E 14

```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC   HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC   HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC   HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCC   HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCC   HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCC   HHH         HHH
```

```
**FILE**ID**PATACT                                H 13
```

```
PPPPPPPP      AAAAAA    TTTTTTTTTT    AAAAAA      CCCCCCCC  TTTTTTTTTT
PPPPPPPP      AAAAAA    TTTTTTTTTT    AAAAAA      CCCCCCCC  TTTTTTTTTT
PP      PP  AA      AA      TT      AA      AA  CC              TT
PP      PP  AA      AA      TT      AA      AA  CC              TT
PP      PP  AA      AA      TT      AA      AA  CC              TT
PP      PP  AA      AA      TT      AA      AA  CC              TT
PPPPPPPP    AA      AA      TT      AA      AA  CC              TT
PPPPPPPP    AA      AA      TT      AA      AA  CC              TT
PP          AAAAAAAAAA      TT      AAAAAAAAAA  CC              TT
PP          AAAAAAAAAA      TT      AAAAAAAAAA  CC              TT
PP          AA      AA      TT      AA      AA  CC              TT          ....
PP          AA      AA      TT      AA      AA  CC              TT          ....
PP          AA      AA      TT      AA      AA    CCCCCCCC      TT          ....
PP          AA      AA      TT      AA      AA    CCCCCCCC      TT          ....

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 PATACT (
0               ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = LONG_RELATIVE),
0               IDENT = 'V04-000') =
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:      PATCH
1 !
1 ! ABSTRACT:
1 !
1 !       End of command line action routine plus a few other parsing
1 !       action routines.
1 !
1 ! ENVIRONMENT:  STARLET, user mode, interrupts disabled.
1 !
1 ! Version:      V02-029
1 !
1 ! History:
1 !       Author:
1 !               Carol Peters, 03 Jul 1976: Version 01
1 !
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-002 MCN0185         Maria del C. Nasr       07-Aug-1984
1 !               Do not execute those commands that are invalid when
1 !               patching in /ABSOLUTE context.  Return error message
1 !               to user.
1 !
1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
1 !               Modify file names to remove duplicate file name useage
1 !               between code and require files.
1 !
1 !       V02-029 MTR0003         Mike Rhodes             03-Feb-1982
```

```
1 !                   Modify the SET PATCH_AREA/INITIALIZE logic to check for
1 !                   a minimum available patch area size of 12 bytes.  This
1 !                   allows 8 bytes for the descriptor plus 1 longword of data.
1 !
1 !           V02-028 MTR0002         Mike Rhodes                 01-Oct-1981
1 !                   Add new qualifier to the SET PATCH_AREA command to allow
1 !                   the user to Initialize a descriptor within the patch area.
1 !
1 !                   SET PATCH_AREA /INITIALIZE=Size_Expression Patch_Area_Address
1 !
1 !                   The code for reconstructing the command line and writing
1 !                   it to the journal and/or command files as well as the
1 !                   PATCH command text area in the image are also included.
1 !
1 !           V02-027 MTR0001         Mike Rhodes                 20-Aug-1981
1 !                   1.  Add new command, HELP.  The associated command verb
1 !                   HELP_CMD and action routine LBR$OUTPUT_HELP (plus its
1 !                   servent routines LIB$GET_INPUT and LIB$PUT_OUTPUT) have
1 !                   been added to the appropriate places.
1 !
1 !                   2.  The procedure for writing command file entries has
1 !                   been modified to selectively make enrties only when the
1 !                   commands correspond to the current ECO level.  Also, the
1 !                   Evaluate, Examine, and Show commands are no longer written
1 !                   to the command file.  The respective action routines in
1 !                   WRITE_CMD have been set to <null> 0.
1 !
1 !           V02-026 KDM0042         Kathleen D. Morse           03-MAR-1981
1 !                   Fix spelling of current patch area message.
1 !
1 !           V02-025 PCG0001         Peter George                02-FEB-1981
1 !                   Add require statement for LIB$:PATDEF.REQ
1 !
1 !--
```

```
0092  1 FORWARD ROUTINE
0093  1         PAT$END_OF_CMD : NOVALUE,                   ! End of command processing routine
0094  1         PAT$END_OF_LINE : NOVALUE,                  ! End of command line processing routine
0095  1         PAT$PERFORM_CMD,                            ! Executes a patch command
0096  1         WRITE_CMD : NOVALUE,                        ! Writes command line to command file
0097  1         PAT$SET_OVERS : NOVALUE,                    ! Sets mode level to local or override level
0098  1         PAT$SET_COMQUAL: NOVALUE,                   ! Sets bit to indicate qualifier in command
0099  1         PAT$GET_COMQUAL : NOVALUE;                  ! Finds all command qualifiers specified
0100  1
0101  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
0102  1 REQUIRE 'SRC$:PATPCT.REQ';
0142  1 REQUIRE 'SRC$:VXSMAC.REQ';
0207  1 REQUIRE 'SRC$:PATGEN.REQ';
0429  1 REQUIRE 'SRC$:BSTRUC.REQ';
0505  1 REQUIRE 'SRC$:DLLNAM.REQ';
0563  1 REQUIRE 'SRC$:LISTEL.REQ';
0605  1 REQUIRE 'LIB$:PATDEF.REQ';                          ! Defines literals
0659  1 REQUIRE 'LIB$:PATMSG.REQ';
0833  1 REQUIRE 'SRC$:PATTER.REQ';
1040  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;       1       SWITCHES LIST (SOURCE);
;       1
;       1       EXTERNAL ROUTINE
;       1               PAT$fao_out;            ! formats a line and outputs to the terminal
;       1
```

