
0000     79 ;               interactive process.
0000     80 ;
0000     81 ;       V03-012 PCG0014         Peter George    13-Jun-1983
0000     82 ;               Fix bug in initial command procedure parameter creation.
0000     83 ;               Call RMS rundown for loginout.
0000     84 ;               Reformat new system service calls.
0000     85 ;               Remove DCL_L_TAB_VEC.
0000     86 ;
0000     87 ;       V03-011 RAS0157         Ron Schaefer    27-May-1983
0000     88 ;               Add support for new logical names structures.
0000     89 ;
0000     90 ;       V03-010 PCG0013         Peter George    27-May-1983
0000     91 ;               Add support for output log flushing and image verification.
0000     92 ;
0000     93 ;       V03-009 PCG0012         Peter George    20-Apr-1983
0000     94 ;               Expect prompt and keypad state to be passed by spawn.
0000     95 ;               Do not treat %XFF from LOGINOUT as a null string.
0000     96 ;
0000     97 ;       V03-008 PCG0011         Peter George    29-Mar-1983
0000     98 ;               Sort out CTX_C_KEY* definitions.
0000     99 ;
0000    100 ;       V03-007 PCG0010         Peter George    15-Feb-1983
0000    101 ;               Convert to new structure level.
0000    102 ;               Init P1-P8 for login command procedures.
0000    103 ;               Create and define $RESTART and BATCH$RESTART.
0000    104 ;               Increase logical name buffers from 64 to LNM$C_NAMLENGTH.
0000    105 ;               Make P1-P8 global symbols in batch jobs.
0000    106 ;               Add support for keypad symbol definitions.
0000    107 ;               Speed up spawn symbol allocation.
0000    108 ;               Hook up terminal XAB.
0000    109 ;
0000    110 ;       V03-006 PCG0009         Peter George    28-Jan-1983
0000    111 ;               Change default PRC_W_ONLEVEL.
0000    112 ;               Override SYS$OUTPUT definition transmitted in a SPAWN.
0000    113 ;
0000    114 ;       V03-005 PCG0008         Peter George    10-Jan-1983
```

```
0000   115 ;               Define MAX_DEPTH symbolically.
0000   116 ;               Remove all references to ERRIFI and ERRRAB.
0000   117 ;               Init PRC_W_OUTISI, PRC_W_OUTIFI, PRC_L_OUTRABCTX, and
0000   118 ;               PRC_T_OUTDVI.  Create initial SYS$OUTPUT.
0000   119 ;               Set up exit handler once here, instead of on every
0000   120 ;               image activation.  Terminate process if captive account
0000   121 ;               and cannot open command procedure.
0000   122 ;
0000   123 ;       V03-004 PCG0007         Peter George    29-Dec-1982
0000   124 ;               Do not reset global symbols in succeeding
0000   125 ;               procedures in a batch job.
0000   126 ;
0000   127 ;       V03-003 PCG0006         Peter George    01-Dec-1982
0000   128 ;               Comment out references to the recall buffer.
0000   129 ;
0000   130 ;       V03-002 PCG0005         Peter George    19-Nov-1982
0000   131 ;               Make prompt string constants global.
0000   132 ;               Allow up to 16 stack levels.
0000   133 ;
0000   134 ;       V03-001 PCG0004         Peter George    20-Aug-1982
0000   135 ;               Initialize PRC_L_NXTCMDPTR.
0000   136 ;               Set initial prompt string.
0000   137 ;---
```

```
                        0000   139 ;
                        0000   140 ; MACRO LIBRARY CALLS
                        0000   141 ;
                        0000   142
                        0000   143         PRCDEF                          ; CLI PROCESS WORK AREA
                        0000   144         PRDDEF                          ; CLI PROCESS RMS AREA
                        0000   145         WRKDEF                          ; COMMAND WORK AREA
                        0000   146         IDFDEF                          ; INDIRECT PROCEDURE FRAME
                        0000   147         SYMDEF                          ; SYMBOL TABLE DEFINITIONS
                        0000   148         CTXDEF                          ; SPAWN CONTEXT RECORD FORMATS
                        0000   149         ITRMDEF                         ; TERMINAL XAB ITEM LIST
                        0000   150         $FABDEF                         ; FILE ACCESS BLOCK
                        0000   151         $RABDEF                         ; RECORD ACCESS BLOCK
                        0000   152         $XABTRMDEF                      ; TERMINAL XAB
                        0000   153         $TRMDEF                         ; TERMINAL DRIVER ITEM LIST SYMBOLS
                        0000   154         $PPDDEF                         ; PROCESS PERMANENT DATA AREA
                        0000   155         $PSLDEF                         ; PROCESSOR STATUS FIELDS
                        0000   156         $DIBDEF                         ; GETDEV CHARACTERISTICS BLOCK
                        0000   157         $LOGDEF                         ; LOGICAL NAME TABLE CODES
                        0000   158         $DEVDEF                         ; DEVICE CHARACTERISTICS DEFINITIONS
                        0000   159         $LNMSTRDEF                      ; NEW LOGICAL NAME STRUCTURES
                        0000   160         $LNMDEF                         ; NEW LOGICAL NAME DEFINITIONS
                        0000   161         $CLIMSGDEF                      ; CLI MESSAGE CODES
                        0000   162         $IODEF                          ; DEFINE QIO CODES
                        0000   163
                00000000   164         .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                        0000   165
                        0000   166 DCL:
            4C 43 44 00' 0000   167         .ASCIC  'DCL'
                     03  0000
                        0004   168 SYS$ERROR:
   52 4F 52 52 45 24 53 59 53 00' 0004   169         .ASCIC  'SYS$ERROR'
                     09  0004
                        000E   170 TRUE:
               45 55 52 54 00' 000E   171         .ASCIC  'TRUE'
                     04  000E
                        0013   172 FALSE:
            45 53 4C 41 46 00' 0013   173         .ASCIC  'FALSE'
                     05  0013
                        0019   174 BATCH$RESTART:
41 54 53 45 52 24 48 43 54 41 42 00' 0019   175         .ASCIC  'BATCH$RESTART'
                  54 52  0025
                     0D  0019
                        0027   176 DEFAULT:
         54 4C 55 41 46 45 44 00' 0027   177         .ASCIC  'DEFAULT'
                     07  0027
                        002F   178 FLUSH_RATE:                             ; ONE MINUTE IN DELTA TIME REPRESENTATION
                DC3CBA00  002F   179         .LONG   ^XDC3CBA00
                FFFFFFFF  0033   180         .LONG   ^XFFFFFFFF
                        0037   181 DCL$CRLF::
                     0D  0037   182         .BYTE   ^X0D
                     0A  0038   183         .BYTE   ^X0A
                        0039   184 DCL$T_PROMPT::
             20 20 24 20  0039   185         .ASCII  ' $  '
                00000005  003D   186 DCL$C_PROMPTLEN == 5
                00000010  003D   187 MAX_DEPTH == 16
                        003D   188
```

```
                                003D   189 ;
                                003D   190 ; TABLE OF RESERVED SYMBOLS
                                003D   191 ;
                                003D   192 RESERVED:                                                    ;
                             05 003D   193          .BYTE   5                                           ;MAXIMUM LENGTH OF RESTART VALUE
                54 52 41 54 53 45 52 24 00' 003E   194          .ASCIC  '$RESTART'                          ;RESTART VALUE SYMBOL
                                08 003E
                             58 0047   195          .BYTE   PRC_L_RESTART                               ;OFFSET TO RESTART VALUE ADDRESS
                             01 0048   196          .BYTE   1                                           ;MAXIMUM LENGTH OF SEVERITY LEVEL
             59 54 49 52 45 56 45 53 24 00' 0049   197          .ASCIC  '$SEVERITY'                         ;ERROR SEVERITY LEVEL SYMBOL
                                09 0049
                             50 0053   198          .BYTE   PRC_L_SEVERITY                              ;OFFSET TO SEVERITY VALUE ADDRESS
                             0A 0054   199          .BYTE   10                                          ;MAXIMUM LENGTH OF STATUS VALUE
                   53 55 54 41 54 53 24 00' 0055   200          .ASCIC  '$STATUS'                           ;STATUS VALUE SYMBOL
                                07 0055
                             54 005D   201          .BYTE   PRC_L_STATUS                                ;OFFSET TO STATUS VALUE ADDRESS
                             00 005E   202          .BYTE   0                                           ; --- END OF TABLE
```

```
                                005F    204          .SBTTL  COMMAND INTERPRETER START UP
                                005F    205 ;+
                                005F    206 ; DCL$STARTUP - COMMAND INTERPRETER START UP
                                005F    207 ;
                                005F    208 ; THIS ENTRY POINT IS JUMPED TO AT THE CONCLUSION OF LOGGING A USER ONTO
                                005F    209 ; THE SYSTEM. ALL INPUT AND OUTPUT FILES ARE OPEN AND THE COMMAND LANGUAGE
                                005F    210 ; INDEPENDENT DATA AREA HAS BEEN INITIALIZED.
                                005F    211 ;-
                                005F    212
                           00000000     213          .PSECT  DCL$$BASE,BYTE,RD,NOWRT
                                0000    214
          5E  00000008'GF   D0  0000    215          MOVL    G^CTL$AL_STACK+8,SP     ;RELOAD SUPERVISOR STACK POINTER
                        5D  D4  0007    216          CLRL    FP                      ;INDICATE NO PREVIOUS FRAME
              005F'CF   6E  FA  0009    217          CALLG   (SP),W^DCL$STARTUP      ;SETUP INITIAL CALL FRAME
                                000E    218          $EXIT_S
                                0017    219
                           0000005F     220          .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                                005F    221 DCL$STARTUP::                           ;COMMAND INTERPRETER START UP
                           0000 005F    222          .WORD   ^M<>                    ;ENTRY MASK
                                7E  D4  0061    223          CLRL    -(SP)           ;SETUP DUMMY PSL (@PRC_L_SAVAP=PRVPSL)
                         68'AF  6E  FA  0063    224          CALLG   (SP),B^10$      ;CREATE DUMMY FP AFTER DUMMY AP
                                04  0067    225          RET
                           0000 0068    226 10$:     .WORD   0
                    6D   0000'CF  9E  006A    227          MOVAB   W^DCL$CONDHAND,(FP)     ;ESTABLISH CONDITION HANDLER
          00000000'GF   00000002'EF  9E  006F    228          MOVAB   L^DCL$UTLSERV+2,G^CTL$AL_CLICALBK ; SET CALL BACK VECTOR
                   5A   00000000'GF  9E  007A    229          MOVAB   G^CTL$AG_CLIDATA,R10    ;GET ADDRESS OF PPD
                                0081    230
                                0081    231 ;
                                0081    232 ; INITIALIZE CLI PROCESS WORK AREA.  IF SECOND OR GREATER STEP OF A BATCH
                                0081    233 ; JOB, THEN DO NOT RESET PRC_Q_ALLOCREG, PRC_Q_GLOBAL, PRC_L_LSTSTATUS,
                                0081    234 ; PRC_L_STATUS, OR PRC_L_SEVERITY.
                                0081    235 ;
                     5B   08 AA  D0  0081    236          MOVL    PPD$Q_CLIREG+4(R10),R11         ; GET ADDRESS OF CLI PRIVATE STORAGE
                           02  E1  0085    237          BBC     #PPD$V_CONTINUE,-               ; FIRST STEP IN BATCH JOB?
                        2B 02 AA        0087    238                  PPD$W_FLAGS(R10),15$            ;
                        7E   20 AB  7D  008A    239          MOVQ    PRC_Q_ALLOCREG(R11),-(SP)       ; NO, SAVE PRC_Q_ALLOCREG
                        7E   28 AB  7D  008E    240          MOVQ    PRC_Q_GLOBAL(R11),-(SP)         ;   AND PRC_Q_GLOBAL
                    7E   00B0 CB    D0  0092    241          MOVL    PRC_L_LSTSTATUS(R11),-(SP)      ;   AND PRC_L_LSTSTATUS
                                0097    242          ASSUME PRC_L_STATUS EQ PRC_L_SEVERITY+4 ;   AND PRC_L_STATUS
                        7E   50 AB  7D  0097    243          MOVQ    PRC_L_SEVERITY(R11),-(SP)       ;   AND PRC_L_SEVERITY
                    00   6E   00  2C  009B    244          MOVC5   #0,(SP),#0,-                    ; ZERO ALL STORAGE
                      6B   04 AA        009F    245                  PPD$Q_CLIREG(R10),(R11)         ;
                      50 AB   8E  7D  00A2    246          MOVQ    (SP)+,PRC_L_SEVERITY(R11)       ; RESTORE PRC FIELDS
                    00B0 CB   8E  D0  00A6    247          MOVL    (SP)+,PRC_L_LSTSTATUS(R11)      ;
                      28 AB   8E  7D  00AB    248          MOVQ    (SP)+,PRC_Q_GLOBAL(R11)         ;
                      20 AB   8E  7D  00AF    249          MOVQ    (SP)+,PRC_Q_ALLOCREG(R11)       ;
                              07  11  00B3    250          BRB     16$                             ;
                    00   6E   00  2C  00B5    251 15$:     MOVC5   #0,(SP),#0,-                    ; ZERO ALL STORAGE
                      6B   04 AA        00B9    252                  PPD$Q_CLIREG(R10),(R11)         ;
                         6B   5C  7D  00BC    253 16$:     MOVQ    AP,PRC_L_SAVAP(R11)             ; SAVE INITIAL AP AND FP
                                00BF    254
                                00BF    255 ;
                                00BF    256 ; SET UP PRC BASED EXIT HANDLER BLOCK (REST OF BLOCK IS ZERO).
                                00BF    257 ;
                         0000'CF  9E  00BF    258          MOVAB   W^DCL$EXITHAND,-                ; SET ADDRESS OF EXIT HANDLER
                         0090 CB      00C3    259                  PRC_L_EXTHND(R11)               ;
                         009C CB  9E  00C6    260          MOVAB   PRC_L_EXTCOD(R11),-             ; SET ADDRESS OF REASON FOR EXIT
```

