```
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDD          DDD  CCC            LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
```

```
HH      HH    AAAAAA    NN      NN  DDDDDDDD    LL          EEEEEEEEEE
HH      HH    AAAAAA    NN      NN  DDDDDDDD    LL          EEEEEEEEEE
HH      HH  AA      AA  NN      NN  DD      DD  LL          EE
HH      HH  AA      AA  NN      NN  DD      DD  LL          EE
HH      HH  AA      AA  NNNN    NN  DD      DD  LL          EE
HH      HH  AA      AA  NNNN    NN  DD      DD  LL          EE
HHHHHHHHHH  AA      AA  NN  NN  NN  DD      DD  LL          EEEEEEEE
HHHHHHHHHH  AA      AA  NN  NN  NN  DD      DD  LL          EEEEEEEE
HH      HH  AAAAAAAAAA  NN    NNNN  DD      DD  LL          EE
HH      HH  AAAAAAAAAA  NN    NNNN  DD      DD  LL          EE
HH      HH  AA      AA  NN      NN  DD      DD  LL          EE              ....
HH      HH  AA      AA  NN      NN  DD      DD  LL          EE              ....
HH      HH  AA      AA  NN      NN  DDDDDDDD  LLLLLLLLLL  EEEEEEEEEE      ....
HH      HH  AA      AA  NN      NN  DDDDDDDD  LLLLLLLLLL  EEEEEEEEEE      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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  HANDLE - CONDITION AND CONTROL/Y AST ROUTINES
0000     2          .IDENT  'V04-002'
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 ; CONDITION AND CONTROL Y AST HANDLER ROUTINES
0000    28 ;
0000    29 ; D. N. CUTLER 29-MAR-77
0000    30 ;
0000    31 ; MODIFIED BY:
0000    32 ;
0000    33 ;       V04-002 HWS0109         Harold Schultz  14-Sep-1984
0000    34 ;               Enable the AST error checking code added by HWS0107
0000    35 ;
0000    36 ;       V04-001 HWS0107         Harold Schultz  07-Sep-1984
0000    37 ;               In the CTRL-Y AST routine, save the AST status. When
0000    38 ;               reenabling the CTRL-Y AST, save the return status and the
0000    39 ;               IOSB. Add code to set hangup pending if any error on reenabling
0000    40 ;               CTRL-Y AST (temporarily branch around error checking code
0000    41 ;               for now).
0000    42 ;
0000    43 ;       V03-011 HWS0085         Harold Schultz  19-Jul-1984
0000    44 ;               In the CTRL-T processing, use value of SCSNODE for node
0000    45 ;               name only if no translation of SYS$NODE.
0000    46 ;
0000    47 ;       V03-010 HWS0048         Harold Schultz  03-Apr-1984
0000    48 ;               Pass cli table name, if any, to the spawned process.
0000    49 ;
0000    50 ;       V03-010 HWS0047         Harold Schultz  02-Apr-1984
0000    51 ;               Fix ^T to handle multiple sets of brackets when formatting
0000    52 ;               the imagename.
0000    53 ;
0000    54 ;       V03-009 HWS0022         Harold Schultz  08-Mar-1984
0000    55 ;               Don't output '::' on ^T when node name not present
0000    56 ;
0000    57 ;       V03-008 PCG0007         Peter George    08-Feb-1984
```

```
0000     58 ;                  Use $GETSYI to get node name in CTRL/T.
0000     59 ;                  Use $BRKTHRU instead of $BRDCST.
0000     60 ;
0000     61 ;          V03-007 PCG0006         Peter George    18-Nov-1983
0000     62 ;                  Finish making WAIT command CTRL/Y interruptable.
0000     63 ;
0000     64 ;          V03-006 PCG0005         Peter George    28-Sep-1983
0000     65 ;                  Correctly align the SP when popping a call frame of the stack.
0000     66 ;
0000     67 ;          V03-005 PCG0004         Peter George    15-Sep-1983
0000     68 ;                  Recognize two versions of CLI$ spawn data structure.
0000     69 ;
0000     70 ;          V03-004 PCG0003         Peter George    18-Aug-1983
0000     71 ;                  Make WAIT command CTRL/Y interruptable.
0000     72 ;
0000     73 ;          V03-003 PCG0002         Peter George    26-Jun-1983
0000     74 ;                  Bring LIB$SPAWN callback up to speed with SPAWN command.
0000     75 ;                  Use event flags more intelligently.
0000     76 ;                  Restructure logical name callbacks to use new system services.
0000     77 ;
0000     78 ;          V03-002 PCG0001         Peter George    28-Dec-1982
0000     79 ;                  Add DCL$DSBCONTRLY routine.
0000     80 ;
0000     81 ;          V03-001 PHL0045         Peter H. Lipman 14-Apr-1982
0000     82 ;                  Control Y rundown of privileged images delayed until
0000     83 ;                  command dispatching to allow CONTINUE, SPAWN, and
0000     84 ;                  ATTACH commands.
0000     85 ;
0000     86 ;                  Set image privileges to process privileges, saving the
0000     87 ;                  image privileges to be restored by CONTINUE
0000     88 ;
0000     89 ;---
```

```
                                    0000   ;
                                    0000   ; MACRO LIBRARY CALLS
                                    0000   ;
                                    0000
                                    0000           PRCDEF                          ; DEFINE PROCESS WORK AREA
                                    0000           WRKDEF                          ; DEFINE COMMAND WORK AREA
                                    0000           SYMDEF                          ; DEFINE TYPES OF SYMBOLS
                                    0000           SPWNDEF                         ; DEFINE SPAWN PARAMETER BLOCK
                                    0000           $BRKDEF                         ; DEFINE BRKTHRU CLASSES
                                    0000           $CLIMSGDEF                      ; DEFINE ERROR/STATUS CODES
                                    0000           $CLIDEF                         ; DEFINE REQUEST BLOCK FORMATS
                                    0000           $CLISERVDEF                     ; DEFINE CLI SERVICE CODE
                                    0000           $DEVDEF                         ; DEFINE DEVICE CHARACTERISTIC BITS
                                    0000           $IODEF                          ; DEFINE I/O FUNCTION CODES
                                    0000           $LNMDEF                         ; DEFINE LOGICAL NAME CODES
                                    0000           $PSLDEF                         ; DEFINE PROCESSOR STATUS FIELDS
                                    0000           $PPDDEF                         ; PROCESS PERMANENT DEFINITIONS
                                    0000           $RABDEF                         ; DEFINE RAB OFFSETS
                                    0000           $SFDEF                          ; DEFINE CALL FRAME OFFSETS
                                    0000           $SYIDEF                         ; DEFINE GETSYI CODES
                                    0000           $$CLITABDEF                     ; DEFINE MAX PROMPT SIZE
                                    0000
                                00000000           .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                                    0000
                                    0000   LNM$PROCESS:
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 0000           .ASCIC  'LNM$PROCESS'
                                 0B 0000
```

```
                     000C   118          .SBTTL  CHANGE MODE TO SUPERVISOR HANDLER
                     000C   119 ;+
                     000C   120 ; DCL$CHANGE_MODE - CHANGE MODE TO SUPERVISOR HANDLER
                     000C   121 ;
                     000C   122 ; THIS ROUTINE IS ENTERED WHEN A CHANGE MODE TO SUPERVISOR INSTRUCTION IS
                     000C   123 ; EXECTED BY THE RESULT PARSER IN USER MODE OR THE CLI PROPER IN SUPER MODE.
                     000C   124 ;
                     000C   125 ; INPUTS:
                     000C   126 ;
                     000C   127 ;       (SP) = CHANGE MODE ARGUMENT
                     000C   128 ;     4(SP) = PC AFTER CHANGE MODE INSTRUCTION
                     000C   129 ;     8(SP) = PSL OF CHANGE MODE INSTRUCTION
                     000C   130 ;
                     000C   131 ; OUTPUTS:
                     000C   132 ;
                     000C   133 ;       A CHECK IS MADE TO SEE IF THE
                     000C   134 ;       PREVIOUS MODE WAS USER OR SUPERVISOR.
                     000C   135 ;
                     000C   136 ;       PREVIOUS MODE USER:
                     000C   137 ;
                     000C   138 ;             THIS IS REQUEST FOR SERVICE FROM THE RUNNING IMAGE.
                     000C   139 ;             THE REQUEST IS DECODED AND PROCESSED, THE RETURN
                     000C   140 ;             IS MADE TO THE POINT OF CALL WITH STATUS OF REQUEST.
                     000C   141 ;
                     000C   142 ;       PREVIOUS MODE SUPERVISOR:
                     000C   143 ;
                     000C   144 ;             THIS IS RESERVED FOR COMMAND PROCESSING ERRORS.
                     000C   145 ;-
                     000C   146
                     000C   147 DCL$CHANGE_MODE::                            ;HANDLE CHANGE MODE TO SUPERVISOR
03 08 AE    18   E0 000C   148          BBS     #PSL$V_CURMOD,8(SP),10$ ;BR IF CHANGE MODE FROM USER
                     0011   149
                     0011   150 ;
                     0011   151 ; CHANGE MODE FROM SUPER
                     0011   152 ;
                     0011   153
        FFEC'    31  0011   154          BRW     DCL$RESTART             ;*** NYI ***
                     0014   155
                     0014   156 ;
                     0014   157 ; BUILD A FRAME THAT LOOKS LIKE AN AST FRAME, EXCEPT THAT IN PLACE OF
                     0014   158 ; THE SAVE R1 IS THE CHANGE MODE ARGUMENT, AND ZERO FOR SAVED R0 AND
                     0014   159 ; THE AST ARGUMENT.
                     0014   160 ;
                     0014   161
        FFE9'    30  0014   162 10$:     BSBW    CLI$GET_PRC             ;GET ADDRESS OF CLI PROCESS WORK AREA
          7E     7C  0017   163          CLRQ    -(SP)                   ;DUMMY SAVED R0 AND AST ARGUMENT
          05     DD  0019   164          PUSHL   #5                      ;NUMBER OF ARGUMENTS IN AST ROUTINE
   32'AF     6E     FA 001B   165          CALLG   (SP),B^30$              ;CREATE A CALL FRAME IN SUPER MODE
       5E     10     C0 001F   166          ADDL    #<4*4>,SP               ;CLEAR ARGUMENTS AND ARG COUNT
          50     D5  0022   167          TSTL    R0                      ;INTERNAL ERROR?
          0B     14  0024   168          BGTR    20$                     ;BR IF NO
       50     50     CE 0026   169          MNEGL   R0,R0                   ;MAKE POSITIVE
   50  E000   8F     A8 0029   170          BISW    #^XE000,R0              ;INCLUDE SUBSYSTEM AND PRIVATE
       50     04     C4 002E   171          MULL    #4,R0                   ;SCALE TO PROPER PLACE
                 02  0031   172 20$:     REI                             ;RETURN TO USER
                     0032   173
                0000 0032   174 30$:     .WORD   0                       ;REGISTERS SAVED BY RESULT PARSER
```

```
                         0034              CASE    12(AP),-                   ;DECODE USER REQUEST
                         0034                      LIMIT = #CLI$K_PAUSE,-     ;LOW LIMIT OF REQUEST
                         0034                      TYPE = W,<-                ;CASE ON 16 BIT VALUE
                         0034                      PAUSE,-                    ;REQUEST IS PAUSE
                         0034                      DEFLOC,-                   ; DEFINE IN LOCAL TABLE
                         0034                      DEFGBL,-                   ; DEFINE IN GLOBAL TABLE
                         0034                      CHAIN,-                    ;IMAGE TO LATER INVOKE
                         0034                      COMMAND,-                  ;COMMAND LINE TO LATER PROCESS
                         0034                      CREALOG,-                  ;CREATE PROCESS LOGICAL NAME
                         0034                      DELELOG,-                  ;DELETE PROCESS LOGICAL NAME
                         0034                      DISACTRLY,-                ;DISABLE CONTROL Y
                         0034                      ENABCTRLY,-                ;RE-ENABLE CONTROL Y
                         0034                      GETSYM,-                   ; GET A SYMBOL VALUE
                         0034                      DELELCL,-                  ; DELETE A LOCAL SYMBOL
                         0034                      DELEGBL,-                  ; DELETE A GLOBAL SYMBOL
                         0034                      DISAOOB,-                  ;DISABLE OUT-OF-BAND CHARACTER(S)
                         0034                      ENABOOB,-                  ;RE-ENABLE OUT-OF-BAND CHARACTER(S)
                         0034                      SPAWN,-                    ;SPAWN A SUBPROCESS
                         0034                      ATTACH,-                   ;ATTACH TO A PROCESS
                         0034                      >                          ;
                         0059
