```
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
CCC              DDD          DDD UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
```

FILEID**TABLE

```
TTTTTTTTTT    AAAAAA    BBBBBBBB    LL          EEEEEEEEEE
TTTTTTTTTT    AAAAAA    BBBBBBBB    LL          EEEEEEEEEE
    TT      AA      AA  BB      BB  LL          EE
    TT      AA      AA  BB      BB  LL          EE
    TT      AA      AA  BB      BB  LL          EE
    TT      AA      AA  BB      BB  LL          EE
    TT      AA      AA  BBBBBBBB    LL          EEEEEEEE
    TT      AA      AA  BBBBBBBB    LL          EEEEEEEE
    TT      AAAAAAAAAA  BB      BB  LL          EE
    TT      AAAAAAAAAA  BB      BB  LL          EE
    TT      AA      AA  BB      BB  LL          EE               ....
    TT      AA      AA  BB      BB  LL          EE               ....
    TT      AA      AA  BBBBBBBB    LLLLLLLLLL  EEEEEEEEEE       ....
    TT      AA      AA  BBBBBBBB    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
```

```
0 MODULE table             (IDENT='V04-000',
0                          ADDRESSING_MODE(EXTERNAL=GENERAL))
1 = BEGIN
1
1 !*****************************************************************************
1 !*                                                                           *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
1 !*  ALL RIGHTS RESERVED.                                                     *
1 !*                                                                           *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
1 !*  TRANSFERRED.                                                             *
1 !*                                                                           *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
1 !*  CORPORATION.                                                             *
1 !*                                                                           *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
1 !*                                                                           *
1 !*                                                                           *
1 !*****************************************************************************
1
1 !++
1 ! Facility:     Command Definition Utility, Table Management Module
1 !
1 ! Abstract:     This module contains all of the routines that manage the
1 !               CLI table.  This includes creation, input, modification, and
1 !               output of the tables.
1 !
1 !               It is recommended that you read over the CLITABDEF.SDL file
1 !               before reading this code.
1 !
1 ! Environment:  Standard CDU Environment.
1 !
1 ! Author:       Paul C. Anagnostopoulos
1 ! Creation:     13 January 1983 (Happy Birthday to me)
1 !
1 ! Modifications:
1 !
1 !       V03-001 MCN0160         Maria del C. Nasr       29-Mar-1984
1 !               Only allow modification of the DCL image table if it matches
1 !               the current version being created by the linker.
1 !
1 !--
1
1
1 library 'sys$library:lib';
1 require 'clitabdef';
1 require 'cdureq';
```

```
  1 !       T A B L E   O F   C O N T E N T S
  1 !       ---------   ---   ---------------
  1
  1 forward routine
  1         cdu$prepare_input_table: novalue,
  1         cdu$prepare_new_table: novalue,
  1         cdu$prepare_image_table: novalue,
  1         cdu$prepare_p1_table: novalue,
  1         cdu$write_output_table: novalue,
  1         cdu$write_image_table: novalue,
  1         cdu$write_p1_table: novalue,
  1         cdu$delete_verb_name,
  1         cdu$add_verb_name: novalue,
  1         cdu$collect_table_blocks: novalue,
  1         long_move: novalue;
  1
  1 !       E X T E R N A L   R E F E R E N C E S
  1 !       ---------------   -------------------
  1
  1 external routine
  1         cdu$report_rms_error,
  1         cdu$upgrade_table,
  1         cli$get_value,
  1         cli$present,
  1         lib$get_vm;
  1
  1 external
  1         ctl$ag_clitable: pointer,
  1         ctl$gl_ctlbasva: pointer;
  1
P 1 $shr_msgdef(cdu,17,local,
P 1         (closeout,severe),
P 1         (openin,severe),
P 1         (openout,severe),
P 1         (writeerr,severe)
  1         );
```

```
   1 !        D C L   T A B L E   C O N T R O L   B L O C K S
   1 !        -----   ---------   -------------   -----------
   1
   1 ! The following items define the RMS control blocks needed to open
   1 ! and map an existing CLI table image.
   1
   1 own
   1          input_xabfhc: $xabfhc(),
   1
   1          input_esa: block[nam$c_maxrss,byte],
   1          input_rsa: block[nam$c_maxrss,byte],
P  1          input_nam: $nam(
P  1                              esa=input_esa,
P  1                              ess=%allocation(input_esa),
P  1                              rsa=input_rsa,
P  1                              rss=%allocation(input_rsa)
   1                              ),
   1
   1          dbuffer(input_spec,nam$c_maxrss),
P  1          input_fab: $fab(
P  1                              dnm='.EXE',
P  1                              fna=input_spec+8,
P  1                              fns=%allocation(input_spec)-8,
P  1                              fop=ufo,
P  1                              nam=input_nam,
P  1                              xab=input_xabfhc
   1                              );
   1
   1 ! The following items define the RMS control blocks needed to create
   1 ! and write a new CLI table image.
   1
   1 own
   1          output_esa: block[nam$c_maxrss,byte],
   1          output_rsa: block[nam$c_maxrss,byte],
P  1          output_nam: $nam(
P  1                              esa=output_esa,
P  1                              ess=%allocation(output_esa),
P  1                              rlf=input_nam,
P  1                              rsa=output_rsa,
P  1                              rss=%allocation(output_rsa)
   1                              ),
   1
   1          dbuffer(output_spec,nam$c_maxrss),
P  1          output_fab: $fab(
P  1                              dnm='.EXE',
P  1                              fac=bio,
P  1                              fna=output_spec+8,
P  1                              fns=%allocation(output_spec)-8,
P  1                              fop=<cbt,sqo,nam,ofp>,
P  1                              mrs=512,
P  1                              nam=output_nam,
P  1                              org=seq,
P  1                              rfm=fix
   1                              ),
   1
P  1          output_rab: $rab(
P  1                              fab=output_fab,
```

```
                        rac=seq,
                        rop=wbh
                        );

! The following items are needed to control the modification of the DCL
! Table on which we will work.  This first item points at the beginning of
! the table, the primary vector block.

global
        cdu$gl_table: pointer;

! This item points at the header for the input table image.  If zero, there
! was no input table image.

own
        input_image_header: pointer initial(0);

! We try to keep unused longwords at the end of the verb name table and
! command block pointer table.  The following item tells us how many.

own
        free_longwords: long initial(0);
```

```
1 !++
1 ! Description:  This routine is called to prepare a CLI table for modification.
1 !               The /TABLE qualifier controls which table is prepared, and
1 !               this is the only table that is modified hereafter.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$prepare_input_table          : novalue
2 = BEGIN
2
2 local
2         status: long,
2         work_ptr: pointer;
2
2
2 ! Determine whether or not the user specified an input table image.
2
2 status = cli$get_value(dtext('TABLES'),input_spec);
2 if .status then
2
2         ! Yes, an input table image was specified.  Call a routine to
2         ! prepare it for modification.
2
2         cdu$prepare_image_table()
2
2 else
2
2         ! No input table image was specified, so the user wants to modify
2         ! the P1 space table.
2
2         cdu$prepare_p1_table();
2
2 ! Upgade the CLI table to the latest format level.  If that fails, then
2 ! signal the resulting CLI status as a fatal error.
2
2 status = cdu$upgrade_table(.cdu$gl_table);
2 check(.status, (.status and not sts$m_severity) + sts$k_severe);
2
2 ! Ensure that the CLI type specified in the primary vector block matches
2 ! the CLI type specified in the command that invoked us.
2
2 if .cdu$gl_table[vec_b_subtype] nequ
2    (if cli$present(dtext('CLI_MCR')) then vec_k_mcr else vec_k_dcl) then
2         signal(msg(cdu$_climismatch));
2
2 return;
2
1 END;
```

```
                                        .TITLE  TABLE
                                        .IDENT  \V04-000\
```

```
                                                   .PSECT  $PLIT$,NOWRT,NOEXE,2

                  45  58  45  2E  00000 P.AAA:    .ASCII  \.EXE\                          ;
                  45  58  45  2E  00004 P.AAB:    .ASCII  \.EXE\                          ;
  00  00  53  45  4C  42  41  54  00008 P.AAD:    .ASCII  \TABLES\<0><0>                  ;
                      010E0006    00010 P.AAC:    .LONG   17694726                        ;
                      00000000'   00014           .ADDRESS P.AAD                          ;
  00  52  43  4D  5F  49  4C  43  00018 P.AAF:    .ASCII  \CLI_MCR\<0>                    ;
                      010E0007    00020 P.AAE:    .LONG   17694727                        ;
                      00000000'   00024           .ADDRESS P.AAF                          ;

                                                  .PSECT  $OWN$,NOEXE,2

                            1D    00000 INPUT_XABFHC:
                                                  .BYTE   29                              ;
                            2C    00001           .BYTE   44                              ;
                          0000    00002           .WORD   0                               ;
                      00000000    00004           .LONG   0                               ;
                      00000000#   00008           .LONG   0[9]                            ;
                                  0002C INPUT_ESA:
                                                  .BLKB   255
                                  0012B           .BLKB   1
                                  0012C INPUT_RSA:
                                                  .BLKB   255
                                  0022B           .BLKB   1
                            02    0022C INPUT_NAM:
                                                  .BYTE   2                               ;
                            60    0022D           .BYTE   96                              ;
                            FF    0022E           .BYTE   -1                              ;
                            00    0022F           .BYTE   0                               ;
                      00000000'   00230           .ADDRESS INPUT_RSA                      ;
                            00    00234           .BYTE   0                               ;
                            00    00235           .BYTE   0                               ;
                            FF    00236           .BYTE   -1                              ;
                            00    00237           .BYTE   0                               ;
                      00000000'   00238           .ADDRESS INPUT_ESA                      ;
                      00000000    0023C           .LONG   0                               ;
                          0000#   00240           .WORD   0[8]                            ;
                          0000#   00250           .WORD   0[3]                            ;
                          0000#   00256           .WORD   0[3]                            ;
                      00000000    0025C           .LONG   0                               ;
                      00000000    00260           .LONG   0                               ;
                            00    00264           .BYTE   0                               ;
                            00    00265           .BYTE   0                               ;
                            00    00266           .BYTE   0                               ;
                            00    00267           .BYTE   0                               ;
                            00    00268           .BYTE   0                               ;
                            00    00269           .BYTE   0                               ;
                            00#   0026A           .BYTE   0[2]                            ;
                      00000000    0026C           .LONG   0                               ;
                      00000000    00270           .LONG   0                               ;
                      00000000    00274           .LONG   0                               ;
                      00000000    00278           .LONG   0                               ;
                      00000000    0027C           .LONG   0                               ;
                      00000000    00280           .LONG   0                               ;
                      00000000#   00284           .LONG   0[2]                            ;
```

```
                 00FF  0028C INPUT_SPEC:
                                         .WORD      255                                                   ;
          00   00  0028E                 .BYTE      0, 0                                                  ;
           00000000'  00290              .ADDRESS   INPUT_SPEC+8                                          ;
                      00294              .BLKB      255
                      00393              .BLKB      1
                   03 00394 INPUT_FAB:
                                         .BYTE      3                                                     ;
                   50 00395              .BYTE      80                                                    ;
                 0000 00396              .WORD      0                                                     ;
             00020000 00398              .LONG      131072                                                ;
             00000000 0039C              .LONG      0                                                     ;
             00000000 003A0              .LONG      0                                                     ;
             00000000 003A4              .LONG      0                                                     ;
                 0000 003A8              .WORD      0                                                     ;
                   02 003AA              .BYTE      2                                                     ;
                   00 003AB              .BYTE      0                                                     ;
             00000000 003AC              .LONG      0                                                     ;
                   00 003B0              .BYTE      0                                                     ;
                   00 003B1              .BYTE      0                                                     ;
                   00 003B2              .BYTE      0                                                     ;
                   02 003B3              .BYTE      2                                                     ;
             00000000 003B4              .LONG      0                                                     ;
            00000000' 003B8              .ADDRESS   INPUT_XABFHC                                          ;
            00000000' 003BC              .ADDRESS   INPUT_NAM                                             ;
            00000000' 003C0              .ADDRESS   INPUT_SPEC+8                                          ;
            00000000' 003C4              .ADDRESS   P.AAA                                                 ;
                   FF 003C8              .BYTE      -1                                                    ;
                   04 003C9              .BYTE      4                                                     ;
                 0000 003CA              .WORD      0                                                     ;
             00000000 003CC              .LONG      0                                                     ;
                 0000 003D0              .WORD      0                                                     ;
                   00 003D2              .BYTE      0                                                     ;
                   00 003D3              .BYTE      0                                                     ;
             00000000 003D4              .LONG      0                                                     ;
             00000000 003D8              .LONG      0                                                     ;
                 0000 003DC              .WORD      0                                                     ;
                   00 003DE              .BYTE      0                                                     ;
                   00 003DF              .BYTE      0                                                     ;
             00000000 003E0              .LONG      0                                                     ;
                      003E4 OUTPUT_ESA:
                                         .BLKB      255
                      004E3              .BLKB      1
                      004E4 OUTPUT_RSA:
                                         .BLKB      255
                      005E3              .BLKB      1
                   02 005E4 OUTPUT_NAM:
                                         .BYTE      2                                                     ;
                   60 005E5              .BYTE      96                                                    ;
                   FF 005E6              .BYTE      -1                                                    ;
                   00 005E7              .BYTE      0                                                     ;
            00000000' 005E8              .ADDRESS   OUTPUT_RSA                                            ;
                   00 005EC              .BYTE      0                                                     ;
                   00 005ED              .BYTE      0                                                     ;
                   FF 005EE              .BYTE      -1                                                    ;
                   00 005EF              .BYTE      0                                                     ;
            00000000' 005F0              .ADDRESS   OUTPUT_ESA                                            ;
```

```
00000000' 005F4             .ADDRESS INPUT_NAM                      ;
    0000# 005F8             .WORD    0[8]                           ;
    0000# 00608             .WORD    0[3]                           ;
    0000# 0060E             .WORD    0[3]                           ;
 00000000 00614             .LONG    0                              ;
 00000000 00618             .LONG    0                              ;
       00 0061C             .BYTE    0                              ;
       00 0061D             .BYTE    0                              ;
       00 0061E             .BYTE    0                              ;
       00 0061F             .BYTE    0                              ;
       00 00620             .BYTE    0                              ;
       00 00621             .BYTE    0                              ;
      00# 00622             .BYTE    0[2]                           ;
 00000000 00624             .LONG    0                              ;
 00000000 00628             .LONG    0                              ;
 00000000 0062C             .LONG    0                              ;
 00000000 00630             .LONG    0                              ;
 00000000 00634             .LONG    0                              ;
 00000000 00638             .LONG    0                              ;
00000000# 0063C             .LONG    0[2]                           ;
     00FF 00644 OUTPUT_SPEC:
                            .WORD    255                            ;
    00 00 00646             .BYTE    0, 0                           ;
00000000' 00648             .ADDRESS OUTPUT_SPEC+8                  ;
          0064C             .BLKB    255
          0074B             .BLKB    1
       03 0074C OUTPUT_FAB:
                            .BYTE    3                              ;
       50 0074D             .BYTE    80                             ;
     0000 0074E             .WORD    0                              ;
 21200040 00750             .LONG    555745344                      ;
 00000000 00754             .LONG    0                              ;
 00000000 00758             .LONG    0                              ;
 00000000 0075C             .LONG    0                              ;
     0000 00760             .WORD    0                              ;
       20 00762             .BYTE    32                             ;
       00 00763             .BYTE    0                              ;
 00000000 00764             .LONG    0                              ;
       00 00768             .BYTE    0                              ;
       00 00769             .BYTE    0                              ;
       00 0076A             .BYTE    0                              ;
       01 0076B             .BYTE    1                              ;
 00000000 0076C             .LONG    0                              ;
 00000000 00770             .LONG    0                              ;
00000000' 00774             .ADDRESS OUTPUT_NAM                     ;
00000000' 00778             .ADDRESS OUTPUT_SPEC+8                  ;
00000000' 0077C             .ADDRESS P.AAB                          ;
       FF 00780             .BYTE    -1                             ;
       04 00781             .BYTE    4                              ;
     0200 00782             .WORD    512                            ;
 00000000 00784             .LONG    0                              ;
     0000 00788             .WORD    0                              ;
       00 0078A             .BYTE    0                              ;
       00 0078B             .BYTE    0                              ;
 00000000 0078C             .LONG    0                              ;
 00000000 00790             .LONG    0                              ;
     0000 00794             .WORD    0                              ;
```

```
                        00  00796           .BYTE   0                                   ;
                        00  00797           .BYTE   0                                   ;
                  00000000  00798           .LONG   0                                   ;
                        01  0079C OUTPUT_RAB:
                                            .BYTE   1                                   ;
                        44  0079D           .BYTE   68                                  ;
                      0000  0079E           .WORD   0                                   ;
                  00000400  007A0           .LONG   1024                                ;
                  00000000  007A4           .LONG   0                                   ;
                  00000000  007A8           .LONG   0                                   ;
                     0000#  007AC           .WORD   0[3]                                ;
                      0000  007B2           .WORD   0                                   ;
                  00000000  007B4           .LONG   0                                   ;
                      0000  007B8           .WORD   0                                   ;
                        00  007BA           .BYTE   0                                   ;
                        00  007BB           .BYTE   0                                   ;
                      0000  007BC           .WORD   0                                   ;
                      0000  007BE           .WORD   0                                   ;
                  00000000  007C0           .LONG   0                                   ;
                  00000000  007C4           .LONG   0                                   ;
                  00000000  007C8           .LONG   0                                   ;
                  00000000  007CC           .LONG   0                                   ;
                        00  007D0           .BYTE   0                                   ;
                        00  007D1           .BYTE   0                                   ;
                        00  007D2           .BYTE   0                                   ;
                        00  007D3           .BYTE   0                                   ;
                  00000000  007D4           .LONG   0                                   ;
                 00000000'  007D8           .ADDRESS OUTPUT_FAB                         ;
                  00000000  007DC           .LONG   0                                   ;
                  00000000  007E0 INPUT_IMAGE_HEADER:
                                            .LONG   0                                   ;
                  00000000  007E4 FREE_LONGWORDS:
                                            .LONG   0                                   ;

                                            .PSECT  $GLOBAL$,NOEXE,2

                            00000 CDU$GL_TABLE::
                                            .BLKB   4

                                            .EXTRN  CDU$REPORT_RMS_ERROR
                                            .EXTRN  CDU$UPGRADE_TABLE
                                            .EXTRN  CLI$GET_VALUE, CLI$PRESENT
                                            .EXTRN  LIB$GET_VM, CTL$AG_CLITABLE
                                            .EXTRN  CTL$GL_CTLBASVA
                                            .EXTRN  CDU$_CLIMISMATCH

                                            .PSECT  $CODE$,NOWRT,2

                      000C  00000           .ENTRY  CDU$PREPARE_INPUT_TABLE, Save R2,R3    ; 0919
        53 00000000G     00  9E  00002      MOVAB   LIB$SIGNAL, R3                         ;
                 0000'   CF  9F  00009      PUSHAB  INPUT_SPEC                             ; 0929
                 0000'   CF  9F  0000D      PUSHAB  P.AAC                                  ;
  00000000G  00          02  FB  00011      CALLS   #2, CLI$GET_VALUE                      ;
             52          50  D0  00018      MOVL    R0, STATUS                             ;
             07          52  E9  0001B      BLBC    STATUS, 1$                             ; 0930
      0000V  CF          00  FB  0001E      CALLS   #0, CDU$PREPARE_IMAGE_TABLE            ; 0935
                         05  11  00023      BRB     2$                                     ;
```

```
                      0000V  CF            00  FB 00025 1$:     CALLS   #0, CDU$PREPARE_P1_TABLE          ; 0942
                                   0000'   CF  DD 0002A 2$:     PUSHL   CDU$GL_TABLE                      ; 0947
                  00000000G  00            01  FB 0002E         CALLS   #1, CDU$UPGRADE_TABLE             ;
                             52            50  D0 00035         MOVL    R0, STATUS                        ;
                             0A            52  E8 00038         BLBS    STATUS, 3$                        ; 0948
                50           52            07  CB 0003B         BICL3   #7, STATUS, R0                    ;
                                    04     A0  9F 0003F         PUSHAB  4(R0)                             ;
                             63            01  FB 00042         CALLS   #1, LIB$SIGNAL                    ;
                             52    0000'   CF  D0 00045 3$:     MOVL    CDU$GL_TABLE, R2                  ; 0953
                                   0000'   CF  9F 0004A         PUSHAB  P.AAE                             ; 0954
                  00000000G  00            01  FB 0004E         CALLS   #1, CLI$PRESENT                   ;
                             05            50  E9 00055         BLBC    R0, 4$                            ;
                             50            02  D0 00058         MOVL    #2, R0                            ;
                                           03  11 0005B         BRB     5$                                ;
                             50            01  D0 0005D 4$:     MOVL    #1, R0                            ;
   50       03   A2          08            00  ED 00060 5$:     CMPZV   #0, #8, 3(R2), R0                 ;
                                           09  13 00066         BEQL    6$                                ;
                                00000000G  8F  DD 00068         PUSHL   #CDU$_CLIMISMATCH                 ; 0955
                             63            01  FB 0006E         CALLS   #1, LIB$SIGNAL                    ;
                                               04 00071 6$:     RET                                       ; 0959
```