```
                0098 CB        00CA   261                 PRC_L_EXTPRM(R11)                ;
                              00CD   262
                              00CD   263 ;
                              00CD   264 ; COPY AND ACT ON INFORMATION IN PPD
                              00CD   265 ;
                  1E AA   B0  00CD   266         MOVW    PPD$W_INPCHAN(R10),-             ; COPY INPUT CHANNEL
                  64 AB       00D0   267                 PRC_W_INPCHAN(R11)               ;
          0C 02 AA    01    E1  00D2   268         BBC     #PPD$V_MODE,PPD$W_FLAGS(R10),20$; COPY JOB MODE
                00C0 8F   A8  00D7   269         BISW    #PRC_M_MODE!PRC_M_VERIFY,-       ; AND TURN VERIFY ON IF BATCH
                  68 AB       00DB   270                 PRC_W_FLAGS(R11)                 ;
      00AF CB    80 8F   88  00DD   271         BISB    #PRC_M_VERIMAGE,PRC_B_FLAGS2(R11);
           02000000 8F   C8  00E3   272 20$:    BISL    #PRC_M_CTRLY,-                   ; ENABLE CTRL/Y
                00B4 CB       00E9   273                 PRC_L_OUTOFBAND(R11)             ;
                      00    E1  00EC   274         BBC     #PPD$V_NOCTLY,-                  ; COPY NOCONTROLY MODE
                   06 02 AA       00EE   275                 PPD$W_FLAGS(R10),25$             ;
                              00F1   276         CLRBIT  PRC_V_CTRLY,PRC_L_OUTOFBAND(R11);
           00AC CB    04   B0  00F7   277 25$:    MOVW    #4,PRC_B_EXMDEPQID(R11)          ; SET EXAMINE MODE TO HEX,WIDTH TO 4
                0133 CB   9E  00FC   278         MOVAB   PRC_G_COMMANDS(R11),-            ; ADDRESS OF RECALL BUFFER
                012F CB       0100   279                 PRC_L_RECALLPTR(R11)             ;
                              0103   280
                              0103   281 ;
                              0103   282 ; SET UP INITIAL PROMPT.
                              0103   283 ;
      00F1 CB    FF30 CF   B0  0103   284         MOVW    DCL$CRLF,PRC_W_PMPTCTRL(R11)     ; SET CARRIAGE CONTROL
                              010A   285         SETBIT  PRC_V_CARRCNTL,PRC_W_FLAGS(R11) ; INDICATE CR LF IN PROMPT
      00F3 CB    FF27 CF   D0  010E   286         MOVL    DCL$T_PROMPT,PRC_B_CONTINUE(R11); SET DEFAULT PROMPT
                      05   90  0115   287         MOVB    #DCL$C_PROMPTLEN,-               ; SET PROMPT LENGTH
                00F0 CB       0117   288                 PRC_B_PROMPTLEN(R11)             ;
                              011A   289
                              011A   290 ;
                              011A   291 ; FOR BATCH JOBS, SETUP TO EXIT ON ERRORS.  FOR INTERACTIVE JOBS,
                              011A   292 ; DO AN IMPLIED "SET NOON".
                              011A   293 ;
      6A AB    0202 8F   B0  011A   294         MOVW    #2@8!2,PRC_W_ONLEVEL(R11)        ; ASSUME "ON ERROR THEN EXIT"
        06 68 AB    06   E0  0120   295         BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),30$; IF BATCH JOB, THIS IS OK
      6A AB    0808 8F   B0  0125   296         MOVW    #8@8!8,PRC_W_ONLEVEL(R11)        ; IF INTERACTIVE, "SET NOON"
```

```
                                F 1
                     012B   298 ;
                     012B   299 ; INITIALIZE CLI SYMBOL TABLE AND GLOBAL, LOCAL, LABEL, AND KEYPAD
                     012B   300 ; TABLE HEADERS.
                     012B   301 ;
        02    E0     012B   302 30$:     BBS     #PPD$V_CONTINUE,-                       ; IF FIRST STEP IN BATCH JOB
  17 02 AA           012D   303                 PPD$W_FLAGS(R10),35$                    ; THEN INIT HEADER AND GLOBAL TABLE
50   0C AA    7D     0130   304             MOVQ    PPD$Q_CLISYMTBL(R10),R0             ; GET LENGTH/ADDRESS OF REGION
20 AB    51    D0     0134   305             MOVL    R1,PRC_Q_ALLOCREG(R11)              ; INIT LISTHEAD OF ALLOCATION REGION
      81    D4     0138   306             CLRL    (R1)+                               ; CLEAR LINK TO NEXT FREE BLOCK
   61    50    D0     013A   307             MOVL    R0,(R1)                             ; SET LENGTH OF THIS FREE BLOCK
50   28 AB    9E     013D   308             MOVAB   PRC_Q_GLOBAL(R11),R0                ; GET ADDRESS OF GLOBAL TABLE LISTHE
   60    50    D0     0141   309             MOVL    R0,(R0)                             ; INIT GLOBAL SYMBOL TABLE EMPTY
   80    80    D0     0144   310             MOVL    (R0)+,(R0)+                         ;
50   30 AB    9E     0147   311 35$:     MOVAB   PRC_Q_LABEL(R11),R0                 ; GET ADDRESS OF LABEL TABLE LISTHEA
   60    50    D0     014B   312             MOVL    R0,(R0)                             ; INIT LABEL TABLE EMPTY
   80    80    D0     014E   313             MOVL    (R0)+,(R0)+                         ;
50   38 AB    9E     0151   314             MOVAB   PRC_Q_LOCAL(R11),R0                 ; GET ADDRESS OF LOCAL TABLE LISTHEA
   60    50    D0     0155   315             MOVL    R0,(R0)                             ; INIT CURRENT LOCAL SYMBOL TABLE EM
   80    80    D0     0158   316             MOVL    (R0)+,(R0)+                         ;
50   40 AB    9E     015B   317             MOVAB   PRC_Q_KEYPAD(R11),R0                ; GET ADDRESS OF KEYPAD TABLE LISTHE
   60    50    D0     015F   318             MOVL    R0,(R0)                             ; INIT KEYPAD TABLE EMPTY
   80    80    D0     0162   319             MOVL    (R0)+,(R0)+                         ;
                     0165   320
                     0165   321 ;
                     0165   322 ; INITIALIZE PRC_B_EXONLYL, PRC_V_SAVCMDV, AND PRC_V_SAVIMGV
                     0165   323 ;
  012D CB    94     0165   324             CLRB    PRC_B_EXONLYL(R11)                  ; CLEAR EXE-ONLY PROCEDURE FLAG
      0C    8A     0169   325             BICB    #PRC_M_SAVCMDV!PRC_M_SAVIMGV,-      ; INIT. SAVED COMM. AND IMG.
  012C CB          016B   326                     PRC_B_OUTFLAGS(R11)                 ;    VERIFICATION FLAGS.
                     016E   327 ;
                     016E   328 ; INITIALIZE PRC_L_CURRKEY AND PRC_L_LASTKEY.
                     016E   329 ;
52   FEB5 CF    9E     016E   330             MOVAB   DEFAULT,R2                          ; GET NORMAL STATE DESCRIPTOR
   51    82    9A     0173   331             MOVZBL  (R2)+,R1                            ;
      FE87'    30     0176   332             BSBW    DCL$ALLOC_STATE                     ; ALLOCATE NORMAL STATE SYMBOL
   48 AB    D0     0179   333             MOVL    PRC_L_CURRKEY(R11),-                ; LOCK THAT STATE
   4C AB          017C   334                     PRC_L_LASTKEY(R11)                  ;
                     017E   335
                     017E   336 ;
                     017E   337 ; CREATE RESERVED SYMBOLS $STATUS, $SEVERITY, AND $RESTART.
                     017E   338 ;
        02    E2     017E   339             BBSS    #PPD$V_CONTINUE,-                   ; IF FIRST STEP IN JOB
  36 02 AA           0180   340                     PPD$W_FLAGS(R10),48$                ; THEN INIT GLOBAL SYMBOLS
56   FEB6 CF    9E     0183   341             MOVAB   RESERVED,R6                         ; GET ADDRESS OF RESERVED SYMBOLS
55   28 AB    9E     0188   342             MOVAB   PRC_Q_GLOBAL(R11),R5                ; GET ADDRESS OF SYMBOL TABLE LISTHE
   51    86    9A     018C   343 40$:     MOVZBL  (R6)+,R1                            ; GET MAXIMUM LENGTH OF VALUE STRING
      2E    13     018F   344             BEQL    50$                                 ; IF EQL END OF TABLE
   52    56    D0     0191   345             MOVL    R6,R2                               ; SET ANY ADDRESS FOR SYMBOL VALUE
   53    86    9A     0194   346             MOVZBL  (R6)+,R3                            ; CONSTRUCT DESCRIPTOR OF SYMBOL NAM
   54    56    D0     0197   347             MOVL    R6,R4                               ;
   56    53    C0     019A   348             ADDL    R3,R6                               ; ADJUST TO ADDRESS OFFSET
   50    01    9A     019D   349             MOVZBL  #SYM_K_PERM,R0                      ; DEFINE SYMBOL AS PERMANENT
      FE5D'    30     01A0   350             BSBW    DCL$ALLOCSYM                        ; ALLOCATE AND INSERT PERMENENT SYMB
50   0C A1    9A     01A3   351             MOVZBL  SYM_T_SYMBOL(R1),R0                 ; GET LENGTH OF SYMBOL NAME
50   0D A140  9E     01A7   352             MOVAB   SYM_T_SYMBOL+1(R1)[R0],R0           ; GET ADDRESS OF SYMBOL VALUE
      80    B4     01AC   353             CLRW    (R0)+                               ; CLEAR LENGTH OF SYMBOL VALUE
   51    86    9A     01AE   354             MOVZBL  (R6)+,R1                            ; GET OFFSET TO ADDRESS OF SYMBOL VA
```

```
                51   5B   C0  01B1    355          ADDL    R11,R1                          ; CALCULATE ADDRESS TO STORE VALUE A
                61   50   D0  01B4    356          MOVL    R0,(R1)                         ; SET ADDRESS OF SYMBOL VALUE
                     D3   11  01B7    357          BRB     40$                             ;
                   007A   31  01B9    358 48$:     BRW     58$                             ; SKIP TO PARAMETER PROCESSING
                   00DE   31  01BC    359 49$:     BRW     70$                             ; SKIP TO END OF BATCH PROCESSING
                              01BF    360
                              01BF    361 ;
                              01BF    362 ; INITIALIZE $RESTART AND BATCH$RESTART SYMBOLS.
                              01BF    363 ;
              56   58 AB    D0  01BF    364 50$:     MOVL    PRC_L_RESTART(R11),R6           ; GET ADDRESS OF RESTART VALUE
            51   FE4C CF    9E  01C3    365          MOVAB   FALSE,R1                        ; GET ADDRESS OF ASCIC FALSE
                     04    E1  01C8    366          BBC     #PPD$V_RESTART,-                ; BRANCH IF SHOULD BE FALSE
                05 02 AA         01CA    367                  PPD$W_FLAGS(R10),55$            ;
            51   FE3D CF    9E  01CD    368          MOVAB   TRUE,R1                         ; GET ADDRESS OF ASCIC TRUE
                 50   81    9A  01D2    369 55$:     MOVZBL  (R1)+,R0                        ; GET LENGTH OF VALUE STRING
              FE A6   50    B0  01D5    370          MOVW    R0,-2(R6)                       ; STORE THE LENGTH
            66   61   50    28  01D9    371          MOVC3   R0,(R1),(R6)                    ; SET VALUE
                              01DD    372
           DA 68 AB    06    E1  01DD    373          BBC     #PRC_V_MODE,PRC_W_FLAGS(R11),49$; BRANCH IF NOT BATCH JOB
                     04    E1  01E2    374          BBC     #PPD$V_RESTART,-                ; BRANCH IF NOT RESTARTED
                4F 02 AA         01E4    375                  PPD$W_FLAGS(R10),58$            ;
              55   28 AB    9E  01E7    376          MOVAB   PRC_Q_GLOBAL(R11),R5            ; SET ADDRESS OF SYMBOL TABLE LISTHE
            5E   FF01 CE    9E  01EB    377          MOVAB   -LNM$C_NAMLENGTH(SP),SP         ; ALLOCATE BUFFER ON STACK
                 5E   DD    PUSHL   SP                                                      ; CONSTRUCT DESCRIPTOR OF BUFFER
             000000FF 8F    DD  01F2    379          PUSHL   #LNM$C_NAMLENGTH                ;
            50   FE1D CF    9E  01F8    380          MOVAB   BATCH$RESTART,R0                ; GET ADDRESS OF LOGICAL NAME
                   01 A0    9F  01FD    381          PUSHAB  1(R0)                           ; CONSTRUCT DESCRIPTOR OF LOGICAL NA
                 7E   60    9A  0200    382          MOVZBL  (R0),-(SP)                      ;
                 56   5E    D0  0203    383          MOVL    SP,R6                           ; GET ADDRESS OF DESCRIPTOR
                              0206    384          $TRNLOG_S LOGNAM=(R6),-                 ; TRANSLATE LOGICAL NAME
                                      385                    RSLBUF=8(R6),-                ; INTO BUFFER ON STACK
                                      386                    RSLLEN=8(R6),-                ;
                                      387                    DSBMSK=#3                     ; DON'T LOOK IN GROUP OR SYSTEM TABL
  00000000'8F    50    D1  021B    388          CMPL    R0,#SS$_NORMAL                  ; SUCCESS?
                 0D    12  0222    389          BNEQ    57$                             ; IF NOT, CLEAN UP
               51   08 A6   7D  0224    390          MOVQ    8(R6),R1                        ; GET DESCRIPTOR OF SYMBOL VALUE
                 53   66    7D  0228    391          MOVQ    (R6),R3                         ; GET DESCRIPTOR OF SYMBOL NAME
                 50   00    9A  022B    392          MOVZBL  #SYM_K_STRING,R0                ; DEFINE SYMBOL AS A STRING
                    FDCF'    30  022E    393          BSBW    DCL$ALLOCSYM                    ; DEFINE SYMBOL IN SYMBOL TABLE
            5E   010F CE    9E  0231    394 57$:     MOVAB   8+8+LNM$C_NAMLENGTH(SP),SP      ; DEALLOCATE SCRATCH STORAGE
                              0236    395
                              0236    396 ;
                              0236    397 ; CREATE LOCAL SYMBOLS P1 THRU P8 AS THE JOB PARAMETERS
                              0236    398 ;
           62 68 AB    06    E1  0236    399 58$:     BBC     #PRC_V_MODE,PRC_W_FLAGS(R11),70$; BRANCH IF NOT BATCH JOB
              55   38 AB    9E  023B    400          MOVAB   PRC_Q_LOCAL(R11),R5             ;SET ADDRESS OF SYMBOL TABLE LISTHEA
            5E   FF01 CE    9E  023F    401          MOVAB   -LNM$C_NAMLENGTH(SP),SP         ;ALLOCATE BUFFER ON STACK
                 5E   DD    PUSHL   SP                                                      ;CONSTRUCT DESCRIPTOR OF BUFFER
             000000FF 8F    DD  0246    403          PUSHL   #LNM$C_NAMLENGTH                ;
             00003050 8F    DD  024C    404          PUSHL   #^A'P0'                         ;PUSH PROTOTYPE LOGICAL/SYMBOL NAME
                 5E   DD    PUSHL   SP                                                      ;AND CONSTRUCT DESCRIPTOR OF IT
                 02   DD    PUSHL   #2                                                      ;
                 56   5E    D0  0256    407          MOVL    SP,R6                           ;GET ADDRESS OF DESCRIPTOR
                 57   08    D0  0259    408          MOVL    #8,R7                           ;LOOP 8 TIMES
        0C A6    000000FF 8F    D0  025C    409 60$:     MOVL    #LNM$C_NAMLENGTH,12(R6)         ;RESET LENGTH OF BUFFER
                   09 A6   96  0264    410          INCB    9(R6)                           ;INCREMENT SYMBOL NAME
                           0267    411          $TRNLOG_S LOGNAM=(R6),-                 ;TRANSLATE LOGICAL NAME P#
```