 50   00038822 8F    D0  0059    INVREQ:   MOVL    #CLI$_INVREQTYP,R0         ;SET ERROR CODE
                     04  0060              RET                                ;
                         0061
      50     08 AB   D0  0061    PAUSE:    MOVL    PRC_L_INPRAB(R11),R0       ;GET PROCESS INPUT RAB
   EF 18 A0     02   E1  0065              BBC     #DEV$V_TRM,RAB$L_CTX(R0),INVREQ ;CAN'T PAUSE IF NOT INTERACTIVE
      5C     08 AD   D0  006A              MOVL    SF$L_SAVE_AP(FP),AP        ;POP CALL FRAME
                0E   EF  006E              EXTZV   #SF$V_STACKOFFS,-          ;GET SP ALIGNMENT
                02       0070                      #SF$S_STACKOFFS,-
      50     06 AD       0071                      SF$W_SAVE_MASK(FP),R0      ;
      5D     0C AD   D0  0074              MOVL    SF$L_SAVE_FP(FP),FP        ;
         5E     14   C0  0078              ADDL    #5*4,SP                    ;POP CALL FRAME OFF THE STACK
         5E     50   C0  007B              ADDL    R0,SP                      ;REALIGN THE STACK
              042D   31  007E              BRW     DCL$SCNTRLY                ;SIMULATE A CONTROL/Y
                         0081
                         0081    ;
                         0081    ; DEFINE A SYMBOL FOR THE PROCESS
                         0081    ;
                         0081
                         0081              .ENABL  LSB
                         0081
      55     38 AB   9E  0081    DEFLOC:   MOVAB   PRC_Q_LOCAL(R11),R5        ;SET ADDRESS OF THE SYMBOL TABLE
                04   11  0085              BRB     10$                        ;
      55     28 AB   9E  0087    DEFGBL:   MOVAB   PRC_Q_GLOBAL(R11),R5       ;SET ADDRESS OF PROPER TABLE
      53     04 A9   7D  008B    10$:      MOVQ    4(R9),R3                   ;SET SYMBOL NAME DESCRIPTOR
         53     53   3C  008F              MOVZWL  R3,R3                      ;GET LENGTH OF SYMBOL NAME
                         0092              IFNORD  R3,(R4),ACCVIO             ;ERROR IF CANNOT READ IT
      51     0C A9   7D  0098              MOVQ    12(R9),R1                  ;SET SYMBOL VALUE DESCRIPTOR
         51     51   3C  009C              MOVZWL  R1,R1                      ;GET LENGTH OF VALUE
                06   13  009F              BEQL    20$                        ;IF NULL VALUE, SKIP PROBE
                         00A1              IFNORD  R1,(R2),ACCVIO             ;ERROR IF CANNOT READ IT
         50     00   D0  00A7    20$:      MOVL    #SYM_K_STRING,R0           ;SET TYPE OF CLI SYMBOL
             FF53'   30  00AA              BSBW    DCL$ALLOCSYMABR            ;CREATE THE SYMBOL
                     04  00AD              RET                                ;ALL DONE
                         00AE
                         00AE              .DSABL  LSB
                         00AE
```

```
                50   0000'8F   3C   00AE   232 ACCVIO: MOVZWL  #SS$_ACCVIO,R0          ;SIGNAL ACCESS VIOLATION
                                04   00B3   233         RET
                                          00B4   234
                                          00B4   235 ;
                                          00B4   236 ; Get a symbol's value
                                          00B4   237 ;
                                          00B4   238 ; WARNING:
                                          00B4   239 ; The returned value string MUST be copied from the area pointed to by
                                          00B4   240 ; the descriptor to a user-defined non-volitile area before the callback
                                          00B4   241 ; facility is used again.  The callback facility may overwrite the area
                                          00B4   242 ; which it uses to build the returned value string.
                                          00B4   243 ;
                                          00B4   244 GETSYM:
                  5A    04 AB    D0   00B4   245         MOVL    PRC_L_SAVFP(R11), R10   ;Get address of work area descriptor.
                  51    04 A9    7D   00B8   246         MOVQ    4(R9), R1               ;Get symbol name to search for.
                        51   51    3C   00BC   247         MOVZWL  R1,R1                   ;Get low order word
                                    00BF   248         IFNORD  R1,(R2),ACCVIO          ;Error if cannot read it
                           FF38'    30   00C5   249         BSBW    DCL$SEARCH              ;Search for it.
                             7D 50  E9   00C8   250         BLBC    R0, NOSUCHSYM           ;Check for symbol not found.
                       03 A9   54   F6   00CB   251         CVTLB   R4, 3(R9)               ;Return local/global table indicator.
                  10 A9   F486 CA   D0   00CF   252         MOVL    WRK_L_EXPANDPTR(R10), 16(R9) ;Return address of string.
                                 52   D5   00D5   253         TSTL    R2                      ;Check for binary valued symbol.
                                 1A   13   00D7   254         BEQL    50$                     ;Branch if binary valued symbol.
                           F892 CA   9F   00D9   255         PUSHAB  WRK_G_BUFFER+WRK_C_CMDBUFSIZ(R10) ;Compute number of characters
                50   8E    F486 CA   C3   00DD   256         SUBL3   WRK_L_EXPANDPTR(R10), (SP)+, R0 ;remaining in expansion buffer.
                           50   51   D1   00E3   257         CMPL    R1, R0                  ;Enough space for symbol string value?
                                 2F   14   00E6   258         BGTR    90$                     ;Branch if not enough space.
                       0C A9   51   D0   00E8   259         MOVL    R1, 12(R9)              ;Return length of string symbol.
                  10 B9   62   51   28   00EC   260         MOVC3   R1, (R2), @16(R9)       ;Copy string to expansion buffer.
                                 20   11   00F1   261         BRB     70$                     ;Go to common exit code.
                                 51   D5   00F3   262 50$:     TSTL    R1                      ;Check for a negative number.
                                 0C   18   00F5   263         BGEQ    55$                     ;If not, skip extra negative stuff.
                           F486 DA   2D   90   00F7   264         MOVB    #^A/-/, @WRK_L_EXPANDPTR(R10) ;For negative numbers, put a
                                F486 CA   D6   00FC   265         INCL    WRK_L_EXPANDPTR(R10)    ;leading minus sign in ASCII string,
                                51   51   CE   0100   266         MNEGL   R1, R1                  ;and negate value before converting.
                                 1A   10   0103   267 55$:     BSBB    100$                    ;Call binary to ASCII converter.
OC A9   F486 CA   10 A9   C3   0105   268         SUBL3   16(R9), WRK_L_EXPANDPTR(R10), - ;Compute number of bytes in
                                          010D   269                 12(R9)                  ;converted value string.
                           F486 CA   10 A9   D0   010D   270         MOVL    16(R9), WRK_L_EXPANDPTR(R10) ;Restore expansion buf. ptr.
                                50   01   D0   0113   271 70$:     MOVL    #1, R0                  ;Signal successful lookup.
                                          04   0116   272         RET                             ;Return to caller.
                                          0117   273 ;
                                          0117   274 ; Return expansion buffer too small status.
                                          0117   275 ;
                  50   00038018 8F   D0   0117   276 90$:     MOVL    #CLI$_BUFOVF, R0
                                          04   011E   277         RET
                                          011F   278 ;
                                          011F   279 ; Recursive routine to output the ASCII number, high order digits first
                                          011F   280 ; without any leading spaces or zeros.
                                          011F   281 ;
                                 52   D4   011F   282 100$:    CLRL    R2                      ;Clear high part of dividend.
                52   51   51   0A   7B   0121   283         EDIV    #10, R1, R1, R2         ;Isolate next digit.
                     7E   52   30   C1   0126   284         ADDL3   #^A/0/, R2, -(SP)       ;Convert digit to ASCII and save it.
                                 51   D5   012A   285         TSTL    R1                      ;Any more digits to convert?
                                 02   13   012C   286         BEQL    130$                    ;Branch if no more digits.
                                 EF   10   012E   287         BSBB    100$                    ;Else convert next digit.
                           F892 CA   9F   0130   288 130$:    PUSHAB  WRK_G_BUFFER+WRK_C_CMDBUFSIZ(R10) ;Is the expansion buffer
```

```
                F486 CA    8E    D1  0134   289          CMPL    (SP)+, WRK_L_EXPANDPTR(R10) ;full?
                           DC    1B  0139   290          BLEQU   90$                         ;Branch if expansion buffer full.
                           51  8ED0  013B   291          POPL    R1                          ;Get next character digit.
                F486 DA    51    90  013E   292          MOVB    R1, @WRK_L_EXPANDPTR(R10) ;Put it in the expansion buffer.
                     F486 CA      D6  0143   293          INCL    WRK_L_EXPANDPTR(R10)    ;Increment expansion buffer pointer.
                                  05  0147   294          RSB
                                      0148   295
                                      0148   296 NOSUCHSYM:
     50    00038140 8F              D0  0148   297          MOVL    #CLI$_UNDSYM, R0        ;Signal symbol not found and
                                    04  014F   298          RET                             ;return error to caller.
                                      0150   299
                                      0150   300 ;
                                      0150   301 ; Delete a local/global symbol.
                                      0150   302 ;
                                      0150   303
                                      0150   304          .ENABL  LSB
                                      0150   305 DELELCL:
                50    38 AB          7E  0150   306          MOVAQ   PRC_Q_LOCAL(R11), R0    ;Setup address of the
                           04         11  0154   307          BRB     10$                     ;proper symbol table.
                                      0156   308 DELEGBL:
                50    28 AB          7E  0156   309          MOVAQ   PRC_Q_GLOBAL(R11), R0
                51    04 A9          7D  015A   310 10$:      MOVQ    4(R9), R1               ;Get symbol name.
                     51    51        3C  015E   311          MOVZWL  R1,R1                   ;Get low order length
                                      0161   312          IFNORD  R1,(R2),ACCVIO2         ;Error if cannot read it
                        FE96'         30  0167   313          BSBW    DCL$SEARCHT             ;Search for the symbol.
                     DB 50            E9  016A   314          BLBC    R0, NOSUCHSYM           ;Branch if symbol not found.
                01   0A A3            91  016D   315          CMPB    SYM_B_TYPE(R3), -       ;Check for a permanent
                                      0171   316                  #SYM_K_PERM             ;symbol (can't delete them).
                           0B         13  0171   317          BEQL    80$                     ;Branch if permanent symbol.
                                      0173   318          DISABLE                         ;Protect from CTRL/Y ASTs.
                        FE84'         30  0179   319          BSBW    DCL$DEALLOCSYM          ;Delete the symbol.
                                      017C   320          ENABLE                          ;Restore CTRL/Y ASTs.
                     50    01         D0  017E   321 80$:      MOVL    #1, R0                  ;Return a successful status
                                    04  0181   322          RET                             ;to the caller.
                                      0182   323          .DSABL  LSB
                                      0182   324
                                      0182   325 ACCVIO2:
                           FF29       31  0182   326          BRW     ACCVIO                  ;SIGNAL ACCESS VIOLATION
                                      0185   327 ;
                                      0185   328 ; ENABLE OR DISABLE PROCESSING OF CONTROL Y OR OUT-OF-BAND AST'S
                                      0185   329 ;
                                      0185   330
                                      0185   331 DISACTRLY:
51   00B4 CB    02000000 8F           CB  0185   332          BICL3   #PRC_M_CTRLY,PRC_L_OUTOFBAND(R11),R1 ;Disable CTRL/Y.
                           069F       30  018F   333          BSBW    DCL$RESETOOB
                             29       11  0192   334          BRB     NORM_EXIT
                                      0194   335 ENABCTRLY:
51   00B4 CB    02000000 8F           C9  0194   336          BISL3   #PRC_M_CTRLY,PRC_L_OUTOFBAND(R11),R1 ;Re-enable CTRL/Y.
                           0690       30  019E   337          BSBW    DCL$RESETOOB
                             1A       11  01A1   338          BRB     NORM_EXIT
                                      01A3   339
                                      01A3   340 DISAOOB:                                 ;Disable out-of-band character(s).
                             1C       10  01A3   341          BSBB    CHECKMASK               ;Check for legal out-of-band mask.
     51   00B4 CB    04 A9           CB  01A5   342          BICL3   4(R9),PRC_L_OUTOFBAND(R11),R1   ;Set mask for reset routine
                           0682       30  01AC   343          BSBW    DCL$RESETOOB            ;Disable appropriate oob AST's
                             0C       11  01AF   344          BRB     NORM_EXIT
                                      01B1   345
```

```
                              01B1  ENABOOB:                                             ;Re-enable out-of-band character(s).
                   0E   10    01B1             BSBB    CHECKMASK                         ;Check for legal out-of-band mask.
51    00B4 CB    04 A9  C9    01B3             BISL3   4(R9),PRC_L_OUTOFBAND(R11),R1     ;Set mask for reset routine
                 0674   30    01BA             BSBW    DCL$RESETOOB                      ;Disable appropriate oob AST's
                              01BD
                              01BD
                              01BD  NORM_EXIT:
           50      01   D0    01BD             MOVL    #1,R0                             ;SET SUCCESS
                              01C0  ERR_EXIT:
                        04    01C0             RET
                              01C1
                              01C1  CHECKMASK:
   08 A9   00B4 CB      D0    01C1             MOVL    PRC_L_OUTOFBAND(R11), -           ;Return current out-of-band
                              01C7                     8(R9)                             ;character(s) enable bits.
04 A9    FDEFFFFF 8F    D3    01C7             BITL    #^C< -                            ;Check for any illegal bits set.
                              01CF                     PRC_M_CTRLT ! -
                              01CF                     PRC_M_CTRLY   -
                              01CF                     >, 4(R9)
                   01   12    01CF             BNEQ    10$                               ;If no illegal bits are set, return
                        05    01D1             RSB                                       ;to caller and finish processing.
   50    000388CA 8F    D0    01D2  10$:       MOVL    #CLI$_BADCTLMSK, R0               ;Otherwise, quit right now returning
                        04    01D9             RET                                       ;an appropriate error status.
                              01DA
                              01DA  ;
                              01DA  ; ACCEPT IMAGE NAME OR COMMAND LINE TO BE EXECUTED AFTER
                              01DA  ; CURRENT IMAGE COMPLETES
                              01DA  ;
                              01DA
                              01DA             .ENABL  LSB
                              01DA  CHAIN:                                               ;ACCEPT IMAGE NAME FOR LATER
           55      02   9A    01DA             MOVZBL  #PRC_M_CHAIN,R5                   ;SET THE BIT MASK FOR CHAIN'S
      56   00D8 CB      7E    01DD             MOVAQ   PRC_Q_IMAGENAME(R11),R6           ; AND GET POINTER TO THE DESCRIPTOR
                   08   11    01E2             BRB     10$                               ;GO JOIN THE COMMON CODE
                              01E4
                              01E4  COMMAND:                                             ;ACCEPT COMMAND LINE FOR LATER
           55      01   9A    01E4             MOVZBL  #PRC_M_CMD,R5                     ;SET THE BIT MASK FOR COMMAND LINE'S
      56   00E0 CB      7E    01E7             MOVAQ   PRC_Q_COMMAND(R11),R6             ; AND GET POINTER TO THE DESCRIPTOR
                              01EC  10$:       IFNORD  8(R9),@12(R9),ACCVIO2             ;ERROR IF CANNOT READ THE STRING
     00AF CB       55   8A    01F4             BICB    R5,PRC_B_FLAGS2(R11)              ;TURN THE FEATURE OFF INITIALLY
   02 A6   08 A9        B0    01F9             MOVW    8(R9),2(R6)                       ;SET NEW SIZE FROM CALLING DESC
                 01F8   30    01FE             BSBW    DCL$ALLDEACMD                     ;GO DEALLOCATE/RE-ALLOCATE SPACE
              BC   50   E9    0201             BLBC    R0,ERR_EXIT                       ;EXIT NOW IF FAILURE...
                   51   D5    0204             TSTL    R1                                ;ANY NEW SIZE?
                   B5   13    0206             BEQL    NORM_EXIT                         ;NOPE, GO SET SUCCESS AND EXIT
     00AF CB       55   88    0208             BISB    R5,PRC_B_FLAGS2(R11)              ;YEP, SO TURN (BACK) ON THE FEATURE
           66      51   7D    020D             MOVQ    R1,(R6)                           ;LOAD DESCRIPTOR
        03 55      01   E1    0210             BBC     #PRC_V_CHAIN,R5,20$               ;IF IMAGE CHAINING, APPEND $ TO IT
           82      24   90    0214             MOVB    #^A'$',(R2)+                      ;SO THAT IT CAN BE TREATED AS FOREIGN
62    0C B9      08 A9  28    0217  20$:       MOVC    8(R9),@12(R9),(R2)                ;MOVE IN THE STRING
                   63   94    021D             CLRB    (R3)                              ;AND ENSURE IT'S TERMINATED
                   9C   11    021F             BRB     NORM_EXIT                         ;SET SUCCESS AND EXIT
                              0221             .DSABL  LSB
                              0221
                              0221  ;
                              0221  ; DEFINE OR DEASSIGN A SUPERVISOR MODE LOGICAL NAME
                              0221  ;
                              0221  CREALOG:                                             ;CREATE A PROCESS LOGICAL NAME
```

```
          7E  D4  0221           CLRL    -(SP)                          ;ALLOCATE ATTRIBUTE LONGWORD
     58   5E  D0  0223           MOVL    SP,R8                          ;SAVE ADDRESS OF STACK
                  0226
  57   1C A9  D0  0226           MOVL    CLI$L_ITMLST(R9),R7            ;ITEM LIST SPECIFIED?
          0D  12  022A           BNEQ    10$                            ;YES, THEN SKIP
          7E  7C  022C           CLRQ    -(SP)                          ;CREATE AN ITEM LIST
  7E   0C A9  7D  022E           MOVQ    CLI$Q_VALDESC(R9),-(SP)        ;SET THE EQUIV NAME DESCR
  02 AE   02  B0  0232           MOVW    #LNM$_STRING,2(SP)             ;SET THE ITEM TYPE
     57   5E  D0  0236           MOVL    SP,R7                          ;SET ADDRESS OF ITEM LIST
                  0239   ;       BISL    #LNM$M_CRELOG,(R8)             ;SET CRELOG ATTRIBUTE
                  0239
  7E   14 A9  7D  0239   10$:    MOVQ    CLI$Q_TABDESC(R9),-(SP)        ;SAVE THE TABLE NAME
          0D  12  023D           BNEQ    19$                            ;BRANCH IF SPECIFIED
  51 FDBD CF  9E  023F           MOVAB   LNM$PROCESS,R1                 ;SET DEFAULT TABLE NAME
     50   81  9A  0244           MOVZBL  (R1)+,R0                       ;
     6E   50  7D  0247           MOVQ    R0,(SP)                        ;
          00  11  024A           BRB     20$                            ;
                  024C   19$:
                  024C   ;       BICL    #LNM$M_CRELOG,(R8)             ;CLEAR CRELOG ATTRIBUTE
                  024C
       20 A9  D5  024C   20$:    TSTL    CLI$L_ATTR(R9)                 ;WERE ATTRIBUTES SPECIFIED
          0B  13  024F           BEQL    30$                            ;NO, THEN BRANCH
                  0251           IFNORD  #4,@CLI$L_ATTR(R9),ACCVIO3     ;CHECK ACCESS TO ATTRIBUTES
  68   20 B9  D0  0258           MOVL    @CLI$L_ATTR(R9),(R8)           ;USE THOSE ATTRIBUTES
                  025C
          02  DD  025C   30$:    PUSHL   #PSL$C_SUPER                   ;SET ACCESS MODE
     51   5E  D0  025E           MOVL    SP,R1                          ;SET ADDRESS OF DATA
                  0261
                  0261           $CRELNM_S  LOGNAM=CLI$Q_NAMDESC(R9),-  ;CREATE THE REQUESTED NAME
                  0261                   ACMODE=(R1),-                  ;
                  0261                   TABNAM=4(R1),-                 ;
                  0261                   ITMLST=(R7),-                  ;
                  0261                   ATTR=(R8)                      ;
                  0274
  5E   04 A8  9E  0274           MOVAB   4(R8),SP                       ;POP THE ITEM LIST
          04      0278           RET                                    ;RETURN STATUS OF SERVICE DIRECTLY
                  0279
                  0279   DELELOG:                                       ;DELETE PROCESS LOGICAL NAME
  7E   14 A9  7D  0279   10$:    MOVQ    CLI$Q_TABDESC(R9),-(SP)        ;SAVE THE TABLE NAME
          0B  12  027D           BNEQ    20$                            ;BRANCH IF SPECIFIED
  51 FD7D CF  9E  027F           MOVAB   LNM$PROCESS,R1                 ;SET DEFAULT TABLE NAME
     50   81  9A  0284           MOVZBL  (R1)+,R0                       ;
     6E   50  7D  0287           MOVQ    R0,(SP)                        ;
          02  DD  028A   20$:    PUSHL   #PSL$C_SUPER                   ;SET ACCESS MODE
     51   5E  D0  028C           MOVL    SP,R1                          ;SET ADDRESS OF DATA
  52   04 A9  7E  028F           MOVAQ   CLI$Q_NAMDESC(R9),R2           ;GET ADDRESS OF LOG NAM DESCR
          62  D5  0293           TSTL    (R2)                           ;IS LENGTH ZERO?
          07  12  0295           BNEQ    30$                            ;NO, THEN BRANCH
       04 A2  D5  0297           TSTL    4(R2)                          ;IS ADDRESS ZERO?
          02  12  029A           BNEQ    30$                            ;NO, THEN BRANCH
          52  D4  029C           CLRL    R2                             ;DEASSIGN/ALL
                  029E
                  029E   30$:    $DELLNM_S  TABNAM=4(R1),-              ;DEASSIGN LOGICAL NAME EQUIVALENCE
                  029E                   LOGNAM=(R2),-                  ;
                  029E                   ACMODE=(R1)                    ;
                  02AC
     5E   0C  C0  02AC           ADDL    #3*4,SP                        ;RESTORE THE STATCK
```

```
                           04  02AF  RET:     RET                                          ;RETURN STATUS OF SERVICE DIRECTLY
                               02B0
                               02B0  ACCVIO3:
                    FDFB  31   02B0           BRW     ACCVIO                      ;REPORT ACCESS VIOLATION
                               02B3
                               02B3  ;
                               02B3  ; SPAWN A SUBPROCESS
                               02B3  ;
                               02B3           .ENABL  LSB
           51    00D6 8F  3C   02B3  SPAWN:   MOVZWL  #SPWN_C_LENGTH,R1                    ;SET LENGTH TO ALLOCATE
                   FD45'  30   02B8           BSBW    DCL$ALLDYNMEM                        ;ALLOCATE STORAGE FOR SPAWN BLOCK
                   F1 50  E9   02BB           BLBC    R0,RET                               ;BRANCH IF ERROR DETECTED
                   56 52  D0   02BE           MOVL    R2,R6                                ;POINT TO BLOCK
  66  00D6 8F  00  61 00  2C   02C1           MOVC5   #0,(R1),#0,#SPWN_C_LENGTH,(R6)       ;ZERO THE BLOCK (WITHOUT DESTROYING
                04 A6 51  B0   02C9           MOVW    R1,SPWN_W_SIZE(R6)                   ;SET SIZE OF BLOCK
                               02CD
                   10 A9  B5   02CD           TSTW    CLI$Q_CMDSTR(R9)                     ;COMMAND STRING PRESENT?
                      10  13   02D0           BEQL    10$                                  ;BRANCH IF NOT
                               02D2           IFNORD  CLI$Q_CMDSTR(R9),-                   ;CHECK ACCESS TO COMMAND STRING
                               02D2                   @CLI$Q_CMDSTR+4(R9),ACCVIO3          ;
                   10 A9  7D   02DA           MOVQ    CLI$Q_CMDSTR(R9),-                   ;PASS CMDSTR DESCRIPTOR
                   30 A6       02DD                   SPWN_Q_CMDSTR(R6)                    ;
                   32 A6  B4   02DF           CLRW    SPWN_Q_CMDSTR+2(R6)                  ;
                               02E2
                   18 A9  B5   02E2  10$:     TSTW    CLI$Q_INPUT(R9)                      ;INPUT FILESPEC PRESENT?
                      15  13   02E5           BEQL    20$                                  ;BRANCH IF NOT
                               02E7           SETBIT  #SPWN_V_INPUT,SPWN_W_FLAGS(R6)       ;INDICATE INPUT SPECIFIED
                               02EC           IFNORD  CLI$Q_INPUT(R9),-                    ;CHECK ACCESS TO INPUT STRING
                               02EC                   @CLI$Q_INPUT+4(R9),ACCVIO3           ;
           20 A6   18 A9  7D   02F4           MOVQ    CLI$Q_INPUT(R9),SPWN_Q_INPUT(R6);PASS INPUT FILESPEC