; Routine Size: 114 bytes, Routine Base: $CODE$ + 0000

```
1 !++
1 ! Description:  This routine is called to prepare a fresh new CLI table to
1 !               receive command definitions.  We have to create a primary
1 !               vector block so that all new blocks can be hung off it.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$prepare_new_table    : novalue
2 = BEGIN
2
2 local
2       status: long;
2
2
2 ! Allocate space for a primary vector block.
2
2 allocate_largest_table_block(vec_k_length,cdu$gl_table);
2
2 ! Initialize the block.  The verb name and command block pointer tables will
2 ! be created later.
2
2 cdu$gl_table[vec_b_type] = block_k_vector;
2 cdu$gl_table[vec_b_subtype] = (if cli$present(dtext('CLI_MCR')) then vec_k_mcr else vec_k_dcl);
2 cdu$gl_table[vec_w_flags] = 0;
2 cdu$gl_table[vec_b_strlvl] = vec_k_strlvl;
2 cdu$gl_table[vec_w_tro_count] = 2;
2 cdu$gl_table[vec_l_verbtbl] = cdu$gl_table[vec_l_comdptr] = 0;
2
2 ! Clear the longword that tells us the overall size of the CLI table.  This
2 ! longword will be adjusted as we allocate table blocks.
2
2 cdu$gl_table[vec_l_table_size] = 0;
2
2 ! Set the size of the block.
2
2 set_table_block_size(vec_k_length,cdu$gl_table);
2
2 return;
2
1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

     00  52  43  4D  5F  49  4C  43  00028 P.AAH:      .ASCII  \CLI_MCR\<0>
                           010E0007  00030 P.AAG:      .LONG   17694727
                           00000000' 00034             .ADDRESS P.AAH


                                                        .PSECT  $CODE$,NOWRT,2
```

```
                                   000C 00000          .ENTRY  CDU$PREPARE_NEW_TABLE, Save R2,R3              ; 0972
               53       0000'   CF   9E 00002          MOVAB   CDU$GL_TABLE, R3                                ;
               5E               04   C2 00007          SUBL2   #4, SP                                          ;
                                53   DD 0000A          PUSHL   R3                                              ; 0981
          04   AE               14   D0 0000C          MOVL    #20, 4(SP)                                      ;
                          04    AE   9F 00010          PUSHAB  4(SP)                                           ;
    00000000G  00               02   FB 00013          CALLS   #2, LIB$GET_VM                                  ;
               09               50   E8 0001A          BLBS    STATUS, 1$                                      ;
                                50   DD 0001D          PUSHL   STATUS                                          ;
    00000000G  00               01   FB 0001F          CALLS   #1, LIB$SIGNAL                                  ;
               52               63   D0 00026 1$:      MOVL    CDU$GL_TABLE, R2                                ; 0986
          02   A2               01   90 00029          MOVB    #1, 2(R2)                                       ;
                        0000'   CF   9F 0002D          PUSHAB  P.AAG                                           ; 0987
    00000000G  00               01   FB 00031          CALLS   #1, CLI$PRESENT                                 ;
               05               50   E9 00038          BLBC    R0, 2$                                          ;
               50               02   D0 0003B          MOVL    #2, R0                                          ;
                                03   11 0003E          BRB     3$                                              ;
               50               01   D0 00040 2$:      MOVL    #1, R0                                          ;
          03   A2               50   90 00043 3$:      MOVB    R0, 3(R2)                                       ;
               50               63   D0 00047          MOVL    CDU$GL_TABLE, R0                                ; 0988
          04   A0 00020000      8F   D0 0004A          MOVL    #131072, 4(R0)                                  ;
          04   A0               06   90 00052          MOVB    #6, 4(R0)                                       ; 0989
                          08    A0   7C 00056          CLRQ    8(R0)                                           ; 0991
                          10    A0   D4 00059          CLRL    16(R0)                                          ; 0996
               60               14   B0 0005C          MOVW    #20, (R0)                                       ; 1000
               51               60   3C 0005F          MOVZWL  (R0), R1                                        ;
          10   A0               51   C0 00062          ADDL2   R1, 16(R0)                                      ;
                                     04 00066          RET                                                     ; 1004

; Routine Size:  103 bytes,    Routine Base:  $CODE$ + 0072
```

```
  1 !++
  1 ! Description:  This routine is called when it has been determined that the
  1 !               user wants to modify an existing CLI table image.  The image
  1 !               is mapped into memory and checked to ensure that it really
  1 !               is a CLI table.
  1 !
  1 ! Parameters:   None.
  1 !
  1 ! Returns:      Nothing.
  1 !
  1 ! Notes:
  1 !--
  1
  1 GLOBAL ROUTINE cdu$prepare_image_table          : novalue
  2 = BEGIN
  2
  2 local
  2         status: long,
  2         return_array: vector[2,long],
  2         isd: pointer,
  2         isd2: pointer;
  2
  2
  2 ! The input table file spec has been placed in the spec buffer.  Open the
  2 ! image file and map it into memory.
  2
  2 status = $open(fab=input_fab);
  2 if not .status then
  2         cdu$report_rms_error(msg(cdu$_openin),input_fab);
P 2 status = $crmpsc(inadr=uplit(0,0),
P 2                  retadr=return_array,
P 2                  flags=sec$m_crf + sec$m_expreg + sec$m_wrt,
  2                  chan=.input_fab[fab$l_stv]);
  2 check(.status, .status);
  2
  2 ! Let's verify that this image really contains a CLI table.  We check the
  2 ! following things:
  2 !
  2 !       Majorid and minorid should match does currently created by linker.
  2 !       Must be a native-mode image.
  2 !       Must be a sharable image.
  2 !       Must be linked without Debug, no transfer, PIC.
  2 !       Must have only one image section.
  2 !       Image section must start at VBN 2, not null, read-only.
  2
  2 input_image_header = .return_array[0];
  2
  2 If .input_image_header[ihd$w_majorid] neq ihd$k_majorid
  2   or .input_image_header[ihd$w_minorid] neq ihd$k_minorid
  2 then
  2     signal ( msg(cli$_oldtab) );
  2
  2 isd = .input_image_header + .input_image_header[ihd$w_size];
  2 isd2 = .isd + .isd[isd$w_size];
  2 if .input_image_header[510,0,16,1] neq -1 or
  2    .input_image_header[ihd$b_imgtype] nequ ihd$k_lim or
  2    .input_image_header[ihd$v_lnkdebug] or
```

```
;  330      1062 2     not .input_image_header[ihd$v_lnknotfr] or
;  331      1063 2     not .input_image_header[ihd$v_picimg] or
;  332      1064 2     .isd2[isd$w_size] nequ 0 or
;  333      1065 2     .isd[isd$l_vbn] nequ 2 or
;  334      1066 2     .isd[isd$w_pagcnt] eqlu 0 or
;  335      1067 2     .isd[isd$v_wrt]          then
;  336      1068 2          signal(msg(cdu$_notclitable),2,.input_nam[nam$b_rsl],.input_nam[nam$l_rsa]);
;  337      1069 2
;  338      1070 2  ! Store the address of the CLI table so other modules can get at it.
;  339      1071 2
;  340      1072 2  cdu$gl_table = .input_image_header + 512;
;  341      1073 2
;  342      1074 2  return;
;  343      1075 2
;  344      1076 1  END;


                                                              .PSECT  $PLIT$,NOWRT,NOEXE,2

                         00000000  00000000  00038 P.AAI:     .LONG   0, 0                                        ;

                                                              .EXTRN  SYS$OPEN, SYS$CRMPSC
                                                              .EXTRN  CLI$_OLDTAB, CDU$_NOTCLITABLE

                                                              .PSECT  $CODE$,NOWRT,2

                                          001C 00000          .ENTRY  CDU$PREPARE_IMAGE_TABLE, Save R2,R3,R4     ; 1018
                      54 00000000G   00    9E 00002           MOVAB   LIB$SIGNAL, R4                              ;
                      53     0000'   CF    9E 00009           MOVAB   INPUT_IMAGE_HEADER, R3                      ;
                      5E             08    C2 0000E           SUBL2   #8, SP                                      ;
                           FBB4      C3    9F 00011           PUSHAB  INPUT_FAB                                   ; 1031
            00000000G  00            01    FB 00015           CALLS   #1, SYS$OPEN                                ;
                       52            50    D0 0001C           MOVL    R0, STATUS                                  ;
                       11            52    E8 0001F           BLBS    STATUS, 1$                                  ; 1032
                           FBB4      C3    9F 00022           PUSHAB  INPUT_FAB                                   ; 1033
                       0011109C      8F    DD 00026           PUSHL   #1118364                                    ;
            00000000G  00            02    FB 0002C           CALLS   #2, CDU$REPORT_RMS_ERROR                    ;
                                     7E    7C 00033 1$:       CLRQ    -(SP)                                       ; 1037
                                     7E    7C 00035           CLRQ    -(SP)                                       ;
                           FBC0      C3    DD 00037           PUSHL   INPUT_FAB+12                                ;
                                     7E    7C 0003B           CLRQ    -(SP)                                       ;
                                     7E    D4 0003D           CLRL    -(SP)                                       ;
                       0002000A      8F    DD 0003F           PUSHL   #131082                                     ;
                                     7E    D4 00045           CLRL    -(SP)                                       ;
                             28      AE    9F 00047           PUSHAB  RETURN_ARRAY                                ;
                          0000'      CF    9F 0004A           PUSHAB  P.AAI                                       ;
            00000000G  00            0C    FB 0004E           CALLS   #12, SYS$CRMPSC                             ;
                       52            50    D0 00055           MOVL    R0, STATUS                                  ;
                       05            52    E8 00058           BLBS    STATUS, 2$                                  ; 1038
                                     52    DD 0005B           PUSHL   STATUS                                      ;
                       64            01    FB 0005D           CALLS   #1, LIB$SIGNAL                              ;
                       63            6E    D0 00060 2$:       MOVL    RETURN_ARRAY, INPUT_IMAGE_HEADER            ; 1050
                       50            63    D0 00063           MOVL    INPUT_IMAGE_HEADER, R0                      ; 1052
                 3230  8F      0C    A0    B1 00066           CMPW    12(R0), #12848                              ;
                                     08    12 0006C           BNEQ    3$                                          ;
                 3530  8F      0E    A0    B1 0006E           CMPW    14(R0), #13616                              ; 1053
```

```
                                     09   13 00074          BEQL    4$                                            :
                          00000000G  8F   DD 00076 3$:      PUSHL   #CLI$_OLDTAB                                  : 1055
                        64           01   FB 0007C          CALLS   #1, LIB$SIGNAL                                :
                        50           63   D0 0007F 4$:      MOVL    INPUT_IMAGE_HEADER, R0                        : 1057
                        51           60   3C 00082          MOVZWL  (R0), ISD                                     :
                        51           50   C0 00085          ADDL2   R0, ISD                                       :
                        52           61   3C 00088          MOVZWL  (ISD), ISD2                                   : 1058
                        52           51   C0 0008B          ADDL2   ISD, ISD2                                     :
                 FFFF   8F      01FE C0   B1 0008E          CMPW    510(R0), #-1                                  : 1059
                                     28   12 00095          BNEQ    5$                                            :
                        02        11 A0   91 00097          CMPB    17(R0), #2                                    : 1060
                                     22   12 0009B          BNEQ    5$                                            :
                        1E        20 A0   E8 0009D          BLBS    32(R0), 5$                                    : 1061
            19     20   A0           01   E1 000A1          BBC     #1, 32(R0), 5$                                : 1062
            14     20   A0           03   E1 000A6          BBC     #3, 32(R0), 5$                                : 1063
                                     62   B5 000AB          TSTW    (ISD2)                                        : 1064
                                     10   12 000AD          BNEQ    5$                                            :
                        02        0C A1   D1 000AF          CMPL    12(ISD), #2                                   : 1065
                                     0A   12 000B3          BNEQ    5$                                            :
                                  02 A1   B5 000B5          TSTW    2(ISD)                                        : 1066
                                     05   13 000B8          BEQL    5$                                            :
            14     08   A1           03   E1 000BA          BBC     #3, 8(ISD), 6$                                : 1067
                                FA50 C3   DD 000BF 5$:      PUSHL   INPUT_NAM+4                                   : 1068
                        7E      FA4F C3   9A 000C3          MOVZBL  INPUT_NAM+3, -(SP)                            :
                                     02   DD 000C8          PUSHL   #2                                            :
                          00000000G  8F   DD 000CA          PUSHL   #CDU$_NOTCLITABLE                             :
                        64           04   FB 000D0          CALLS   #4, LIB$SIGNAL                                :
      0000'  CF         63 00000200  8F   C1 000D3 6$:      ADDL3   #512, INPUT_IMAGE_HEADER, CDU$GL_TABLE        : 1072
                                          04 000DD          RET                                                   : 1076
```