```
                                0267   412                      RSLBUF=12(R6),-                 ;INTO BUFFER ON STACK
                                0267   413                      RSLLEN=12(R6),-                 ;
                                0267   414                      DSBMSK=#3                       ;DON'T LOOK IN GROUP OR SYSTEM TABLE
00000000'8F    50    D1  0279   415              CMPL    R0,#SS$_NORMAL                  ;SUCCESS?
                 03    13  027D   416              BEQL    65$                             ;IF NOT,
               0C A6   D4  027F   417              CLRL    12(R6)                          ;SET THE SYMBOL TO NULL STRING
          51   0C A6   7D  0282   418 65$:         MOVQ    12(R6),R1                       ;GET DESCRIPTOR OF SYMBOL VALUE
             53   66   7D  0286   419              MOVQ    (R6),R3                         ;GET DESCRIPTOR OF SYMBOL NAME
             50   00   9A  0289   420              MOVZBL  #SYM_K_STRING,R0                ;DEFINE SYMBOL AS A STRING
                FD6B'  30  028C   421              BSBW    DCL$ALLOCSYM                    ;DEFINE SYMBOL IN SYMBOL TABLE
              C4 57    F5  028F   422              SOBGTR  R7,60$                          ;LOOP UNTIL ALL SYMBOLS DONE
          5E  0113 CE   9E  0292   423              MOVAB   8+4+8+LNM$C_NAMLENGTH(SP),SP    ;DEALLOCATE SCRATCH STORAGE
```

```
                        029D    425 ;
                        029D    426 ; INITIALIZE PROCESS RMS DATA AREA
                        029D    427 ;
                        029D    428 ; INIT AND CONNECT FAB, NAM, INPRAB, OUTRAB.
                        029D    429 ;
   58   0534 CB   9E    029D    430 70$:    MOVAB   PRC_C_LENGTH'R11),R8                    ;SET ADDRESS OF RMS STRUCTUR
     1C AB   68   9E    02A2    431         MOVAB   PRD_G_FAB(R8),PRC_L_INDFAB(R11)         ;ADDRESS OF GENERAL PURPOSE
                        02A6    432         ASSUME  PRD_G_FAB EQ 0
                        02A6    433         ASSUME  FAB$B_BID EQ 0
                        02A6    434         ASSUME  FAB$B_BLN EQ 1
   68   5003 8F   B0    02A6    435         MOVW    #FAB$C_BID+<FAB$C_BLN@8>,PRD_G_FAB(R8)  ;SET FAB ID/LENGTH
   28 A8   50   A8   9E    02AB    436         MOVAB   PRD_G_NAM(R8),FAB$L_NAM(R8)             ;SET ADDRESS OF NAM BLOCK
50 A8   0000'8F   B0    02B0    437         MOVW    #NAM$C_BID+<NAM$C_BLN@8>,PRD_G_NAM(R8)  ;SET NAM ID/LENGTH
   59   00B0 C8   9E    02B6    438         MOVAB   PRD_G_INPRAB(R8),R9                     ;SET ADDRESS OF INPUT RAB
   57   017C C8   9E    02BB    439         MOVAB   PRD_G_OUTRAB(R8),R7                     ;SET ADDRESS OF OUTPUT RAB
   69   4401 8F   B0    02C0    440         MOVW    #RAB$C_BID+<RAB$C_BLN@8>,RAB$B_BID(R9)  ;SET RAB ID/LENGTH
        67   69   B0    02C5    441         MOVW    RAB$B_BID(R9),RAB$B_BID(R7)             ;
     3C A9   58   D0    02C8    442         MOVL    R8,RAB$L_FAB(R9)                        ;SET ADDRESS OF FAB
     3C A7   58   D0    02CC    443         MOVL    R8,RAB$L_FAB(R7)                        ;
                        02D0    444
                        02D0    445 ;
                        02D0    446 ; SET ISI'S OF INITIAL INPUT/OUTPUT FILES.
                        02D0    447 ;
 02 A9   22 AA   B0     02D0    448         MOVW    PPD$W_INPISI(R10),RAB$W_ISI(R9)         ;SET INPUT ISI
 02 A7   26 AA   B0     02D5    449         MOVW    PPD$W_OUTISI(R10),RAB$W_ISI(R7)         ;SET OUTPUT ISI
                        02DA    450
                        02DA    451 ;
                        02DA    452 ; SET PPF DIRECT ACCESS, SO THAT RMS USER-MODE EOF CHECKING IS NOT DONE
                        02DA    453 ;
                        02DA    454         CLRBIT  RAB$V_PPF_IND,RAB$W_ISI(R9)             ;ENABLE DIRECT ACCESS
                        02DF    455         CLRBIT  RAB$V_PPF_IND,RAB$W_ISI(R7)             ;TO INPUT STREAMS
                        02E4    456
                        02E4    457 ;
                        02E4    458 ; STORE DEVICE CHARACTERISTICS IN THE RAB$L_CTX FIELD SO THAT RAB IS ENOUGH
                        02E4    459 ;
18 A9   44 AA   D0      02E4    460         MOVL    PPD$L_INPDEV(R10),RAB$L_CTX(R9)         ;INPUT DEVICE CHARACTERISTIC
18 A7   64 AA   D0      02E9    461         MOVL    PPD$L_OUTDEV(R10),RAB$L_CTX(R7)         ;OUTPUT DEVICE CHARACTERISTI
                        02EE    462
                        02EE    463 ;
                        02EE    464 ; INIT ALTINPRAB, ALTOUTRAB.
                        02EE    465 ;
                        02EE    466         ASSUME  RAB$W_ISI EQ RAB$B_BLN+1
 00F4 C8   69   D0      02EE    467         MOVL    RAB$B_BID(R9),PRD_G_ALTINPRAB(R8)       ;SET RAB ID/LENGTH/ISI
 0138 C8   67   D0      02F3    468         MOVL    RAB$B_BID(R7),PRD_G_ALTOUTRAB(R8)       ;SET RAB ID/LENGTH/ISI
                        02F8    469
                        02F8    470 ;
                        02F8    471 ; LIMIT ALLOCATION OF BLOCKS/BUFFERS ON PPF STREAM (PIOSEG SPACE IS LIMITED)
                        02F8    472 ; IN ALL RABS.
                        02F8    473 ;
   37 A9   01   90      02F8    474         MOVB    #1,RAB$B_MBC(R9)                        ;ONLY ALLOCATE 1 BLOCK/BUFFE
36 A9   FF 8F   90      02FC    475         MOVB    #-1,RAB$B_MBF(R9)                       ;ONLY ALLOCATE 1 BUFFER/STRE
                        0301    476         ASSUME  RAB$B_MBC EQ RAB$B_MBF+1
36 A7   36 A9   B0      0301    477         MOVW    RAB$B_MBF(R9),RAB$B_MBF(R7)             ; SET MBC/MBF FOR ALL RABS
     36 A9   B0         0306    478         MOVW    RAB$B_MBF(R9),-                         ;
   012A C8              0309    479                 PRD_G_ALTINPRAB+RAB$B_MBF(R8)           ;
     36 A9   B0         030C    480         MOVW    RAB$B_MBF(R9),-                         ;
   016E C8              030F    481                 PRD_G_ALTOUTRAB+RAB$B_MBF(R8)           ;
```

```
                                0312    482
                                0312    483 ;
                                0312    484 ; INIT XABTRM.
                                0312    485 ;
        241F  8F    B0          0312    486         MOVW    #XAB$C_TRM+<XAB$C_TRMLEN@8>,-           ;SET XABTRM ID/LENGTH
        01C0  C8                0316    487                 PRD_G_XABTRM(R8)                        ;
    50  01C0  C8    9E          0319    488         MOVAB   PRD_G_XABTRM(R8),R0                     ;GET ADDRESS OF XABTRM BLOCK
08  A0  01E4  C8    9E          031E    489         MOVAB   PRD_G_TRMLIST(R8),XAB$L_ITMLST(R0)      ;SET ADDRESS OF ITEM LIST
    0C  A0    18    B0          0324    490         MOVW    #ITRM_K_MINLEN,XAB$W_ITMLST_LEN(R0)     ;SET MIN SIZE OF ITEM LIST
                                0328    491
                                0328    492 ;
                                0328    493 ; CONNECT XABTRM TO THE INPUT RABS.
                                0328    494 ;
    00F0  C8    50    D0        0328    495         MOVL    R0,PRD_G_INPRAB+RAB$L_XAB(R8)           ;SET ADDRESS OF XABTRM BLOCK
    0134  C8    50    D0        032D    496         MOVL    R0,PRD_G_ALTINPRAB+RAB$L_XAB(R8)        ;
                                0332    497         SETBIT  RAB$V_ETO,PRD_G_ALTINPRAB+RAB$L_ROP(R8) ;SET READ WITH XABTRM
                                0338    498         SETBIT  RAB$V_ETO,PRD_G_INPRAB+RAB$L_ROP(R8)    ;
                                033E    499
                                033E    500 ;
                                033E    501 ; INIT XABTRM ITEM LIST.
                                033E    502 ;
    50  01E4  C8    9E          033E    503         MOVAB   PRD_G_TRMLIST(R8),R0                    ;GET ADDRESS OF ITEM LIST
      10  AB    50    D0        0343    504         MOVL    R0,PRC_L_TRMLIST(R11)                   ;SAVE ADDRESS IN PRC
                60    B4        0347    505         CLRW    ITRM_W_MODLEN(R0)                       ;SET READ MODIFIERS
      02  A0    00    B0        0349    506         MOVW    #TRM$_MODIFIERS,ITRM_W_MODCODE(R0)      ;
   00014000  8F    D0          034D    507         MOVL    #TRM$M_TM_ESCAPE!TRM$M_TM_NORECALL,-    ;
          04  A0                0353    508                 ITRM_L_MODIFIERS(R0)                    ;
      0E  A0    04    B0        0355    509         MOVW    #TRM$_PROMPT,ITRM_W_PMPTCODE(R0)        ;SET READ WITH PROMPT
      1A  A0    05    B0        0359    510         MOVW    #TRM$_INISTRNG,ITRM_W_INICODE(R0)       ;SET USE INITIAL STRING
      26  A0    08    B0        035D    511         MOVW    #TRM$_INIOFFSET,ITRM_W_OFFCODE(R0)      ;SET USE INITIAL STRING OFFS
                                0361    512
                                0361    513 ;
                                0361    514 ; USE SAME RAB FOR BOTH INPUT AND OUTPUT IF THEY BOTH POINT TO THE SAME STREAM
                                0361    515 ; (THIS IS FOR CODE SEGMENTS WHICH DECIDE IF THEY ARE THE SAME BY COMPARING
                                0361    516 ; THE INPUT AND OUTPUT RAB ADDRESSES RATHER THAN THEIR ISI'S)
                                0361    517 ;
  02  A9    02    A7    B1      0361    518         CMPW    RAB$W_ISI(R7),RAB$W_ISI(R9)             ;ARE INPUT AND OUTPUT THE SA
              03    12          0366    519         BNEQ    75$                                     ;BRANCH IF NOT
        57    59    D0          0368    520         MOVL    R9,R7                                   ;USE INPUT RAB FOR OUTPUT
                                0368    521
                                0368    522 ;
                                0368    523 ; STORE FAB/RAB ADDRESSES IN PRC AREA FOR EASY ACCESS
                                0368    524 ;
      0C  AB    57    D0        0368    525 75$:    MOVL    R7,PRC_L_OUTRAB(R11)                    ;SET ADDRESS OF OUTPUT RAB
      08  AB    59    D0        036F    526         MOVL    R9,PRC_L_INPRAB(R11)                    ;SET ADDRESS OF INPUT RAB
      18  AB    57    D0        0373    527         MOVL    R7,PRC_L_INDOUTRAB(R11)                 ;SET ADDRESS OF INDIRECT OUT
      14  AB    59    D0        0377    528         MOVL    R9,PRC_L_INDINPRAB(R11)                 ;SET ADDRESS OF INDIRECT INP
                                037B    529
                                037B    530 ;
                                037B    531 ; IF IN BATCH JOB, SET THE DEFAULT FLUSH RATE.
                                037B    532 ;
  0A  68  AB    06    E1        037B    533         BBC     #PRC_V_MODE,PRC_W_FLAGS(R11),76$        ;BRANCH IF NOT BATCH JOB
00D0  CB    FCAB  CF    7D      0380    534         MOVQ    FLUSH_RATE,PRC_Q_FLUSHTIME(R11)         ;SET DEFAULT FLUSH RATE
              FC76' 30          0387    535         BSBW    DCL$SET_TIMER                           ;SET FLUSH TIMER
```