                   22 A6  B4   02F9           CLRW    SPWN_Q_INPUT+2(R6)                   ;
                               02FC
                   20 A9  B5   02FC  20$:     TSTW    CLI$Q_OUTPUT(R9)                     ;OUTPUT FILESPEC PRESENT?
                      15  13   02FF           BEQL    30$                                  ;BRANCH IF NOT
                               0301           SETBIT  #SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)      ;INDICATE OUTPUT SPECIFIED
                               0306           IFNORD  CLI$Q_OUTPUT(R9),-                   ;CHECK ACCESS TO OUTPUT STRING
                               0306                   @CLI$Q_OUTPUT+4(R9),ACCVIO3          ;
                   20 A9  7D   030E           MOVQ    CLI$Q_OUTPUT(R9),-                   ;PASS OUTPUT FILESPEC
                   28 A6       0311                   SPWN_Q_OUTPUT(R6)
                   2A A6  B4   0313           CLRW    SPWN_Q_OUTPUT+2(R6)                  ;
                               0316
                   28 A9  B5   0316  30$:     TSTW    CLI$Q_PRCNAM(R9)                     ;PROCESS NAME PRESENT?
                      15  13   0319           BEQL    40$                                  ;BRANCH IF NOT
                               031B           SETBIT  #SPWN_V_PRCNAM,SPWN_W_FLAGS(R6)      ;INDICATE PRCNAM SPECIFIED
                               0320           IFNORD  CLI$Q_PRCNAM(R9),-                   ;CHECK ACCESS TO PROCESS NAME
                               0320                   @CLI$Q_PRCNAM+4(R9),ACCVIO3          ;
                   28 A9  7D   0328           MOVQ    CLI$Q_PRCNAM(R9),-                   ;PASS PROCESS NAME
                   18 A6       032B                   SPWN_Q_PRCNAM(R6)
                   1A A6  B4   032D           CLRW    SPWN_Q_PRCNAM+2(R6)                  ;
                               0330
                   39 A9  95   0330  40$:     TSTB    CLI$B_VERSION(R9)                    ;IF VERSION 0
                      5D  13   0333           BEQL    70$                                  ;THEN SKIP
                   3C A9  B5   0335           TSTW    CLI$Q_PROMPT(R9)                     ;PROMPT STRING PRESENT?
                      1B  13   0338           BEQL    50$                                  ;BRANCH IF NOT
                               033A           SETBIT  #SPWN_V_PROMPT,SPWN_W_FLAGS(R6)      ;INDICATE PROMPT SPECIFIED
                               033F           IFNORD  CLI$Q_PROMPT(R9),-                   ;CHECK ACCESS TO PROMPT
                               033F                   @CLI$Q_PROMPT+4(R9),ACCVIO4          ;
```

```
                3C A9    03    81   0347   517          ADDB3   #3,CLI$Q_PROMPT(R9),-                   ;GET PROMPT
                    00A2 C6          034B   518                  SPWN_B_PROMPTLEN(R6)                    ;
                         20    28   034E   519          MOVC3   #ENT_K_MAX_PROMPT,-                     ;
                      40 B9          0350   520                  @CLI$Q_PROMPT+4(R9),-                   ;
                    00A6 C6          0352   521                  SPWN_G_PROMPT(R6)                       ;
                                     0355   522
                      44 A9    B5    0355   523 50$:     TSTW    CLI$Q_CLI(R9)                           ;CLI PRESENT?
                         17    13    0358   524          BEQL    60$                                     ;BRANCH IF NOT
                                     035A   525          SETBIT  #SPWN_V_CLI,SPWN_W_FLAGS(R6)            ;INDICATE CLI SPECIFIED
                                     035F   526          IFNORD  CLI$Q_CLI(R9),-                         ;CHECK ACCESS TO CLI STRING
                                     035F   527                  @CLI$Q_CLI+4(R9),ACCVIO4                ;
                00C6 C6    44 A9    7D    0367   528          MOVQ    CLI$Q_CLI(R9),SPWN_Q_CLI(R6)    ;PASS CLI NAME
                    00C8 C6          B4    036D   529          CLRW    SPWN_Q_CLI+2(R6)                ;
                                     0371   530
                      4C A9    B5    0371   531 60$:     TSTW    CLI$Q_TABLE(R9)                         ;CLI TABLE PRESENT
                         1C    13    0374   532          BEQL    70$                                     ;BRANCH IF NOT
                                     0376   533          SETBIT  #SPWN_V_TABLE,SPWN_W_FLAGS(R6)          ;INDICATE CLI TABLE SPECIFIED
                                     037B   534          IFNORD  CLI$Q_TABLE(R9),-                       ;CHECK ACCESS TO CLI TABLE STRING
                                     037B   535                  @CLI$Q_TABLE(R9),ACCVIO4                ;
                00CE C6    4C A9    7D    0383   536          MOVQ    CLI$Q_TABLE(R9),SPWN_Q_TABLE(R6);PASS CLI TABLE NAME
                    00D0 C6          B4    0389   537          CLRW    SPWN_Q_TABLE+2(R6)              ;
                         03    11    038D   538          BRB     70$                                     ;
                                     038F   539
                                     038F   540 ACCVIO4:
                        FD1C   31    038F   541          BRW     ACCVIO                  ;REPORT ACCESS VIOLATION
                                     0392   542
                05 04 A9    00    E0   0392   543 70$:     BBS     #CLI$V_NOWAIT,CLI$B_FLAGS(R9),71$       ;BRANCH IF FLAG SET
                                     0397   544          SETBIT  #SPWN_V_WAIT,SPWN_W_FLAGS(R6)           ;INDICATE IF WE SHOULD WAIT
                05 04 A9    01    E0   039C   545 71$:     BBS     #CLI$V_NOCLISYM,CLI$B_FLAGS(R9),72$     ;BRANCH IF FLAG SET
                                     03A1   546          SETBIT  #SPWN_V_CLISYM,SPWN_W_FLAGS(R6)         ;INDICATE TO COPY CLI SYMBOL
                05 04 A9    03    E0   03A6   547 72$:     BBS     #CLI$V_NOKEYPAD,CLI$B_FLAGS(R9),73$     ;BRANCH IF FLAG SET
                                     03AB   548          SETBIT  #SPWN_V_KEYPAD,SPWN_W_FLAGS(R6)         ;INDICATE TO COPY KEYPAD STA
                05 04 A9    02    E0   03B0   549 73$:     BBS     #CLI$V_NOLOGNAM,CLI$B_FLAGS(R9),74$     ;BRANCH IF FLAG SET
                                     03B5   550          SETBIT  #SPWN_V_LOGNAM,SPWN_W_FLAGS(R6)         ;INDICATE TO COPY LOGNAMES
                05 04 A9    04    E1   03BA   551 74$:     BBC     #CLI$V_NOTIFY,CLI$B_FLAGS(R9),75$       ;BRANCH IF FLAG CLEAR
                                     03BF   552          SETBIT  #SPWN_V_NOTIFY,SPWN_W_FLAGS(R6)         ;INDICATE TO NOTIFY
                09 04 A9    05    E0   03C4   553 75$:     BBS     #CLI$V_NOCONTROL,CLI$B_FLAGS(R9),80$    ;BRANCH IF FLAG SET
        00A3 C6    00000000'EF    B0   03C9   554          MOVW    DCL$CRLF,SPWN_W_PMPTCTRL(R6)            ;SET DEFAULT PROMPT CONTROL
                                     03D2   555
                4C A6    30 A9    D0   03D2   556 80$:     MOVL    CLI$L_ASTADR(R9),SPWN_L_ASTADR(R6)      ;COPY AST ADDRESS
                50 A6    34 A9    D0   03D7   557          MOVL    CLI$L_ASTPRM(R9),SPWN_L_ASTPRM(R6)      ;COPY AST PARAMETER
                0F A6    38 A9    90   03DC   558          MOVB    CLI$B_EFN(R9),SPWN_B_EFN(R6)            ;COPY EVENT FLAG #
                54 A6    0C A9    D0   03E1   559          MOVL    CLI$L_LSTSTATUS(R9),SPWN_L_STSADR(R6)   ;RECEIVES FINAL STATUS
                                     03E6   560
                          FC17'  30    03E6   561          BSBW    DCL$SPAWN2                              ;SPAWN THE SUBPROCESS
                08 A9    40 A6    D0   03E9   562          MOVL    SPWN_L_SUBPID(R6),CLI$L_OUTPID(R9)      ;PASS SUBPROCESS PID (IN CAS
                                  04   03EE   563          RET
                                     03EF   564          .DSABL  LSB
                                     03EF   565
                                     03EF   566 ;
                                     03EF   567 ; ATTACH THE TERMINAL TO ANOTHER PROCESS (ESSENTIALLY A CO-ROUTINE CALL)
                                     03EF   568 ;
                                     03EF   569
                             56    D4   03EF   570 ATTACH: CLRL    R6                      ;MARK NO PROCESS NAME SUPPLIED
                    58    04 A9    D0   03F1   571          MOVL    CLI$L_PID(R9),R8        ;GET PID OF DESTINATION PROCESS
                           FC08'  30    03F5   572          BSBW    DCL$ATTACH2             ;ATTACH TO SPECIFIED PROCESS
                                  04   03F8   573          RET
```

```
                        03F9     575          .SBTTL  ALLOCATE CHAIN STRING STORAGE
                        03F9     576 ;+
                        03F9     577 ; DCL$ALLDEACMD - DEALLOCATE/RE-ALLOCATE CHAIN/COMMAND STRING STORAGE
                        03F9     578 ;
                        03F9     579 ; INPUTS:
                        03F9     580 ;
                        03F9     581 ;       R6 -> DESC W/ NEW SIZE @ 2(R6)
                        03F9     582 ;
                        03F9     583 ; OUTPUTS:
                        03F9     584 ;
                        03F9     585 ;       R0 =  STATUS
                        03F9     586 ;       R1 =  NEW SIZE
                        03F9     587 ;       R2 -> NEW BLOCK
                        03F9     588 ;       R3,R4 = UNDEFINED
                        03F9     589 ;-
                        03F9     590 DCL$ALLDEACMD::                             ;DEALLOCATE/RE-ALLOCATE CHAIN/COMMAND
                        03F9     591          DISABLE                            ;DISABLE CONTROL/Y & C AST'S
          51   66   3C  03FF     592          MOVZWL  (R6),R1                    ;GET CURRENT ALLOCATED SIZE
               09   13  0402     593          BEQL    10$                        ;NONE
     50   04 A6   D0  0404     594          MOVL    4(R6),R0                   ;SOME, GET POINTER TO BLOCK TO RETURN
            FBF5'  30  0408     595          BSBW    DCL$DEADYNMEM              ; AND GO RETURN IT
               66   B4  040B     596          CLRW    (R6)                       ;  THEN SAY IT'S NOW NULL
     50   01   D0  040D     597 10$:     MOVL    #1,R0                      ;PRESET SUCCESS STATUS
          51   02 A6   3C  0410     598          MOVZWL  2(R6),R1                   ;GET NEW DESCRIPTOR'S SIZE
               16   13  0414     599          BEQL    20$                        ;ZERO LENGTH, JUST EXIT STATUS=SUCCESS
               66   D4  0416     600          CLRL    (R6)                       ;REAL LENGTH, BUT DON'T KEEP SAYING SO
50   0003883A 8F   D0  0418     601          MOVL    #CLI$_ILLVAL,R0            ;PRE-SET ERROR CODE
     0100 8F   51   B1  041F     602          CMPW    R1,#WRK_C_INPBUFSIZ        ;DOES TEXT FIT WITH ROOM TO SPARE?
               06   1A  0424     603          BGTRU   20$                        ;BRANCH IF NOT
          51   02   C0  0426     604          ADDL    #2,R1                      ;ADD IN ROOM FOR '$' + TRAILING EOL
            FBD4'  30  0429     605          BSBW    DCL$ALLDYNMEM              ;GET THE DYNAMIC MEMORY SPACE
                       042C     606 20$:     ENABLE                             ;ENABLE CONTROL/Y&C
                   05  042E     607          RSB                                ;EXIT
```

```
                                        .SBTTL  CONTROL Y AST HANDLER
                                ;+
                                ; DCL$CONTRLY - CONTROL Y AST HANDLER
                                ;
                                ; THIS ROUTINE IS CALLED WHEN A CONTROL Y AST OCCURS WHILE RUNNING IN USER
                                ; OR SUPERVISOR MODE.
                                ;
                                ; INPUTS:
                                ;
                                ;       AP = ADDRESS OF AST ARGUMENT LIST.
                                ;
                                ; OUTPUTS:
                                ;
                                ;       THE CONTROL Y AST IS RE-ENABLED AND A CHECK IS MADE TO SEE IF THE
                                ;       PREVIOUS MODE WAS USER OR SUPERVISOR.
                                ;
                                ;       PREVIOUS MODE USER:
                                ;
                                ;               A COMMAND WORK AREA IS ALLOCATED ON THE STACK, THE PROCESS
                                ;               SAVED ARGUMENT AND FRAME POINTERS ARE MOVED TO THE COMMAND
                                ;               WORK AREA, THE CURRENT ARGUMENT AND FRAME POINTERS ARE SAVED
                                ;               IN THE PROCESS SAVE AREA, AST'S ARE ENABLED, AND THE COMMAND
                                ;               INTERPRETER RESTART POINT IS JUMPED TO.
                                ;
                                ;       PREVIOUS MODE SUPERVISOR:
                                ;
                                ;               IF CONTROL Y AST'S ARE CURRENTLY SOFTWARE DISABLED, THEN THE