; Routine Size: 222 bytes,   Routine Base: $CODE$ + 00D9

```
1 !++
1 ! Description:  This routine is responsible for preparing the CLI table in
1 !               P1 space for modification.  This is the table that is
1 !               currently being used by the CLI for its command
1 !               definitions.  We allow the user to modify the table and put
1 !               it back in P1 space later.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$prepare_p1_table             : novalue
2 = BEGIN
2
2 local
2       status: long;
2
2
2 ! Allocate memory to contain a modifiable copy of the P1 CLI table.  Place
2 ! its address in the global table pointer.
2
2 status = lib$get_vm(%ref(round_up(.ctl$ag_clitable[vec_l_table_size],512)), cdu$gl_table);
2 check(.status, .status);
2
2 ! Copy the P1 table into the memory we just allocated.
2
2 long_move(.ctl$ag_clitable[vec_l_table_size],.ctl$ag_clitable, .cdu$gl_table);
2
2 return;
2
1 END;
```

```
                                  0004 00000        .ENTRY  CDU$PREPARE_P1_TABLE, Save R2          ; 1091
             52 00000000G    00   9E 00002        MOVAB   CTL$AG_CLITABLE, R2                     ;
             5E              04   C2 00009        SUBL2   #4, SP                                  ;
                    0000'    CF   9F 0000C        PUSHAB  CDU$GL_TABLE                            ; 1101
             50              62   D0 00010        MOVL    CTL$AG_CLITABLE, R0                     ;
      50     10 A0 000001FF  8F   C1 00013        ADDL3   #511, 16(R0), R0                        ;
             50 00000200     8F   C6 0001C        DIVL2   #512, R0                                ;
  04  AE     50              09   78 00023        ASHL    #9, R0, 4(SP)                           ;
                       04    AE   9F 00028        PUSHAB  4(SP)                                   ;
        00000000G 00         02   FB 0002B        CALLS   #2, LIB$GET_VM                          ;
             09              50   E8 00032        BLBS    STATUS, 1$                              ; 1102
                             50   DD 00035        PUSHL   STATUS                                  ;
        00000000G 00         01   FB 00037        CALLS   #1, LIB$SIGNAL                          ;
                    0000'    CF   DD 0003E 1$:    PUSHL   CDU$GL_TABLE                            ; 1106
             50              62   D0 00042        MOVL    CTL$AG_CLITABLE, R0                     ;
                             50   DD 00045        PUSHL   R0                                      ;
                       10    A0   DD 00047        PUSHL   16(R0)                                  ;
          0000V CF           03   FB 0004A        CALLS   #3, LONG_MOVE                           ;
```

```
                                04 0004F        RET                                   ; 1110

; Routine Size:  80 bytes,    Routine Base:  $CODE$ + 01B7
```

```
1 !++
1 ! Description:  This routine is responsible for writing out a freshly modified
1 !               CLI table to the destination specified by the /OUTPUT
1 !               qualifier.  The table can be thrown away, written to P1
1 !               space, or written as an image.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$write_output_table            : novalue
2 = BEGIN
2
2 local
2         status: long;
2
2
2 ! If the user doesn't want us to write the table, then just quit.
2
2 if not cli$present(dtext('OUTPUT')) then
2         return;
2
2 ! Determine whether or not the user specified an output table image.
2
2 status = cli$get_value(dtext('OUTPUT'),output_spec);
2 if .status then
2
2         ! Yes, an output table image is desired.  Call a routine to do it.
2
2         cdu$write_image_table()
2
2 else
2
2         ! No output table image was specified, so the user wants to replace
2         ! the P1 space table.
2
2         cdu$write_p1_table();
2
2 return;
2
1 END;


                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

     00  00  54  55  50  54  55  4F   00040 P.AAK:  .ASCII  \OUTPUT\<0><0>              ;
                            010E0006  00048 P.AAJ:  .LONG   17694726                    ;
                           00000000'  0004C         .ADDRESS P.AAK                      ;
     00  00  54  55  50  54  55  4F   00050 P.AAM:  .ASCII  \OUTPUT\<0><0>              ;
                            010E0006  00058 P.AAL:  .LONG   17694726                    ;
                           00000000'  0005C         .ADDRESS P.AAM                      ;
```

```
                                                        .PSECT  $CODE$,NOWRT,2

                                     0000 00000         .ENTRY  CDU$WRITE_OUTPUT_TABLE, Save nothing          ; 1124
                      0000'   CF   9F 00002             PUSHAB  P.AAJ                                         ; 1133
        00000000G  00         01   FB 00006             CALLS   #1, CLI$PRESENT                               ;
                   1D         50   E9 0000D             BLBC    R0, 2$                                        ;
                      0000'   CF   9F 00010             PUSHAB  OUTPUT_SPEC                                   ; 1138
                      0000'   CF   9F 00014             PUSHAB  P.AAL                                         ;
        00000000G  00         02   FB 00018             CALLS   #2, CLI$GET_VALUE                             ;
                   06         50   E9 0001F             BLBC    STATUS, 1$                                    ; 1139
            0000V  CF         00   FB 00022             CALLS   #0, CDU$WRITE_IMAGE_TABLE                     ; 1143
                                   04 00027             RET                                                   ;
            0000V  CF         00   FB 00028 1$:         CALLS   #0, CDU$WRITE_P1_TABLE                        ; 1150
                                   04 0002D 2$:         RET                                                   ; 1154

; Routine Size:  46 bytes,    Routine Base:  $CODE$ + 0207
```

```
1 !++
1 ! Description:  This routine is called to write out a new image containing the
1 !               CLI table we have just compiled.  The new image contains
1 !               the header and symbols from the input table (/TABLE), but
1 !               contains the new table blocks.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$write_image_table            : novalue
2 = BEGIN
2
2 local
2         status: long,
2         final_area: pointer,
2         input_image_gst_vbn: long,
2         length: long,
2         channel: word;
2
2 own
2         acp_fib: block[fib$k_length,byte],
2         acp_fib_dsc: descriptor preset(
2                         [len] = fib$k_length,
2                         [ptr] = acp_fib
2                         ),
2         acp_record_attributes: block[fat$k_length,byte];
2 bind
2         acp_attributes_list = uplit(word(atr$s_recattr),word(atr$c_recattr),
2                                     long(acp_record_attributes),
2                                     long(0));
2
2 builtin
2         rot;
2
2
2 ! An output image can only be created if an input image was specified.
2 ! Make sure this is the case.
2
2 if .input_image_header eqlu 0 then
2         signal(msg(cdu$_nointable));
2
2 ! Begin by creating the output image file.  The output spec has already
2 ! been stored in the spec buffer.
2
2 status = $create(fab=output_fab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_openout),output_fab);
2 status = $connect(rab=output_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_openout),output_rab);
2
2 ! Allocate a large area to contain the final CLI table.  Collect all of the
2 ! table blocks into that area.
```

```
2
2 status = lib$get_vm(cdu$gl_table[vec_l_table_size], final_area);
2 check(.status, .status);
2 cdu$collect_table_blocks(.final_area);
2
2 ! Update the length of the table image section.  Save the VBN of the global
2 ! symbol table and update it, since it resides after che image section.
2
3 begin
3 bind
3         isd = .input_image_header + .input_image_header[ihd$w_size]: block[,byte],
3         ihs = .input_image_header + .input_image_header[ihd$w_symdbgoff]: block[,byte];
3
3 isd[isd$w_pagcnt] = (.cdu$gl_table[vec_l_table_size] + 511) / 512;
3 input_image_gst_vbn = .ihs[ihs$l_gstvbn];
3 ihs[ihs$l_gstvbn] = 2 + .isd[isd$w_pagcnt];
2 end;
2
2 ! Write the image header into the output image.
2
2 output_rab[rab$l_rbf] = .input_image_header;
2 output_rab[rab$w_rsz] = 512;
2 status = $write(rab=output_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),output_rab);
2
2 ! Now we want to write out the new CLI table blocks as the image section.
2
2 output_rab[rab$l_rbf] = .cdu$gl_table;
2 length = round_up(.cdu$gl_table[vec_l_table_size],512);
3 while .length gtru 0 do (
3
3         ! Write out 63K, or fewer bytes if there aren't 63K left.
3
3         output_rab[rab$w_rsz] = minu(63*1024, .length);
3         status = $write(rab=output_rab);
3         if not .status then
3                 cdu$report_rms_error(msg(cdu$_writeerr),output_rab);
3
3         ! Advance the table address and reduce the number of bytes
3         ! remaining to be written.
3
3         output_rab[rab$l_rbf] = .output_rab[rab$l_rbf] + .output_rab[rab$w_rsz];
3         length = .length - .output_rab[rab$w_rsz];
2 );
2
2 ! Now we must copy the global symbol records from the input table image to
2 ! the new image.  These records begin at the VBN specified in the input image
2 ! header, and extend through the last block of the file.
2
2 output_rab[rab$l_rbf] = .input_image_header + (.input_image_gst_vbn-1) * 512;
2 output_rab[rab$w_rsz] = (.input_xabfhc[xab$l_ebk] - .input_image_gst_vbn + 1) * 512;
2 status = $write(rab=output_rab);
2 if not .status then
2         cdu$report_rms_error(msg(cdu$_writeerr),output_rab);
2
2 ! Close the new image file.
```

```
;  540      1269  2
;  541      1270  2 status = $close(fab=output_fab);
;  542      1271  2 if not .status then
;  543      1272  2         cdu$report_rms_error(msg(cdu$_closeout),output_fab);
```

```
     2 ! Now we have to set the end-of-file on the new image so that it points
     2 ! after the end of the global symbol records.  This must be done with the
     2 ! ACP, since there is no such RMS function.  Begin by opening a channel to
     2 ! the device containing the file.
     2
     3 begin
     3 local
     3         device_dsc: descriptor;
     3
     3 build_descriptor(device_dsc,.output_nam[nam$b_dev],.output_nam[nam$l_dev]);
P    3 status = $assign(devnam=device_dsc,
     3                  chan=channel);
     3 check(.status, .status);
     2 end;
     2
     2 ! Read in the record attribute area from the file header of the new image.
     2 ! This is done using the file identification (FID) from the output NAM block.
     2
     2 acp_fib[fib$v_write] = true;
     2 ch$move(fib$s_fid,output_nam[nam$w_fid], acp_fib[fib$w_fid]);
P    2 status = $qiow(chan=.channel,
P    2                func=io$_access + io$m_access,
P    2                p1=acp_fib_dsc,
     2                p5=acp_attributes_list);
     2 check(.status, .status);
     2
     2 ! Back up the end-of-file so that it points into the last block at the same
     2 ! byte offset as the input file.  The two words of the end-of-file VBN are
     2 ! stored reversed.
     2
     2 acp_record_attributes[fat$l_efblk] = rot(rot(.acp_record_attributes[fat$l_efblk],-16)-1,-16);
     2 acp_record_attributes[fat$w_ffbyte] = .input_xabfhc[xab$w_ffb];
     2
     2 ! Update the record attributes area.
     2
P    2 status = $qiow(chan=.channel,
P    2                func=io$_deaccess,
P    2                p1=acp_fib_dsc,
     2                p5=acp_attributes_list);
     2 check(.status, .status);
     2
     2 ! Now we can deassign the channel.
     2
     2 status = $dassgn(chan=.channel);
     2 check(.status, .status);
     2
     2 return;
     2
     1 END;
```

```
                                     .PSECT  $PLIT$,NOWRT,NOEXE,2

         0020  00060 P.AAN:          .WORD   32                           ;
         0004  00062                 .WORD   4                            ;
    00000000'  00064                 .ADDRESS ACP_RECORD_ATTRIBUTES       ;
```

```
                        00000000  00068           .LONG    0                                              ;

                                                  .PSECT   $OWN$,NOEXE,2

                                  007E8 ACP_FIB:.BLKB     64
                              0040 00828 ACP_FIB_DSC:
                                                  .WORD    64                                             ;
                               00# 0082A          .BYTE    0[2]                                           ;
                        00000000' 0082C           .ADDRESS ACP_FIB                                        ;
                                  00830 ACP_RECORD_ATTRIBUTES:
                                                  .BLKB    32

                                        ACP_ATTRIBUTES_LIST=P.AAN
                                                  .EXTRN   CDU$_NOINTABLE, SYS$CREATE
                                                  .EXTRN   SYS$CONNECT, SYS$WRITE
                                                  .EXTRN   SYS$CLOSE, SYS$ASSIGN
                                                  .EXTRN   SYS$QIOW, SYS$DASSGN

                                                  .PSECT   $CODE$,NOWRT,2

                              0FFC 00000          .ENTRY   CDU$WRITE_IMAGE_TABLE, Save R2,R3,R4,R5,R6,-; 1168
                                                           R7,R8,R9,R10,R11                               ;
              5B 00000000G   00  9E 00002          MOVAB    SYS$QIOW, R11                                  ;
              5A 00000000G   00  9E 00009          MOVAB    SYS$WRITE, R10                                 ;
              59 00000000G   00  9E 00010          MOVAB    CDU$REPORT_RMS_ERROR, R9                       ;
              58 00000000G   00  9E 00017          MOVAB    LIB$SIGNAL, R8                                 ;
              57       0000'  CF  9E 0001E          MOVAB    OUTPUT_RAB, R7                                 ;
              5E              10  C2 00023          SUBL2    #16, SP                                        ;
                        44    A7  D5 00026          TSTL     INPUT_IMAGE_HEADER                             ; 1197
                              09  12 00029          BNEQ     1$                                             ;
                 00000000G    8F  DD 0002B          PUSHL    #CDU$_NOINTABLE                                ; 1198
              68              01  FB 00031          CALLS    #1, LIB$SIGNAL                                 ;
                        B0    A7  9F 00034 1$:      PUSHAB   OUTPUT_FAB                                     ; 1203
     00000000G   00           01  FB 00037          CALLS    #1, SYS$CREATE                                 ;
                 56           50  D0 0003E          MOVL     R0, STATUS                                     ;
                 0C           56  E8 00041          BLBS     STATUS, 2$                                     ; 1204
                        B0    A7  9F 00044          PUSHAB   OUTPUT_FAB                                     ; 1205
                 001110A4     8F  DD 00047          PUSHL    #1118372                                       ;
                 69           02  FB 0004D          CALLS    #2, CDU$REPORT_RMS_ERROR                       ;
                              57  DD 00050 2$:      PUSHL    R7                                             ; 1206
     00000000G   00           01  FB 00052          CALLS    #1, SYS$CONNECT                                ;
                 56           50  D0 00059          MOVL     R0, STATUS                                     ;
                 0B           56  E8 0005C          BLBS     STATUS, 3$                                     ; 1207
                              57  DD 0005F          PUSHL    R7                                             ; 1208
                 001110A4     8F  DD 00061          PUSHL    #1118372                                       ;
                 69           02  FB 00067          CALLS    #2, CDU$REPORT_RMS_ERROR                       ;
                              5E  DD 0006A 3$:      PUSHL    SP                                             ; 1213
     7E      0000'   CF       10  C1 0006C          ADDL3    #16, CDU$GL_TABLE, -(SP)                       ;