```
                                038A    537 ;
                                038A    538 ; INITIALIZE INDIRECT FRAME STACK & LEVEL 0 FRAME.  IF WE HAVE ENOUGH SPACE,
                                038A    539 ; ALLOCATE ROOM FOR EXACTLY 16 FRAMES.  IF NOT, GET AS MANY FRAMES AS POSSIBLE.
                                038A    540 ;
                    0778 CB     9E  038A    541 76$:    MOVAB   PRC_C_LENGTH+PRD_C_XLENGTH(R11),-       ;SET LIMIT OF INDIRECT FRAME
                    00A4 CB         038E    542                 PRC_L_STACKLM(R11)                      ;TO WHAT IS AVAILABLE
                      04 AA     C1  0391    543         ADDL3   PPD$Q_CLIREG(R10),-                     ;GET END+1 OF REGION
                50    08 AA         0394    544                 PPD$Q_CLIREG+4(R10),R0
                                0397    545 ;*****
                                0397    546 ;***** DO NOT USE LAST PAGE - TO CATCH BUG IN DCL WHICH REFERENCES OFF
                                0397    547 ;***** THE END OF THE INDIRECT STACK BEYOND OUR STORAGE AREA.
                                0397    548 ;*****
              50    FE00 C0     9E  0397    549         MOVAB   -512(R0),R0
                59    8C A0     9E  039C    550         MOVAB   -IDF_K_LENGTH(R0),R9                    ;ADDRESS OF LEVEL 0 FRAME
              50    F8C0 C9     9E  03A0    551         MOVAB   -MAX_DEPTH*IDF_K_LENGTH(R9),R0          ;COMPUTE BASE OF STACK W/16
              00A4 CB     50    D1  03A5    552         CMPL    R0,PRC_L_STACKLM(R11)                   ;ROOM FOR EXACTLY 16 FRAMES?
                          05    1B  03AA    553         BLEQU   77$                                     ;IF NOT, USE WHATEVER ROOM W
              00A4 CB     50    D0  03AC    554         MOVL    R0,PRC_L_STACKLM(R11)                   ;ELSE, LIMIT TO EXACTLY 16 F
              00A0 CB     59    D0  03B1    555 77$:    MOVL    R9,PRC_L_STACKPT(R11)                   ;SET INDIRECT FRAME STACK PO
              00BC CB     59    D0  03B6    556         MOVL    R9,PRC_L_IDFLNK(R11)                    ;SET LINK POINTER
   69 0074 8F     00    6E    00  2C  03BB    557         MOVC5   #0,(SP),#0,#IDF_L_LENGTH,(R9)           ;ZERO LEVEL 0 FRAME
                                03C3    558                                                             ;(IDF_L_LNK=0 TO TERMINATE L
                                03C3    559                                                             ;(IDF_L_SEARCHCTX=0 TO INITI
                68 A9     68 AA     9E  03C3    560         MOVAB   PPD$T_FILENAME(R10),IDF_L_FILENAME(R9)  ;GET ADDRESS OF INPUT FILE N
                20 A9     24 AA     B0  03C8    561         MOVW    PPD$W_OUTIFI(R10),IDF_W_OUTIFI(R9)      ;SAVE OUTPUT IFI
              0114 CB     24 AA     B0  03CD    562         MOVW    PPD$W_OUTIFI(R10),PRC_W_OUTIFI(R11)     ;SAVE OUTPUT IFI
                04 A9     20 AA     B0  03D3    563         MOVW    PPD$W_INPIFI(R10),IDF_W_INPIFI(R9)      ;SAVE INPUT IFI
                22 A9     02 A7     B0  03D8    564         MOVW    RAB$W_ISI(R7),IDF_W_OUTISI(R9)          ;SAVE OUTPUT ISI
              0116 CB     02 A7     B0  03DD    565         MOVW    RAB$W_ISI(R7),PRC_W_OUTISI(R11)         ;SAVE OUTPUT ISI
                24 A9     64 AA     D0  03E3    566         MOVL    PPD$L_OUTDEV(R10),IDF_L_OUTRABCTX(R9)   ;SAVE OUTPUT DEVCHAR
              0118 CB     64 AA     D0  03E8    567         MOVL    PPD$L_OUTDEV(R10),PRC_L_OUTRABCTX(R11)  ;SAVE OUTPUT DEVCHAR
                0C A9     44 AA     D0  03EE    568         MOVL    PPD$L_INPDEV(R10),IDF_L_INPRABCTX(R9)   ;SAVE INPUT DEVCHAR
                   28 AA    1C     28  03F3    569         MOVC    #PPD$C_DVIFID,PPD$T_INPDVI(R10),-       ;COPY INPUT DEVICE NAME/IDS
                         3C A9         03F7    570                 IDF_T_INPDVI(R9)
          28 A9     48 AA    10     28  03F9    571         MOVC    #16,PPD$T_OUTDVI(R10),IDF_T_OUTDVI(R9)  ;COPY OUTPUT DEVICE NAME
        011C CB     48 AA    10     28  03FF    572         MOVC    #16,PPD$T_OUTDVI(R10),PRC_T_OUTDVI(R11) ;COPY OUTPUT DEVICE NAME
                         38 A9     94  0406    573         CLRB    IDF_B_OUTFLAGS(R9)                      ;CLEAR CCL BITS
                04 02 AA    05     E1  0409    574         BBC     #PPD$V_INPCCL,PPD$W_FLAGS(R10),771$     ;SKIP IF INPUT NOT CCL
                                040E    575         SETBIT  IDF_V_INPCCL,IDF_B_OUTFLAGS(R9)         ;SET INPCCL BIT
                04 02 AA    06     E1  0412    576 771$:   BBC     #PPD$V_OUTCCL,PPD$W_FLAGS(R10),772$     ;SKIP IF OUTPUT NOT CCL
                                0417    577         SETBIT  IDF_V_OUTCCL,IDF_B_OUTFLAGS(R9)         ;SET OUTCCL BIT
              012C CB     38 A9     90  041B    578 772$:   MOVB    IDF_B_OUTFLAGS(R9),PRC_B_OUTFLAGS(R11)  ;COPY CCL BITS
                                0421    579
                                0421    580 ;
                                0421    581 ; CREATE SYS$OUTPUT LOGICAL NAME AND ENABLE IMAGE VERIFICATION IF IN BATCH JOB.
                                0421    582 ;
                   58    59     D0  0421    583         MOVL    R9,R8                                   ;SET ADDRESS OF IDF BLOCK
                      FBD9'     30  0424    584         BSBW    DCL$CREATE_OUTPUT                       ;CREATE SYS$OUTPUT LOGICAL N
                0F 68 AB    06     E1  0427    585         BBC     #PRC_V_MODE,PRC_W_FLAGS(R11),78$        ;BRANCH IF NOT BATCH JOB
                   51    1C AB     D0  042C    586         MOVL    PRC_L_INDFAB(R11),R1                    ;GET ADDRESS OF FAB
                02 A1    04 A8     B0  0430    587         MOVW    IDF_W_INPIFI(R8),FAB$W_IFI(R1)          ;SET INPUT IFI
                  00000000'EF     16  0435    588         JSB     DCL$VERIFY_IMAGE                        ;SET IMAGE VERIFICATION
                                043B    589
                                043B    590 ;
                                043B    591 ; STACK INITIALIZATION PROCEDURES PROC1 THRU PROC(PPD$B_NPROCS)
                                043B    592 ;
                5E    FF01 CE     9E  043B    593 78$:    MOVAB   -LNM$C_NAMLENGTH(SP),SP                 ;ALLOCATE BUFFER ON STACK
```

```
                      5E   DD  0440    594          PUSHL   SP                                  ;CONSTRUCT DESCRIPTOR OF BUFFER
                   7E   FF 8F   9A  0442    595          MOVZBL  #LNM$C_NAMLENGTH,-(SP)              ;
      7E    00000030 434F5250 8F   7D  0446    596          MOVQ    #^A'PROCO',-(SP)                    ;PUSH PROTOTYPE PROCEDURE NAME
                      5E   DD  0451    597          PUSHL   SP                                  ;AND CONSTRUCT DESCRIPTOR OF IT
                      05   DD  0453    598          PUSHL   #5                                  ;
                   56    5E   D0  0455    599          MOVL    SP,R6                               ;GET ADDRESS OF DESCRIPTOR
                57    1C AA   9A  0458    600          MOVZBL  PPD$B_NPROCS(R10),R7                ;GET NUMBER OF INITIAL PROCEDURES
                0C A6    57   80  045C    601          ADDB    R7,12(R6)                           ;SET TO LAST PROCEDURE NAME
             10 A6    FF 8F   9A  0460    602 80$:     MOVZBL  #LNM$C_NAMLENGTH,16(R6)             ;RESET LENGTH OF BUFFER
                              0465    603          $TRNLOG_S LOGNAM=(R6),-                     ;TRANSLATE LOGICAL NAME PROC#
                              0465    604                    RSLBUF=16(R6),-                   ;INTO BUFFER ON STACK
                              0465    605                    RSLLEN=16(R6),-
                              0465    606                    DSBMSK=#3                         ;DON'T LOOK IN GROUP OR SYSTEM TABLE
          0000'8F    50   B1  047A    607          CMPW    R0,#SS$_NOTRAN                      ;NO TRANSLATION?
                     2B   13  047F    608          BEQL    86$                                 ;IF SO, SKIP THIS ONE
             52    10 A6   7D  0481    609          MOVQ    16(R6),R2                           ;GET DESCRIPTOR OF PROCEDURE NAME
                      54   D4  0485    610          CLRL    R4                                  ;SPECIFY PRIMARY OUTPUT FILE
                51    01   D0  0487    611          MOVL    #1,R1                               ;SUPPRESS RMS ERROR MESSAGES
                   FB73'   30  048A    612          BSBW    DCL$PUSHPROC                        ;PUSH PROCEDURE ONTO INDIRECT STACK
                   0F 50   E8  048D    613          BLBS    R0,85$                              ;BRANCH IF SUCCESS
                      03   E1  0490    614          BBC     #PPD$V_CAPTIVE,-                    ;CAPTIVE ACCOUNT?
                17 02 AA        0492    615                  PPD$W_FLAGS(R10),86$               ;NO, THEN SKIP
          50    0003890A 8F   D0  0495    616          MOVL    #CLI$_NOCMDPROC,R0                  ;SET ERROR STATUS
                   011A   31   049C    617          BRW     INITIAL_ERROR                       ;TERMINATE THE PROCESS
                01FE 8F   BB  049F    618 85$:     PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8>        ;SAVE REGISTERS
                      56   D4  04A3    619          CLRL    R6                                  ;SET NO INITIAL VALUES
                   FB58'   30  04A5    620          BSBW    DCL$DEFINE_P1_TO_P8                 ;DEFINE P1 THROUGH P8
                01FE 8F   BA  04A8    621          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8>        ;RESTORE REGISTERS
                   0C A6   97  04AC    622 86$:     DECB    12(R6)                              ;DECREMENT PROCEDURE NUMBER
                   AE 57   F5  04AF    623          SOBGTR  R7,80$                              ;LOOP UNTIL ALL SYMBOLS DONE
             5E    0117 CE   9E  04B2    624          MOVAB   8+8+8+LNM$C_NAMLENGTH(SP),SP        ;DEALLOCATE SCRATCH S"ORAGE
                              04B7    625
```

```
                        04B7   627 ;
                        04B7   628 ; RUNDOWN LOGINOUT IMAGE, THUS DELETING ALL USER-MODE LOGICAL NAMES
                        04B7   629 ;
FB46'  30               04B7   630         BSBW    DCL$RMSRUNDWN                   ;RUNDOWN RMS FILES
                        04BA   631         $RUNDWN_S #PSL$C_USER                   ;RUNDOWN LOGINOUT IMAGE
                        04C3   632
```