                                ;               AST IS DISMISSED IMMEDIATELY. OTHERWISE THE SAVED PROCESS
                                ;               ARGUMENT AND FRAME POINTERS ARE RESTORED, AST'S ARE ENABLED,
                                ;               AND THE COMMAND INTERPRETER RESTART POINT IS JUMPED TO.
                                ;-
                                        .ENABL  LSB
                                ;
                        OFFC    ;       .ENTRY  DCL$CONTRLY,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                ;
                FBCC'   30      ;       BSBW    CLI$GET_PRC             ;GET ADDRESS OF CLI PROCESS WORK AREA
00C4 CB   04 AC   B0    ;       MOVW    4(AP),PRC_W_ASTSTATUS(R11)  ;SAVE AST STATUS
04 AC   0000'8F   B1    ;       CMPW    #SS$_HANGUP,4(AP)       ;TERMINAL LINE HANGUP?
          07    12      ;       BNEQ    10$                     ;IF NEQ NO
                        ;       SETBIT  PRC_V_HANGUP,PRC_W_FLAGS(R11) ;SET HANGUP PENDING
          09    11      ;       BRB     15$                     ;NO MORE CONTROL Y'S ALLOWED
        030B    30      ;10$:   BSBW    DCL$ENBCONTRLY          ;RE-ENABLE CONTROL Y AST
3F 00B4 CB    19    E1  ;       BBC     #PRC_V_CTRLY,PRC_L_OUTOFBAND(R11),35$ ;BR IF NOT ALLOWED
                        ;15$:   $SETEF_S  EFN=#31               ;TERMINATE CURRENT WAIT COMMAND
    66 AB     01    B0  ;       MOVW    #1,PRC_W_WAITIOSB(R11)  ;
 58 14 AC     18    E0  ;       BBS     #PSL$V_CURMOD,20(AP),60$ ;IF SET, PREVIOUS MODE USER
        00B8 CB     D5  ;       TSTL    PRC_L_ONCTLY(R11)       ;USER DEFINED ACTION
          23    12      ;       BNEQ    30$                     ;BR IF YES - EXCUTE THE COMMAND
      0000'CF   9F      ;       PUSHAB  W^DCL$LOW_LIMIT         ;GET ADDRESS OF LOWER ADDRESS LIMIT
    10 AC       8E    D1  ;       CMPL    (SP)+,16(AP)            ;ADDRESS WITHIN LIMITS?
          0A    1A      ;       BGTRU   20$                     ;IF GTRU NO
      0000'CF   9F      ;       PUSHAB  W^DCL$HIGH_LIMIT        ;GET ADDRESS OF HIGH ADDRESS LIMIT
    10 AC       8E    D1  ;       CMPL    (SP)+,16(AP)            ;ADDRESS WITHIN LIMITS?
          1D    1A      ;       BGTRU   50$                     ;IF GTRU YES
 0A 68 AB     02    E0  ;20$:   BBS     #PRC_V_DISABL,PRC_W_FLAGS(R11),30$ ;IF SET, CONTROL Y/C AST'S DISABL
 0A 68 AB     0B    E0  ;       BBS     #PRC_V_YLEVEL,PRC_W_FLAGS(R11),40$ ;IF SET, AT CONTROL Y/C LEVEL
       5C AB    D5      ;       TSTL    PRC_L_INDEPTH(R11)      ;INDIRECT LEVEL ZERO?
```

```
                 05     13  048B        BEQL    40$                     ;IF EQL YES
       68 AB     02     A8  048D 30$:   BISW    #PRC_M_CNTRLY,PRC_W_FLAGS(R11) ;SET CONTROL Y/C REQUEST
                        04  0491 35$:   RET                             ;
                            0492
                            0492 ;
                            0492 ; PREVIOUS MODE SUPERVISOR
                            0492 ;
                            0492
       5D    04 AB      D0  0492 40$:   MOVL    PRC_L_SAVFP(R11),FP     ;RESTORE SAVED FRAME POINTER
             5A  5D     D0  0496        MOVL    FP,R10                  ;SET ADDRESS OF WRK AREA
                 33     11  0499        BRB     70$                     ;
                            049B
                            049B ;
                            049B ; WE HAVE DETECTED A CONTROL/Y WHILE ACTIVATING AN IMAGE BUT BEFORE
                            049B ; THE IMAGE WAS ACTUALLY STARTED IN USER MODE.
                            049B ;
                            049B ; CREATE DUMMY CONTROL Y/C AST FRAME WHICH CAN EVENTUALLY BE PLUGGED
                            049B ; WITH A MODIFIED R0 AND PC/PSL (EXE$EXIT_IMAGE) BY IMAGE RUNDOWN.
                            049B ;
                            049B
       5D    04 AB      D0  049B 50$:   MOVL    PRC_L_SAVFP(R11),FP     ;RESTORE SAVED FRAME POINTER
             5A  5D     D0  049F        MOVL    FP,R10                  ;SET ADDRESS OF WRK AREA
  56   F4 AA     18     C3  04A2        SUBL3   #6*4,WRK_L_SAVSP(R10),R6 ;ALLOCATE DUMMY AST ARGUMENT LIST
     66   6C     18     28  04A7        MOVC3   #6*4,(AP),(R6)          ;MOVE REAL LIST INTO ALLOCATED SPACE
          5E     56     D0  04AB        MOVL    R6,SP                   ;RESET STACK POINTER
                            04AE
                            04AE ;
                            04AE ; ASSUME DUMMY CONTROL Y/C AST FRAME IS ON TOP OF STACK.
                            04AE ;
                            04AE
                            04AE DCL$SCNTRLY::                          ;SUPERVISOR CONTROL Y/C
             5C  5E     D0  04AE        MOVL    SP,AP                   ;SET ARGUMENT POINTER
              F5'AF     9F  04B1        PUSHAB  B^80$                   ;SET RETURN ADDRESS
             7E  5C     7D  04B4        MOVQ    AP,-(SP)                ;SAVE ARGUMENT AND FRAME POINTERS
                 7E     7C  04B7        CLRQ    -(SP)                   ;CLEAR PSW, MASK, AND HANDLER ADDRESS
             5D  5E     D0  04B9        MOVL    SP,FP                   ;SET NEW FRAME POINTER
                            04BC
                            04BC ;
                            04BC ; PREVIOUS MODE USER
                            04BC ;
                            04BC
       5E   F486 CD     9E  04BC 60$:   MOVAB   WRK_K_LENGTH(FP),SP     ;ALLOCATE COMMAND WORK AREA
             5A  5D     D0  04C1        MOVL    FP,R10                  ;SET ADDRESS OF WRK AREA
       F8 AA     6B     7D  04C4        MOVQ    PRC_L_SAVAP(R11),WRK_L_SAVAP(R10) ;SAVE ARGUMENT AND FRAME POINTERS
             5C  14     C0  04C8        ADDL    #20,AP                  ;POINT TO SAVED PSL
             6B  5C     7D  04CB        MOVQ    AP,PRC_L_SAVAP(R11)     ;SAVE CURRENT ARGUMENT AND FRAME POINTERS
  68 AB  0800 8F        A8  04CE 70$:   BISW    #PRC_M_YLEVEL,PRC_W_FLAGS(R11) ;SET CONTROL Y/C LEVEL
                 00     BC  04D4        CHMK    #0                      ;ENABLE AST'S
                            04D6        SETBIT  WRK_V_COMMAND,WRK_W_FLAGS(R10) ;SET COMMAND IN EXECUTION
          00B8 CB       D5  04DA        TSTL    PRC_L_ONCTLY(R11)       ;USER DEFINED ACTION?
                 0B     12  04DE        BNEQ    72$                     ;BRANCH IF YES
  0C 00AF CB     04     E1  04E0        BBC     #PRC_V_PRIV,PRC_B_FLAGS2(R11),75$ ;BRANCH IF NOT PRIVILEGED IMAGE
                            04E6 ;
                            04E6 ; SAVE THE IMAGE PRIVILEGES FOR THE CONTINUE COMMAND TO RESTORE
                            04E6 ; SET THE IMAGE PRIVILEGES TO THE PROCESS PRIVILEGES
                            04E6 ;
             FB17'      30  04E6        BSBW    DCL$SAVE_PRIVS          ;
```

```
                07   11   04E9    723                BRB     75$
             FB12'   30   04EB    724 72$:           BSBW    DCL$RUNDWNI                     ;RUNDOWN BUT PRESERVE INDIRECT LEVELS
          68 AB   02   A8   04EE    725                BISW    #PRC_M_CNTRLY,PRC_W_FLAGS(R11) ;SET CONTROL Y/C REQUEST
             FB0B'   31   04F2    726 75$:           BRW     DCL$RESTART                     ;
                          04F5    727
                          04F5    728 ;
                          04F5    729 ; CONTINUE AFTER SIMULATED CONTROL Y/C AST FROM USER MODE
                          04F5    730 ;
                          04F5    731
             5E   08   C0   04F5    732 80$:           ADDL    #8,SP                           ;REMOVE DUMMY AST COUNT AND ASTPRM
                03   BA   04F8    733                POPR    #^M<R0,R1>                      ;RESTORE SAVED R0 AND R1 (PLUGGED)
                     02   04FA    734                REI                                     ;RETURN TO EXE$EXIT_IMAGE (PLUGGED)
                          04FB    735                .DSABL  LSB
```

```
                     04FB   737          .SBTTL  CONTROL T AST HANDLER
                     04FB   738 ;+
                     04FB   739 ; DCL$CONTRLT - CONTROL T AST HANDLER
                     04FB   740 ;
                     04FB   741 ; THIS ROUTINE IS CALLED WHEN A CONTROL T AST OCCURS.
                     04FB   742 ;
                     04FB   743 ; INPUTS:
                     04FB   744 ;
                     04FB   745 ;       AP = ADDRESS OF AST ARGUMENT LIST.
                     04FB   746 ;
                     04FB   747 ; OUTPUTS:
                     04FB   748 ;
                     04FB   749 ;       THE CONTROL T AST IS AUTOMATICALLY RE-ENABLED AND A LINE OF PROCESS
                     04FB   750 ;       STATUS INFORMATION IS OUTPUT.
                     04FB   751 ;-
           00000000  04FB   752 CTRLT_ARGS = 0
                     04FB   753
                     04FB   754 .MACRO  CTRLT   NAME,LENGTH=4
                     04FB   755 .WORD   LENGTH
                     04FB   756 .WORD   JPI$_'NAME
                     04FB   757 CTRLT_ARGS = CTRLT_ARGS+1
                     04FB   758 ITEM_'NAME' = 12 * <8-CTRLT_ARGS>
                     04FB   759 BUFF_'NAME' = -4 * CTRLT_ARGS
                     04FB   760 .ENDM
                     04FB   761
                     04FB   762 CTRLT_TABLE:
                     04FB   763          CTRLT   PAGEFLTS
                     04FF   764          CTRLT   GPGCNT
                     0503   765          CTRLT   PPGCNT
                     0507   766          CTRLT   CPUTIM
                     050B   767          CTRLT   DIRIO
                     050F   768          CTRLT   BUFIO
                     0513   769          CTRLT   PRCNAM,16
                     0517   770          CTRLT   IMAGNAME,64
                     051B   771
                     051B   772 CTRLTMSG:
21 20 53 41 21 20 53 41 21 53 41 21  051B   773          .ASCII  &!AS!AS !AS !9AS CPU=!%T PF=!UL IO=!UL MEM=!UL&
20 54 25 21 3D 55 50 43 20 53 41 39  0527
55 21 3D 4F 49 20 4C 55 21 3D 46 50  0533
          4C 55 21 3D 4D 45 4D 20 4C  053F
                     0548   774 CTRLTMSGEND:
          20 20 29 4C 43 44 28 20 20 0548   775 DCL:     .ASCII  /  (DCL)  /
                     0551   776 DCLEND:
                     0551   777
                     0551   778 LNM$SYSTEM_TABLE:
5F 4D 45 54 53 59 53 24 4D 4E 4C 00' 0551   779          .ASCIC  /LNM$SYSTEM_TABLE/
                        45 4C 42 41 54  055D
                                    10  0551
                     0562   780
                     0562   781 SYS$NODE:
          45 44 4F 4E 24 53 59 53 00' 0562   782          .ASCIC  /SYS$NODE/
                                    08  0562
                     056B   783
                     056B   784
                     056B   785          .ENTRY  DCL$CONTRLT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     056D   786
                  5B   5E   D0  056D   787          MOVL    SP,R11                          ;SAVE STACK POINTER
```

```
                     0570   788
                     0570   789 ;
                     0570   790 ; CHECK TRANSLATION OF SYS$NODE FIRST.
                     0570   791 ;
                     0570   792 ; BUILD NECESSARY DESCRIPTORS AND ITEM LISTS
                     0570   793 ;
      5E   12   C2   0570   794          SUBL    #18,SP                          ;SPACE FOR NODE NAME
           7E   D4   0573   795          CLRL    -(SP)                           ;MARK END OF LIST
        F4 AE   9F   0575   796          PUSHAB  -12(SP)                         ;SET ADDRESS TO RETURN LENGTH
        0C AE   9F   0578   797          PUSHAB  12(SP)                          ;SET ADDRESS OF BUFFER
00020010 8F   DD   057B   798          PUSHL   #LNM$_STRING@16+16              ;SET ITEM TYPE AND LENGTH
      5A   5E   D0   0581   799          MOVL    SP,R10                          ;SAVE ADDR. OF ITEM LIST
                     0584   800
    7E   CB AF   9E   0584   801          MOVAB   LNM$SYSTEM_TABLE+1,-(SP)        ;GET ADDR. OF TABLE NAME
    7E   C6 AF   9A   0588   802          MOVZBL  LNM$SYSTEM_TABLE,-(SP)          ;GET LENGTH OF TABLE NAME
      55   5E   D0   058C   803          MOVL    SP,R5                           ;FORM A DESCRIPTOR
                     058F   804
    7E   D1 AF   9E   058F   805          MOVAB   SYS$NODE+1,-(SP)                ;FORM A DISCRIPTOR LOGIC. NAME
    7E   CC AF   9A   0593   806          MOVZBL  SYS$NODE,-(SP)                  ;GET LENGTH
      56   5E   D0   0597   807          MOVL    SP,R6                           ;FORM A DESCRIPTOR
                     059A   808 ;
                     059A   809 ; GET TRANSLATION OF SYS$NODE
                     059A   810 ;
                     059A   811          $TRNLNM_S       TABNAM=(R5),-           ;TABLE NAME ADDR.
                     059A   812                          LOGNAM=(R6),-           ;"SYS$NODE"
                     059A   813                          ITMLST=(R10)            ;ITEM LIST
   50   0000'8F   B1   05AB   814          CMPW    #SS$_NOLOGNAM,R0                ;DID TRANSLATION OCCUR?
           08   13   05B0   815          BEQL    1$                              ;IF EQ, DON'T HAVE TRANS.
        37 50   E9   05B2   816          BLBC    R0,2$                           ;EXIT IF ERROR
                     05B5   817 ;