     00000000G   00           02  FB 00072          CALLS    #2, LIB$GET_VM                                 ;
                 56           50  D0 00079          MOVL     R0, STATUS                                     ;
                 05           56  E8 0007C          BLBS     STATUS, 4$                                     ; 1214
                              56  DD 0007F          PUSHL    STATUS                                         ;
                 68           01  FB 00081          CALLS    #1, LIB$SIGNAL                                 ;
                              6E  DD 00084 4$:      PUSHL    FINAL_AREA                                     ; 1215
         0000V   CF           01  FB 00086          CALLS    #1, CDU$COLLECT_TABLE_BLOCKS                   ;
                 50     44    A7  D0 0008B          MOVL     INPUT_IMAGE_HEADER, R0                         ; 1222
                 51           60  3C 0008F          MOVZWL   (R0), R1                                       ;
```

```
      52                 50            51  C1 00092          ADDL3    R1, R0, R2                                 ;
                         51       04   A0  3C 00096          MOVZWL   4(R0), R1                                  ; 1223
                         51            50  C0 0009A          ADDL2    R0, R1                                     ;
                         53     0000'  CF  D0 0009D          MOVL     CDU$GL_TABLE, R3                           ; 1225
      53          10     A3 000001FF   8F  C1 000A2          ADDL3    #511, 16(R3), R3                           ;
      54                 53 00000200   8F  C7 000AB          DIVL3    #512, R3, R4                               ;
                  02     A2            54  B0 000B3          MOVW     R4, 2(R2)                                  ;
                         53       04   A1  D0 000B7          MOVL     4(R1), INPUT_IMAGE_GST_VBN                 ; 1226
                  04     A1       02   A2  3C 000BB          MOVZWL   2(R2), 4(R1)                               ; 1227
                  04     A1            02  C0 000C0          ADDL2    #2, 4(R1)                                  ;
                  28     A7            50  D0 000C4          MOVL     R0, OUTPUT_RAB+40                          ; 1232
                  22     A7     0200   8F  B0 000C8          MOVW     #512, OUTPUT_RAB+34                        ; 1233
                                       57  DD 000CE          PUSHL    R7                                         ; 1234
                         6A            01  FB 000D0          CALLS    #1, SYS$WRITE                              ;
                         56            50  D0 000D3          MOVL     R0, STATUS                                 ;
                         0B            56  E8 000D6          BLBS     STATUS, 5$                                 ; 1235
                                       57  DD 000D9          PUSHL    R7                                         ; 1236
                            001110D4   8F  DD 000DB          PUSHL    #1118420                                   ;
                         69            02  FB 000E1          CALLS    #2, CDU$REPORT_RMS_ERROR                   ;
                         50     0000'  CF  D0 000E4    5$:   MOVL     CDU$GL_TABLE, R0                           ; 1240
                  28     A7            50  D0 000E9          MOVL     R0, OUTPUT_RAB+40                          ;
      50          10     A0 000001FF   8F  C1 000ED          ADDL3    #511, 16(R0), R0                           ; 1241
                         50 00000200   8F  C6 000F6          DIVL2    #512, R0                                   ;
      52                 50            09  78 000FD          ASHL     #9, R0, LENGTH                             ;
                                       3C  13 00101    6$:   BEQL     9$                                         ; 1242
                         50            52  D0 00103          MOVL     LENGTH, R0                                 ; 1246
            0000FC00     8F            50  D1 00106          CMPL     R0, #64512                                 ;
                                       05  1B 0010D          BLEQU    7$                                         ;
                         50     FC00   8F  3C 0010F          MOVZWL   #64512, R0                                 ;
                  22     A7            50  B0 00114    7$:   MOVW     R0, OUTPUT_RAB+34                          ;
                                       57  DD 00118          PUSHL    R7                                         ; 1247
                         6A            01  FB 0011A          CALLS    #1, SYS$WRITE                              ;
                         56            50  D0 0011D          MOVL     R0, STATUS                                 ;
                         0B            56  E8 00120          BLBS     STATUS, 8$                                 ; 1248
                                       57  DD 00123          PUSHL    R7                                         ; 1249
                            001110D4   8F  DD 00125          PUSHL    #1118420                                   ;
                         69            02  FB 0012B          CALLS    #2, CDU$REPORT_RMS_ERROR                   ;
                         50       22   A7  3C 0012E    8$:   MOVZWL   OUTPUT_RAB+34, R0                          ; 1254
                  28     A7            50  C0 00132          ADDL2    R0, OUTPUT_RAB+40                          ;
                         50       22   A7  3C 00136          MOVZWL   OUTPUT_RAB+34, R0                          ; 1255
                         52            50  C2 0013A          SUBL2    R0, LENGTH                                 ;
                                       C2  11 0013D          BRB      6$                                         ; 1242
      50                 53            09  78 0013F    9$:   ASHL     #9, INPUT_IMAGE_GST_VBN, R0                ; 1262
                         50       44   A7  C0 00143          ADDL2    INPUT_IMAGE_HEADER, R0                     ;
                  28     A7     FE00   C0  9E 00147          MOVAB    -512(R0), OUTPUT_RAB+40                    ;
      50        F874     C7            53  C3 0014D          SUBL3    INPUT_IMAGE_GST_VBN, INPUT_XABFHC+16, R0   ; 1263
      50                 50            09  78 00153          ASHL     #9, R0, R0                                 ;
22    A7                 50     0200   8F  A1 00157          ADDW3    #512, R0, OUTPUT_RAB+34                    ;
                                       57  DD 0015E          PUSHL    R7                                         ; 1264
                         6A            01  FB 00160          CALLS    #1, SYS$WRITE                              ;
                         56            50  D0 00163          MOVL     R0, STATUS                                 ;
                         0B            56  E8 00166          BLBS     STATUS, 10$                                ; 1265
                                       57  DD 00169          PUSHL    R7                                         ; 1266
                            001110D4   8F  DD 0016B          PUSHL    #1118420                                   ;
                         69            02  FB 00171          CALLS    #2, CDU$REPORT_RMS_ERROR                   ;
                                  B0   A7  9F 00174   10$:   PUSHAB   OUTPUT_FAB                                 ; 1270
            00000000G    00            01  FB 00177          CALLS    #1, SYS$CLOSE                              ;
```

```
              56                50 D0 0017E         MOVL    R0, STATUS                                          ;
              0C                56 E8 00181         BLBS    STATUS, 11$                                         ; 1271
                          B0    A7 9F 00184         PUSHAB  OUTPUT_FAB                                          ; 1272
                   0011105C     8F DD 00187         PUSHL   #1118300                                            ;
              69                02 FB 0018D         CALLS   #2, CDU$REPORT_RMS_ERROR                            ;
         08   AE        FE81    C7 9B 00190  11$:   MOVZBW  OUTPUT_NAM+57, DEVICE_DSC                           ; 1282
                          0A    AE B4 00196         CLRW    DEVICE_DSC+2                                        ;
         0C   AE        FE8C    C7 D0 00199         MOVL    OUTPUT_NAM+68, DEVICE_DSC+4                         ;
                                7E 7C 0019F         CLRQ    -(SP)                                               ; 1284
                          0C    AE 9F 001A1         PUSHAB  CHANNEL                                             ;
                          14    AE 9F 001A4         PUSHAB  DEVICE_DSC                                          ;
       00000000G      00        04 FB 001A7         CALLS   #4, SYS$ASSIGN                                      ;
              56                50 D0 001AE         MOVL    R0, STATUS                                          ;
              05                56 E8 001B1         BLBS    STATUS, 12$                                         ; 1285
                                56 DD 001B4         PUSHL   STATUS                                              ;
              68                01 FB 001B6         CALLS   #1, LIB$SIGNAL                                      ;
         4D   A7                01 88 001B9  12$:   BISB2   #1, ACP_FIB+1                                       ; 1291
  50     A7   FE6C   C7         06 28 001BD         MOVC3   #6, OUTPUT_NAM+36, ACP_FIB+4                        ; 1292
                                7E D4 001C4         CLRL    -(SP)                                               ; 1296
                        0000'   CF 9F 001C6         PUSHAB  ACP_ATTRIBUTES_LIST                                 ;
                                7E 7C 001CA         CLRQ    -(SP)                                               ;
                                7E D4 001CC         CLRL    -(SP)                                               ;
                        008C    C7 9F 001CE         PUSHAB  ACP_FIB_DSC                                         ;
                                7E 7C 001D2         CLRQ    -(SP)                                               ;
                                7E D4 001D4         CLRL    -(SP)                                               ;
              7E          72    8F 9A 001D6         MOVZBL  #114, -(SP)                                         ;
              7E          2C    AE 3C 001DA         MOVZWL  CHANNEL, -(SP)                                      ;
                                7E D4 001DE         CLRL    -(SP)                                               ;
              6B                0C FB 001E0         CALLS   #12, SYS$QIOW                                       ;
              56                50 D0 001E3         MOVL    R0, STATUS                                          ;
              05                56 E8 001E6         BLBS    STATUS, 13$                                         ; 1297
                                56 DD 001E9         PUSHL   STATUS                                              ;
              68                01 FB 001EB         CALLS   #1, LIB$SIGNAL                                      ;
       50   009C   C7     F0    8F 9C 001EE  13$:   ROTL    #-16, ACP_RECORD_ATTRIBUTES+8, R0                   ; 1303
                                50 D7 001F5         DECL    R0                                                  ;
  009C     C7     50      F0    8F 9C 001F7         ROTL    #-16, R0, ACP_RECORD_ATTRIBUTES+8                   ;
          00A0     C7     F878  C7 B0 001FE         MOVW    INPUT_XABFHC+20, ACP_RECORD_ATTRIBUTES+12           ; 1304
                                7E D4 00205         CLRL    -(SP)                                               ; 1311
                        0000'   CF 9F 00207         PUSHAB  ACP_ATTRIBUTES_LIST                                 ;
                                7E 7C 0020B         CLRQ    -(SP)                                               ;
                                7E D4 0020D         CLRL    -(SP)                                               ;
                        008C    C7 9F 0020F         PUSHAB  ACP_FIB_DSC                                         ;
                                7E 7C 00213         CLRQ    -(SP)                                               ;
              7E                34 7D 00215         MOVQ    #52, -(SP)                                          ;
              7E          2C    AE 3C 00218         MOVZWL  CHANNEL, -(SP)                                      ;
                                7E D4 0021C         CLRL    -(SP)                                               ;
              6B                0C FB 0021E         CALLS   #12, SYS$QIOW                                       ;
              56                50 D0 00221         MOVL    R0, STATUS                                          ;
              05                56 E8 00224         BLBS    STATUS, 14$                                         ; 1312
                                56 DD 00227         PUSHL   STATUS                                              ;
              68                01 FB 00229         CALLS   #1, LIB$SIGNAL                                      ;
              7E          04    AE 3C 0022C  14$:   MOVZWL  CHANNEL, -(SP)                                      ; 1316
       00000000G      00        01 FB 00230         CALLS   #1, SYS$DASSGN                                      ;
              56                50 D0 00237         MOVL    R0, STATUS                                          ;
              05                56 E8 0023A         BLBS    STATUS, 15$                                         ; 1317
                                56 DD 0023D         PUSHL   STATUS                                              ;
              68                01 FB 0023F         CALLS   #1, LIB$SIGNAL                                      ;
```

```
                                              04 00242 15$:     RET                                        ; 1321

; Routine Size:  579 bytes,    Routine Base:  $CODE$ + 0235
```

```
1 !++
1 ! Description:  This routine is called to write out the new CLI table into
1 !               P1 space, making it the current table for this process.  If
1 !               the new table is no bigger than the current one, we can
1 !               just overwrite it.  If it's bigger, we have to allocate new
1 !               memory in P1 space.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$write_p1_table                     : novalue
2 = BEGIN
2
2 own
2         existing_table: pointer,
2         existing_table_size: long;
2
2 local
2         status: long,
2         final_area: pointer;
```

```
;  620        1346  2 ! This internal routine resets the base address of process-permanent P1
;  621        1347  2 ! space.  The argument list address is the new base.
;  622        1348  2 !
;  623        1349  2 ! Note that we are in kernel mode.
;  624        1350  2
;  625        1351  2 ROUTINE reset_p1_base
;  626        1352  3 = BEGIN
;  627        1353  3
;  628        1354  3 builtin
;  629        1355  3         ap;
;  630        1356  3
;  631        1357  3
;  632        1358  3 ! Just set the new base address in the global longword.
;  633        1359  3
;  634        1360  3 ctl$gl_ctlbasva = .ap;
;  635        1361  3
;  636        1362  3 return true;
;  637        1363  3
;  638        1364  2 END;


                                                      .PSECT  $OWN$,NOEXE,2

                                    00850 EXISTING_TABLE:
                                                      .BLKB   4
                                    00854 EXISTING_TABLE_SIZE:
                                                      .BLKB   4


                                                      .PSECT  $CODE$,NOWRT,2

                               0000 00000 RESET_P1_BASE:
                                                      .WORD   Save nothing                          ; 1351
             00000000G  00          5C  D0 00002      MOVL    AP, CTL$GL_CTLBASVA                   ; 1360
                        50          01  D0 00009      MOVL    #1, R0                                ; 1362