```
                                                     N 1
                        04C3    634 ;
                        04C3    635 ; IF THIS PROCESS IS A SUBPROCESS WHICH WAS CREATED BY THE DCL SPAWN
                        04C3    636 ; COMMAND, THEN OPEN THE MAILBOX CONTAINING THE PROCESS CONTEXT AND
                        04C3    637 ; INITIALIZE THIS PROCESS.
                        04C3    638 ;
          7E    7C      04C3    639         CLRQ    -(SP)                           ;CREATE GETJPI ITEM LIST
        F8 AE   9F      04C5    640         PUSHAB  -2*4(SP)                        ;SET BUFFER ADDRESS
00000004'8F     DD      04C8    641         PUSHL   #JPI$_OWNER@16+4                ;REQUEST PARENT PID, SET BUFFER LENGTH
          7E    7C      04CE    642         CLRQ    -(SP)                           ;ALLOCATE AN IOSB
     50   5E    D0      04D0    643         MOVL    SP,R0                           ;
                        04D3    644         $GETJPIW_S  ITMLST=8(R0),-              ;GET PID OF PARENT PROCESS
                        04D3    645                 EFN=#EXE$C_SYSEFN,-             ;
                        04D3    646                 IOSB=(R0)                       ;
     5E   08    C0      04EB    647         ADDL    #2*4,SP                         ;POP IOSB
          59 8ED0       04EE    648         POPL    R9                              ;GET PID
     5E   0C    C0      04F1    649         ADDL    #3*4,SP                         ;CLEANUP STACK
          59    D5      04F4    650         TSTL    R9                              ;IS THIS A SUBPROCESS?
          59    13      04F6    651         BEQL    120$                            ;IF NOT, SKIP SPAWN INITIALIZATION
5E   FF01 CE    9E      04F8    652         MOVAB   -LNM$C_NAMLENGTH(SP),SP         ;ALLOCATE BUFFER ON STACK
          5E    DD      04FD    653         PUSHL   SP                              ;CONSTRUCT DESCRIPTOR OF BUFFER
  7E    FF 8F   9A      04FF    654         MOVZBL  #LNM$C_NAMLENGTH,-(SP)          ;
     56   5E    D0      0503    655         MOVL    SP,R6                           ;POINT TO BUFFER DESCRIPTOR
     5E   04    C2      0506    656         SUBL    #4,SP                           ;ALLOCATE SCRATCH LONGWORD FOR ACMODE
52   FAF7 CF    9E      0509    657         MOVAB   SYS$ERROR,R2                    ;GET ADDRESS OF ASCIC STRING
     51   82    9A      050E    658         MOVZBL  (R2)+,R1                        ;CONSTRUCT DESCRIPTOR OF STRING
     7E   51    7D      0511    659         MOVQ    R1,-(SP)                        ;PUSH DESCRIPTOR OF SYS$ERROR
                        0514    660         $TRNLOG_S LOGNAM=-12(R6),-              ;TRANSLATE SYS$ERROR
                        0514    661                 RSLBUF=(R6),-
                        0514    662                 RSLLEN=(R6),-
                        0514    663                 DSBMSK=#3,-                     ;ONLY SEARCH PROCESS LOGNAME TABLE
                        0514    664                 ACMODE=-4(R6)
        02   FC A6      91      0529    665         CMPB    -4(R6),#PSL$C_SUPER     ;SUPERVISOR MODE SYS$ERROR?
               1D       12      052D    666         BNEQ    100$                    ;IF NOT, THEN NO CONTEXT WAS PASSED
41424D5F 8F    04 B6    D1      052F    667         CMPL    @4(R6),#^A'_MBA'        ;IS MAILBOX NAME A PHYSICAL DEVICE?
               03       12      0537    668         BNEQ    90$                     ;IF NOT, INVALID SPAWN - DELETE LOGNAM
             008F       30      0539    669         BSBW    SPAWN_CONTEXT           ;INITIALIZE BASED ON SPAWN CONTEXT
          50   5E    D0      053C    670 90$:    MOVL    SP,R0                       ;GET ADDRESS OF 'SYS$ERROR' DESCRIPTOR
                        053F    671         $DELLOG_S LOGNAM=(R0),-                 ;DELETE SUPERVISOR MODE SYS$ERROR
                        053F    672                 TBLFLG=#LOG$C_PROCESS,-         ;LEAVING EXECUTIVE MODE SYS$ERROR
                        053F    673                 ACMODE=#PSL$C_SUPER             ;(WHICH IS EQUIVALENT TO SYS$OUTPUT)
        5E   0113 CE    9E      054C    674 100$:   MOVAB   8+4+8+LNM$C_NAMLENGTH(SP),SP    ;DEALLOCATE SCRATCH STORAGE
                        0551    675
```

```
                                0551   677 ;
                                0551   678 ; MAKE SURE COMMAND TABLES HAVE A VALID STRUCTURE LEVEL NUMBER
                                0551   679 ;
      00000000'GF   DD          0551   680 120$:   PUSHL   G^CTL$AG_CLITABLE        ; GET ADDRESS OF COMMAND TABLES
00000000'GF    01   FB          0557   681         CALLS   #1,G^CDU$UPGRADE_TABLE  ; VALIDATE TABLE STRUCTURE
          58 50     E9          055E   682         BLBC    R0,INITIAL_ERROR        ; SIGNAL ERROR STATUS AND ABORT
                                0561   683 ;
                                0561   684 ;
                                0561   685 ; TELL THE TERMINAL DRIVER THAT WE NOW OWN THE TERMINAL.
                                0561   686 ;
   2F 68 AB   0F   E5           0561   687         BBCC    #PRC_V_DETACHED,PRC_W_FLAGS(R11),125$ ; BRANCH IF NOWAIT SUBPROCESS
   2A 68 AB   06   E0           0566   688         BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),125$ ; BRANCH IF NOT INTERACTIVE
              7E   7C           056B   689         CLRQ    -(SP)                   ; ALLOCATE AN IOSB
         50   5E   D0           056D   690         MOVL    SP,R0                   ; GET ADDRESS OF IOSB
                                0570   691         $QIOW_S FUNC=#IO$_SETMODE!IO$M_TT_PROCESS,- ; ASSUME TERMINAL OWNERSHIP
                                0570   692                 CHAN=PRC_W_INPCHAN(R11),-  ;
                                0570   693                 IOSB=(R0),-             ;
                                0570   694                 EFN=#EXE$C_SYSEFN       ;
         5E   08   C0           0592   695         ADDL    #8,SP                   ; RESTORE THE STACK
                                0595   696 ;
                                0595   697 ;
                                0595   698 ; ENABLE CONTROL/Y AST ROUTINE AND OUT-OF-BAND AST ROUTINES.
                                0595   699 ;
            FA68'   30          0595   700 125$:   BSBW    DCL$ENBCONTRLY          ; ENABLE CONTROL Y AST'S
   51    00B4 CB   D0           0598   701         MOVL    PRC_L_OUTOFBAND(R11),R1 ; GET AST CHARACTER MASK
         00B4 CB   D4           059D   702         CLRL    PRC_L_OUTOFBAND(R11)    ; FORCE INITIALIZATION
            FA5C'   30          05A1   703         BSBW    DCL$RESETOOB            ; ENABLE/DISABLE APPROPRIATE AST'S
                                05A4   704 ;
                                05A4   705 ;
                                05A4   706 ; ENABLE CHANGE MODE TO SUPERVISOR HANDLING ROUTINE
                                05A4   707 ;
                                05A4   708         $DCLCMH_S W^DCL$CHANGE_MODE     ; SET CHANGE MODE TO SUPER HANDLER
          03 50   E9            05B3   709         BLBC    R0,INITIAL_ERROR        ; BRANCH IF ERROR DETECTED
                                05B6   710 ;
                                05B6   711 ;
                                05B6   712 ; PROCESS FIRST COMMAND LINE
                                05B6   713 ;
            FA47'   31          05B6   714         BRW     DCL$RESTART             ; START COMMAND INTERPRETATION
                                05B9   715 ;
                                05B9   716 ;
                                05B9   717 ; ERROR OCCURED SOMEWHERE IN STARTUP PROCEDURE.  SIGNAL IT AND ABORT.
                                05B9   718 ;
                                05B9   719 INITIAL_ERROR:
      5A   5E   D0              05B9   720         MOVL    SP,R10                  ;SET BASE ADDRESS OF WRK
   5E   F486 CE   9E            05BC   721         MOVAB   WRK_C_LENGTH(SP),SP     ;ALLOCATE WRK TO SIGNAL MESSAGE
     F0 AA   02   B0            05C1   722         MOVW    #WRK_M_COMMAND,WRK_W_FLAGS(R10) ;CLEAR FLAGS, SET WRK_V_COMMAND
                                05C5   723                                         ;TO MARK NO ERROR TEXT SEGMENT
            FA38'   30          05C5   724         BSBW    DCL$ERRORMSG            ;ISSUE ERROR MESSAGE
            FA35'   31          05C8   725         BRW     DCL$ABORT               ;AND EXIT PROCESS
```

```
                     05CB   727          .SBTTL  SPAWN_CONTEXT, INITIALIZE BASED ON SPAWN CONTEXT
                     05CB   728 ;---
                     05CB   729 ;
                     05CB   730 ; THIS ROUTINE IS CALLED TO INITIALIZE THE SUBPROCESS FROM THE CONTEXT
                     05CB   731 ; PASSED FROM THE PARENT PROCESS DURING EXECUTION OF A SPAWN COMMAND.
                     05CB   732 ;
                     05CB   733 ; INPUTS:
                     05CB   734 ;
                     05CB   735 ;       R6 = ADDRESS OF DESCRIPTOR OF MAILBOX DEVICE NAME
                     05CB   736 ;            (PASSED TO SUBPROCESS AS SYS$ERROR TRANSLATION)
                     05CB   737 ;       R9 = PID OF PARENT PROCESS
                     05CB   738 ;       R11 = ADDRESS OF PRC AREA
                     05CB   739 ;
                     05CB   740 ; OUTPUTS:
                     05CB   741 ;
                     05CB   742 ;       THE CONTEXT IS INITIALIZED, IF POSSIBLE.
                     05CB   743 ;
                     05CB   744 ;       R0-R8 DESTROYED.
                     05CB   745 ;---
                     05CB   746
                     05CB   747 SPAWN_CONTEXT:
                     05CB   748          ASSUME  DIB$C_LENGTH LE CTX_C_MAXLEN
   5E   FC00 CE   9E 05CB   749          MOVAB   -CTX_C_MAXLEN(SP),SP           ;ALLOCATE DIB/CTX STORAGE
        5E    DD   05D0   750          PUSHL   SP                              ;CREATE DESCRIPTOR OF DIB BUFFER
   7E   0400 8F   3C 05D2   751          MOVZWL  #CTX_C_MAXLEN,-(SP)             ;
        52   7E    DE 05D7   752          MOVAL   -(SP),R2                        ;ALLOCATE CHANNEL LONGWORD
                     05DA   753          $ASSIGN_S DEVNAM=(R6),-                 ;ASSIGN A CHANNEL TO THE MAILBOX
                     05DA   754                    CHAN=(R2)                     ;
          7A 50    E9 05E7   755          BLBC    R0,90$                          ;IF ERROR, SKIP IT
                     05EA   756          $GETCHN_S CHAN=(R2),-                   ;GET DEVICE CHARACTERISTICS
                     05EA   757                    PRIBUF=4(R2)                  ;
          5A 50    E9 05FD   758          BLBC    R0,50$                          ;IF ERROR, SKIP IT
55 0C A2    14    E1 0600   759          BBC     #DEV$V_MBX,DIB$L_DEVCHAR+12(R2),50$ ;IF NOT MAILBOX, SKIP IT
                     0605   760
                     0605   761 ;
                     0605   762 ; READ NEXT RECORD FROM MAILBOX AND PROCESS IT
                     0605   763 ;
        52   5E    D0 0605   764 10$:    MOVL    SP,R2                           ;GET ADDRESS OF CHAN,IOSB,BUFFER
                     0608   765          $QIOW_S FUNC=#IO$_READVBLK,-             ;READ NEXT RECORD
                     0608   766                  CHAN=(R2),-                     ;
                     0608   767                  IOSB=4(R2),-                    ;RE-USE DESCRIPTOR AS IOSB
                     0608   768                  EFN=#EXE$C_SYSEFN,-             ;USE SYSTEM EFN
                     0608   769                  P1=12(R2),-                     ;ADDRESS OF RECEIVE BUFFER
                     0608   770                  P2=#CTX_C_MAXLEN                ;SIZE OF RECEIVE BUFFER
          2A 50    E9 062D   771          BLBC    R0,50$                          ;IF SUBMIT ERROR, SKIP IT
      26 04 A2     E9 0630   772          BLBC    4(R2),50$                       ;IF I/O ERROR (OR EOF), EXIT
   59 08 A2     D1 0634   773          CMPL    8(R2),R9                        ;IS TRANSMITTER OUR PARENT PROCESS?
          20     12 0638   774          BNEQ    50$                             ;IF NOT, SKIP IT
   54 06 A2     3C 063A   775          MOVZWL  6(R2),R4                        ;GET SIZE OF RECORD
   55 0C A2     9E 063E   776          MOVAB   12(R2),R5                       ;POINT TO CTX RECORD
        C0 AF    9F 0642   777          PUSHAB  10$                             ;RETURN TO TOP OF LOOP AFTER PROCESS
                     0645   778
                     0645   779          ;
                     0645   780          ; THESE ROUTINES MAY DESTROY R0-R8
                     0645   781          ; ON ENTRY, R4/R5 = DESCRIPTOR OF CTX RECORD
                     0645   782          ;
                     0645   783          CASE    CTX_W_TYPE(R5),TYPE=W,<-        ;CASE ON TYPE OF RECORD
```