                     05B5   818 ; WILL USE TRANSLATION OF SYS$NODE FOR NODE NAME
                     05B5   819 ;
      02 AA   B4   05B5   820          CLRW    2(R10)                          ;CLEAN UP DESCRIPTOR
         4A   11   05B8   821          BRB     5$                              ;GET SYSTEM TIME
                     05BA   822 ;
                     05BA   823 ;
                     05BA   824 ; GET NODE NAME WITH $GETSYI.
                     05BA   825 ;
      5E   12   C2   05BA   826 1$:       SUBL    #18,SP                          ;SPACE FOR NODE NAME
           7E   D4   05BD   827          CLRL    -(SP)                           ;MARK END OF LIST
        F4 AE   9F   05BF   828          PUSHAB  -12(SP)                         ;SET ADDRESS TO RETURN LENGTH
        0C AE   9F   05C2   829          PUSHAB  12(SP)                          ;SET ADDRESS OF BUFFER
10D90010 8F   DD   05C5   830          PUSHL   #SYI$_NODENAME@16+16            ;SET ITEM TYPE AND LENGTH
      5A   5E   D0   05CB   831          MOVL    SP,R10                          ;SAVE ADDRESS OF DESCR
           7E   7C   05CE   832          CLRQ    -(SP)                           ;ALLOCATE AN IOSB
      50   5E   D0   05D0   833          MOVL    SP,R0                           ;
                     05D3   834          $GETSYIW_S  ITMLST=(R10),-              ;GET SYSTEM INFO
                     05D3   835                      IOSB=(R0),-                 ;
                     05D3   836                      EFN=#31                     ;
      03 50   E9   05E6   837          BLBC    R0,2$                           ;IF PROBLEM WITH GETJPI, THEN EXIT
      50   6E   3C   05E9   838          MOVZWL  (SP),R0                         ;GET IOSB STATUS
        2F 50   E9   05EC   839 2$:       BLBC    R0,10$                          ;EXIT IF ERROR
      02 AA   B4   05EF   840          CLRW    2(R10)                          ;INIT THE DESCRIPTOR
                     05F2   841
      50   6A   3C   05F2   842          MOVZWL  (R10),R0                        ;NODE NAME PRESENT?
           0D   13   05F5   843          BEQL    5$                              ;NO, DON'T INSERT '::'
                     05F7   844
```

```
                50   04 BA40   9E  05F7   845          MOVAB   @4(R10)[R0],R0                   ;R0 = ADDR. WHERE TO INSERT '::'
                60   3A3A 8F   B0  05FC   846          MOVW    #^A'::',(R0)                    ;INSERT '::'
                     6A   02   A0  0601   847          ADDW    #2,(R10)                        ;ADJUST NODE NAME LENGTH
                                    0604   848 ;
                                    0604   849 ; GET SYSTEM TIME.
                                    0604   850 ;
                          5E   DD   0604   851 5$:      PUSHL   SP                              ;PUSH ADDR OF LEFT OVER IOSB
                     7E   08   9A   0606   852          MOVZBL  #8,-(SP)                        ;PUSH BUFFER LENGTH
                     59   5E   D0   0609   853          MOVL    SP,R9
                                    060C   854          $ASCTIM_S       TIMLEN=(R9),-           ;GET CURRENT TIME
                                    060C   855                          TIMBUF=(R9),-
                                    060C   856                          CVTFLG=#1
                       03 50   E8   061B   857          BLBS    R0,20$                          ;IF PROBLEM, THEN EXIT
                        012B   31   061E   858 10$:     BRW     150$
                                    0621   859
                                    0621   860 ;
                                    0621   861 ; GET JPI INFORMATION.
                                    0621   862 ;
                          7E   D4   0621   863 20$:     CLRL    -(SP)                           ;MARK END OF LIST
                     55   5E   D0   0623   864          MOVL    SP,R5                           ;INIT LIST PTR
           5E  00000060 8F   C2   0626   865          SUBL    #12*CTRLT_ARGS,SP               ;INIT BUFFER PTR
                52   FECA CF   9E   062D   866          MOVAB   CTRLT_TABLE,R2                  ;INIT TABLE PTR
                          53   D4   0632   867          CLRL    R3                              ;INIT ARG COUNT
                                    0634   868
                     75   F4 A5   3E   0634   869 30$:     MOVAW   -12(R5),-(R5)                   ;SET RETURN LEN ADDR
                          50   62   3C   0638   870          MOVZWL  (R2),R0                 ;GET BUFFER LENGTH
                          5E   50   C2   063B   871          SUBL    R0,SP                   ;ALLOCATE BUFFER
                          75   5E   D0   063E   872          MOVL    SP,-(R5)                ;SET BUFFER ADDR
                          75   82   D0   0641   873          MOVL    (R2)+,-(R5)             ;SET LEN AND TYPE
                     EC   53   07   F3   0644   874          AOBLEQ  #CTRLT_ARGS-1,R3,30$    ;LOOP TILL END OF LIST
                                    0648   875
                          7E   7C   0648   876          CLRQ    -(SP)                           ;ALLOCATE AN IOSB
                               50   5E   D0   064A   877          MOVL    SP,R0                   ;
                                    064D   878          $GETJPIW_S  ITMLST=(R5),-               ;GET JOB/PROCESS INFO
                                    064D   879                      IOSB=(R0),-             ;
                                    064D   880                      EFN=#31                 ;
                            03 50   E9   0660   881          BLBC    R0,32$                  ;IF PROBLEM WITH GETJPI, THEN EXIT
                               50   6E   3C   0663   882          MOVZWL  (SP),R0                 ;GET IOSB STATUS
                               5E   08   C0   0666   883 32$:     ADDL    #8,SP                   ;POP IOSB
                                 B2 50   E9   0669   884          BLBC    R0,10$                  ;EXIT IF ERROR
                                    066C   885
7E   00   F0 A5   FFFE7960 8F   7A   066C   886          EMUL    #-100000,BUFF_CPUTIM(R5),#0,-(SP)   ;CONVERT CPUTIME TO 100NS UNITS
                               56   5E   D0   0676   887          MOVL    SP,R6
                     F8 A5   F4 A5   C0   0679   888          ADDL    BUFF_PPGCNT(R5),BUFF_GPGCNT(R5) ;CALCULATE PAGE COUNT
                     E8 A5   EC A5   C0   067E   889          ADDL    BUFF_DIRIO(R5),BUFF_BUFIO(R5)   ;CALCULATE I/O TOTAL
                             0E A5   B4   0683   890          CLRW    ITEM_PRCNAM+2(R5)               ;CLEAR JPI CODE
                          58 0C A5   9E   0686   891          MOVAB   ITEM_PRCNAM(R5),R8              ;STORE ADDRESS OF DESC
                             02 A5   B4   068A   892          CLRW    ITEM_IMAGNAME+2(R5)             ;CLEAR JPI CODE
                               57   65   9E   068D   893          MOVAB   ITEM_IMAGNAME(R5),R7            ;STORE ADDRESS OF DESC
                                    0690   894
                                    0690   895 ;
                                    0690   896 ; IF THE IMAGNAME IS NULL, THEN USE "(DCL)".  OTHERWISE, GET THE NINE
                                    0690   897 ; CHARACTER FILE NAME FROM THE IMAGE NAME.
                                    0690   898 ;
                          67   B5   0690   899          TSTW    (R7)                            ;IS IMAGE NAME NULL?
                          0B   12   0692   900          BNEQ    40$                             ;NO, THEN EXTRACT NAME
                               67   09   9A   0694   901          MOVZBL  #DCLEND-DCL,(R7)                ;INSERT DEFAULT STRING
```

```
04 A7    FEAD CF   9E 0697         MOVAB   DCL,4(R7)
               40   11 069D         BRB     100$
                       069F
         52    67   7D 069F  40$:   MOVQ    (R7),R2                        ;GET LENGTH AND ADDRESS
    63   52    3A   3A 06A2  50$:   LOCC    #^A/:/,R2,(R3)                 ;FIND COLON
               0A   13 06A6         BEQL    60$                            ;BRANCH IF NOT FOUND
    52   50    01   C3 06A8         SUBL3   #1,R0,R2                       ;GET NEW LENGTH
    53   51    01   C1 06AC         ADDL3   #1,R1,R3                       ;GET NEW ADDRESS
               F0   11 06B0         BRB     50$                            ;LOOK FOR ANOTHER COLON
 63   52    5D 8F   3A 06B2  60$:   LOCC    #^A/]/,R2,(R3)                 ;FIND CLOSING BRACKET
               0A   13 06B7         BEQL    65$                            ;BRANCH IF NOT FOUND
    52   50    01   C3 06B9         SUBL3   #1,R0,R2                       ;GET NEW LENGTH
    53   51    01   C1 06BD         ADDL3   #1,R1,R3                       ;GET NEW ADDRESS
               EF   11 06C1         BRB     60$                            ;LOOK FOR ANOTHER ']'
                       06C3
    63   52    3E   3A 06C3  65$:   LOCC    #^A/>/,R2,(R3)                 ;FIND CLOSING BRACKET
               0A   13 06C7         BEQL    80$                            ;BRANCH IF NOT FOUND
    52   50    01   C3 06C9         SUBL3   #1,R0,R2                       ;GET NEW LENGTH
    53   51    01   C1 06CD         ADDL3   #1,R1,R3                       ;GET NEW ADDRESS
               F0   11 06D1         BRB     65$                            ;LOOK FOR ANOTHER '>'
                       06D3
    63   52    2E   3A 06D3  80$:   LOCC    #^A/./,R2,(R3)                 ;FIND PERIOD
               03   13 06D7         BEQL    90$                            ;BRANCH IF NOT FOUND
         52    50   C2 06D9         SUBL    R0,R2                          ;REMOVE FILE TYPE
         67    52   7D 06DC  90$:   MOVQ    R2,(R7)                        ;STORE LENGTH AND ADDRESS
                       06DF
                       06DF
                       06DF  ;
                       06DF  ; CALL FAO TO FORMAT THE MESSAGE.
                       06DF  ;
5E    00000084 8F   C2 06DF  100$:  SUBL    #132,SP                        ;ALLOCATE SPACE FOR FAO RESULT
               5E   DD 06E6         PUSHL   SP                             ;PUSH BUFFER ADDR
      7E   84 8F    9A 06E8         MOVZBL  #132,-(SP)                     ;PUSH BUFFER LENGTH
         52    5E   D0 06EC         MOVL    SP,R2
                       06EF
         FE28 CF    9F 06EF         PUSHAB  CTRLTMSG                       ;ADDRESS OF CTRL STRING
         7E    2D   9A 06F3         MOVZBL  #CTRLTMSGEND-CTRLTMSG,-(SP)    ;LENGTH OF CTRL STRING
         53    5E   D0 06F6         MOVL    SP,R3
                       06F9
                       06F9         $FAO_S  CTRSTR = (R3),-
                       06F9                 OUTLEN = (R2),-
                       06F9                 OUTBUF = (R2),-
                       06F9                 P1 = R10,-                     ;NODE NAME
                       06F9                 P2 = R8,-                      ;PROCESS NAME
                       06F9                 P3 = R9,-                      ;CURRENT TIME
                       06F9                 P4 = R7,-                      ;IMAGE NAME
                       06F9                 P5 = R6,-                      ;CPU TIME
                       06F9                 P6 = BUFF_PAGEFLTS(R5),-       ;PAGE FAULTS
                       06F9                 P7 = BUFF_BUFIO(R5),-          ;I/O TOTAL
                       06F9                 P8 = BUFF_GPGCNT(R5)           ;MEMORY USAGE
            30 50   E9 0719         BLBC    R0,150$                        ;IF PROBLEM, THEN EXIT
                       071C
         50    8E   7D 071C         MOVQ    (SP)+,R0                       ;POP CTRL STRING DESC
                       071F
                       071F  ;
                       071F  ; BROADCAST THE MESSAGE.
                       071F  ;
```

```
 51   00000028'8F   D0  071F         MOVL    #CTL$AG_CLIDATA+PPD$T_INPDVI,R1 ;GET ADDR OF DEVICE NAME
        50    81    9A  0726         MOVZBL  (R1)+,R0                        ;LENGTH OF DEVICE NAME
        7E    50    7D  0729         MOVQ    R0,-(SP)                        ;CREATE DESCRIPTOR
              7E    7C  072C         CLRQ    -(SP)                           ;ALLOCATE AN IOSB
        50    5E    D0  072E         MOVL    SP,R0                           ;
                        0731
                        0731         $BRKTHRUW_S      MSGBUF=(R2),-          ;BROADCAST THE MESSAGE
                        0731                          SENDTO=(R3),-
                        0731                          SNDTYP=#BRK$C_DEVICE,-
                        0731                          REQID=#BRK$C_DCL,-
                        0731                          EFN=#31,-
                        0731                          IOSB=(R0)
                        074C
        5E    5B    D0  074C   150$: MOVL    R11,SP                          ;RESTORE STACK PTR
                        074F         STATUS  NORMAL                          ;SET SUCCESS
                    04  0756         RET
                        0757
```

```
                     0757   977          .SBTTL  ENABLE CONTROL Y AST