                                        04 0000C      RET                                           ; 1364

; Routine Size:  13 bytes,    Routine Base:  $CODE$ + 0478
```

```
;  640        1365  2 ! This internal routine will unmap the existing CLI table in P1 space.
;  641        1366  2 ! Since we are about to copy a new table, there is no point in keeping the
;  642        1367  2 ! old one around.
;  643        1368  2 !
;  644        1369  2 ! Note that we are in executive mode.
;  645        1370  2
;  646        1371  2 ROUTINE unmap_existing_table
;  647        1372  3 = BEGIN
;  648        1373  3
;  649        1374  3 local
;  650        1375  3         status: long,
;  651        1376  3         range: vector[2,long];
;  652        1377  3
;  653        1378  3
;  654        1379  3 ! Delete the virtual memory containing the existing table.
;  655        1380  3
;  656        1381  3 range[0] = .existing_table;
;  657        1382  3 range[1] = .existing_table + .existing_table_size - 1;
;  658        1383  3 status = $deltva(inadr=range);
;  659        1384  3 if not .status then
;  660        1385  3         return .status;
;  661        1386  3
;  662        1387  3 ! If we just deleted virtual memory at the end of process-permanent
;  663        1388  3 ! P1 space, the we must reset the P1 base address so that it is accurate.
;  664        1389  3
;  665        1390  4 if .existing_table eqla .ctl$gl_ctlbasva then (
;  666      P 1391  4         status = $cmkrnl(routin=reset_p1_base,
;  667        1392  4                          arglst=.existing_table+.existing_table_size);
;  668        1393  4         if not .status then
;  669        1394  4                 return .status;
;  670        1395  3 );
;  671        1396  3
;  672        1397  3 return true;
;  673        1398  3
;  674        1399  2 END;


                                                          .EXTRN  SYS$DELTVA, SYS$CMKRNL

                                        0004 00000 UNMAP_EXISTING_TABLE:
                                                          .WORD   Save R2                                          ; 1371
                     52        0000'   CF  9E 00002       MOVAB   EXISTING_TABLE, R2                               ;
                     5E           04   C2 00007           SUBL2   #4, SP                                           ;
                                  62   DD 0000A           PUSHL   EXISTING_TABLE                                   ; 1381
         50          62     04    A2   C1 0000C           ADDL3   EXISTING_TABLE_SIZE, EXISTING_TABLE, R0          ; 1382
              04     AE     FF    A0   9E 00011           MOVAB   -1(R0), RANGE+4                                  ;
                                  7E   7C 00016           CLRQ    -(SP)                                            ; 1383
                            08    AE   9F 00018           PUSHAB  RANGE                                            ;
      00000000G      00           03   FB 0001B           CALLS   #3, SYS$DELTVA                                   ;
                     1E           50   E9 00022           BLBC    STATUS, 2$                                       ; 1384
      00000000G      00           62   D1 00025           CMPL    EXISTING_TABLE, CTL$GL_CTLBASVA                  ; 1390
                                  12   12 0002C           BNEQ    1$                                               ;
         7E          62     04    A2   C1 0002E           ADDL3   EXISTING_TABLE_SIZE, EXISTING_TABLE, -(SP)       ; 1392
                            BD    AF   9F 00033           PUSHAB  RESET_P1_BASE                                    ;
      00000000G      00           02   FB 00036           CALLS   #2, SYS$CMKRNL                                   ;
                     03           50   E9 0003D           BLBC    STATUS, 2$                                       ; 1393
```

```
                    50           01  D0 00040 1$:     MOVL    #1, R0                                          ; 1397
                                     04 00043 2$:     RET                                                     ; 1399

; Routine Size:  68 bytes,    Routine Base:  $CODE$ + 0485
```

```
;      2 ! This routine will copy the new CLI table into the same virtual address
;      2 ! space as the existing one occupied.  This is done if the new table is no
;      2 ! larger than the existing one.
;      2 !
;      2 ! Note that we are in executive mode.
;      2
;      2 ROUTINE overmap_existing_table
;      3 = BEGIN
;      3
;      3 local
;      3         status: long,
;      3         range: vector[2,long];
;      3
;      3
;      3 ! Create virtual address space in the place where the existing table was
;      3 ! (we already deleted it).  The space must be owned by supervisor mode.
;      3
;      3 range[0] = .existing_table;
;      3 range[1] = .existing_table + .existing_table_size - 1;
;    P 3 status = $cretva(inadr=range,
;      3                  acmode=psl$c_super);
;      3 if not .status then
;      3         return .status;
;      3
;      3 ! If we just created virtual address space at the end of the process-permanent
;      3 ! portion of P1 space, then reset the base address.
;      3
;      4 if .range[1]+1 eqla .ctl$gl_ctlbasva then (
;    P 4         status = $cmkrnl(routin=reset_p1_base,
;      4                          arglst=.existing_table);
;      4         if not .status then
;      4                 return .status;
;      3 );
;      3
;      3 ! Move the new CLI table into the space.
;      3
;      3 long_move(.cdu$gl_table[vec_l_table_size],.cdu$gl_table, .existing_table);
;      3
;      3 ! Make the new memory read-only to user mode.
;      3
;    P 3 status = $setprt(inadr=range,
;      3                  prot=prt$c_ur);
;      3 if not .status then
;      3         return .status;
;      3
;      3 return true;
;      3
;      2 END;


                                                  .EXTRN  SYS$CRETVA, SYS$SETPRT

                                 000C 00000 OVERMAP_EXISTING_TABLE:
                                                  .WORD   Save R2,R3                              ; 1406
              53      0000'   CF  9E 00002        MOVAB   EXISTING_TABLE, R3                      ;
              5E              04  C2 00007        SUBL2   #4, SP                                  ;
```

```
                                     63  DD 0000A          PUSHL   EXISTING_TABLE                              ; 1417
   50          63          04        A3  C1 0000C          ADDL3   EXISTING_TABLE_SIZE, EXISTING_TABLE, R0     ; 1418
          04   AE          FF        A0  9E 00011          MOVAB   -1(R0), RANGE+4                             ;
                                     02  DD 00016          PUSHL   #2                                          ; 1420
                                     7E  D4 00018          CLRL    -(SP)                                       ;
                           08        AE  9F 0001A          PUSHAB  RANGE                                       ;
      00000000G  00                  03  FB 0001D          CALLS   #3, SYS$CRETVA                              ;
                 52                  50  D0 00024          MOVL    R0, STATUS                                  ;
                 47                  52  E9 00027          BLBC    STATUS, 2$                                  ; 1421
   50     04     AE                  01  C1 0002A          ADDL3   #1, RANGE+4, R0                             ; 1427
      00000000G  00                  50  D1 0002F          CMPL    R0, CTL$GL_CTLBASVA                         ;
                                     13  12 00036          BNEQ    1$                                          ;
                                     63  DD 00038          PUSHL   EXISTING_TABLE                              ; 1429
                           FF71      CF  9F 0003A          PUSHAB  RESET_P1_BASE                               ;
      00000000G  00                  02  FB 0003E          CALLS   #2, SYS$CMKRNL                              ;
                 52                  50  D0 00045          MOVL    R0, STATUS                                  ;
                 26                  52  E9 00048          BLBC    STATUS, 2$                                  ; 1430
                                     63  DD 0004B 1$:      PUSHL   EXISTING_TABLE                              ; 1436
                 50        0000'     CF  D0 0004D          MOVL    CDU$GL_TABLE, R0                            ;
                                     50  DD 00052          PUSHL   R0                                          ;
                           10        A0  DD 00054          PUSHL   16(R0)                                      ;
          0000V  CF                  03  FB 00057          CALLS   #3, LONG_MOVE                               ;
                 7E                  0F  7D 0005C          MOVQ    #15, -(SP)                                  ; 1441
                                     7E  7C 0005F          CLRQ    -(SP)                                       ;
                           10        AE  9F 00061          PUSHAB  RANGE                                       ;
      00000000G  00                  05  FB 00064          CALLS   #5, SYS$SETPRT                              ;
                 52                  50  D0 0006B          MOVL    R0, STATUS                                  ;
                 04                  52  E8 0006E          BLBS    STATUS, 3$                                  ; 1442
                 50                  52  D0 00071 2$:      MOVL    STATUS, R0                                  ; 1443
                                         04 00074          RET                                                 ;
                 50                  01  D0 00075 3$:      MOVL    #1, R0                                      ; 1445
                                         04 00078          RET                                                 ; 1447

; Routine Size:  121 bytes,    Routine Base:  $CODE$ + 04C9
```

```
! This routine will copy a new CLI table into a hunk of virtual memory in
! P1 space.  The address of this memory has nothing to do with the address
! of the existing P1 table, because the new table is bigger.  The existing
! table has already been deleted.
!
! Note that we are in executive mode.

ROUTINE map_bigger_table
= BEGIN

local
        status: long,
        range: vector[2,long];


! Expand P1 space so it can accomodate the new table.  The new memory must be
! owned by supervisor mode.

status = $expreg(pagcnt=(.cdu$gl_table[vec_l_table_size]+511) / 512,
                 retadr=range,
                 acmode=psl$c_super,
                 region=1);
if not .status then
        return .status;

! Set the base address of process-permanent P1 space. since it has just
! been moved with the expand region.

status = $cmkrnl(routin=reset_p1_base,
                 arglst=.range[1]);
if not .status then
        return .status;

! Move the new CLI table into the memory we just created.

long_move(.cdu$gl_table[vec_l_table_size],.cdu$gl_table, .range[1]);

! Make the new memory read-only to user mode.

status = $setprt(inadr=range,
                 prot=prt$c_ur);
if not .status then
        return .status;

! Reset the address range of the table in P1 space.

ctl$ag_clitable = .range[1];
begin
bind
        end_address = ctl$ag_clitable+4: pointer;

end_address = .range[0];
end;

return true;

END;
```

```
                                                        .EXTRN  SYS$EXPREG

                                   0004 00000 MAP_BIGGER_TABLE:
                                                        .WORD   Save R2                         ; 1455
                   5E              08  C2 00002         SUBL2   #8, SP
                                   01  DD 00005         PUSHL   #1                              ; 1469
                                   02  DD 00007         PUSHL   #2                              ;
                              08   AE  9F 00009         PUSHAB  RANGE                           ;
                   50      0000'   CF  D0 0000C         MOVL    CDU$GL_TABLE, R0                ;
      50      10   A0 000001FF     8F  C1 00011         ADDL3   #511, T6(R0), R0                ;
      7E           50 00000200     8F  C7 0001A         DIVL3   #512, R0, -(SP)                 ;
         00000000G 00              04  FB 00022         CALLS   #4, SYS$EXPREG                  ;
                   52              50  D0 00029         MOVL    R0, STATUS                      ;
                   3B              52  E9 0002C         BLBC    STATUS, 1$                      ; 1470
                              04   AE  DD 0002F         PUSHL   RANGE+4                         ; 1477
                            FF00   CF  9F 00032         PUSHAB  RESET_P1_BASE                   ;
         00000000G 00              02  FB 00036         CALLS   #2, SYS$CMKRNL                  ;
                   52              50  D0 0003D         MOVL    R0, STATUS                      ;
                   27              52  E9 00040         BLBC    STATUS, 1$                      ; 1478
                              04   AE  DD 00043         PUSHL   RANGE+4                         ; 1483
                   50      00[illegible]'   CF  D0 00046         MOVL    CDU$GL_TABLE, R0                ;
                                   50  DD 0004B         PUSHL   R0                              ;
                              10   A0  DD 0004D         PUSHL   16(R0)                          ;
              0000V CF             03  FB 00050         CALLS   #3, LONG_MOVE                   ;
                   7E              0F  7D 00055         MOVQ    #15, -(SP)                      ; 1488
                                   7E  7C 00058         CLRQ    -(SP)                           ;
                              10   AE  9F 0005A         PUSHAB  RANGE                           ;
         00000000G 00              05  FB 0005D         CALLS   #5, SYS$SETPRT                  ;
                   52              50  D0 00064         MOVL    R0, STATUS                      ;
                   04              52  E8 00067         BLBS    STATUS, 2$                      ; 1489
                   50              52  D0 0006A 1$:     MOVL    STATUS, R0                      ; 1490
                                       04 0006D         RET                                     ;
         00000000G 00         04   AE  D0 0006E 2$:     MOVL    RANGE+4, CTL$AG_CLITABLE        ; 1494
         00000000G 00              6E  D0 00076         MOVL    RANGE, END_ADDRESS              ; 1499
                   50              01  D0 0007D         MOVL    #1, R0                          ; 1502
                                       04 00080         RET                                     ; 1504

; Routine Size:  129 bytes,    Routine Base:  $CODE$ + 0542
```

```
;  783        1505  2 ! Main routine.
;  784        1506  2 ! Allocate a large area to contain the final CLI tables, before we move
;  785        1507  2 ! them into P1 space.  Collect all of the table blocks into the area.  We
;  786        1508  2 ! can't collect directly into P1 space, because we don't know where the
;  787        1509  2 ! final table will fit.
;  788        1510  2
;  789        1511  2 status = lib$get_vm(cdu$gl_table[vec_l_table_size], final_area);
;  790        1512  2 check(.status, .status);
;  791        1513  2 cdu$collect_table_blocks(.final_area);
;  792        1514  2
;  793        1515  2 ! Place the address and size of the existing P1 table into own storage,
;  794        1516  2 ! so that the internal routines can get at it.
;  795        1517  2
;  796        1518  2 existing_table = .ctl$ag_clitable;
;  797        1519  2 existing_table_size = round_up(.existing_table[vec_l_table_size],512);
;  798        1520  2
;  799        1521  2 ! Unmap the existing CLI table from P1 space.
;  800        1522  2
;  801        1523  2 status = $cmexec(routin=unmap_existing_table);
;  802        1524  2 check(.status, .status);
;  803        1525  2
;  804        1526  2 ! If the new table is no larger than the old one, then we can use the same
;  805        1527  2 ! memory in P1 space to contain it.  Otherwise, we have to allocate new
;  806        1528  2 ! memory.
;  807        1529  2
;  808        1530  2 if .cdu$gl_table[vec_l_table_size] lequ .existing_table_size then