```
1 REQUIRE 'SRC$:PREFIX.REQ';
1 REQUIRE 'SRC$:PATPRE.REQ';
1 REQUIRE 'SRC$:PATRTS.REQ';
1 REQUIRE 'HELPDEF';                                              ! Help options value definitions.
1
1 EXTERNAL ROUTINE
1         LBR$OUTPUT_HELP,                                        ! Interactive help facility
1         LIB$GET_INPUT,                                          ! Not currently required...here for future u
1         LIB$PUT_OUTPUT,                                         ! Writes the help text for LBR$OUTPUT_HELP
1         PAT$ADD_PAL,                                            ! Adds patch area to list
1         PAT$ALIGN_CMD,                                          ! Align command
1         PAT$DELETE_PATH,                                        ! Free up pathname storage
1         PAT$CANC_MODULE,                                        ! Cancels symbols for modules
1         PAT$DEFINE_SYM,                                         ! Define command
1         PAT$DEPOSIT_CMD,                                        ! Deposit command
1         PAT$ECO_CMDS,                                           ! Set eco level and check eco level commands
1         PAT$EXAMINE_CMD,                                        ! Examine command
1         PAT$FAO_PUT,                                            ! Formats an FAO line
1         PAT$FILL_BUF,                                           ! Updates and enlarges a buffer from a strin
1         PAT$FREE_ARG,                                           ! Frees elements of a command argument list
1         PAT$FREERELEASE,                                        ! Releases storage in dynamic allocation are
1         PAT$INIT_MODES,                                         ! Initializes modes
1         PAT$MAP_ADDR : NOVALUE,                                 ! Maps a virtual address
1         PAT$OPEN_COMFIL : NOVALUE,                              ! Opens command file for output
1         PAT$OUT_MEM_LOC,                                        ! Outputs values to output device
1         PAT$OUT_PAL_EXP,                                        ! Outputs PATCH Area address and size expres
1         PAT$REPLACE_CMD,                                        ! Replace command
1         PAT$RESET_DEF,                                          ! Resets modes to initialization mode
1         PAT$SAVE_SCOPE,                                         ! Saves a current path name
1         PAT$SET_CONTEXT,                                        ! Initializes context bits
1         PAT$SET_MODULE,                                         ! Sets up symbols for modules
1         PAT$SET_MOD_LST,                                        ! Sets mode list
1         PAT$SET_MOD_LVL,                                        ! Sets mode pointer
1         PAT$SET_NEW_MOD,                                        ! Sets new modes
1         PAT$SHOW_DEFAL,                                         ! Show default command
1         PAT$SHOW_MODULE,                                        ! Show module command
1         PAT$SHOW_SCOPE,                                         ! Show scope command
1         PAT$WRITE_EXP1 : NOVALUE,                               ! Writes expressions to the command file
1         PAT$WRITEFILE : NOVALUE,                                ! Writes data to a file
1         PAT$WRITE_INS : NOVALUE,                                ! Writes instruction-type command arguments
1         PAT$WRITE_NAME : NOVALUE,                               ! Writes names to the command file
1         PAT$WRTIMG : NOVALUE;                                   ! Writes out new patched image
1
1 EXTERNAL
1         PAT$GL_HELP_LIN : BLOCK [8,BYTE],                       ! Global descriptor to remainder of command
1         PAT$GB_MOD_PTR : REF VECTOR[,BYTE],                     ! Current mode pointer
1         PAT$GL_ECO_UPD : BITVECTOR,                             ! Update qualifier eco mask
1         PAT$GB_EXEC_CMD : BYTE,                                 ! Indicator whether or not to execute patch
1         PAT$GL_CSP_PTR : REF PATHNAME_VECTOR,                   ! Current scope position
1         PAT$GL_COMQUAL : BITVECTOR,                             ! Command qualifier indicators
1         PAT$GL_IHPPTR : REF BLOCK[,BYTE],                       ! Pointer to patch section of image header
1         PAT$CP_OUT_STR,                                         ! Pointer to output buffer
1         PAT$GL_BUF_SIZ,                                         ! Size of data written into output buffer
1         PAT$GL_COMRAB,                                          ! Command file RAB
1         PAT$GL_FLAGS : BITVECTOR [32],                                ! CLI flags
1         PAT$GL_RLOC_BUF : BLOCK[,BYTE],                         ! Descriptor for relocation buffer
1         PAT$GL_TEMP_BUF : BLOCK[,BYTE],                         ! Descriptor temporary deposit buffer
```

```
1               PAT$GL_OLD_ASD : BLOCK[,BYTE],                          ! Descriptor for old contents assembler dire
1               PAT$GL_NEW_ASD : BLOCK[,BYTE],                          ! Descriptor for new contents assembler dire
1               PAT$GB_SUBST_IN : VECTOR[,BYTE],                        ! Buffer for substitution instructions
1               PAT$GL_FWRLHD,                                          ! FoWard Reference table listhead
1               PAT$CP_INP_DSCS : REF VECTOR [, LONG],                  ! Table of input string descriptors
1               PAT$GB_TAKE_CMD: BYTE,                                  ! Flag which says continue to accept command
1               PAT$GL_CONTEXT: BITVECTOR,                              ! Context word
1               PAT$GL_HEAD_LST,                                        ! Head of command argument list
1               PAT$GL_JNLRAB,                                          ! Journal file RAB
1               PAT$GL_SEMAN1 : VECTOR,                                 ! Token stack for parser
1               PAT$GL_IMGHDR : REF BLOCK[,BYTE],                       ! Image header pointer
1               PAT$GL_PATAREA : REF BLOCK[,BYTE],                      ! Patch area descriptor pointer
1               PAT$GL_OLDLABLS,                                        ! Pointer to listhead for old contents label
1               PAT$GL_NEWLABLS,                                        ! Pointer to listhead for new contents un-re
1               PAT$GL_RLCLABLS,                                        ! Pointer to listhead for new contents reloc
1               PAT$GL_SYMTBPTR,                                        ! Pointer to current symbol table listhead
1               PAT$GL_SYMHEAD;                                         ! Listhead for user-defined symbol table
1
1       !
1       ! COMMAND VERB STRINGS
1       !
1       BIND
1               ALIGN_CMD       =       UPLIT BYTE (%ASCIC 'AL ') : VECTOR[,BYTE],
1               CANCEL_MODE_CMD =       UPLIT BYTE (%ASCIC 'CA M') : VECTOR[,BYTE],
1               CANCEL_MODU_CMD =       UPLIT BYTE (%ASCIC 'CA MODU') : VECTOR[,BYTE],
1               CAN_MOD_ALL_CMD =       UPLIT BYTE (%ASCIC 'CA MODU /ALL') : VECTOR[,BYTE],
1               CANCEL_SCO_CMD  =       UPLIT BYTE (%ASCIC 'CA SC') : VECTOR[,BYTE],
1               CANCEL_PAT_CMD  =       UPLIT BYTE (%ASCIC 'CA PAT') : VECTOR[,BYTE],
1               CHECK_N_ECO_CMD =       UPLIT BYTE (%ASCIC 'CH NOT EC') : VECTOR[,BYTE],
1               CHECK_ECO_CMD   =       UPLIT BYTE (%ASCIC 'CH EC') : VECTOR[,BYTE],
1               DEFINE_CMD      =       UPLIT BYTE (%ASCIC 'DEF') : VECTOR[,BYTE],
1               DELETE_CMD      =       UPLIT BYTE (%ASCIC 'DEL ') : VECTOR[,BYTE],
1               DEPOSIT_CMD     =       UPLIT BYTE (%ASCIC 'D ') : VECTOR[,BYTE],
1               EXAMINE_CMD     =       UPLIT BYTE (%ASCIC 'E ') : VECTOR[,BYTE],
1               EVALUATE_CMD    =       UPLIT BYTE (%ASCIC 'EV') : VECTOR[,BYTE],
1               EXIT_CMD        =       UPLIT BYTE (%ASCIC 'EXI') : VECTOR[,BYTE],
1               HELP_CMD        =       UPLIT BYTE (%ASCIC 'H ') : VECTOR[,BYTE],
1               INSERT_CMD      =       UPLIT BYTE (%ASCIC 'INSE ') : VECTOR[,BYTE],
1               NAME_CMD        =       UPLIT BYTE (%ASCIC '!AD') : VECTOR[,BYTE],
1               REPLACE_CMD     =       UPLIT BYTE (%ASCIC 'RE ') : VECTOR[,BYTE],
1               SCO_NAM_CMD     =       UPLIT BYTE (%ASCIC '!AC') : VECTOR[,BYTE],
1               SET_ECO_CMD     =       UPLIT BYTE (%ASCIC 'SE EC') : VECTOR[,BYTE],
1               SET_MODE_CMD    =       UPLIT BYTE (%ASCIC 'SE M') : VECTOR[,BYTE],
1               SET_MODU_CMD    =       UPLIT BYTE (%ASCIC 'SE MODU') : VECTOR[,BYTE],
1               SET_MOD_ALL_CMD =       UPLIT BYTE (%ASCIC 'SE MODU /ALL') : VECTOR[,BYTE],
1               SET_PAT_CMD     =       UPLIT BYTE (%ASCIC 'SE PAT') : VECTOR[,BYTE],
1               SET_SCO_CMD     =       UPLIT BYTE (%ASCIC 'SE SC') : VECTOR[,BYTE],
1               SHOW_MODE_CMD   =       UPLIT BYTE (%ASCIC 'SH M') : VECTOR[,BYTE],
1               SHOW_MODU_CMD   =       UPLIT BYTE (%ASCIC 'SH MODU') : VECTOR[,BYTE],
1               SHOW_SCO_CMD    =       UPLIT BYTE (%ASCIC 'SH SC') : VECTOR[,BYTE],
1               UPDATE_CMD      =       UPLIT BYTE (%ASCIC 'U') : VECTOR[,BYTE],
1               VALUE_CMD       =       UPLIT BYTE (%ASCIC '^X!XL') : VECTOR[,BYTE],
1               VERIFY_CMD      =       UPLIT BYTE (%ASCIC 'V ') : VECTOR[,BYTE],
1
1       !++
1       ! Qualifiers for align command.
1       !--
```

```
        ALIGN_QUAL_TBL  =       UPLIT BYTE (
                                                %ASCII '/BYT',
                                                %ASCII '/WOR',
                                                %ASCII '/LON',
                                                %ASCII '/QUA',
                                                %ASCII '/PAG'
                                                        ) : VECTOR[,BYTE];

LITERAL
        ALIGN_QUAL_LNG = 4,                                     ! Length of align qualifiers
        NO_CASE_TABLE = 0,                                      ! Don't print CASE dispatch tables
        CASE_TABLE = 1,                                         ! Print CASE dispatch tables
        HELP_FLAGS =    HLP$M_PROCESS   OR                      ! Disallow HELP prompting only.