```
                     0645   784                     SPAWN_HEADER,-                  ;HEADER RECORD
                     0645   785                     SPAWN_CMDSTR,-                  ;COMMAND STRING
                     0645   786                     SPAWN_LOGNAM,-                  ;LOGICAL NAME
                     0645   787                     SPAWN_CLISYM,-                  ;CLI SYMBOL
                     0645   788                     SPAWN_KEYSTATE,-                ;KEYPAD STATE
                     0645   789                     SPAWN_LNMTABLE,-                ;LOGICAL NAME TABLE
                     0645   790                     SPAWN_LNMNAME,-                 ;SIMPLE LOGICAL NAME
                     0645   791                     SPAWN_LNMTRAN>                  ;ADDITIONAL TRANSLATIONS
               05    0659   792             RSB                                     ;IF UNKNOWN, SKIP RECORD
                     065A   793
                     065A   794 50$:         $DASSGN_S CHAN=(R2)                     ;DEASSIGN CHANNEL TO MAILBOX
      5E  040C CE    9E     0664   795 90$:  MOVAB   4+8+CTX_C_MAXLEN(SP),SP        ;DEALLOCATE SCRATCH STORAGE
               05    0669   796             RSB
                     066A   797
                     066A   798 ;
                     066A   799 ; PROCESS A SPAWN HEADER CONTEXT RECORD
                     066A   800 ;
                     066A   801 SPAWN_HEADER:
         7E  01  CE  066A   802             MNEGL   #1,-(SP)                        ;CREATE PRIVILEGE MASK WITH ALL
         7E  01  CE  066D   803             MNEGL   #1,-(SP)                        ;THE PRIVILEGE BITS SET
         50  5E  D0  0670   804             MOVL    SP,R0                           ;GET ADDRESS OF IT
                     0673   805             $SETPRV_S PRVADR=(R0),-                 ;DISABLE ALL PRIVILEGES
                     0673   806                       PRMFLG=#1,-                   ;
                     0673   807                       ENBFLG=#0                     ;
         5E  08  C0  0682   808             ADDL    #8,SP                           ;DEALLOCATE PRIVILEGE MASK ON STACK
                     0685   809             $SETPRV_S PRVADR=CTX_Q_PROCPRIV(R5),-    ;ENABLE NEW PRIVILEGES
                     0685   810                       PRMFLG=#1,-                   ;
                     0685   811                       ENBFLG=#1                     ;
05 0E A5   04    E1  0695   812             BBC     #CTX_V_WAIT,CTX_B_FLAGS(R5),5$          ;COPY WAIT FLAG
                     069A   813             SETBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11)
05 0E A5   00    E1  069F   814 5$:         BBC     #CTX_V_AUTOLOGO,CTX_B_FLAGS(R5),10$     ;COPY SILENT LOGOUT FLAG
                     06A4   815             SETBIT  PRC_V_AUTOLOGO,PRC_Q_FLAGS(R11)
05 0E A5   01    E1  06A9   816 10$:        BBC     #CTX_V_MODE,CTX_B_FLAGS(R5),20$         ;IF BATCH, SET EOF SILENT LO
                     06AE   817             SETBIT  PRC_V_EOFLOGO,PRC_W_FLAGS(R11)
                     06B3   818 20$:        CLRBIT  PRC_V_VERIFY,PRC_Q_FLAGS(R11)           ;ASSUME FLAG OFF
05 0E A5   02    E1  06B8   819             BBC     #CTX_V_VERIFY,CTX_B_FLAGS(R5),25$       ;COPY VERIFICATION FLAG
                     06BD   820             SETBIT  PRC_V_VERIFY,PRC_Q_FLAGS(R11)
      56   01    D0  06C2   821 25$:        MOVL    #1,R6                           ;ASSUME FLAG ON
02 0E A5   03    E0  06C5   822             BBS     #CTX_V_VERIMAGE,CTX_B_FLAGS(R5),30$     ;IF SET, FLAG OK AS IS
         56      D4  06CA   823             CLRL    R6                              ;SET FLAG TO TURN OFF VERIIF
  00000000'EF    16  06CC   824 30$:        JSB     DCL$SETVERIFY_IMAGE             ;TURN IMAGE VERIFY ON/OFF
        0A A5    D0  06D2   825             MOVL    CTX_L_OUTOFBAND(R5),-           ;COPY OUT-OF-BAND AST MASK
      00B4 CB        06D5   826                     PRC_L_OUTOFBAND(R11)
   50   0F A5    9A  06D8   827             MOVZBL  CTX_B_PROMPTLEN(R5),R0          ;COPY PROMPT STRING LENGTH
 00F0 CB   50    90  06DC   828             MOVB    R0,PRC_B_PROMPTLEN(R11)         ;
   10 A5   50    28  06E1   829             MOVC    R0,CTX_W_PMPTCTRL(R5),-         ;COPY PROMPT STRING
      00F1 CB        06E5   830                     PRC_W_PMPTCTRL(R11)             ;
 00F1 CB   00    B1  06E8   831             CMPW    #0,PRC_W_PMPTCTRL(R11)          ;CR/LF CURRENTLY IN USE?
           04    12  06ED   832             BNEQ    35$                             ;YES, CARRCNTL FLAG OK AS IS
                     06EF   833             CLRBIT  PRC_V_CARRCNTL,PRC_W_FLAGS(R11) ;NO, INDICATE NO CR/LF IN USE.
               05    06F3   834 35$:        RSB
                     06F4   835
                     06F4   836 ;
                     06F4   837 ; PROCESS COMMAND STRING RECORD
                     06F4   838 ;
                     06F4   839 SPAWN_CMDSTR:
 56  00E0 CB    9E  06F4   840             MOVAB   PRC_Q_COMMAND(R11),R6   ;POINT TO COMMAND DESCRIPTOR AREA
```

```
                57   54    02    A3  06F9    841          SUBW3   #CTX_T_CMDSTR,R4,R7     ;COMPUTE SIZE OF COMMAND STRING
                  02 A6    57    B0  06FD    842          MOVW    R7,2(R6)                ;SET SIZE OF COMMAND STRING
                        FEFC'   30  0701    843          BSBW    DCL$ALLDEACMD           ;ALLOCATE BUFFER TO HOLD COMMAND
                     0F  50     E9  0704    844          BLBC    R0,90$                  ;IF ERROR, SKIP IT
                     66  51     7D  0707    845          MOVQ    R1,(R6)                 ;STORE DESCRIPTOR OF BUFFER
            62   02 A5    57    28  070A    846          MOVC    R7,CTX_T_CMDSTR(R5),(R2) ;STORE COMMAND STRING IN BUFFER
                        63    94  070F    847          CLRB    (R3)                    ;AND ENSURE THAT ITS TERMINATED
                              0711    848          SETBIT  PRC_V_CMD,PRC_B_FLAGS2(R11) ;MARK COMMAND PENDING
                        05    0716    849 90$:     RSB
                              0717    850
                              0717    851 ;
                              0717    852 ; PROCESS SPAWN KEYPAD STATE
                              0717    853 ;
                              0717    854 SPAWN_KEYSTATE:
                51   02 A5    9A  0717    855          MOVZBL  CTX_B_KEYLENGTH(R5),R1          ;GET LENGTH OF STATE
                52   03 A5    9E  071B    856          MOVAB   CTX_T_KEYSTATE(R5),R2           ;GET ADDRESS OF STRING
                      F8DE'   30  071F    857          BSBW    DCL$ALLOC_STATE                 ;ALLOCATE STATE SYMBOL
                     48 AB    D0  0722    858          MOVL    PRC_L_CURRKEY(R11),-            ;LOCK THAT STATE
                     4C AB            0725    859                  PRC_L_LASTKEY(R11)              ;
                        05    0727    860          RSB
                              0728    861
                              0728    862 ;
                              0728    863 ; PROCESS SPAWN LOGICAL NAME RECORDS
                              0728    864 ;
                              0728    865 SPAWN_LOGNAM:
                50   04 A5    9A  0728    866          MOVZBL  CTX_B_ACMODE(R5),R0     ;GET ACMODE OF LOGICAL NAME
                   02   50    91  072C    867          CMPB    R0,#PSL$C_SUPER         ;IS ACMODE HIGHER THAN SUPER?
                        27    19  072F    868          BLSS    90$                     ;IF EXEC OR KERNEL, CANNOT DEFINE IT
                     06 A5    9F  0731    869          PUSHAB  CTX_T_LOGNAM+1(R5)      ;PUSH DESCRIPTOR OF LOGNAM
                7E   05 A5    9A  0734    870          MOVZBL  CTX_T_LOGNAM(R5),-(SP)  ;
            52   04 AE    6E    C1  0738    871          ADDL3   (SP),4(SP),R2           ;POINT TO WORD COUNTED EQLNAM
                   51    82    3C  073D    872          MOVZWL  (R2)+,R1                ;CONSTRUCT DESCRIPTOR OF EQLNAM
                         06    BB  0740    873          PUSHR   #^M<R1,R2>              ;PUSH DESCRIPTOR OF EQLNAM
                   53    5E    D0  0742    874          MOVL    SP,R3                   ;GET ADDRESS OF STACK
                                   0745    875          $CRELOG_S LOGNAM=8(R3),-        ;CREATE LOGICAL NAME
                                   0745    876                  EQLNAM=(R3),-
                                   0745    877                  ACMODE=R0,-
                                   0745    878                  TBLFLG=#LOG$C_PROCESS
                   5E    10    C0  0755    879          ADDL    #4*4,SP                 ;CLEANUP STACK
                         05    0758    880 90$:     RSB
                              0759    881
                              0759    882 ;
                              0759    883 ; PROCESS SPAWN CLI SYMBOL RECORDS
                              0759    884 ;
                              0759    885 SPAWN_CLISYM:
                              0759    886          ASSUME  CTX_C_STRING EQ SYM_K_STRING
                              0759    887          ASSUME  CTX_C_PERM EQ SYM_K_PERM
                              0759    888          ASSUME  CTX_C_BINARY EQ SYM_K_BINARY
                              0759    889          ASSUME  CTX_C_KEYPAD EQ SYM_K_KEYPAD
             50   05 A5    9A  0759    890          MOVZBL  CTX_B_SYMTYPE(R5),R0    ; R0 = SYMBOL TYPE
             54   07 A5    9E  075D    891          MOVAB   CTX_T_SYMBOL(R5),R4     ; GET ADDRESS OF ASCIC NAME
                53   84    9A  0761    892          MOVZBL  (R4)+,R3                ; R3/R4 = DESCRIPTOR OF SYMBOL NAME
            52   54   53    C1  0764    893          ADDL3   R3,R4,R2                ; POINT TO JUST AFTER SYMBOL NAME
                              0768    894          ASSUME  CTX_C_STRING EQ 0
                              0768    895          ASSUME  CTX_C_PERM EQ 1
                              0768    896          ASSUME  CTX_C_BINARY EQ 2
                              0768    897          ASSUME  CTX_C_KEYPAD EQ 4
```