;  809        1531  3         status = $cmexec(routin=overmap_existing_table)
;  810        1532  2 else
;  811        1533  2         status = $cmexec(routin=map_bigger_table);
;  812        1534  2 check(.status, .status);
;  813        1535  2
;  814        1536  2 return;
;  815        1537  2
;  816        1538  1 END;


                                                               .EXTRN  SYS$CMEXEC

                                        003C 00000             .ENTRY  CDU$WRITE_P1_TABLE, Save R2,R3,R4,R5             ; 1336
                     55 00000000G  00    9E 00002             MOVAB   SYS$CMEXEC, R5                                   ;
                     54     0000'  CF    9E 00009             MOVAB   EXISTING_TABLE, R4                               ;
                     53 00000000G  00    9E 0000E             MOVAB   LIB$SIGNAL, R3                                   ;
                     5E            04    C2 00015             SUBL2   #4, SP                                           ;
                                   5E    DD 00018             PUSHL   SP                                               ; 1511
            7E      0000'  CF      10    C1 0001A             ADDL3   #16, CDU$GL_TABLE, -(SP)                         ;
               00000000G   00      02    FB 00020             CALLS   #2, LIB$GET_VM                                   ;
                           52      50    D0 00027             MOVL    R0, STATUS                                       ;
                           05      52    E8 0002A             BLBS    STATUS, 1$                                       ; 1512
                                   52    DD 0002D             PUSHL   STATUS                                           ;
                           63      01    FB 0002F             CALLS   #1, LIB$SIGNAL                                   ;
                                   6E    DD 00032 1$:         PUSHL   FINAL_AREA                                       ; 1513
                   0000V   CF      01    FB 00034             CALLS   #1, CDU$COLLECT_TABLE_BLOCKS                     ;
                     64 00000000G  00    D0 00039             MOVL    CTL$AG_CLITABLE, EXISTING_TABLE                  ; 1518
                     50            64    D0 00040             MOVL    EXISTING_TABLE, R0                               ; 1519
            50        10   A0 000001FF  8F    C1 00043             ADDL3   #511, 16(R0), R0                                 ;
                     50 00000200   8F    C6 0004C             DIVL2   #512, R0                                         ;
```

```
        04  A4        50         09  78 00053        ASHL    #9, R0, EXISTING_TABLE_SIZE     ;
                                 7E  D4 00058        CLRL    -(SP)                           ; 1523
                           FE64  CF  9F 0005A        PUSHAB  UNMAP_EXISTING_TABLE            ;
                      65         02  FB 0005E        CALLS   #2, SYS$CMEXEC                  ;
                      52         50  D0 00061        MOVL    R0, STATUS                      ;
                      05         52  E8 00064        BLBS    STATUS, 2$                      ; 1524
                                 52  DD 00067        PUSHL   STATUS                          ;
                      63         01  FB 00069        CALLS   #1, LIB$SIGNAL                  ;
                      50   0000' CF  D0 0006C 2$:    MOVL    CDU$GL_TABLE, R0                ; 1530
              04  A4         10  A0  D1 00071        CMPL    16(R0), EXISTING_TABLE_SIZE     ;
                                 08  1A 00076        BGTRU   3$                              ;
                                 7E  D4 00078        CLRL    -(SP)                           ; 1531
                           FE88  CF  9F 0007A        PUSHAB  OVERMAP_EXISTING_TABLE          ;
                                 06  11 0007E        BRB     4$                              ;
                                 7E  D4 00080 3$:    CLRL    -(SP)                           ; 1533
                           FEF9  CF  9F 00082        PUSHAB  MAP_BIGGER_TABLE                ;
                      65         02  FB 00086 4$:    CALLS   #2, SYS$CMEXEC                  ;
                      52         50  D0 00089        MOVL    R0, STATUS                      ;
                      05         52  E8 0008C        BLBS    STATUS, 5$                      ; 1534
                                 52  DD 0008F        PUSHL   STATUS                          ;
                      63         01  FB 00091        CALLS   #1, LIB$SIGNAL                  ;
                                     04 00094 5$:    RET                                     ; 1538

; Routine Size:  149 bytes,    Routine Base:  $CODE$ + 05C3
```

```
1 !++
1 ! Description:  This routine is responsible for deleting a verb name from a
1 !               CLI table.  This is done by removing the verb name and
1 !               its corresponding command block pointer from the two
1 !               vector blocks in which they reside.  The blocks which
1 !               define the verb simply fall into disuse.
1 !
1 ! Parameters:   verb_name       By descriptor, the name of the verb.
1 !
1 ! Returns:      By value, a boolean which is true if the verb name existed.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$delete_verb_name(verb_name: ref descriptor)
2 = BEGIN
2
2 local
2         key: long,
2         entry: pointer,
2         length: long;
2
2
2 ! If there is no verb name table now, then the command can't exist.
2
2 if .cdu$gl_table[vec_l_verbtbl] eqla 0 then
2         return false;
2
2 ! The verb name table contains longwords, each of which is the first four
2 ! characters of a verb name, padded with NULs.
2
2 ch$copy(minu(.verb_name[len],4),.verb_name[ptr], NUL,4,key);
2
2 ! Search the verb name table for the key just built.  If not found,
2 ! forget it.
2
3 begin
3 bind
3         vector_verb = .cdu$gl_table + .cdu$gl_table[vec_l_verbtbl]: block[,byte],
3         verb_names = vector_verb + vec_k_header_length: vector[,long],
3         vector_command = .cdu$gl_table + .cdu$gl_table[vec_l_comdptr]: block[,byte],
3         command_block_pointers = vector_command + vec_k_header_length: vector[,long];
3
4 entry = (incr i from 0 to .vector_command[vec_w_tro_count]-1 do
3                 if ch$eql(4,key, 4,verb_names[.i],NUL) then exitloop .i);
3 if .entry eql -1 then
3         return false;
3
3 ! Eliminate the found entry and the corresponding entry in the command
3 ! block pointer table by closing up the tables around the entry.
3
3 length = (.vector_command[vec_w_tro_count] - .entry - 1) * 4;
3 ch$move(.length,verb_names[.entry+1], verb_names[.entry]);
3 ch$move(.length,command_block_pointers[.entry+1], command_block_pointers[.entry]);
3
3 ! Adjust the length of the verb name table, along with the length and TRO
3 ! count in the command block pointer table.
```

```
;  875        1596 3
;  876        1597 3 vector_verb[vec_w_size] = .vector_verb[vec_w_size] - 4;
;  877        1598 3 vector_command[vec_w_size] = .vector_command[vec_w_size] - 4;
;  878        1599 3 decrement(vector_command[vec_w_tro_count]);
;  879        1600 2 end;
;  880        1601 2
;  881        1602 2 ! Increment the count of unused longwords at the ends of the tables.
;  882        1603 2
;  883        1604 2 increment(free_longwords);
;  884        1605 2 return true;
;  885        1606 2
;  886        1607 1 END;



                                       03FC 00000          .ENTRY  CDU$DELETE_VERB_NAME, Save R2,R3,R4,R5,R6,-  ; 1553
                                                                   R7,R8,R9                                     ;
                     5E             04   C2 00002          SUBL2   #4, SP                                       ;
                     57       0000'  CF   D0 00005          MOVL    CDU$GL_TABLE, R7                             ; 1564
                              08     A7   D5 0000A          TSTL    8(R7)                                        ;
                                     75   13 0000D          BEQL    5$                                           ;
                     50       04     AC   D0 0000F          MOVL    VERB_NAME, R0                                ; 1570
                     51              60   3C 00013          MOVZWL  (R0), R1                                     ;
                     04              51   B1 00016          CMPW    R1, #4                                       ;
                                     03   1B 00019          BLEQU   1$                                           ;
                     51              04   D0 0001B          MOVL    #4, R1                                       ;
    04         00       04   B0      51   2C 0001E  1$:     MOVC5   R1, @4(R0), #0, #4, KEY                      ;
                                     6E      00024                                                              ;
               58       57    08     A7   C1 00025          ADDL3   8(R7), R7, R8                                ; 1577
                        57    0C     A7   C0 0002A          ADDL2   12(R7), R7                                   ; 1579
                        50    06     A7   3C 0002E          MOVZWL  6(R7), R0                                    ; 1582
                        56           01   CE 00032          MNEGL   #1, I                                        ; 1583
                                     07   11 00035          BRB     3$                                           ;
                   08 A846           6E   D1 00037  2$:     CMPL    KEY, 8(R8)[I]                                ;
                                     07   13 0003C          BEQL    4$                                           ;
               F5       56           50   F2 0003E  3$:     AOBLSS  R0, I, 2$                                    ;
                        56           01   CE 00042          MNEGL   #1, ENTRY                                    ; 1582
                 FFFFFFFF   8F       56   D1 00045  4$:     CMPL    ENTRY, #-1                                   ; 1584
                                     36   13 0004C          BEQL    5$                                           ;
                        50    06     A7   3C 0004E          MOVZWL  6(R7), R0                                    ; 1590
                        50           56   C2 00052          SUBL2   ENTRY, R0                                    ;
                                     50   D7 00055          DECL    R0                                           ;
               59       50           02   78 00057          ASHL    #2, R0, LENGTH                               ;
                              08 A846     DF 0005B          PUSHAL  8(R8)[ENTRY]                                 ; 1591
                              0C A846     DF 0005F          PUSHAL  12(R8)[ENTRY]                                ;
               9E       9E           59   28 00063          MOVC3   LENGTH, @(SP)+, @(SP)+                       ;
                              08 A746     DF 00067          PUSHAL  8(R7)[ENTRY]                                 ; 1592
                              0C A746     DF 0006B          PUSHAL  12(R7)[ENTRY]                                ;
               9E       9E           59   28 0006F          MOVC3   LENGTH, @(SP)+, @(SP)+                       ;
                        68           04   A2 00073          SUBW2   #4, (R8)                                     ; 1597
                        67           04   A2 00076          SUBW2   #4, (R7)                                     ; 1598
                              06     A7   B7 00079          DECW    6(R7)                                        ; 1599
                            0000'    CF   D6 0007C          INCL    FREE_LONGWORDS                               ; 1604
                        50           01   D0 00080          MOVL    #1, R0                                       ; 1605
                                     04      00083          RET                                                  ;
```

```
                 50  D4 00084 5$:      CLRL    R0                                      ; 1607
                     04 00086          RET                                             ;

; Routine Size:  135 bytes,    Routine Base:  $CODE$ + 0658
```

```
!++
! Description:  This routine is responsible for adding a verb name to a
!               CLI table.  First any existing verb name is removed.
!               Then the verb name is added, along with its corresponding
!               command block pointer.
!
! Parameters:   verb_name       By descriptor, the name of the verb.
!               command_block   By reference, the command block for the verb.
!
! Returns:      Nothing.
!
! Notes:
!--

GLOBAL ROUTINE cdu$add_verb_name(verb_name: ref descriptor,
                                 command_block: long)          : novalue
= BEGIN

local
        status: long,
        vector_verb: pointer,
        vector_command: pointer,
        length: long,
        new_block: pointer,
        key: long,
        entry: long;


! Decide if we have an existing verb name table.

if .cdu$gl_table[vec_l_verbtbl] eqla 0 then (

        ! No, we don't.  This must be a new CLI table, so we want to
        ! allocate a verb name and command block table and initialize
        ! them.  Allocate enough space for 128 free longwords.  Hang
        ! the two tables off of the primary vector block.

        length = vec_k_header_length + 128*4;
        allocate_largest_table_block(.length,vector_verb);
        vector_verb[vec_b_type] = block_k_vector;
        vector_verb[vec_b_subtype] = vec_k_verb;
        vector_verb[vec_w_flags] = 0;
        vector_verb[vec_w_tro_count] = 0;
        set_table_block_size(vec_k_header_length,vector_verb);
        cdu$gl_table[vec_l_verbtbl] = .vector_verb - .cdu$gl_table;

        allocate_largest_table_block(.length,vector_command);
        vector_command[vec_b_type] = block_k_vector;
        vector_command[vec_b_subtype] = vec_k_command;
        vector_command[vec_w_flags] = 0;
        vector_command[vec_w_tro_count] = 0;
        set_table_block_size(vec_k_header_length,vector_command);
        cdu$gl_table[vec_l_comdptr] = .vector_command - .cdu$gl_table;

        free_longwords = 128;

) else
```

```

		! A verb name table already exists.  If there is an entry for the
		! new verb name, delete it.

		cdu$delete_verb_name(.verb_name);

	! There must be at least one free longword after the end of the tables,
	! or else we need to make the tables bigger.  If we must, then extend
	! them by 32 entries.

	if .free_longwords eqlu 0 then (
		bind
			vector_verb = .cdu$gl_table + .cdu$gl_table[vec_l_verbtbl]: block[,byte],
			vector_command = .cdu$gl_table + .cdu$gl_table[vec_l_comdptr]: block[,byte];

		allocate_largest_table_block(.vector_verb[vec_w_size] + 32*4, new_block);
		ch$move(.vector_verb[vec_w_size],vector_verb, .new_block);
		cdu$gl_table[vec_l_verbtbl] = .new_block - .cdu$gl_table;

		allocate_largest_table_block(.vector_command[vec_w_size] + 32*4, new_block);
		ch$move(.vector_command[vec_w_size],vector_command, .new_block);
		cdu$gl_table[vec_l_comdptr] = .new_block - .cdu$gl_table;

		free_longwords = 32;
	);

	! The verb name table contains longwords, each of which is the first four
	! characters of a verb name, padded with NULs.

	ch$copy(minu(.verb_name[len],4),.verb_name[ptr], NUL,4,key);

	! Search the verb name table to find the first verb name that collates
	! higher than the new one.

	begin
	bind
		vector_verb = .cdu$gl_table + .cdu$gl_table[vec_l_verbtbl]: block[,byte],
		verb_names = vector_verb + vec_k_header_length: vector[,long],
		vector_command = .cdu$gl_table + .cdu$gl_table[vec_l_comdptr]: block[,byte],
		command_block_pointers = vector_command + vec_k_header_length: vector[,long];

	entry = (incr i from 0 to .vector_command[vec_w_tro_count]-1 do
			if ch$gtr(4,verb_names[.i], 4,key,NUL) then exitloop .i);
	if .entry eql -1 then
		entry = .vector_command[vec_w_tro_count];

	! Make a hole in the verb name and command block pointer tables to
	! accomodate the new verb name.

	length = (.vector_command[vec_w_tro_count] - .entry) * 4;
	ch$move(.length,verb_names[.entry], verb_names[.entry+1]);
	ch$move(.length,command_block_pointers[.entry], command_block_pointers[.entry+1]);

	! Put the verb name and command block pointer into the table.  Adjust
	! the verb name table size, and the size and TRO count in the command
	! block pointer table.