                        HLP$M_GROUP     OR                      ! Default Logical Name Table searching to
                        HLP$M_SYSTEM;                           !       Process, Group, and System.
```

```
GLOBAL ROUTINE PAT$END_OF_CMD (SEMSP) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Resets all PATCH context that is exclusive to a single PATCH command.
!       This includes resetting default modes from single line overrides back
!       to the actual default modes and resetting a large number of context bits.
!
!       This routine also releases any storage associated with parameters
!       stored for this command, more specifically for commands which
!       build descriptors for symbolic names.  It also releases any storage
!       used for assembler directive tables, forward reference tables, and
!       temporary deposit buffers.
!
! CALLING SEQUENCE:
!
!       PAT$END_OF_CMD (SEMSP)
!
! INPUTS:
!
!       SEMSP - Offset to command verb token on parse stack
!
! IMPLICIT INPUTS:
!
!       PAT$GL_RLOC_BUF - Descriptor for relocation buffer, if used
!       PAT$GL_TEMP_BUF - Descriptor for temporary buffer used on depositing
!                         new values into memory
!       PAT$GL_OLD_ASD - Descriptor for old contents assembler directive table
!       PAT$GL_NEW_ASD - Descriptor for new contents assembler directive table
!       PAT$GL_FWRLHD - Listhead for ForWard Reference table for instructions
!       PAT$GL_OLDLABLS - Pointer to listhead for old contents label list
!       PAT$GL_NEWLABLS - Pointer to listhead for new contents un-relocated label list
!       PAT$GL_RLCLABLS - Pointer to listhead for new contents relocated label list
!       PAT$GL_SYMTBPTR - Pointer to current symbol table listhead
!       PAT$GL_SYMHEAD - Pointer to user-defined symbol table listhead
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       none
!
! SIDE EFFECTS:
!
!       Defaults are re-established.
!       Any free storage used in symbolic name descriptors, forward reference
!       tables, and symbolic label lists is released.
!
!--
```

```
2 BEGIN
2
2 LOCAL
2         POINTER,                                                        ! Pointer to current command parameter
2         DESC_PTR : REF BLOCK[,BYTE];                                    ! Pointer to symbolic name descriptor
2
2 !++
2 ! This routine guarantees the internal consistency
2 ! of PATCH, and must succeed or give up.
2 !--
2 PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;                                      ! Reset the current symbol table to be user-
2 PAT$INIT_MODES (OVERRIDE_MODE, USER_DEF_MODE);
2 PAT$SET_MOD_LVL (USER_DEF_MODE);
2 PAT$SET_CONTEXT ();
2 PAT$GB_SUBST_IN[0] = 0;                                                 ! Allow no substitution instructions
2 PAT$GL_COMQUAL = 0;                                                     ! Set no qualifiers specified
2
2 !++
2 ! Now release any symbolic name descriptors used for this command.  The commands
2 ! which have these string descriptors are:  ALIGN, SET MODULE, CANCEL MODULE,
2 ! and DEFINE.
2 !--
2 IF (.PAT$GL_SEMAN1[.SEMSP] EQL ALIGN_TOKEN) OR
2    (.PAT$GL_SEMAN1[.SEMSP] EQL DEFINE_TOKEN) OR
3    (.PAT$GL_CONTEXT[MODULE_BIT])
2 THEN
3         BEGIN
3         POINTER = .PAT$GL_HEAD_LST;
3         WHILE .POINTER NEQA 0
3         DO
4                 BEGIN
4                 DESC_PTR = .LIST_ELEM_EXP1(.POINTER);
4                 PAT$FREERELEASE(.DESC_PTR, ((.DESC_PTR[DSC$W_LENGTH] + 3) /A_LONGWORD) + 2);
4                 POINTER = .LIST_ELEM_FLINK(.POINTER);
3                 END;
2         END;
2
2 !++
2 ! Free all storage used in argument accumulation and pathname building.
2 !--
2 PAT$FREE_ARG ();
2 PAT$DELETE_PATH ();
2
2 !++
2 ! Now release any temporary buffer storage used to deposit new values
2 ! into memory.  This is for commands REPLACE, INSERT, and DEPOSIT.
2 !--
3 IF (.PAT$GL_TEMP_BUF[DSC$W_LENGTH] NEQ 0)
2 THEN
3         BEGIN
3         PAT$FREERELEASE ( .PAT$GL_TEMP_BUF[DSC$A_POINTER],
3                         (.PAT$GL_TEMP_BUF[DSC$W_LENGTH] + 3)/4);
3         PAT$GL_TEMP_BUF[DSC$W_LENGTH] = 0;
3         PAT$GL_TEMP_BUF[DSC$A_POINTER] = 0;
2         END;
2
2 !++
2 ! Now release any relocation buffer storage used to deposit new instructions
2 ! into memory.  This is for commands REPLACE and INSERT.
2 !--
3 IF (.PAT$GL_RLOC_BUF[DSC$W_LENGTH] NEQ 0)
2 THEN
3         BEGIN
3         PAT$FREERELEASE ( .PAT$GL_RLOC_BUF[DSC$A_POINTER],
3                         (.PAT$GL_RLOC_BUF[DSC$W_LENGTH] + 3)/4);
3         PAT$GL_RLOC_BUF[DSC$W_LENGTH] = 0;
3         PAT$GL_RLOC_BUF[DSC$A_POINTER] = 0;
2         END;
2
2 !++
2 ! Now release any temporary buffer storage used for the new contents assembler
2 ! directive table.
2 !--
3 IF (.PAT$GL_NEW_ASD[DSC$W_LENGTH] NEQ 0)
2 THEN
3         BEGIN
3         PAT$FREERELEASE ( .PAT$GL_NEW_ASD[DSC$A_POINTER],
3                         (.PAT$GL_NEW_ASD[DSC$W_LENGTH] + 3)/4);
3         PAT$GL_NEW_ASD[DSC$W_LENGTH] = 0;
3         PAT$GL_NEW_ASD[DSC$A_POINTER] = 0;
2         END;
2
2 !++
2 ! Now release any temporary buffer storage used for the old contents assembler
2 ! directive table.
2 !--
3 IF (.PAT$GL_OLD_ASD[DSC$W_LENGTH] NEQ 0)
2 THEN
3         BEGIN
3         PAT$FREERELEASE ( .PAT$GL_OLD_ASD[DSC$A_POINTER],
3                         (.PAT$GL_OLD_ASD[DSC$W_LENGTH] + 3)/4);
3         PAT$GL_OLD_ASD[DSC$W_LENGTH] = 0;
3         PAT$GL_OLD_ASD[DSC$A_POINTER] = 0;
2         END;
2
2 !++
2 ! There may also be some ForWard Reference table (FWR) to be released.
2 !--
3 WHILE (.PAT$GL_FWRLHD NEQA 0)
2 DO
3         BEGIN
3         LOCAL
3                 TEMP_PTR : REF BLOCK[,BYTE];
3         TEMP_PTR = .PAT$GL_FWRLHD;
3         PAT$GL_FWRLHD = .TEMP_PTR[FWR$L_FLINK];
3         PAT$FREERELEASE(.TEMP_PTR, (FWR$C_SIZE + 3)/4);
2         END;
2
2 !++
2 ! Now release any space used temporarily for symbolic instruction labels on
2 ! old contents of locations.
2 !--
3 WHILE (.DLL_RLINK(.PAT$GL_OLDLABLS) NEQA .PAT$GL_OLDLABLS)
2 DO
```

```
3               BEGIN
3               POINTER = .DLL_RLINK(.PAT$GL_OLDLABLS);
3               DLL_RLINK(.PAT$GL_OLDLABLS) = .DLL_RLINK(.POINTER);
3               PAT$FREERELEASE(.POINTER, (.SYM_CHCOUNT(.POINTER) + 1 + 3)/4 + OVERHEAD_SYM - 1);
2               END;
2
2       !++
2       ! Now release any space used temporarily for un-relocated symbolic instruction
2       ! labels on new contents of locations.