                     0757   978 ;+
                     0757   979 ; DCL$ENBCONTRLY - ENABLE CONTROL Y AST
                     0757   980 ;
                     0757   981 ; THIS ROUTINE IS CALLED TO ENABLE CONTROL Y AST'S ON THE INPUT CHANNEL.
                     0757   982 ;
                     0757   983 ; INPUTS:
                     0757   984 ;
                     0757   985 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                     0757   986 ;
                     0757   987 ; OUTPUTS:
                     0757   988 ;
                     0757   989 ;       R0 = FINAL REQUEST STATUS.
                     0757   990 ;-
                     0757   991
                     0757   992 DCL$ENBCONTRLY::                                  ;ENABLE CONTROL Y AST
50 68 AB    06   E0 0757   993          BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),90$ ;IF SET, NOT INTERACTIVE JOB
   50    08 AB   D0 075C   994          MOVL    PRC_L_INPRAB(R11),R0             ;GET ADDRESS OF INPUT RAB
47 18 A0    02   E1 0760   995          BBC     #DEV$V_TRM,RAB$L_CTX(R0),90$     ;IF CLR, 'INPUT' NOT FROM TERMINAL
         7E      7C 0765   996          CLRQ    -(SP)                            ;ALLOCATE IOSB
      50    5E   D0 0767   997          MOVL    SP,R0                            ;
                     076A   998          $QIOW_S EFN=#EXE$C_SYSEFN,-              ;EVENT FLAG
                     076A   999                  IOSB=(R0),-                      ;IOSB
                     076A  1000                  CHAN=PRC_W_INPCHAN(R11),-        ;INPUT CHANNEL
                     076A  1001                  FUNC=#IO$_SETMODE!IO$M_CTRLYAST,- ;FUNCTION CODE
                     076A  1002                  P1=W^DCL$CONTRLY,-               ;AST ROUTINE ADDRESS
                     076A  1003                  P3=#PSL$C_SUPER                  ;ACCESS MODE
                     0790  1004
00C8 CB     50   B0 0790  1005          MOVW    R0,PRC_W_ASTRETN(R11)            ;SAVE RETURN STATUS
00C6 CB     6E   B0 0795  1006          MOVW    (SP),PRC_W_ASTIOSB(R11)          ;SAVE IOSB
         02      11 079A  1007          BRB     67$                              ;CHECK FOR REENABLE ERRORS
         0B      11 079C  1008          BRB     85$                              ;TEMPORARILY SKIP ERROR CHECKING
                     079E  1009
      03 50      E9 079E  1010 67$:     BLBC    R0,70$                           ;SET HANGUP PENDING IF ERROR
      05 6E      E8 07A1  1011          BLBS    (SP),85$                         ;SKIP IF OK
                     07A4  1012 70$:     SETBIT  PRC_V_HANGUP,PRC_W_FLAGS(R11)    ;SET HANGUP PENDING IF ERROR
                     07A9  1013
   5E    08      C0 07A9  1014 85$:     ADDL    #8,SP                            ;POP IOSB
         05      05 07AC  1015 90$:     RSB
```

```
                     07AD  1017          .SBTTL  DISABLE CONTROL Y AST
                     07AD  1018 ;+
                     07AD  1019 ; DCL$DSBCONTRLY - DISABLE CONTROL Y AST
                     07AD  1020 ;
                     07AD  1021 ; THIS ROUTINE IS CALLED TO DISABLE CONTROL Y AST'S ON THE INPUT CHANNEL.
                     07AD  1022 ;
                     07AD  1023 ; INPUTS:
                     07AD  1024 ;
                     07AD  1025 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                     07AD  1026 ;
                     07AD  1027 ; OUTPUTS:
                     07AD  1028 ;
                     07AD  1029 ;       R0 = FINAL REQUEST STATUS.
                     07AD  1030 ;-
                     07AD  1031
                     07AD  1032 DCL$DSBCONTRLY::                                ;DISABLE CONTROL Y AST
35 68 AB    06   E0  07AD  1033          BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),90$ ;IF SET, NOT INTERACTIVE JOB
   50    08 AB   D0  07B2  1034          MOVL    PRC_[_INPRAB(R11),R0             ;GET ADDRESS OF INPUT RAB
2C 18 A0    02   E1  07B6  1035          BBC     #DEV$V_TRM,RAB$L_CTX(R0),90$     ;IF CLR, 'INPUT' NOT FROM TERMINAL
         7E     7C  07BB  1036          CLRQ    -(SP)                            ;ALLOCATE IOSB
      50    5E   D0  07BD  1037          MOVL    SP,R0                            ;
                     07C0  1038          $QIOW_S EFN=#EXE$C_SYSEFN,-              ;EVENT FLAG
                     07C0  1039                  IOSB=(R0),-                      ;IOSB
                     07C0  1040                  CHAN=PRC_W_INPCHAN(R11),-        ;INPUT CHANNEL
                     07C0  1041                  FUNC=#IO$_SETMODE!IO$M_CTRLYAST,- ;FUNCTION CODE
                     07C0  1042                  P1=0,-                           ;AST ROUTINE ADDRESS
                     07C0  1043                  P3=#PSL$C_SUPER                  ;ACCESS MODE
      5E    08   C0  07E4  1044          ADDL    #8,SP                            ;POP IOSB
                  05  07E7  1045 90$:     RSB
```

```
                        07E8  1047          .SBTTL  ENABLE/DISABLE CTRL/T AST'S
                        07E8  1048 ;+
                        07E8  1049 ; DCL$ENBCONTRLT - ENABLE/DISABLE CTRL/T AST'S
                        07E8  1050 ;
                        07E8  1051 ; THIS ROUTINE IS CALLED TO ENABLE/DISABLE CTRL/T AST'S ON THE
                        07E8  1052 ; INPUT CHANNEL.
                        07E8  1053 ;
                        07E8  1054 ; INPUTS:
                        07E8  1055 ;
                        07E8  1056 ;       R1  = CONTROL MASK
                        07E8  1057 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                        07E8  1058 ;
                        07E8  1059 ; OUTPUTS:
                        07E8  1060 ;
                        07E8  1061 ;       R0 = FINAL REQUEST STATUS.
                        07E8  1062 ;-
                        07E8  1063
                        07E8  1064 DCL$ENBCONTRLT:                              ;ENABLE/DISABLE CONTROL T AST
      7E    52    7D    07E8  1065          MOVQ    R2,-(SP)                    ;GET TWO REGISTERS TO WORK WITH
52  FD7C    CF    DE    07EB  1066          MOVAL   DCL$CONTRLT,R2              ;GET ADDRESS OF AST ROUTINE
 00100000    8F    DD    07F0  1067          PUSHL   #PRC_M_CTRLT                ;SET CHARACTER MASK
      7E    D4    07F6  1068          CLRL    -(SP)                       ;USE SHORT FORM OF MASK
      53    5E    D0    07F8  1069          MOVL    SP,R3                       ;GET ADDRESS OF MASK BLOCK
   02 51    14    E0    07FB  1070          BBS     #PRC_V_CTRLT,R1,10$         ;SKIP IF ENABLING CTRL/T'S
      52    D4    07FF  1071          CLRL    R2                          ;CLEAR ADDRESS OF AST ROUTINE
      7E    7C    0801  1072 10$:     CLRQ    -(SP)                       ;ALLOCATE IOSB
      50    5E    D0    0803  1073          MOVL    SP,R0                       ;
                        0806  1074          $QIOW_S EFN=#EXE$C_SYSEFN,-             ;EVENT FLAG
                        0806  1075                  IOSB=(R0),-                 ;IOSB
                        0806  1076                  CHAN=PRC_W_INPCHAN(R11),-   ;INPUT CHANNEL
                        0806  1077                  FUNC=#IO$_SETMODE!IO$M_OUTBAND,- ;FUNCTION CODE
                        0806  1078                  P1=(R2),-                   ;AST ROUTINE ADDRESS
                        0806  1079                  P2=R3,-                     ;ADDRESS OF CHARACTER MASK
                        0806  1080                  P3=#PSL$C_SUPER             ;ACCESS MODE
   5E    10    C0    082A  1081          ADDL    #16,SP                      ;POP STACK
   52    8E    7D    082D  1082          MOVQ    (SP)+,R2                    ;RESTORE REGISTERS
               05    0830  1083          RSB
```

```
                     0831  1085          .SBTTL  RESET OUT-OF-BAND AST'S
                     0831  1086 ;+
                     0831  1087 ; DCL$RESETOOB - RESET OUT-OF-BAND AST'S
                     0831  1088 ;
                     0831  1089 ; THIS ROUTINE IS CALLED TO ENABLE OR DISABLE OUT-OF-BAND AST'S ON THE INPUT
                     0831  1090 ; CHANNEL.
                     0831  1091 ;
                     0831  1092 ; INPUTS:
                     0831  1093 ;
                     0831  1094 ;       R1  = CONTROL MASK.  BITS ARE SET IF AST SHOULD BE ENABLED, CLEAR
                     0831  1095 ;             IF AST SHOULD BE DISABLED.
                     0831  1096 ;
                     0831  1097 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                     0831  1098 ;
                     0831  1099 ;-
                     0831  1100
                     0831  1101 DCL$RESETOOB::                                       ;ENABLE OR DISABLE OUT-OF-BAND AST
 25 68 AB    06   E0 0831  1102          BBS     #PRC_V_MODE,PRC_W_FLAGS(R11),90$ ;IF SET, NOT INTERACTIVE JOB
    50    08 AB   D0 0836  1103          MOVL    PRC_L_INPRAB(R11),R0             ;GET ADDRESS OF INPUT RAB
 1C 18 A0    02   E1 083A  1104          BBC     #DEV$V_TRM,RAB$L_CTX(R0),90$     ;IF CLR, 'INPUT' NOT FROM TERMINAL
                     083F  1105
       51         DD 083F  1106          PUSHL   R1                               ;SAVE AST CHARACTER MASK
    08 51         14   E1 0841  1107          BBC     #PRC_V_CTRLT,R1,10$              ;SKIP IF DISABLING CTRL/T'S
0B 00B4 CB        14   E0 0845  1108          BBS     #PRC_V_CTRLT,PRC_L_OUTOFBAND(R11),30$  ;SKIP IF ALREADY ENABLED
                  06   11 084B  1109          BRB     20$                              ;ENABLE CTRL/T AST
03 00B4 CB        14   E1 084D  1110 10$:     BBC     #PRC_V_CTRLT,PRC_L_OUTOFBAND(R11),30$  ;SKIP IF ALREADY DISABLED
              FF92    30 0853  1111 20$:     BSBW    DCL$ENBCONTRLT                   ;ENABLE/DISABLE CTRL/T AST'S
                     0856  1112
   00B4 CB        8E   D0 0856  1113 30$:     MOVL    (SP)+,PRC_L_OUTOFBAND(R11)       ;SET OUT-OF-BAND MASK
                  05   085B  1114 90$:     RSB
                     085C  1115
```

```
                085C  1117          .SBTTL  COMMAND INTERPRETER CONDITION HANDLER
                085C  1118 ;+
                085C  1119 ; DCL$CONDHAND - COMMAND INTERPRETER CONDITION HANDLER
                085C  1120 ;
                085C  1121 ; THIS ROUTINE IS CALLED AS THE RESULT OF AN EXCEPTION CONDITION THAT OCCURS
                085C  1122 ; WHILE EXECUTING IN THE COMMAND INTERPRETER.
                085C  1123 ;
                085C  1124 ; INPUTS:
                085C  1125 ;
                085C  1126 ;       MECHANISM AND SIGNAL VECTORS
                085C  1127 ;
                085C  1128 ; OUTPUTS:
                085C  1129 ;
                085C  1130 ;       ANY EXIT HANDLERS ARE CANCELLED AND THE CONDITION IS RESIGNALLED.
                085C  1131 ;-
                085C  1132
    0000        085C  1133          .ENTRY  DCL$CONDHAND,^M<>
                085E  1134          $CANEXH_S                       ;CANCEL ANY EXIT HANDLERS
    50    D4    0867  1135          CLRL    R0                      ;RESIGNAL THE CONDITION
          04    0869  1136          RET
                086A  1137
                086A  1138          .END
```

```
$$T1                    =  00000001
$$T2                    =  0000000B
ACCVIO                     000000AE  R     02
ACCVIO2                    00000182  R     02
ACCVIO3                    000002B0  R     02
ACCVIO4                    0000038F  R     02
ATTACH                     000003EF  R     02
BRK$C_DCL               =  00000006
BRK$C_DEVICE            =  00000001
BUFF_BUFIO              =  FFFFFFE8
BUFF_CPUTIM             =  FFFFFFF0
BUFF_DIRIO              =  FFFFFFEC
BUFF_GPGCNT             =  FFFFFFF8
BUFF_IMAGNAME           =  FFFFFFE0
BUFF_PAGEFLTS           =  FFFFFFFC
BUFF_PPGCNT             =  FFFFFFF4
BUFF_PRCNAM             =  FFFFFFE4
CHAIN                      000001DA  R     02
CHECKMASK                  000001C1  R     02
CLI$B_EFN               =  00000038
CLI$B_FLAGS             =  00000004
CLI$B_VERSION           =  00000039
CLI$GET_PRC                ********    X   02
CLI$K_PAUSE             =  00000001
CLI$L_ASTADR            =  00000030
CLI$L_ASTPRM            =  00000034
CLI$L_ATTR              =  00000020
CLI$L_ITMLST            =  0000001C
CLI$L_LSTSTATUS         =  0000000C
CLI$L_OUTPID            =  00000008
CLI$L_PID               =  00000004
CLI$Q_CLI               =  00000044
CLI$Q_CMDSTR            =  00000010
CLI$Q_INPUT             =  00000018
CLI$Q_NAMDESC           =  00000004
CLI$Q_OUTPUT            =  00000020
CLI$Q_PRCNAM            =  00000028
CLI$Q_PROMPT            =  0000003C
CLI$Q_TABDESC           =  00000014
CLI$Q_TABLE             =  0000004C
CLI$Q_VALDESC           =  0000000C
CLI$V_NOCLISYM          =  00000001
CLI$V_NOCONTROL         =  00000005
CLI$V_NOKEYPAD          =  00000003
CLI$V_NOLOGNAM          =  00000002
CLI$V_NOTIFY            =  00000004
CLI$V_NOWAIT            =  00000000
CLI$_BADCTLMSK          =  000388CA
CLI$_BUFOVF             =  00038018
CLI$_ILLVAL             =  0003883A
CLI$_INVREQTYP          =  00038822
CLI$_NORMAL             =  00030001
CLI$_UNDSYM             =  00038140