```

```
3 verb_names[.entry] = .key;
3 extend_table_block_size(4,vector_verb);
3 command_block_pointers[.entry] = .command_block - .cdu$gl_table;
3 extend_table_block_size(4,vector_command);
3 increment(vector_command[vec_w_tro_count]);
2 end;
2
2 ! Decrement the count of free longwords at the end of the tables.
2
2 decrement(free_longwords);
2 return;
2
1 END;



                                                   0FFC 00000          .ENTRY  CDU$ADD_VERB_NAME, Save R2,R3,R4,R5,R6,R7,- ; 1622
                                                                               R8,R9,R10,R11                               ;
                          5B 00000000G   00   9E 00002          MOVAB   LIB$GET_VM, R11                             ;
                          5E             14   C2 00009          SUBL2   #20, SP                                     ;
                          50      0000'  CF   D0 0000C          MOVL    CDU$GL_TABLE, R0                            ; 1638
                                    08   A0   D5 00011          TSTL    8(R0)                                       ;
                                         03   13 00014          BEQL    1$                                          ;
                                       0081   31 00016          BRW     4$                                          ;
                          57      0208   8F   3C 00019 1$:      MOVZWL  #520, LENGTH                                ; 1645
                                    04   AE   9F 0001E          PUSHAB  VECTOR_VERB                                 ; 1646
                   04     AE             57   D0 00021          MOVL    LENGTH, 4(SP)                               ;
                                    04   AE   9F 00025          PUSHAB  4(SP)                                       ;
                          6B             02   FB 00028          CALLS   #2, LIB$GET_VM                              ;
                          09             50   E8 0002B          BLBS    STATUS, 2$                                  ;
                                         50   DD 0002E          PUSHL   STATUS                                      ;
             00000000G    00             01   FB 00030          CALLS   #1, LIB$SIGNAL                              ;
                          51        04   AE   D0 00037 2$:      MOVL    VECTOR_VERB, R1                             ; 1647
                   02     A1      0301   8F   3C 0003B          MOVZWL  #769, 2(R1)                                 ;
                                    06   A1   B4 00041          CLRW    6(R1)                                       ; 1650
                          61             08   B0 00044          MOVW    #8, (R1)                                    ; 1651
                          50      0000'  CF   D0 00047          MOVL    CDU$GL_TABLE, R0                            ;
                          52             61   3C 0004C          MOVZWL  (R1), R2                                    ;
                   10     A0             52   C0 0004F          ADDL2   R2, 16(R0)                                  ;
      08     A0           51             50   C3 00053          SUBL3   R0, R1, 8(R0)                               ; 1652
                                    08   AE   9F 00058          PUSHAB  VECTOR_COMMAND                              ; 1654
                   04     AE             57   D0 0005B          MOVL    LENGTH, 4(SP)                               ;
                                    04   AE   9F 0005F          PUSHAB  4(SP)                                       ;
                          6B             02   FB 00062          CALLS   #2, LIB$GET_VM                              ;
                          09             50   E8 00065          BLBS    STATUS, 3$                                  ;
                                         50   DD 00068          PUSHL   STATUS                                      ;
             00000000G    00             01   FB 0006A          CALLS   #1, LIB$SIGNAL                              ;
                          51        08   AE   D0 00071 3$:      MOVL    VECTOR_COMMAND, R1                          ; 1655
                   02     A1      0401   8F   3C 00075          MOVZWL  #1025, 2(R1)                                ;
                                    06   A1   B4 0007B          CLRW    6(R1)                                       ; 1658
                          61             08   B0 0007E          MOVW    #8, (R1)                                    ; 1659
                          50      0000'  CF   D0 00081          MOVL    CDU$GL_TABLE, R0                            ;
                          52             61   3C 00086          MOVZWL  (R1), R2                                    ;
                   10     A0             52   C0 00089          ADDL2   R2, 16(R0)                                  ;
      0C     A0           51             50   C3 0008D          SUBL3   R0, R1, 12(R0)                              ; 1660
```

```
                    0000'  CF       80     8F  9A 00092           MOVZBL  #128, FREE_LONGWORDS                     ; 1662
                                           08  11 00098           BRB     5$                                       ; 1638
                                    04     AC  DD 0009A  4$:      PUSHL   VERB_NAME                                ; 1669
                     FED7  CF              01  FB 0009D           CALLS   #1, CDU$DELETE_VERB_NAME
                                 0000'     CF  D5 000A2  5$:      TSTL    FREE_LONGWORDS                           ; 1675
                                           76  12 000A6           BNEQ    8$
                           50    0000'     CF  D0 000A8           MOVL    CDU$GL_TABLE, R0                         ; 1677
              52           50       08     A0  C1 000AD           ADDL3   8(R0), R0, R2
              56           50       0C     A0  C1 000B2           ADDL3   12(R0), R0, R6                           ; 1678
                                    0C     AE  9F 000B7           PUSHAB  NEW_BLOCK                                ; 1680
                   04      AE              62  3C 000BA           MOVZWL  (R2), 4(SP)
                   04      AE 00000080     8F  C0 000BE           ADDL2   #128, 4(SP)
                                    04     AE  9F 000C6           PUSHAB  4(SP)
                           6B              02  FB 000C9           CALLS   #2, LIB$GET_VM
                           09              50  E8 000CC           BLBS    STATUS, 6$
                                           50  DD 000CF           PUSHL   STATUS
                0000000G   00              01  FB 000D1           CALLS   #1, LIB$SIGNAL
       0C    BE            62              62  28 000D8  6$:      MOVC3   (R2), (R2), @NEW_BLOCK                   ; 1681
                           50    0000'     CF  D0 000DD           MOVL    CDU$GL_TABLE, R0                         ; 1682
       08    A0      0C    AE              50  C3 000E2           SUBL3   R0, NEW_BLOCK, 8(R0)
                                    0C     AE  9F 000E8           PUSHAB  NEW_BLOCK                                ; 1684
                   04      AE              66  3C 000EB           MOVZWL  (R6), 4(SP)
                   04      AE 00000080     8F  C0 000EF           ADDL2   #128, 4(SP)
                                    04     AE  9F 000F7           PUSHAB  4(SP)
                           6B              02  FB 000FA           CALLS   #2, LIB$GET_VM
                           09              50  E8 000FD           BLBS    STATUS, 7$
                                           50  DD 00100           PUSHL   STATUS
                0000000G   00              01  FB 00102           CALLS   #1, LIB$SIGNAL
       0C    BE            66              66  28 00109  7$:      MOVC3   (R6), (R6), @NEW_BLOCK                   ; 1685
                           50    0000'     CF  D0 0010E           MOVL    CDU$GL_TABLE, R0                         ; 1686
       0C    A0      0C    AE              50  C3 00113           SUBL3   R0, NEW_BLOCK, 12(R0)
                    0000'  CF              20  D0 00119           MOVL    #32, FREE_LONGWORDS                      ; 1688
                           50       04     AC  D0 0011E  8$:      MOVL    VERB_NAME, R0                            ; 1694
                           51              60  3C 00122           MOVZWL  (R0), R1
                           04              51  B1 00125           CMPW    R1, #4
                                           03  1B 00128           BLEQU   9$
                           51              04  D0 0012A           MOVL    #4, R1
 04          00      04    B0              51  2C 0012D  9$:      MOVC5   R1, @4(R0), #0, #4, KEY
                                    10     AE     00133
                           59    0000'     CF  D0 00135           MOVL    CDU$GL_TABLE, R9                         ; 1701
              5A           59       08     A9  C1 0013A           ADDL3   8(R9), R9, R10
              58           59       0C     A9  C1 0013F           ADDL3   12(R9), R9, R8                           ; 1703
                           54       06     A8  3C 00144           MOVZWL  6(R8), R4                                ; 1706
                           56              01  CE 00148           MNEGL   #1, I
                                           0B  11 0014B           BRB     11$
                                 08      AA46  DF 0014D  10$:     PUSHAL  8(R10)[I]                                ; 1707
       10    AE            9E              04  29 00151           CMPC3   #4, @(SP)+, KEY
                                           07  1A 00156           BGTRU   12$
             F1            56              54  F2 00158  11$:     AOBLSS  R4, I, 10$
                           56              01  CE 0015C           MNEGL   #1, ENTRY                                ; 1706
          FFFFFFFF         8F              56  D1 0015F  12$:     CMPL    ENTRY, #-1                               ; 1708
                                           04  12 00166           BNEQ    13$
                           56       06     A8  3C 00168           MOVZWL  6(R8), ENTRY                             ; 1709
                           50       06     A8  3C 0016C  13$:     MOVZWL  6(R8), R0                                ; 1714
                           50              56  C2 00170           SUBL2   ENTRY, R0
             57            50              02  78 00173           ASHL    #2, R0, LENGTH
                                 0C      AA46  DF 00177           PUSHAL  12(R10)[ENTRY]                           ; 1715
```

```
                                        08 AA46  DF 0017B          PUSHAL  8(R10)[ENTRY]                                ;
          9E            9E                   57  28 0017F          MOVC3   LENGTH, @(SP)+, @(SP)+                       ;
                                        0C A846  DF 00183          PUSHAL  12(R8)[ENTRY]                                ; 1716
                                        08 A846  DF 00187          PUSHAL  8(R8)[ENTRY]                                 ;
          9E            9E                   57  28 0018B          MOVC3   LENGTH, @(SP)+, @(SP)+                       ;
                   08 AA46          10       AE  D0 0018F          MOVL    KEY, 8(R10)[ENTRY]                           ; 1722
                        6A                   04  A0 00195          ADDW2   #4, (R10)                                    ; 1723
                   10   A9                   04  C0 00198          ADDL2   #4, 16(R9)                                   ;
  08 A846          08   AC                   59  C3 0019C          SUBL3   R9, COMMAND_BLOCK, 8(R8)[ENTRY]              ; 1724
                        68                   04  A0 001A3          ADDW2   #4, (R8)                                     ; 1725
                   10   A9                   04  C0 001A6          ADDL2   #4, 16(R9)                                   ;
                                    06       A8  B6 001AA          INCW    6(R8)                                        ; 1726
                                 0000'       CF  D7 001AD          DECL    FREE_LONGWORDS                               ; 1731
                                                 04 001B1          RET                                                  ; 1734

; Routine Size:  434 bytes,    Routine Base:  $CODE$ + 06DF
```

```
1 !++
1 ! Description:  This routine is called to collect all of the table blocks
1 !               and copy them into the final CLI table area.  After the CLD
1 !               files have been compiled, the table blocks are scattered
1 !               throughout memory and must be collected in a contiguous
1 !               area before being written.
1 !
1 !               The blocks are collected by traversing the tree of blocks
1 !               which is rooted in the primary vector block.  As the blocks
1 !               are copied into the final area, all inter-block pointers
1 !               (TROs) are adjusted accordingly.
1 !
1 ! Parameters:   final_area      By reference, an area of memory to contain
1 !                               the final tables.  It is assumed to be
1 !                               large enough.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$collect_table_blocks(final_area: pointer)    : novalue
2 = BEGIN
2
2 own
2         own_final_area: pointer,
2         allocation_offset: long;
```

```
2 ! This internal routine collects a block and all of its children.  It is passed
2 ! the address of the block and returns the TRO of the block in the final
2 ! area.
2
2 ROUTINE collect_block(a_block: pointer)
3 = BEGIN
3
3 local
3         new_tro: long;
3
3
3 ! If the table block has already been visited for collection, a special type
3 ! code will exist in its header.  Simply return the saved TRO.
3
3 if .a_block[vec_b_type] eqlu block_k_cdu_visited then
3         return .a_block[4,0,32,0];
3
3 ! Remember the new TRO of this block and copy it there.  Advance the allocation
3 ! offset to account for the block.
3
3 new_tro = .allocation_offset;
3 ch$move(.a_block[vec_w_size],.a_block, .own_final_area+.new_tro);
3 allocation_offset = .allocation_offset + .a_block[vec_w_size];
3
3 ! Mark this block as collected by storing a special type code in its
3 ! header.  Also save its new TRO in the second longword.  This will ensure
3 ! that we don't collect the block more than once if it is referenced
3 ! multiple times.
3
3 a_block[vec_b_type] = block_k_cdu_visited;
3 a_block[4,0,32,0] = .new_tro;
3
3 ! Loop through each of the blocks that are directly referenced by this
3 ! block.  Collect them and update the TROs in this block.
3
4 begin
4 bind
4         new_block = .own_final_area + .new_tro: block[,byte],
4         tro_list = new_block + vec_k_header_length: vector[,long];
4
4 incr i from 0 to .new_block[vec_w_tro_count]-1 do
4         if .tro_list[.i] nequ 0 then
4                 tro_list[.i] = collect_block(.cdu$gl_table+.tro_list[.i]);
3 end;
3
3 ! Return the new TRO of this block in the final area.
3
3 return .new_tro;
3
2 END;


                                        .PSECT  $OWN$,NOEXE,2

                         00858 OWN_FINAL_AREA:
                                        .BLKB   4
```

```
                                        0085C ALLOCATION_OFFSET:
                                                        .BLKB    4


                                                        .PSECT   $CODE$,NOWRT,2

                                  01FC 00000 COLLECT_BLOCK:
                                                        .WORD    Save R2,R3,R4,R5,R6,R7,R8                        ; 1766
                56        04    AC  D0 00002           MOVL     A_BLOCK, R6                                      ; 1776
                06        02    A6  91 00006           CMPB     2(R6), #6                                        ;
                                05  12 0000A           BNEQ     1$                                               ;
                50        04    A6  D0 0000C           MOVL     4(R6), R0                                        ; 1777
                                04  00010              RET                                                       ;
                58     0000'    CF  D0 00011 1$:       MOVL     ALLOCATION_OFFSET, NEW_TRO                       ; 1782
      57        58     0000'    CF  C1 00016           ADDL3    OWN_FINAL_AREA, NEW_TRO, R7                      ; 1783
      67        66              66  28 0001C           MOVC3    (R6), (R6), (R7)                                 ;
                50              66  3C 00020           MOVZWL   (R6), R0                                         ; 1784
          0000' CF              50  C0 00023           ADDL2    R0, ALLOCATION_OFFSET                            ;
             02 A6              06  90 00028           MOVB     #6, 2(R6)                                        ; 1791
             04 A6              58  D0 0002C           MOVL     NEW_TRO, 4(R6)                                   ; 1792
                53        08    A7  9E 00030           MOVAB    8(R7), R3                                        ; 1800
                54        06    A7  3C 00034           MOVZWL   6(R7), R4                                        ; 1802
                52              01  CE 00038           MNEGL    #1, I                                            ;
                                14  11 0003B           BRB      3$                                               ;
                              6342  D5 0003D 2$:       TSTL     (R3)[I]                                          ; 1803
                                0F  13 00040           BEQL     3$                                               ;
      7E  0000' CF            6342  C1 00042           ADDL3    (R3)[I], CDU$GL_TABLE, -(SP)                     ; 1804
             B3 AF              01  FB 00049           CALLS    #1, COLLECT_BLOCK                                ;
              6342              50  D0 0004D           MOVL     R0, (R3)[I]                                      ;
      E8        52              54  F2 00051 3$:       AOBLSS   R4, I, 2$                                        ; 1803
                50              58  D0 00055           MOVL     NEW_TRO, R0                                      ; 1809
                                04  00058              RET                                                       ; 1811

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 0891
```

```
2 ! Main routine.
2 ! Call an internal routine to collect all of the blocks.
2
2 own_final_area = .final_area;
2 allocation_offset = 0;
2 collect_block(.cdu$gl_table);
2
2 ! Reset the global variable that points at the beginning of the table.
2 ! The uncollected table will fade into oblivion.
2
2 cdu$gl_table = .final_area;
2
2 ! Store the exact size of the CLI table in the primary vector block.
2
2 cdu$gl_table[vec_l_table_size] = .allocation_offset;
2
2 return;
2
1 END;
```

```
                                    0004 00000          .ENTRY  CDU$COLLECT_TABLE_BLOCKS, Save R2     ; 1756
                 52      0000'  CF  9E 00002            MOVAB   CDU$GL_TABLE, R2                      ;
         0000'   CF         04  AC  D0 00007            MOVL    FINAL_AREA, OWN_FINAL_AREA            ; 1815
                         0000'  CF  D4 0000D            CLRL    ALLOCATION_OFFSET                     ; 1816
                                62  DD 00011            PUSHL   CDU$GL_TABLE                          ; 1817
            90   AF             01  FB 00013            CALLS   #1, COLLECT_BLOCK                     ;
                 62         04  AC  D0 00017            MOVL    FINAL_AREA, CDU$GL_TABLE              ; 1822
                 50             62  D0 0001B            MOVL    CDU$GL_TABLE, R0                      ; 1826
            10   A0      0000'  CF  D0 0001E            MOVL    ALLOCATION_OFFSET, 16(R0)             ;
                                    04 00024            RET                                           ; 1830

; Routine Size:  37 bytes,    Routine Base:  $CODE$ + 08EA
```

```
1 !++
1 ! Description:  This internal routine is called to move an arbitrarily-long
1 !               block of bytes from one place to another.  A loop must be
1 !               used since the stupid VAX instructions can't move more than
1 !               65K-1 bytes at a time.
1 !
1 ! Parameters:   length            By value, the length of the block.
1 !               from_address      By reference, the source address.
1 !               to_address        By reference, the destination address.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE long_move(length: long,
1                         from_address: pointer,
1                         to_address: pointer)            : novalue
2 = BEGIN
2
2 local
2       remaining_length: long;
2
2
2 ! Just move the block in 65K-1 chunks.
2
2 remaining_length = .length;
3 incru offset from 0 to .length by %x'0ffff' do (
3       ch$move(minu(.remaining_length,%x'0ffff'),.from_address+.offset, .to_address+.offset);
3       remaining_length = .remaining_length - minu(.remaining_length,%x'0ffff');
2 );
2
2 return;
2
1 END;
```

```
                                   01FC 00000        .ENTRY  LONG_MOVE, Save R2,R3,R4,R5,R6,R7,R8           ; 1846
                  56        04  AC  D0 00002        MOVL    LENGTH, REMAINING_LENGTH                          ; 1857
                                57  D4 00006        CLRL    OFFSET                                            ; 1859
                                23  11 00008        BRB     3$                                                ;
                  58            56  D0 0000A 1$:    MOVL    REMAINING_LENGTH, R8                              ;
        0000FFFF  8F            58  D1 0000D        CMPL    R8, #65535                                        ;
                                05  1B 00014        BLEQU   2$                                                ;
                  58      FFFF  8F  3C 00016        MOVZWL  #65535, R8                                        ;
     0C BC47           08 BC47  58  28 0001B 2$:    MOVC3   R8, @FROM_ADDRESS[OFFSET], @TO_ADDRESS-           ;
                                                            [OFFSET]                                          ;
                  56            58  C2 00023        SUBL2   R8, REMAINING_LENGTH                              ; 1860
                  57 0000FFFF   E7  9E 00026        MOVAB   65535(R7), OFFSET                                 ; 1858
              04  AC            57  D1 0002D 3$:    CMPL    OFFSET, LENGTH                                    ;
                                D7  1B 00031        BLEQU   1$                                                ;
                                    04 00033        RET                                                       ; 1865
```