2       !--
3       WHILE (.DLL_RLINK(.PAT$GL_NEWLABLS) NEQA .PAT$GL_NEWLABLS)
2       DO
3               BEGIN
3               PCINTER = .DLL_RLINK(.PAT$GL_NEWLABLS);
3               DLL_RLINK(.PAT$GL_NEWLABLS) = .DLL_RLINK(.POINTER);
3               PAT$FREERELEASE(.POINTER, (.SYM_CHCOUNT(.POINTER) + 1 + 3)/4 + OVERHEAD_SYM - 1);
2               END;
2
2       !++
2       ! Now release any space used temporarily for relocated symbolic instruction
2       ! labels on old contents of locations.
2       !--
3       WHILE (.DLL_RLINK(.PAT$GL_RLCLABLS) NEQA .PAT$GL_RLCLABLS)
2       DO
3               BEGIN
3               POINTER = .DLL_RLINK(.PAT$GL_RLCLABLS);
3               DLL_RLINK(.PAT$GL_RLCLABLS) = .DLL_RLINK(.POINTER);
3               PAT$FREERELEASE(.POINTER, (.SYM_CHCOUNT(.POINTER) + 1 + 3)/4 + OVERHEAD_SYM - 1);
2               END;
1       END;


                                                         .TITLE  PATACT
                                                         .IDENT  \V04-000\

                                                         .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                20  4C  41  03  00000 P.AAA:  .ASCII  <3>\AL \                   ;
                            4D  20  41  43  04  00004 P.AAB:  .ASCII  <4>\CA M\                  ;
                55  44  4F  4D  20  41  43  07  00009 P.AAC:  .ASCII  <7>\CA MODU\               ;
4C  4C  41  2F  20  55  44  4F  4D  20  41  43  0C  00011 P.AAD:  .ASCII  <12>\CA MODU /ALL\     ;
                            43  53  20  41  43  05  0001E P.AAE:  .ASCII  <5>\CA SC\             ;
                    54  41  50  20  41  43  06  00024 P.AAF:  .ASCII  <6>\CA PAT\                ;
        43  45  20  54  4F  4E  20  48  43  09  0002B P.AAG:  .ASCII  <9>\CH NOT EC\             ;
                        43  45  20  48  43  05  00035 P.AAH:  .ASCII  <5>\CH EC\                 ;
                                46  45  44  03  0003B P.AAI:  .ASCII  <3>\DEF\                   ;
                            20  4C  45  44  04  0003F P.AAJ:  .ASCII  <4>\DEL \                  ;
                                    20  44  02  00044 P.AAK:  .ASCII  <2>\D \                    ;
                                    20  45  02  00047 P.AAL:  .ASCII  <2>\E \                    ;
                                    56  45  02  0004A P.AAM:  .ASCII  <2>\EV\                    ;
                                49  58  45  03  0004D P.AAN:  .ASCII  <3>\EXI\                   ;
                                    20  48  02  00051 P.AAO:  .ASCII  <2>\H \                    ;
                        20  45  53  4E  49  05  00054 P.AAP:  .ASCII  <5>\INSE \                 ;
                                44  41  21  03  0005A P.AAQ:  .ASCII  <3>\!AD\                   ;
                                20  45  52  03  0005E P.AAR:  .ASCII  <3>\RE \                   ;
                                43  41  21  03  00062 P.AAS:  .ASCII  <3>\!AC\                   ;
                        43  45  20  45  53  05  00066 P.AAT:  .ASCII  <5>\SE EC\                 ;
```

```
                                    4D  20  45  53  04  0006C P.AAU:  .ASCII  <4>\SE M\
                        55  44  4F  4D  20  45  53  07  00071 P.AAV:  .ASCII  <7>\SE MODU\
4C  4C  41  2F  20      55  44  4F  4D  20  45  53  0C  00079 P.AAW:  .ASCII  <12>\SE MODU /ALL\
                                54  41  50  20  45  53  06  00086 P.AAX:  .ASCII  <6>\SE PAT\
                                    43  53  20  45  53  05  0008D P.AAY:  .ASCII  <5>\SE SC\
                                    4D  20  48  53  04  00093 P.AAZ:  .ASCII  <4>\SH M\
                        55  44  4F  4D  20  48  53  07  00098 P.ABA:  .ASCII  <7>\SH MODU\
                                    43  53  20  48  53  05  000A0 P.ABB:  .ASCII  <5>\SH SC\
                                                55  01  000A6 P.ABC:  .ASCII  <1>\U\
                            4C  58  21  58  5E  05  000A8 P.ABD:  .ASCII  <5>\^X!XL\
                                            20  56  02  000AE P.ABE:  .ASCII  <2>\V \
                                    54  59  42  2F  000B1 P.ABF:  .ASCII  \/BYT\
                                    52  4F  57  2F  000B5         .ASCII  \/WOR\
                                    4E  4F  4C  2F  000B9         .ASCII  \/LON\
                                    41  55  51  2F  000BD         .ASCII  \/QUA\
                                    47  41  50  2F  000C1         .ASCII  \/PAG\

                                                      ISE$C_SIZE==           20
                                                      TXT$C_SIZE==           4
                                                      PAL$C_SIZE==           16
                                                      ASD$C_SIZE==           9
                                                      FWR$C_SIZE==           24
                                                      ALIGN_CMD=             P.AAA
                                                      CANCEL_MODE_CMD=       P.AAB
                                                      CANCEL_MODU_CMD=       P.AAC
                                                      CAN_MOD_ALL_CMD=       P.AAD
                                                      CANCEL_SCO_CMD=        P.AAE
                                                      CANCEL_PAT_CMD=        P.AAF
                                                      CHECK_N_ECO_CMD=       P.AAG
                                                      CHECK_ECO_CMD=         P.AAH
                                                      DEFINE_CMD=            P.AAI
                                                      DELETE_CMD=            P.AAJ
                                                      DEPOSIT_CMD=           P.AAK
                                                      EXAMINE_CMD=           P.AAL
                                                      EVALUATE_CMD=          P.AAM
                                                      EXIT_CMD=              P.AAN
                                                      HELP_CMD=              P.AAO
                                                      INSERT_CMD=            P.AAP
                                                      NAME_CMD=              P.AAQ
                                                      REPLACE_CMD=           P.AAR
                                                      SCO_NAM_CMD=           P.AAS
                                                      SET_ECO_CMD=           P.AAT
                                                      SET_MODE_CMD=          P.AAU
                                                      SET_MODU_CMD=          P.AAV
                                                      SET_MOD_ALL_CMD=       P.AAW
                                                      SET_PAT_CMD=           P.AAX
                                                      SET_SCO_CMD=           P.AAY
                                                      SHOW_MODE_CMD=         P.AAZ
                                                      SHOW_MODU_CMD=         P.ABA
                                                      SHOW_SCO_CMD=          P.ABB
                                                      UPDATE_CMD=            P.ABC
                                                      VALUE_CMD=             P.ABD
                                                      VERIFY_CMD=            P.ABE
                                                      ALIGN_QUAL_TBL=        P.ABF
                                                              .EXTRN  PAT$FAO_OUT, LBR$OUTPUT_HELP
                                                              .EXTRN  LIB$GET_INPUT, LIB$PUT_OUTPUT
                                                              .EXTRN  PAT$ADD_PAL, PAT$ALIGN_CMD
```

```
                                                   .EXTRN   PAT$DELETE_PATH
                                                   .EXTRN   PAT$CANC_MODULE
                                                   .EXTRN   PAT$DEFINE_SYM, PAT$DEPOSIT_CMD
                                                   .EXTRN   PAT$ECO_CMDS, PAT$EXAMINE_CMD
                                                   .EXTRN   PAT$FAO_PUT, PAT$FILL_BUF
                                                   .EXTRN   PAT$FREE_ARG, PAT$FREERELEASE
                                                   .EXTRN   PAT$INIT_MODES, PAT$MAP_ADDR
                                                   .EXTRN   PAT$OPEN_COMFIL
                                                   .EXTRN   PAT$OUT_MEM_LOC
                                                   .EXTRN   PAT$OUT_PAL_EXP
                                                   .EXTRN   PAT$REPLACE_CMD
                                                   .EXTRN   PAT$RESET_DEF, PAT$SAVE_SCOPE
                                                   .EXTRN   PAT$SET_CONTEXT
                                                   .EXTRN   PAT$SET_MODULE, PAT$SET_MOD_LST
                                                   .EXTRN   PAT$SET_MOD_LVL
                                                   .EXTRN   PAT$SET_NEW_MOD
                                                   .EXTRN   PAT$SHOW_DEFAL, PAT$SHOW_MODULE
                                                   .EXTRN   PAT$SHOW_SCOPE, PAT$WRITE_EXP1
                                                   .EXTRN   PAT$WRITEFILE, PAT$WRITE_INS
                                                   .EXTRN   PAT$WRITE_NAME, PAT$WRTIMG
                                                   .EXTRN   PAT$GL_HELP_LIN
                                                   .EXTRN   PAT$GB_MOD_PTR, PAT$GL_ECO_UPD
                                                   .EXTRN   PAT$GB_EXEC_CMD