COMMAND                    000001E4  R     02
CREALOG                    00000221  R     02
CTL$AG_CLIDATA             ********    X   02
CTRLTMSG                   0000051B  R     02
CTRLTMSGEND                00000548  R     02
CTRLT_ARGS              =  00000008
CTRLT_TABLE                000004FB  R     02
DCL                        00000548  R     02
DCL$ALLDEACMD              000003F9  RG    02
DCL$ALLDYNMEM              ********      X 02
DCL$ALLOCSYMABR            ********      X 02
DCL$ATTACH2                ********      X 02
DCL$CHANGE_MODE            0000000C  RG    02
DCL$CONDHAND               0000085C  RG    02
DCL$CONTRLT                0000056B  RG    02
DCL$CONTRLY                0000042F  RG    02
DCL$CRLF                   ********      X 02
DCL$DEADYNMEM              ********      X 02
DCL$DEALLOCSYM             ********      X 02
DCL$DISABLE                ********      X 02
DCL$DSBCONTRLY             000007AD  RG    02
DCL$ENBCONTRLT             000007E8  R     02
DCL$ENBCONTRLY             00000757  RG    02
DCL$HIGH_LIMIT             ********      X 02
DCL$LOW_LIMIT              ********      X 02
DCL$RESETOOB               00000831  RG    02
DCL$RESTART                ********      X 02
DCL$RUNDWNI                ********      X 02
DCL$SAVE_PRIVS             ********      X 02
DCL$SCNTRLY                000004AE  RG    02
DCL$SEARCH                 ********      X 02
DCL$SEARCHT                ********      X 02
DCL$SPAWN2                 ********    X   02
DCLEND                     00000551  R     02
DEFGBL                     00000087  R     02
DEFLOC                     00000081  R     02
DELEGBL                    00000156  R     02
DELELCL                    00000150  R     02
DELELOG                    00000279  R     02
DEV$V_TRM               =  00000002
DISACTRLY                  00000185  R     02
DISAOOB                    000001A3  R     02
ENABCTRLY                  00000194  R     02
ENABOOB                    000001B1  R     02
ENT_K_MAX_PROMPT        =  00000020
ERR_EXIT                   000001C0  R     02
EXE$C_SYSEFN               ********      X 02
GETSYM                     000000B4  R     02
INVREQ                     00000059  R     02
IO$M_CTRLYAST           =  00000080
IO$M_OUTBAND            =  00000400
IO$_SETMODE             =  00000023
ITEM_BUFIO              =  00000018
ITEM_CPUTIM             =  00000030
ITEM_DIRIO              =  00000024
ITEM_GPGCNT             =  00000048
ITEM_IMAGNAME           =  00000000
ITEM_PAGEFLTS           =  00000054
ITEM_PPGCNT             =  0000003C
ITEM_PRCNAM             =  0000000C
JPI$_BUFIO                 ********      X 02
```

```
JPI$_CPUTIM           ********    X   02      PRC_L_IDFLNK                    000000BC
JPI$_DIRIO            ********    X   02      PRC_L_IMGACTSTS                 00000080
JPI$_GPGCNT           ********    X   02      PRC_L_INDCLOCK                  0000007C
JPI$_IMAGNAME         ********    X   02      PRC_L_INDEPTH                   0000005C
JPI$_PAGEFLTS         ********    X   02      PRC_L_INDFAB                    0000001C
JPI$_PPGCNT           ********    X   02      PRC_L_INDINPRAB                 00000014
JPI$_PRCNAM           ********    X   02      PRC_L_INDOUTRAB                 00000018
LNM$PROCESS             00000000  R   02      PRC_L_INPRAB                    00000008
LNM$SYSTEM_TABLE        00000551  R   02      PRC_L_LASTKEY                   0000004C
LNM$_STRING           = 00000002              PRC_L_LSTSTATUS                 000000B0
NORM_EXIT               000001BD  R   02      PRC_L_ONCTLY                    000000B8
NOSUCHSYM               00000148  R   02      PRC_L_ONERROR                   0000006C
PAUSE                   00000061  R   02      PRC_L_OUTOFBAND                 000000B4
PPD$B_NPROCS            0000001C              PRC_L_OUTRAB                    0000000C
PPD$C_LENGTH            00000168              PRC_L_OUTRABCTX                 00000118
PPD$K_LENGTH            00000168              PRC_L_PPFLIST                   00000070
PPD$L_INPDEV            00000044              PRC_L_RECALLPTR                 0000012F
PPD$L_LGI               00000014              PRC_L_RESTART                   00000058
PPD$L_LSTSTATUS         00000018              PRC_L_SAVAP                     00000000
PPD$L_OUTDEV            00000064              PRC_L_SAVFP                     00000004
PPD$L_PRC               00000008              PRC_L_SEVERITY                  00000050
PPD$Q_CLIREG            00000004              PRC_L_SPWN                      000000C0
PPD$Q_CLISYMTBL         0000000C              PRC_L_STACKLM                   000000A4
PPD$T_FILENAME          00000068              PRC_L_STACKPT                   000000A0
PPD$T_INPDVI            00000028              PRC_L_STATUS                    00000054
PPD$T_OUTDVI            00000048              PRC_L_STS                       00000084
PPD$W_FLAGS             00000002              PRC_L_STV                       00000088
PPD$W_INPCHAN           0000001E              PRC_L_SYMBOL                    00000060
PPD$W_INPDID            0000003E              PRC_L_TMBX                      00000074
PPD$W_INPFID            00000038              PRC_L_TRMLIST                   00000010
PPD$W_INPIFI            00000020              PRC_M_CHAIN           = 00000002
PPD$W_INPISI            00000022              PRC_M_CMD             = 00000001
PPD$W_OUTDID            0000005E              PRC_M_CNTRLY          = 00000002
PPD$W_OUTFID            00000058              PRC_M_CTRLT           = 00100000
PPD$W_OUTIFI            00000024              PRC_M_CTRLY           = 02000000
PPD$W_OUTISI            00000026              PRC_M_YLEVEL          = 00000800
PPD$W_SIZE              00000000              PRC_Q_ALLOCREG                  00000020
PRC_B_CONTINUE          000000F3              PRC_Q_COMMAND                   000000E0
PRC_B_DEFRADIX          000000AE              PRC_Q_FLUSHTIME                 000000D0
PRC_B_EXMDEPMOD         000000AD              PRC_Q_GLOBAL                    00000028
PRC_B_EXMDEPWID         000000AC              PRC_Q_IMAGENAME                 000000D8
PRC_B_EXONLYL           0000012D              PRC_Q_KEYPAD                    00000040
PRC_B_FLAGS2            000000AF              PRC_Q_LABEL                     00000030
PRC_B_IMGFLAG           00000078              PRC_Q_LOCAL                     00000038
PRC_B_OUTFLAGS          0000012C              PRC_Q_SAVEPRIV                  000000E8
PRC_B_PROMPTLEN         000000F0              PRC_T_OUTDVI                    0000011C
PRC_C_LENGTH            00000534              PRC_V_CHAIN           = 00000001
PRC_G_COMMANDS          00000133              PRC_V_CTRLT           = 00000014
PRC_G_PROMPT            000000F4              PRC_V_CTRLY           = 00000019
PRC_K_LENGTH            00000534              PRC_V_DISABL          = 00000002
PRC_L_CURRKEY           00000048              PRC_V_HANGUP          = 0000000C
PRC_L_EXMDEPADR         000000A8              PRC_V_MODE            = 00000006
PRC_L_EXTARG            00000094              PRC_V_PRIV            = 00000004
PRC_L_EXTBLK            0000008C              PRC_V_YLEVEL          = 0000000B
PRC_L_EXTCOD            0000009C              PRC_W_ASTIOSB                   000000C6
PRC_L_EXTHND            00000090              PRC_W_ASTRETN                   000000C8
PRC_L_EXTPRM            00000098              PRC_W_ASTSTATUS                 000000C4
```

```
PRC_W_ATTMBX                    0000007A
PRC_W_FLAGS                     00000068
PRC_W_INPCHAN                   00000064
PRC_W_ONLEVEL                   0000006A
PRC_W_OUTIFI                    00000114
PRC_W_OUTISI                    00000116
PRC_W_OUTMBXCHN                 000000CA
PRC_W_OUTMBXREF                 000000CE
PRC_W_OUTMBXSIZ                 000000CC
PRC_W_PMPTCTRL                  000000F1
PRC_W_WAITIOSB                  00000066
PSL$C_SUPER                 =   00000002
PSL$V_CURMOD                =   00000018
RAB$L_CTX                   =   00000018
RET                             000002AF R      02
SF$L_SAVE_AP                =   00000008
SF$L_SAVE_FP                =   0000000C
SF$S_STACKOFFS              =   00000002
SF$V_STACKOFFS              =   0000000E
SF$W_SAVE_MASK              =   00000006
SPAWN                           000002B3 R      02
SPWN_B_ACMODE                   0000000E
SPWN_B_CONTINUE                 000000A5
SPWN_B_EFN                      0000000F
SPWN_B_PROMPTLEN                000000A2
SPWN_C_LENGTH                   000000D6
SPWN_G_PROMPT                   000000A6
SPWN_G_QUOTAS                   00000060
SPWN_K_LENGTH                   000000D6
SPWN_L_ASTADR                   0000004C
SPWN_L_ASTPRM                   00000050
SPWN_L_IMAGCNT                  0000005C
SPWN_L_LINK                     00000000
SPWN_L_OUTOFBAND                00000058
SPWN_L_PRIB                     00000048
SPWN_L_STATUS                   00000044
SPWN_L_STSADR                   00000054
SPWN_L_SUBPID                   00000040
SPWN_Q_CLI                      000000C6
SPWN_Q_CMDSTR                   00000030
SPWN_Q_INPUT                    00000020
SPWN_Q_IOSB                     00000038
SPWN_Q_MBXNAM                   00000010
SPWN_Q_OUTPUT                   00000028
SPWN_Q_PRCNAM                   00000018
SPWN_Q_TABLE                    000000CE
SPWN_T_PROCESS                  00000092
SPWN_V_CLI                  =   0000000D
SPWN_V_CLISYM               =   00000005
SPWN_V_INPUT                =   0000000A
SPWN_V_KEYPAD               =   0000000C
SPWN_V_LOGNAM               =   00000006
SPWN_V_NOTIFY               =   00000008
SPWN_V_OUTPUT               =   0000000B
SPWN_V_PRCNAM               =   00000001
SPWN_V_PROMPT               =   00000009
SPWN_V_TABLE                =   0000000F
SPWN_V_WAIT                 =   00000002
SPWN_W_CHAN                     0000000A
SPWN_W_FLAGS                    0000000C
SPWN_W_PMPTCTRL                 000000A3
SPWN_W_SIZE                     00000004
SPWN_W_UNIT                     00000008
SS$_ACCVIO                      ********   X   02
SS$_HANGUP                      ********   X   02
SS$_NOLOGNAM                    ********   X   02
SYI$_NODENAME               =   000010D9
SYM_B_FLAGS                     0000000B
SYM_B_NONUNIQUE                 0000000B
SYM_B_TYPE                      0000000A
SYM_K_PERM                  =   00000001
SYM_K_STRING                =   00000000
SYM_L_BL                        00000004
SYM_L_FL                        00000000
SYM_T_SYMBOL                    0000000C
SYM_W_SIZE                      00000008
SYS$ASCTIM                      ********  GX   02
SYS$BRKTHRUW                    ********  GX   02
SYS$CANEXH                      ********  GX   02
SYS$CRELNM                      ********  GX   02
SYS$DELLNM                      ********  GX   02
SYS$FAO                         ********   X   02
SYS$GETJPIW                     ********  GX   02
SYS$GETSYIW                     ********  GX   02
SYS$NODE                        00000562 R      02
SYS$QIOW                        ********  GX   02
SYS$SETEF                       ********  GX   02
SYS$TRNLNM                      ********  GX   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_CMDBUFSIZ             =   00000400
WRK_C_INPBUFSIZ             =   00000100
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_V_COMMAND                 = 00000001
WRK_W_FLAGS                     FFFFFFF0
WRK_W_FLAGS2                    FFFFFFF2
WRK_W_IMGCHAN                   FFFFFFEE
WRK_W_PMPTLEN                   FFFFF99E
_$$_                          = 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	0000086A (2154.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.03	00:00:01.52
Command processing	80	00:00:00.72	00:00:06.42
Pass 1	520	00:00:23.20	00:01:13.01
Symbol table sort	0	00:00:03.22	00:00:08.43
Pass 2	212	00:00:04.75	00:00:12.69
Symbol table output	44	00:00:00.32	00:00:00.88
Psect synopsis output	1	00:00:00.02	00:00:00.18
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	866	00:00:32.26	00:01:43.13

```
The working set limit was 1500 pages.
125114 bytes (245 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 2034 non-local and 77 local symbols.
1138 source lines were read in Pass 1, producing 28 object records in Pass 2.
68 pages of virtual memory were used to define 50 macros.
```

```
                              +----------------------------+
                              ! Macro library statistics !
                              +----------------------------+

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[SYSLIB]SYSBLDMLB.MLB;1                        0
_$255$DUA28:[DCL.OBJ]DCL.MLB;1                            12
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                             2
_$255$DUA28:[SYSLIB]STARLET.MLB;2                         28
TOTALS (all libraries)                                    42

2306 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

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

GETKEYNAM
LIS

GOTO
LIS

EXPRESS
LIS

HANDLE
LIS

IMAGECTRL
LIS

INITIAL
LIS

INDIRECT
LIS

FILECMDS
LIS

IF
LIS

IMAGEXECT
LIS