; Routine Size: 52 bytes,   Routine Base: $CODE$ + 090F

```
; 1150          1866  1
; 1151          1867  1 END
; 1152          1868  0 ELUDOM



                                                          .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;    $OWN$                         2144  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $PLIT$                         108  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $GLOBAL$                         4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;    $CODE$                        2371  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                        Library Statistics
;
;                                          -------- Symbols --------      Pages      Processing
;       File                               Total    Loaded   Percent     Mapped     Time
;
;    _$255$DUA28:[SYSLIB]LIB.L32;1         18619      112         0       1000        00:02.0



;                               COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:TABLE/OBJ=OBJ$:TABLE MSRC$:TABLE/UPDATE=(ENH$:TABLE)

; Size:           2371 code + 2256 data bytes
; Run Time:          00:49.3
; Elapsed Time:      01:45.4
; Lines/CPU Min:      2272
; Lexemes/CPU-Min: 27392
; Memory Used:  284 pages
; Compilation Complete
```

0044 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

NODES
LIS

OBJECT
LIS

PARSE1
LIS

PARSE3
LIS

ROUTINES
LIS

LISTING
LIS

MAIN
LIS

TABLE
LIS

PARSE2
LIS

0045 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

JNLBUSR
R32

JNLDEFINT
SDL

CJFV4.

JNLPREFIX
R32

CJFRUFMAC
SDL

RUFUSR
SDL

JNLFILE
SDL

UPGRADE
LIS

BOPTIONS
R32

JNLDEF
SDL