```
                     0768   898          CASE    R0,TYPE=B,<-                    ; CASE ON SYMBOL TYPE
                     0768   899                  10$,-                           ; STRING
                     0768   900                  5$,-                            ; IGNORE PERMANENT STRINGS
                     0768   901                  20$,-                           ; BINARY
                     0768   902                  5$,-                            ; UNKNOWN
                     0768   903                  10$>                            ; KEYPAD SYMBOL
                  05 0776   904 5$:      RSB                                     ; SKIP UNKNOWN TYPES
          51   82  3C 0777   905 10$:     MOVZWL  (R2)+,R1                        ; R1/R2 = DESCRIPTOR OF STRING VALUE
               03  11 077A   906         BRB     40$
          51   62  D0 077C   907 20$:     MOVL    (R2),R1                         ; R1 = LONGWORD BINARY VALUE
          56   55  D0 077F   908 40$:     MOVL    R5,R6                           ; COPY ADDRESS OF CTX RECORD
                     0782   909          ASSUME  CTX_C_GLOBAL EQ 0
                     0782   910          ASSUME  CTX_C_LOCAL EQ 1
                     0782   911          ASSUME  CTX_C_KEYTABL EQ 2
                     0782   912          CASE    CTX_B_SYMTAB(R5),TYPE=B,<-      ; CASE ON SYMBOL TABLE
                     0782   913                  50$,-                           ; GLOBAL SYMBOL TABLE
                     0782   914                  60$,-                           ; LOCAL SYMBOL TABLE
                     0782   915                  65$>                            ; KEYPAD SYMBOL TABLE
                  05 078D   916          RSB                                     ; IGNORE UNKNOWN SYMBOL TABLE NUMBER
     55   28 AB  9E 078E   917 50$:     MOVAB   PRC_Q_GLOBAL(R11),R5            ; R5 = ADDRESS OF GLOBAL SYMBOL LISTHEAD
               0A  11 0792   918         BRB     70$
     55   38 AB  9E 0794   919 60$:     MOVAB   PRC_Q_LOCAL(R11),R5             ; R5 = ADDRESS OF LOCAL TABLE LISTHEAD
               04  11 0798   920         BRB     70$
     55   40 AB  9E 079A   921 65$:     MOVAB   PRC_Q_KEYPAD(R11),R5            ; R5 = ADDRESS OF KEYPAD SYMBOL TABLE
           F85F'  30 079E   922 70$:     BSBW    DCL$RESTORE_SYM                 ; ADD SYMBOL (R1-R6 ARE ARGUMENTS)
                  05 07A1   923          RSB
                     07A2   924
                     07A2   925 ;
                     07A2   926 ; PROCESS SPAWN LOGICAL NAME TABLE RECORDS
                     07A2   927 ;
                     07A2   928 SPAWN_LNMTABLE:
           04   A5  91 07A2   929          CMPB    CTX_B_ACMODE(R5),-              ; IS ACMODE HIGHER THAN SUPER?
                02         07A5   930                  #PSL$C_SUPER                ;
                   3E  19 07A6   931          BLSS    90$                         ; IF EXEC OR KERNEL, CANNOT DEFINE IT
     7E    04   A5  9A 07A8   932          MOVZBL  CTX_B_ACMODE(R5),-(SP)      ; GET ACMODE OF LOGICAL NAME
     7E    05   A5  9A 07AC   933          MOVZBL  CTX_B_TFLAGS(R5),-(SP)      ; GET ATTR FLAGS
     54    0C   A5  DE 07B0   934          MOVAL   CTX_T_LNMTABLE(R5),R4       ; PTR TO NAME
          51   84  9A 07B4   935          MOVZBL  (R4)+,R1                    ; GET SIZE
               54  DD 07B7   936          PUSHL   R4                          ; PUSH ADDR
               51  DD 07B9   937          PUSHL   R1                          ; AND SIZE
          54   51  C0 07BB   938          ADDL2   R1,R4                       ; ADVANCE TO PARENT TABLE NAME
          51   84  9A 07BE   939          MOVZBL  (R4)+,R1                    ; GET SIZE
               54  DD 07C1   940          PUSHL   R4                          ; PUSH ADDR
               51  DD 07C3   941          PUSHL   R1                          ; AND SIZE
          51   5E  D0 07C5   942          MOVL    SP,R1                       ; REMEMBER DESCR ADDR
                     07C8   943
                     07C8   944          $CRELNT_S       -
                     07C8   945                  ATTR=16(R1),-               ; ATTRIBUTES
                     07C8   946                  QUOTA=CTX_L_QUOTA(R5),-     ; QUOTA
                     07C8   947                  ACMODE=20(R1),-             ; ACCESS MODE
                     07C8   948                  TABNAM=8(R1),-              ; DESC OF TABLE NAME
                     07C8   949                  PARTAB=(R1)                 ; DESC OF PARENT TABLE NAME
          5E   18  C0 07E3   950          ADDL2   #<6*4>,SP                   ; REMOVE DESCRIPTORS
                  05 07E6   951 90$:     RSB
                     07E7   952
                     07E7   953 ;
                     07E7   954 ; THE FOLLOWING ASSUMES ARE NEEDED BECAUSE THIS CODE PROCESSES LOGICAL
```

```
                        07E7   955 ; NAME BLOCKS IN THEIR INTERNAL FORMAT.
                        07E7   956 ;
                        07E7   957         ASSUME  LNMX$B_FLAGS    EQ      0
                        07E7   958         ASSUME  LNMX$B_INDEX    EQ      LNMX$B_FLAGS+1
                        07E7   959         ASSUME  LNMX$W_HASH     EQ      LNMX$B_INDEX+1
                        07E7   960         ASSUME  LNMX$T_XLATION  EQ      LNMX$W_HASH+2
                        07E7   961
                        07E7   962 ;
                        07E7   963 ; PROCESS SPAWNED LOGICAL NAMES
                        07E7   964 ;
                        07E7   965 $SPAWN_LNMNAME:
02   04 A5      91      07E7   966         CMPB    CTX_B_ACMODE(R5),#PSL$C_SUPER   ;IS ACMODE HIGHER THAN SUPER?
      01      18      07EB   967         BGEQ    5$
                      07ED   968 3$:      RSB                                     ;IF EXEC OR KERNEL, CANNOT DEFINE IT
51   06 A5      9A      07EE   969 5$:      MOVZBL  CTX_B_TRANCNT(R5),R1    ; COUNT OF XLATIONS
  51    24      C4      07F2   970         MULL2   #<3*3*4>,R1             ; 3 DESCRIPTORS OF 3 LONGWORDS E.
  51    04      C0      07F5   971         ADDL2   #4,R1                   ; ZERO TERMINATOR
     F805'      30      07F8   972         BSBW    DCL$ALLDYNMEM           ; GET SPACE FOR ITMLST
    EF  50      E9      07FB   973         BLBC    R0,3$                   ; GIVE UP IF NO ROOM
  56    51      7D      07FE   974         MOVQ    R1,R6                   ; SAVE SPACE DESCRIPTION
50   07 A5      9E      0801   975         MOVAB   CTX_T_LNMNAME(R5),R0    ; PTR TO NAME
   01 A0      9F      0805   976         PUSHAB  1(R0)                   ; MAKE DESCRIPTOR OF TABLE
  7E    80      9A      0808   977         MOVZBL  (R0)+,-(SP)             ;  NAME
  50    6E      C0      080B   978         ADDL2   (SP),R0                 ; SKIP OVER TABLE NAME
   01 A0      9F      080E   979         PUSHAB  1(R0)                   ; MAKE DESCRIPTOR OF LOGICAL
  7E    80      9A      0811   980         MOVZBL  (R0)+,-(SP)             ;  NAME
  50    6E      C0      0814   981         ADDL2   (SP),R0                 ; SKIP OVER LOGICAL NAME
  54    5E      D0      0817   982         MOVL    SP,R4                   ; REMEMBER ADDRESS
   38 60      02      E0      081A   983 10$:      BBS     #LNMX$V_XEND,LNMX$B_FLAGS(R0),20$ ; END OF LIST
     7E      80      9A      081E   984         MOVZBL  (R0)+,-(SP)             ; PUSH ATTRIBUTE VALUE (LNMX$B_FLAGS)
6E   6E      08      9C      0821   985         ROTL    #8,(SP),(SP)            ; PUT ATTRIBUTES IN 2ND BYTE
82   00030004 8F      D0      0825   986         MOVL    #<<LNM$_ATTRIBUTES@16>+4>,(R2)+ ; MAKE ITMLST ENTRY FOR XL ATTR
     82      6E      DE      082C   987         MOVAL   (SP),(R2)+              ; ADDR OF XL ATTR
     82      D4      082F   988         CLRL    (R2)+                   ; IGNORE RETURN LENGTH
     7E      80      9A      0831   989         MOVZBL  (R0)+,-(SP)             ; PUSH INDEX VALUE (LNMX$B_INDEX)
82   00010004 8F      D0      0834   990         MOVL    #<<LNM$_INDEX@16>+4>,(R2)+ ; MAKE ITMLST ENTRY FOR XL INDEX
     82      6E      DE      083B   991         MOVAL   (SP),(R2)+              ; ADDR OF XL INDEX
     82      D4      083E   992         CLRL    (R2)+                   ; IGNORE RETURN LENGTH
     50      02      C0      0840   993         ADDL2   #2,R0                   ; SKIP OVER THE HASH CODE FIELD
     51      80      9A      0843   994         MOVZBL  (R0)+,R1                ; STRING LENGTH
     82      51      B0      0846   995         MOVW    R1,(R2)+                ; MAKE ITMLST ENTRY FOR XL STRING
     82      02      B0      0849   996         MOVW    #LNM$_STRING,(R2)+      ;  (LNMX$T_XLATION)
     82      60      DE      084C   997         MOVAL   (R0),(R2)+              ; ADDR OF XL STRING
     82      D4      084F   998         CLRL    (R2)+                   ; IGNORE RETURN LENGTH
     50      51      C0      0851   999         ADDL2   R1,R0                   ; SKIP OVER STRING
     C4      11      0854  1000         BRB     10$                     ; GET NEXT STRING
     82      D4      0856  1001 20$:      CLRL    (R2)+                   ; MARK END OF ITMLST
7E   04 A5      9A      0858  1002         MOVZBL  CTX_B_ACMODE(R5),-(SP)  ; GET ACCESS MODE
7E   05 A5      9A      085C  1003         MOVZBL  CTX_B_NFLAGS(R5),-(SP)  ; GET NAME ATTRIBUTES
  50    5E      D0      0860  1004         MOVL    SP,R0                   ; GET ADDRESS OF STACK
                        0863  1005
                        0863  1006         $CRELNM_S       -               ; CREATE THE NAME
                        0863  1007                 ATTR=(R0),-             ; ATTRIBUTES
                        0863  1008                 TABNAM=8(R4),-          ; TABLE NAME
                        0863  1009                 LOGNAM=(R4),-           ; LOGICAL NAME
                        0863  1010                 ACMODE=4(R0),-          ; ACCESS MODE
                        0863  1011                 ITMLST=(R7)             ; ITMLST
```

```
5E   10 A4    9E  0876  1012          MOVAB   16(R4),SP               ; CLEAN THE STACK
   51    56    D0  087A  1013          MOVL    R6,R1                  ; SIZE OF DYN MEMORY
   50    57    D0  087D  1014          MOVL    R7,R0                  ; ADDR OF DYN MEMORY
      F77D'    30  0880  1015          BSBW    DCL$DEADYNMEM          ; FREE THE SPACE
               05  0883  1016 90$:     RSB
                   0884  1017
                   0884  1018 SPAWN_LNMTRAN:
               05  0884  1019          RSB                            ; RETURN
                   0885  1020
```

```
                     0885  1022          .SBTTL  CLI$GET_PRC, GET ADDRESS OF PRC STRUCTURE
                     0885  1023 ;---
                     0885  1024 ;
                     0885  1025 ;       THIS ROUTINE IS CALLED TO GET THE ADDRESS OF THE CLI
                     0885  1026 ;       OWN STORAGE AREA (PRC).
                     0885  1027 ;
                     0885  1028 ; INPUTS:
                     0885  1029 ;
                     0885  1030 ;       NONE
                     0885  1031 ;
                     0885  1032 ; OUTPUTS:
                     0885  1033 ;
                     0885  1034 ;       R11 = ADDRESS OF PRC AREA
                     0885  1035 ;---
                     0885  1036
                     0885  1037 CLI$GET_PRC::
5B  00000000'GF   9E 0885  1038          MOVAB   G^CTL$AG_CLIDATA,R11    ;GET ADDRESS OF PPD
     5B   08 AB   D0 088C  1039          MOVL    PPD$L_PRC(R11),R11      ;SET ADDRESS OF CLI OWN STORAGE
                  05 0890  1040          RSB
                     0891  1041
                     0891  1042          .END    DCL$STARTUP
```

```
$$T1                            = 00000000
BATCH$RESTART                     00000019 R      02
CDU$UPGRADE_TABLE                 ********   X    02
CLI$GET_PRC                       00000885 RG     02
CLI$_NOCMDPROC                  = 0003890A
CTL$AG_CLIDATA                    ********   X    02
CTL$AG_CLITABLE                   ********   X    02
CTL$AL_CLICALBK                   ********   X    02
CTL$AL_STACK                      ********   X    03
CTX_B_ACMODE                      00000004
CTX_B_CONTINUE                    00000012
CTX_B_FLAGS                       0000000E
CTX_B_KEYLENGTH                   00000002
CTX_B_NFLAGS                      00000005
CTX_B_NONUNIQUE                   00000006
CTX_B_PROMPTLEN                   0000000F
CTX_B_SYMTAB                      00000004
CTX_B_SYMTYPE                     00000005
CTX_B_TFLAGS                      00000005
CTX_B_TRANCNT                     00000006
CTX_C_BINARY                    = 00000002
CTX_C_GLOBAL                    = 00000000
CTX_C_HDRLEN                      00000033
CTX_C_KEYPAD                    = 00000004
CTX_C_KEYTABL                   = 00000002
CTX_C_LOCAL                     = 00000001
CTX_C_MAXLEN                    = 00000400
CTX_C_PERM                      = 00000001
CTX_C_STRING                    = 00000000
CTX_G_PROMPT                      00000013
CTX_K_HDRLEN                      00000033
CTX_L_OUTOFBAND                   0000000A
CTX_L_QUOTA                       00000008
CTX_Q_PROCPRIV                    00000002
CTX_T_CMDSTR                      00000002
CTX_T_KEYSTATE                    00000003
CTX_T_LNMNAME                     00000007
CTX_T_LNMTABLE                    0000000C
CTX_T_LOGNAM                      00000005
CTX_T_SYMBOL                      00000007
CTX_V_AUTOLOGO                  = 00000000
CTX_V_MODE                      = 00000001
CTX_V_VERIFY                    = 00000002
CTX_V_VERIMAGE                  = 00000003
CTX_V_WAIT                      = 00000004
CTX_W_ENTSIZE                     00000002
CTX_W_PMPTCTRL                    00000010
CTX_W_PROT                        00000006
CTX_W_TYPE                        00000000
DCL                               00000000 R      02
DCL$ABORT                         ********   X    02
DCL$ALLDEACMD                     ********   X    02
DCL$ALLDYNMEM                     ********   X    02
DCL$ALLOCSYM                      ********   X    02
DCL$ALLOC_STATE                   ********   X    02
DCL$CHANGE_MODE                   ********   X    02
DCL$CONDHAND                      ********   X    02
DCL$CREATE_OUTPUT                 ********   X    02
DCL$CRLF                          00000037 RG     02
DCL$C_PROMPTLEN                 = 00000005   G
DCL$DEADYNMEM                     ********   X    02
DCL$DEFINE_P1_TO_P8               ********   X    02
DCL$ENBCONTRLY                    ********   X    02
DCL$ERRORMSG                      ********   X    02
DCL$EXITHAND                      ********   X    02
DCL$PUSHPROC                      ********   X    02
DCL$RESETOOB                      ********   X    02
DCL$RESTART                       ********   X    02
DCL$RESTORE_SYM                   ********   X    02
DCL$RMSRUNDWN                     ********   X    02
DCL$SETVERIFY_IMAGE               ********   X    02
DCL$SET_TIMER                     ********   X    02
DCL$STARTUP                       0000005F RG     02
DCL$T_PROMPT                      00000039 RG     02
DCL$UTLSERV                       ********   X    02
DCL$VERIFY_IMAGE                  ********   X    02
DEFAULT                           00000027 R      02
DEV$V_MBX                       = 00000014
DIB$C_LENGTH                    = 00000074
DIB$L_DEVCHAR                   = 00000000
EXE$C_SYSEFN                      ********   X    02
FAB$B_BID                       = 00000000
FAB$B_BLN                       = 00000001
FAB$C_BID                       = 00000003
FAB$C_BLN                       = 00000050
FAB$L_NAM                       = 00000028
FAB$W_IFI                       = 00000002
FALSE                             00000013 R      02
FLUSH_RATE                        0000002F R      02
IDF_B_OUTFLAGS                    00000038
IDF_C_LENGTH                      00000074
IDF_K_LENGTH                      00000074
IDF_L_FILENAME                    00000068
IDF_L_INPRABCTX                   0000000C
IDF_L_LNK                         00000000
IDF_L_ONCTLY                      00000060
IDF_L_ONERROR                     00000008
IDF_L_OUTRABCTX                   00000024
IDF_L_SEARCHCTX                   00000064
IDF_Q_LABEL                       00000018
IDF_Q_LOCAL                       00000010
IDF_T_INPDVI                      0000003C
IDF_T_OUTDVI                      00000028
IDF_V_INPCCL                    = 00000001
IDF_V_OUTCCL                    = 00000000
IDF_W_FLAG                        0000005E
IDF_W_INPDID                      00000052
IDF_W_INPFID                      0000004C
IDF_W_INPIFI                      00000004
IDF_W_INPRFA                      00000058
IDF_W_ONLEVEL                     00000006
IDF_W_OUTIFI                      00000020
IDF_W_OUTISI                      00000022
INITIAL_ERROR                     000005B9 R      02
```