                                                   .EXTRN   PAT$GL_CSP_PTR, PAT$GL_COMQUAL
                                                   .EXTRN   PAT$GL_IHPPTR, PAT$CP_OUT_STR
                                                   .EXTRN   PAT$GL_BUF_SIZ, PAT$GL_COMRAB
                                                   .EXTRN   PAT$GL_FLAGS, PAT$GL_RLOC_BUF
                                                   .EXTRN   PAT$GL_TEMP_BUF
                                                   .EXTRN   PAT$GL_OLD_ASD, PAT$GL_NEW_ASD
                                                   .EXTRN   PAT$GB_SUBST_IN
                                                   .EXTRN   PAT$GL_FWRLHD, PAT$CP_INP_DSCS
                                                   .EXTRN   PAT$GB_TAKE_CMD
                                                   .EXTRN   PAT$GL_CONTEXT, PAT$GL_HEAD_LST
                                                   .EXTRN   PAT$GL_JNLRAB, PAT$GL_SEMANT
                                                   .EXTRN   PAT$GL_IMGHDR, PAT$GL_PATAREA
                                                   .EXTRN   PAT$GL_OLDLABLS
                                                   .EXTRN   PAT$GL_NEWLABLS
                                                   .EXTRN   PAT$GL_RLCLABLS
                                                   .EXTRN   PAT$GL_SYMTBPTR
                                                   .EXTRN   PAT$GL_SYMHEAD
                                                   .WEAK    ACCESS_CHECK

                                                   .PSECT   _PAT$CODE,NOWRT,2

                                 03FC 00000        .ENTRY   PAT$END_OF_CMD, Save R2,R3,R4,R5,R6,R7,R8,- ; 3285
                                                            R9
              59 00000000G  00   9E 00002        MOVAB    PAT$GL_FWRLHD, R9                             ;
              58 00000000G  00   9E 00009        MOVAB    PAT$GL_OLD_ASD, R8                            ;
              57 00000000G  00   9E 00010        MOVAB    PAT$GL_NEW_ASD, R7                            ;
              56 00000000G  00   9E 00017        MOVAB    PAT$GL_RLOC_BUF, R6                           ;
              55 00000000G  00   9E 0001E        MOVAB    PAT$GL_TEMP_BUF, R5                           ;
              54 00000000G  00   9E 00025        MOVAB    PAT$FREERELEASE, R4                           ;
 00000000G  00 00000000G  00   D0 0002C        MOVL     PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR               ; 3352
                            01   DD 00037        PUSHL    #1                                            ; 3353
                            02   DD 00039        PUSHL    #2                                            ;
 00000000G  00              02   FB 0003B        CALLS    #2, PAT$INIT_MODES                            ;
                            01   DD 00042        PUSHL    #1                                            ; 3354
```

```
          00000000G  00             01  FB 00044           CALLS    #1, PAT$SET_MOD_LVL                    ;
          00000000G  00             00  FB 0004B           CALLS    #0, PAT$SET_CONTEXT                    ; 3355
                      00000000G     00  94 00052           CLRB     PAT$GB_SUBST_IN                        ; 3356
                      00000000G     00  D4 00058           CLRL     PAT$GL_COMQUAL                         ; 3357
                     50        04   AC  D0 0005E           MOVL     SEMSP, R0                              ; 3364
                     50 00000000G0040  D0 00062           MOVL     PAT$GL_SEMAN1[R0], R0                  ;
                     01             50  D1 0006A           CMPL     R0, #1                                 ;
                                    0D  13 0006D           BEQL     1$                                     ;
                     05             50  D1 0006F           CMPL     R0, #5                                 ; 3365
                                    08  13 00072           BEQL     1$                                     ;
                      00000000G     00  95 00074           TSTB     PAT$GL_CONTEXT                         ; 3366
                                    23  18 0007A           BGEQ     3$                                     ;
                     52 00000000G   00  D0 0007C 1$:       MOVL     PAT$GL_HEAD_LST, POINTER               ; 3369
                                    1A  13 00083 2$:       BEQL     3$                                     ; 3370
                     53        04   A2  D0 00085           MOVL     4(POINTER), DESC_PTR                   ; 3373
                     50             63  3C 00089           MOVZWL   (DESC_PTR), R0                         ; 3374
                     50             03  C0 0008C           ADDL2    #3, R0                                 ;
                     50             04  C6 0008F           DIVL2    #4, R0                                 ;
                               02   A0  9F 00092           PUSHAB   2(R0)                                  ;
                                    53  DD 00095           PUSHL    DESC_PTR                               ;
                     64             02  FB 00097           CALLS    #2, PAT$FREERELEASE                    ;
                     52             62  D0 0009A           MOVL     (POINTER), POINTER                     ; 3375
                                    E4  11 0009D           BRB      2$                                     ; 3370
          00000000G  00             00  FB 0009F 3$:       CALLS    #0, PAT$FREE_ARG                       ; 3382
          00000000G  00             00  FB 000A6           CALLS    #0, PAT$DELETE_PATH                    ; 3383
                     50             65  3C 000AD           MOVZWL   PAT$GL_TEMP_BUF, R0                    ; 3389
                                    12  13 000B0           BEQL     4$                                     ;
                     50             03  C0 000B2           ADDL2    #3, R0                                 ; 3393
      7E             50             04  C7 000B5           DIVL3    #4, R0, -(SP)                          ;
                               04   A5  DD 000B9           PUSHL    PAT$GL_TEMP_BUF+4                      ; 3392
                     64             02  FB 000BC           CALLS    #2, PAT$FREERELEASE                    ;
                                    65  B4 000BF           CLRW     PAT$GL_TEMP_BUF                        ; 3394
                               04   A5  D4 000C1           CLRL     PAT$GL_TEMP_BUF+4                      ; 3395
                     50             66  3C 000C4 4$:       MOVZWL   PAT$GL_RLOC_BUF, R0                    ; 3402
                                    12  13 000C7           BEQL     5$                                     ;
                     50             03  C0 000C9           ADDL2    #3, R0                                 ; 3406
      7E             50             04  C7 000CC           DIVL3    #4, R0, -(SP)                          ;
                               04   A6  DD 000D0           PUSHL    PAT$GL_RLOC_BUF+4                      ; 3405
                     64             02  FB 000D3           CALLS    #2, PAT$FREERELEASE                    ;
                                    66  B4 000D6           CLRW     PAT$GL_RLOC_BUF                        ; 3407
                               04   A6  D4 000D8           CLRL     PAT$GL_RLOC_BUF+4                      ; 3408
                     50             67  3C 000DB 5$:       MOVZWL   PAT$GL_NEW_ASD, R0                     ; 3415
                                    12  13 000DE           BEQL     6$                                     ;
                     50             03  C0 000E0           ADDL2    #3, R0                                 ; 3419
      7E             50             04  C7 000E3           DIVL3    #4, R0, -(SP)                          ;
                               04   A7  DD 000E7           PUSHL    PAT$GL_NEW_ASD+4                       ; 3418
                     64             02  FB 000EA           CALLS    #2, PAT$FREERELEASE                    ;
                                    67  B4 000ED           CLRW     PAT$GL_NEW_ASD                         ; 3420
                               04   A7  D4 000EF           CLRL     PAT$GL_NEW_ASD+4                       ; 3421
                     50             68  3C 000F2 6$:       MOVZWL   PAT$GL_OLD_ASD, R0                     ; 3428
                                    12  13 000F5           BEQL     7$                                     ;
                     50             03  C0 000F7           ADDL2    #3, R0                                 ; 3432
      7E             50             04  C7 000FA           DIVL3    #4, R0, -(SP)                          ;
                               04   A8  DD 000FE           PUSHL    PAT$GL_OLD_ASD+4                       ; 3431
                     64             02  FB 00101           CALLS    #2, PAT$FREERELEASE                    ;
                                    68  B4 00104           CLRW     PAT$GL_OLD_ASD                         ; 3433
                               04   A8  D4 00106           CLRL     PAT$GL_OLD_ASD+4                       ; 3434
```

```
                   50            69 D0 00109 7$:        MOVL    PAT$GL_FWRLHD, R0                                      ; 3440
                                 0C 13 0010C            BEQL    8$                                                     ;
                   69            60 D0 0010E            MOVL    (TEMP_PTR), PAT$GL_FWRLHD                              ; 3446