```
IO$M_TT_PROCESS                 = 00002000
IO$_READVBLK                    = 00000031
IO$_SETMODE                     = 00000023
ITRM_C_LENGTH                     00000030
ITRM_C_MINLEN                     00000018
ITRM_K_LENGTH                     00000030
ITRM_K_MINLEN                     00000018
ITRM_L_INIADDR                    0000001C
ITRM_L_INIRET                     00000020
ITRM_L_MODIFIERS                  00000004
ITRM_L_MODRET                     00000008
ITRM_L_OFFRET                     0000002C
ITRM_L_OFFSET                     00000028
ITRM_L_PMPTADDR                   00000010
ITRM_L_PMPTRET                    00000014
ITRM_W_INICODE                    0000001A
ITRM_W_INILEN                     00000018
ITRM_W_MODCODE                    00000002
ITRM_W_MODLEN                     00000000
ITRM_W_OFFCODE                    00000026
ITRM_W_OFFLEN                     00000024
ITRM_W_PMPTCODE                   0000000E
ITRM_W_PMPTLEN                    0000000C
JPI$_OWNER                        ******** X   02
LNM$C_NAMLENGTH                 = 000000FF
LNM$_ATTRIBUTES                 = 00000003
LNM$_INDEX                      = 00000001
LNM$_STRING                     = 00000002
LNMX$B_FLAGS                    = 00000000
LNMX$B_INDEX                    = 00000001
LNMX$T_XLATION                  = 00000004
LNMX$V_XEND                     = 00000002
LNMX$W_HASH                     = 00000002
LOG$C_PROCESS                   = 00000002
MAX_DEPTH                       = 00000010 G
NAM$C_BID                         ******** X   02
NAM$C_BLN                         ******** X   02
PPD$B_NPROCS                      0000001C
PPD$C_DVIFID                    = 0000001C
PPD$C_LENGTH                      00000168
PPD$K_LENGTH                      00000168
PPD$L_INPDEV                      00000044
PPD$L_LGI                         00000014
PPD$L_LSTSTATUS                   00000018
PPD$L_OUTDEV                      00000064
PPD$L_PRC                         00000008
PPD$Q_CLIREG                      00000004
PPD$Q_CLISYMTBL                   0000000C
PPD$T_FILENAME                    00000068
PPD$T_INPDVI                      00000028
PPD$T_OUTDVI                      00000048
PPD$V_CAPTIVE                   = 00000003
PPD$V_CONTINUE                  = 00000002
PPD$V_INPCCL                    = 00000005
PPD$V_MODE                      = 00000001
PPD$V_NOCTLY                    = 00000000
PPD$V_OUTCCL                    = 00000006
PPD$V_RESTART                   = 00000004
PPD$W_FLAGS                       00000002
PPD$W_INPCHAN                     0000001E
PPD$W_INPDID                      0000003E
PPD$W_INPFID                      00000038
PPD$W_INPIFI                      000C0020
PPD$W_INPISI                      00000022
PPD$W_OUTDID                      0000005E
PPD$W_OUTFID                      00000058
PPD$W_OUTIFI                      00000024
PPD$W_OUTISI                      00000026
PPD$W_SIZE                        00000000
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_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                      000000BC
PRC_L_IMGACTSTS                   00000080
PRC_L_INDCLOCK                    0000007C
PRC_L_INDEPTH                     0000005C
PRC_L_INDFAB                      0000001C
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	PRD_W_OUTDID		0000022A			
PRC_L_STV		00000088	PRD_W_OUTFID		00000224			
PRC_L_SYMBOL		00000060	PSL$C_SUPER	=	00000002			
PRC_L_TMBX		00000074	PSL$C_USER	=	00000003			
PRC_L_TRMLIST		00000010	RAB$B_BID	=	00000000			
PRC_M_CTRLY	=	02000000	RAB$B_BLN	=	00000001			
PRC_M_MODE	=	00000040	RAB$B_MBC	=	00000037			
PRC_M_SAVCMDV	=	00000004	RAB$B_MBF	=	00000036			
PRC_M_SAVIMGV	=	00000008	RAB$C_BID	=	00000001			
PRC_M_VERIFY	=	00000080	RAB$C_BLN	=	00000044			
PRC_M_VERIMAGE	=	00000080	RAB$L_CTX	=	00000018			
PRC_Q_ALLOCREG		00000020	RAB$L_FAB	=	0000003C			
PRC_Q_COMMAND		000000E0	RAB$L_ROP	=	00000004			
PRC_Q_FLUSHTIME		000000D0	RAB$L_XAB	=	00000040			
PRC_Q_GLOBAL		00000028	RAB$V_ETO	=	0000001C			
PRC_Q_IMAGENAME		000000D8	RAB$V_PPF_IND	=	0000000E			
PRC_Q_KEYPAD		00000040	RAB$W_ISI	=	00000002			
PRC_Q_LABEL		00000030	RESERVED		0000003D	R		02
PRC_Q_LOCAL		00000038	SPAWN_CLISYM		00000759	R		02
PRC_Q_SAVEPRIV		000000E8	SPAWN_CMDSTR		000006F4	R		02
PRC_T_OUTDVI		0000011C	SPAWN_CONTEXT		000005CB	R		02
PRC_V_AUTOLOGO	=	00000008	SPAWN_HEADER		0000066A	R		02
PRC_V_CARRCNTL	=	00000000	SPAWN_KEYSTATE		00000717	R		02
PRC_V_CMD	=	00000000	SPAWN_LNMNAME		000007E7	R		02
PRC_V_CTRLY	=	00000019	SPAWN_LNMTABLE		000007A2	R		02
PRC_V_DETACHED	=	0000000F	SPAWN_LNMTRAN		00000884	R		02
PRC_V_EOFLOGO	=	0000000E	SPAWN_LOGNAM		00000728	R		02
PRC_V_MODE	=	00000006	SS$_NORMAL		********		X	02
PRC_V_VERIFY	=	00000007	SS$_NOTRAN		********		X	02
PRC_W_ASTIOSB		000000C6	SYM_B_FLAGS		0000000B			
PRC_W_ASTRETN		000000C8	SYM_B_NONUNIQUE		0000000B			
PRC_W_ASTSTATUS		000000C4	SYM_B_TYPE		0000000A			
PRC_W_ATTMBX		0000007A	SYM_K_BINARY	=	00000002			
PRC_W_FLAGS		00000068	SYM_K_KEYPAD	=	00000004			
PRC_W_INPCHAN		00000064	SYM_K_PERM	=	00000001			
PRC_W_ONLEVEL		0000006A	SYM_K_STRING	=	00000000			
PRC_W_OUTIFI		00000114	SYM_L_BL		00000004			
PRC_W_OUTISI		00000116	SYM_L_FL		00000000			
PRC_W_OUTMBXCHN		000000CA	SYM_T_SYMBOL		0000000C			
PRC_W_OUTMBXREF		000000CE	SYM_W_SIZE		00000008			
PRC_W_OUTMBXSIZ		000000CC	SYS$ASSIGN		********		GX	02
PRC_W_PMPTCTRL		000000F1	SYS$CRELNM		********		GX	02
PRC_W_WAITIOSB		00000066	SYS$CRELNT		********		GX	02
PRD_C_LENGTH		00000214	SYS$CRELOG		********		GX	02
PRD_C_XLENGTH		00000244	SYS$DASSGN		********		GX	02
PRD_G_ALTINPRAB		000000F4	SYS$DCLCMH		********		GX	02
PRD_G_ALTOUTRAB		00000138	SYS$DELLOG		********		GX	02
PRD_G_FAB		00000000	SYS$ERROR		00000004	R		02
PRD_G_INPRAB		000000B0	SYS$EXIT		********		GX	03
PRD_G_NAM		00000050	SYS$GETCHN		********		GX	02
PRD_G_OUTRAB		0000017C	SYS$GETJPIW		********		GX	02
PRD_G_TRMLIST		000001E4	SYS$QIOW		********		GX	02
PRD_G_XABTRM		000001C0	SYS$RUNDWN		********		GX	02
PRD_K_LENGTH		00000214	SYS$SETPRV		********		GX	02
PRD_K_XLENGTH		00000244	SYS$TRNLOG		********		GX	02
PRD_T_OUTDVI		00000214	TRM$M_TM_ESCAPE	=	00004000			
PRD_T_OUTFNM		00000230	TRM$M_TM_NORECALL	=	00010000			

```
TRM$_INIOFFSET                  =  00000008
TRM$_INISTRNG                   =  00000005
TRM$_MODIFIERS                  =  00000000
TRM$_PROMPT                     =  00000004
TRUE                               0000000E R      02
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_LENGTH                       FFFFF486
WRK_G_BUFFER                       FFFFF492
WRK_G_INPBUF                       FFFFF896
WRK_G_RESULT                       FFFFF9B6
WRK_K_LENGTH                       FFFFF486
WRK_L_CHARPTR                      FFFFF48E
WRK_L_DISALLOW                     FFFFFFE6
WRK_L_ERRORRTN                     FFFFF9AE
WRK_L_EXPANDPTR                    FFFFF486
WRK_L_IMAGE                        FFFFFFE2
WRK_L_MARKPTR                      FFFFF48A
WRK_L_PAROUT                       FFFFFFD2
WRK_L_PMPTADDR                     FFFFF9A2
WRK_L_PROMPTRTN                    FFFFF9A6
WRK_L_PROPTR                       FFFFFFC6
WRK_L_QUABLK                       FFFFFFCA
WRK_L_READRTN                      FFFFF9AA
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                      FFFFF9B2
WRK_L_TAB_VEC                      FFFFFFDE
WRK_L_VERB                         FFFFFFBE
WRK_M_COMMAND                   =  00000002
WRK_W_FLAGS                        FFFFFFF0
WRK_W_FLAGS2                       FFFFFFF2
WRK_W_IMGCHAN                      FFFFFFEE
WRK_W_PMPTLEN                      FFFFF99E
XAB$C_TRM                       =  0000001F
XAB$C_TRMLEN                    =  00000024
XAB$L_ITMLST                    =  00000008
XAB$W_ITMLST_LEN                =  0000000C
_$$_                            =  000000EF
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL$ZCODE	00000891 (2193.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
DCL$$BASE	00000017 (23.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.08	00:00:00.60
Command processing	99	00:00:00.71	00:00:04.65
Pass 1	484	00:00:21.52	00:01:02.81
Symbol table sort	0	00:00:02.48	00:00:09.20
Pass 2	191	00:00:04.24	00:00:13.49
Symbol table output	46	00:00:00.29	00:00:00.91
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	838	00:00:29.35	00:01:31.69

The working set limit was 1800 pages.
110642 bytes (217 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1671 non-local and 61 local symbols.
1042 source lines were read in Pass 1, producing 22 object records in Pass 2.
76 pages of virtual memory were used to define 55 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
_255DUA28:[DCL.OBJ]DCL.MLB;1	11
_255DUA28:[SYS.OBJ]LIB.MLB;1	3
_255DUA28:[SYSLIB]STARLET.MLB;2	31
TOTALS (all libraries)	45

2046 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:INITIAL/OBJ=OBJ$:INITIAL MSRC$:INITIAL/UPDATE=(ENH$:INITIAL)+EXECML$/LIB+LIB$:DCL/LIB+SYS$LIBRARY:SYSBLDMLB/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

GETKEYNAM
LIS

GOTO
LIS

EXPRESS
LIS

HANDLE
LIS

IMAGECTRL
LIS

INITIAL
LIS

INDIRECT
LIS

FILECMDS
LIS

IF
LIS

IMAGEXECT
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INQUIRE
LIS

LEXICON
LIS

KEYPAD
LIS

LOGICAL
LIS