                                 06 DD 00111            PUSHL   #6                                                     ; 3447
                                 50 DD 00113            PUSHL   TEMP_PTR                                               ;
                   64            02 FB 00115            CALLS   #2, PAT$FREERELEASE                                    ;
                                 EF 11 00118            BRB     7$                                                     ; 3440
                   50 00000000G  00 D0 0011A 8$:        MOVL    PAT$GL_OLDLABLS, R0                                    ; 3454
                   50            60 D1 00121            CMPL    (R0), R0                                               ;
                                 1A 13 00124            BEQL    9$                                                     ;
                   52            60 D0 00126            MOVL    (R0), POINTER                                          ; 3457
                   60            62 D0 00129            MOVL    (POINTER), (R0)                                        ; 3458
                   50        0C  A2 9A 0012C            MOVZBL  12(POINTER), R0                                        ; 3459
                   50            04 C0 00130            ADDL2   #4, R0                                                 ;
                   50            04 C6 00133            DIVL2   #4, R0                                                 ;
                             03  A0 9F 00136            PUSHAB  3(R0)                                                  ;
                                 52 DD 00139            PUSHL   POINTER                                                ;
                   64            02 FB 0013B            CALLS   #2, PAT$FREERELEASE                                    ;
                                 DA 11 0013E            BRB     8$                                                     ; 3454
                   50 00000000G  00 D0 00140 9$:        MOVL    PAT$GL_NEWLABLS, R0                                    ; 3466
                   50            60 D1 00147            CMPL    (R0), R0                                               ;
                                 1A 13 0014A            BEQL    10$                                                    ;
                   52            60 D0 0014C            MOVL    (R0), POINTER                                          ; 3469
                   60            62 D0 0014F            MOVL    (POINTER), (R0)                                        ; 3470
                   50        0C  A2 9A 00152            MOVZBL  12(POINTER), R0                                        ; 3471
                   50            04 C0 00156            ADDL2   #4, R0                                                 ;
                   50            04 C6 00159            DIVL2   #4, R0                                                 ;
                             03  A0 9F 0015C            PUSHAB  3(R0)                                                  ;
                                 52 DD 0015F            PUSHL   POINTER                                                ;
                   64            02 FB 00161            CALLS   #2, PAT$FREERELEASE                                    ;
                                 DA 11 00164            BRB     9$                                                     ; 3466
                   50 00000000G  00 D0 00166 10$:       MOVL    PAT$GL_RLCLABLS, R0                                    ; 3478
                   50            60 D1 0016D            CMPL    (R0), R0                                               ;
                                 1A 13 00170            BEQL    11$                                                    ;
                   52            60 D0 00172            MOVL    (R0), POINTER                                          ; 3481
                   60            62 D0 00175            MOVL    (POINTER), (R0)                                        ; 3482
                   50        0C  A2 9A 00178            MOVZBL  12(POINTER), R0                                        ; 3483
                   50            04 C0 0017C            ADDL2   #4, R0                                                 ;
                   50            04 C6 0017F            DIVL2   #4, R0                                                 ;
                             03  A0 9F 00182            PUSHAB  3(R0)                                                  ;
                                 52 DD 00185            PUSHL   POINTER                                                ;
                   64            02 FB 00187            CALLS   #2, PAT$FREERELEASE                                    ;
                                 DA 11 0018A            BRB     10$                                                    ; 3478
                                    04 0018C 11$:       RET                                                            ; 3485

; Routine Size:  397 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
1 GLOBAL ROUTINE PAT$END_OF_LINE (SEMSP) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Calls the PAT$END_OF_CMD to reset all patch context that is
1 !       exclusive to a singe PATCH command.  This includes resetting default
1 !       modes from single line overrides back to the actual default modes and
1 !       resetting a large number of context bits.  In addition, any free
1 !       storage required temporarily is released.
1 !
1 !       Also, the command line buffer is released.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$END_OF_LINE (SEMSP)
1 !
1 ! INPUTS:
1 !
1 !       SEMSP - Offset to command verb on parse stack
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$CP_INP_DSCS - Address of vector of command line buffer descriptors,
1 !                         first longword of which is count of descriptors
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Defaults are reestablished.  The command line buffer space is released.
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       temp_loc;
2
2 !++
2 ! This routine guarantees the internal consistency
2 ! of PATCH, and must succeed or give up.
2 !--
2 PAT$END_OF_CMD(.SEMSP);
2
2 !++
2 ! Now release the command line buffer space.
```

```
;  502          3543  2 !--
;  503          3544  2 INCR LOOP FROM 1 TO .PAT$CP_INP_DSCS[0]*2 BY 2
;  504          3545  2 DO
;  505          3546  2         IF .PAT$CP_INP_DSCS[.LOOP] NEQ 0
;  506          3547  2         THEN
;  507          3548  3                 BEGIN
;  508          3549  3                 PAT$FREERELEASE (.PAT$CP_INP_DSCS [.LOOP+1],
;  509          3550  3                                 (.PAT$CP_INP_DSCS [.LOOP] + 3) / 4);
;  510          3551  3                 PAT$CP_INP_DSCS [.LOOP] = 0;
;  511          3552  3                 PAT$CP_INP_DSCS [.LOOP+1] = 0;
;  512          3553  3                 END
;  513          3554  2         ELSE
;  514          3555  2                 RETURN;
;  515          3556  1 END;



                                               001C 00000        .ENTRY  PAT$END_OF_LINE, Save R2,R3,R4                     ; 3486
                         54 00000000G  00   9E 00002        MOVAB   PAT$CP_INP_DSCS, R4
                                  04   AC   DD 00009        PUSHL   SEMSP                                                      ; 3539
                      FE62   CF        01   FB 0000C        CALLS   #1, PAT$END_OF_CMD
                             50        64   D0 00011        MOVL    PAT$CP_INP_DSCS, R0                                        ; 3544
                 53          60        01   78 00014        ASHL    #1, (R0), R3
                             52        01   CE 00018        MNEGL   #1, LOOP
                                       26   11 0001B        BRB     2$
                             50        64   D0 0001D 1$:    MOVL    PAT$CP_INP_DSCS, R0                                        ; 3546
                                     6042   D5 00020        TSTL    (R0)[LOOP]
                                       24   13 00023        BEQL    3$
                 51        6042        03   C1 00025        ADDL3   #3, (R0)[LOOP], R1                                         ; 3550
                 7E          51        04   C7 0002A        DIVL3   #4, R1, -(SP)
                                  04 A042   DD 0002E        PUSHL   4(R0)[LOOP]                                                ; 3549
                  00000000G  00        02   FB 00032        CALLS   #2, PAT$FREERELEASE
                             50        64   D0 00039        MOVL    PAT$CP_INP_DSCS, R0                                        ; 3551
                                     6042   D4 0003C        CLRL    (R0)[LOOP]
                                  04 A042   D4 0003F        CLRL    4(R0)[LOOP]                                                ; 3552
      FFD4       52          02        53   F1 00043 2$:    ACBL    R3, #2, LOOP, 1$                                           ; 3546
                                       04 00049 3$:         RET                                                                ; 3556

; Routine Size:  74 bytes,    Routine Base:  _PAT$CODE + 018D
```

```
GLOBAL ROUTINE PAT$PERFORM_CMD (SEMSP) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Action routine for a single PATCH command.  Based on the command verb
!       various routines are called to execute the command.  After the command
!       is executed, a cleanup is done to reset the "one line" modes to the
!       default modes and reset the context switches.  The command line is
!       written to the output command file, if one is being created.
!
! CALLING SEQUENCE:
!
!       PAT$PERFORM_CMD ()
!
! INPUTS:
!
!       SEMSP - Offset in parse stack which holds current verb token
!
! IMPLICIT INPUTS:
!
!       none
!
! OUTPUTS:
!
!       TRUE or FALSE, depending on whether parsing is to continue or not.
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!        TRUE or FALSE
!
! SIDE EFFECTS:
!
!       A PATCH command is actually executed.
!
!--

BEGIN

LOCAL
        BIT_NUMBER,                                     ! ECO bit number
        ECOLVL_PTR : REF BITVECTOR,                     ! Pointer to ECO bits in image header
        OUTPUT_BUF : VECTOR[NO_OF_INP_CHARS,BYTE],      ! Buffer for journal file output
        ISE_PTR;                                        ! Pointer to image section descriptor

!++
! If the /UPDATE qualifier was specified, then the execute command indicator,
! PAT$GB_EXEC_CMD, may be set to FALSE indicating the current patch session
! should be skipped.  If this is the case, then don't bother to execute the
! command unless it is a new "SET ECO" level, indicating a new patch session.
! If /UPDATE was not specified, then the execute command indicator is always
! TRUE.  In this case, execute the complete command.  In all cases, the "EXIT"
! command must be executed.
```

```
;  2 !--
;  2 IF (.PAT$GB_EXEC_CMD) OR
;  2     (.PAT$GL_CONTEXT[SET_ECO]) OR
;  3     (.PAT$GL_SEMAN1[.SEMSP] EQL EXIT_TOKEN)
;  2 THEN
;  3         BEGIN
;  3         CASE .PAT$GL_SEMAN1 [.SEMSP] FROM ALIGN_TOKEN TO VERIFY_TOKEN OF
;  3
;  3         SET
;  3
;  3         [ALIGN_TOKEN]:
;  3                 IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]
;  3                 THEN
;  3                     SIGNAL (PAT$_INVCMDABS)
;  3                 ELSE
;  3                     PAT$ALIGN_CMD ();
;  3
;  3         [CANCEL_TOKEN]:
;  3                 IF .PAT$GL_CONTEXT[MODE_BIT]
;  3                 THEN
;  3                     PAT$RESET_DEF()
;  3                 ELSE
;  3
;  3                 IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]
;  3                 THEN
;  3                     SIGNAL (PAT$_INVCMDABS)
;  3                 ELSE
;  3
;  3                 SELECTONE TRUE OF
;  3                 SET
;  3
;  3                 [.PAT$GL_CONTEXT[PAT_AREA_BIT]]:
;  3                         PAT$GL_PATAREA = CH$PTR(PAT$GL_IHPPTR[IHP$L_RW_PATSIZ], 0);
;  3
;  3                 [.PAT$GL_CONTEXT[MODULE_BIT]]:
;  3                         PAT$CANC_MODULE();
;  3
;  3                 [.PAT$GL_CONTEXT[SCOPE_BIT]]:
;  3                         PAT$SAVE_SCOPE(FALSE);
;  3                 TES;
;  3
;  3         [CHECK_TOKEN]:
;  3                 IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]
;  3                 THEN
;  3                     SIGNAL (PAT$_INVCMDABS)
;  3                 ELSE
;  3                     PAT$ECO_CMDS ();
;  3
;  3         [CREATE_TOKEN]:
;  3                 PAT$OPEN_COMFIL(0, 0);
;  3
;  3         [DEFINE_TOKEN]:
;  4                 BEGIN
;  4
;  4                 LOCAL
;  4                         POINTER;
;  4
```

```
4                  POINTER = .PAT$GL_HEAD_LST;
5                  WHILE (.POINTER NEQ 0)
4                  DO
5                          BEGIN
5                          PAT$DEFINE_SYM (.LIST_ELEM_EXP1 (.POINTER), .LIST_ELEM_EXP2 (.POINTER), TRUE);
5                          POINTER = .LIST_ELEM_FLINK (.POINTER);
4                          END;
3                  END;
3
3          [DELETE_TOKEN]:
4                  BEGIN
4                  PAT$GL_CONTEXT [DELETE_BIT] = TRUE;
4                  PAT$DEPOSIT_CMD ();
3                  END;
3
3          [DEPOSIT_TOKEN]:
3                  PAT$DEPOSIT_CMD ();
3
3          [EXAMINE_TOKEN]:
4                  BEGIN
4                  PAT$GL_CONTEXT [EXAMINE_BIT] = TRUE;
4                  PAT$EXAMINE_CMD ();
3                  END;
3
3          [EVALUATE_TOKEN]:
4                  BEGIN
4                  LOCAL
4                          POINTER;
4                  POINTER = .PAT$GL_HEAD_LST;
5                  WHILE (.POINTER NEQ 0)
4                  DO
5                          BEGIN
5                          PAT$OUT_MEM_LOC (LIST_ELEM_EXP1 (.POINTER), 0, CASE_TABLE);
5                          POINTER = .LIST_ELEM_FLINK (.POINTER);
4                          END;
3                  END;
3
3          [EXIT_TOKEN]:
4                  BEGIN
4                  PAT$GB_TAKE_CMD = FALSE;
4                  IF (.PAT$GL_FLAGS AND PAT$M_UPDATE) NEQ 0
4                  THEN
5                          BEGIN
5                          ECOLVL_PTR = CH$PTR(PAT$GL_IHPPTR[IHP$L_ECO1], 0);
5                          INCR BIT_NUMBER FROM PAT$K_MIN_ECO-1 TO PAT$K_MAX_ECO-1
5                          DO
5                                  IF .PAT$GL_ECO_UPD[.BIT_NUMBER]
5                                  THEN
5                                          IF NOT .ECOLVL_PTR[.BIT_NUMBER]
5                                          THEN
5                                                  SIGNAL(PAT$_NOUPDATE, 1, .BIT_NUMBER+1);
4                          END;
3                  END;
3
3          [HELP_TOKEN]:
3                  LBR$OUTPUT_HELP (LIB$PUT_OUTPUT,,PAT$GL_HELP_LIN, %ASCID 'PATCHHELP', %REF (HELP_FLAGS), LIB
3
```

```
[INSERT_TOKEN]:
        BEGIN
        IF (NOT .PAT$GB_MOD_PTR[MODE_INSTRUC])
        THEN
                SIGNAL(PAT$_INVCMDABS);
        PAT$GL_CONTEXT [INSERT_BIT] = TRUE;
        PAT$REPLACE_CMD ();
        END;

[REPLACE_TOKEN]:
        PAT$REPLACE_CMD ();

[SET_TOKEN]:
        IF .PAT$GL_CONTEXT[MODE_BIT]
        THEN
                BEGIN
                !++
                ! The "SET MODE" command verb must be written to the
                ! indirect command file here as the modes to be "SET"
                ! are output in PAT$SET_MOD_LST and the information
                ! lost.  Therefore, only the "EXIT" to the "NEW>" prompt
                ! will be output in the routine, WRITE_CMD.
                !--
                PAT$WRITEFILE(.SET_MODE_CMD[0], SET_MODE_CMD[1], PAT$GL_COMRAB);
                PAT$SET_MOD_LST (USER_DEF_MODE);
                END
        ELSE

        IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]
        THEN
            SIGNAL (PAT$_INVCMDABS)
        ELSE

        SELECTONE TRUE OF
        SET

        [.PAT$GL_CONTEXT[SCOPE_BIT]]:
                PAT$SAVE_SCOPE (TRUE);

        [.PAT$GL_CONTEXT[SET_ECO]]:
                PAT$ECO_CMDS ();

        [.PAT$GL_CONTEXT[PAT_AREA_BIT]]:

                BEGIN
                PAT$MAP_ADDR(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST),
                                PAT$GL_PATAREA, ISE_PTR);
                !++
                !The SET PATCH_AREA command may have a /INITIALIZE=size expression
                !qualifier included.  If its present, then check first that the size
                !value is not larger than the patch area.  If size is to big then, we
                !assure that sufficient space exists to accomodate the patch area
                !descriptor plus a longword (12 bytes).  If space does exists then we
                !set the default size to the size of the unused portion of the patch
                !area image section, informing the user of course.  Else, we signal
                !an informative error message stating the address and amount of space
                !available.  Next, check to make sure that the patch area has not already
                !been initialized.  If it has, issue a warning to the user and set up the
                !descriptor info.  If it has not been previously initialized then take the
                !size value and insert it into the first long word of the patch area and
                !set the second long word to point to the succeeding long word (eg. .+4).
                !
                !*** NOTE ***  The size value that is inserted into the first long word
                ! is reduced by 8 (the size of the descriptor) to reflect the fact that
                ! we have eaten up this space with the descriptor.
                !
                !       Also note, that since the address of the patch area is synomous
                ! with the address of the patch area descriptor, updating the pointer
                ! PAT$GL_PATAREA is not necessary.
                !--

        IF (.PAT$GL_CONTEXT [INIT_PAT_BIT]) THEN
                BEGIN
                BIND    PATCH_AREA = .PAT$GL_PATAREA : VECTOR [, LONG],
                        FIRST_AVAIL_ADR = LIST_ELEM_EXP1[.PAT$GL_HEAD_LST],
                        INITIAL_SIZE = LIST_ELEM_EXP2[.PAT$GL_HEAD_LST];

                LOCAL
                        AVAIL_BYTE_CNT,                         !Number of available
                        ISD_PTR : REF BLOCK [, BYTE];           !Points to the curre

                ISD_PTR = CH$PTR (.ISE_PTR, ISE$C_SIZE);
                AVAIL_BYTE_CNT = (.ISD_PTR[ISD$W_PAGCNT] * 512)
                                - (.FIRST_AVAIL_ADR - (.ISD_PTR[ISD$L_VPNPFC] * 512)

                IF (.AVAIL_BYTE_CNT LSS 12) THEN                !Can we accomodate t
                        BEGIN                                   ! a longword (total
                        SIGNAL (PAT$_NOPATAREA, 2, .FIRST_AVAIL_ADR, .AVAIL_BYTE_CNT
                        PAT$END_OF_LINE (.SEMSP);               !Clean up after ours
                        RETURN FALSE                            !Go process next com
                        END;

                IF ((.INITIAL_SIZE LEQ 0) OR (.INITIAL_SIZE GTR .AVAIL_BYTE_CNT)) TH
                        BEGIN                                   !Set the default pat
                        INITIAL_SIZE = .AVAIL_BYTE_CNT;         !available space in
                        IF (.PATCH_AREA[0] LEQ 0) THEN          !Should the user be
                         SIGNAL(PAT$_BADINITSZ, 1, .INITIAL_SIZE - 8);!YES, they wil
                        END;                                    !signalling the adju


                IF (.PATCH_AREA[0] LEQ 0) THEN
                        BEGIN                                   !Initialize a descri
                        PATCH_AREA[0] = .INITIAL_SIZE - 8;      !area in the first t
                        PATCH_AREA[1] = .FIRST_AVAIL_ADR + 8;   !patch area.  Adjust
                        END                                     !address values to r
                ELSE
                        SIGNAL (PAT$_PREVINIT);                 !Patch Area was prev

                END;

PAT$ADD_PAL(.PAT$GL_PATAREA[DSC$A_POINTER],
        .PAT$GL_PATAREA[DSC$A_POINTER]+.PAT$GL_PATAREA[DSC$W_LENGTH],
        PAL$K_ADD_PAREA);
END;
```

```
                        [.PAT$GL_CONTEXT[MODULE_BIT]]:
                                PAT$SET_MODULE(0);
                        TES;

                [SHOW_TOKEN]:

                        IF .PAT$GL_CONTEXT[MODE_BIT]
                        THEN
                            PAT$SHOW_DEFAL ()
                        ELSE

                        IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]
                        THEN
                            SIGNAL (PAT$_INVCMDABS)
                        ELSE

                        SELECTONE TRUE OF
                        SET

                        [.PAT$GL_CONTEXT[SCOPE_BIT]]:
                                PAT$SHOW_SCOPE ();

                        [.PAT$GL_CONTEXT[MODULE_BIT]]:
                                PAT$SHOW_MODULE();

                        [.PAT$GL_CONTEXT[PAT_AREA_BIT]]:
                                BEGIN
                                $FAO_TT_OUT('current patch area size:   !XL',
                                                .PAT$GL_PATAREA[DSC$W_LENGTH]);
                                $FAO_TT_OUT('current patch area address:        !XL',
                                                .PAT$GL_PATAREA[DSC$A_POINTER]);
                                END;
                        TES;

                [UPDATE_TOKEN]:
                        PAT$WRTIMG();

                [VERIFY_TOKEN]:
                        BEGIN
                        PAT$GL_CONTEXT[VERIFY_BIT] = TRUE;
                        PAT$REPLACE_CMD ();
                        END;

                [OUTRANGE]:
                        IF .PAT$GL_SEMAN1[.SEMSP] EQL EOL_TOKEN
                        THEN
                                BEGIN
                                PAT$END_OF_LINE (.SEMSP);
                                RETURN FALSE
                                END;

                TES;
                END;

        !++
        ! Now output the command to the appended patch command text.  Since the command
```

```
;  859          3899  2 ! has already been successfully executed, call WRITE_CMD to reconstruct the
;  860          3900  2 ! command and write it to the command file, if desired.  PAT$WRITEFILE
;  861          3901  2 ! handles output to the command file and to the appended patch command text
;  862          3902  2 ! buffers, PAT$GL_TXTxxxx.
;  863          3903  2 !--
;  864          3904  2 WRITE_CMD(.SEMSP);
;  865          3905  2
;  866          3906  2 !++
;  867          3907  2 ! Check for end of command line.  If this is the end of the command line, then
;  868          3908  2 ! prompt for another command otherwise process the next command in this command
;  869          3909  2 ! line.
;  870          3910  2 !--
;  871          3911  3 IF (.PAT$GL_SEMAN1 [.SEMSP + PAT$K_SPOS_ONE] EQL EOL_TOKEN)
;  872          3912  2 THEN
;  873          3913  3         BEGIN
;  874          3914  3         PAT$END_OF_LINE(.SEMSP);
;  875          3915  3         RETURN FALSE;
;  876          3916  3         END
;  877          3917  2 ELSE
;  878          3918  2         PAT$END_OF_CMD (.SEMSP);
;  879          3919  2
;  880          3920  2 RETURN TRUE;
;  881          3921  1 END;
; INFO#212              L1:3726
; Null expression appears in value-required context


                                                                                   .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                                             000C5                 .BLKB   3
         00  00  00  50  4C  45  48  48  43  54  41  50      000C8 P.ABH:  .ASCII  \PATCHHELP\<0><0><0>                ;
                                              010E0009       000D4 P.ABG:  .LONG   17694729                            ;
                                              00000000'      000D8         .ADDRESS P.ABH                              ;
                                                    1C       000DC P.ABI:  .BYTE   28                                  ;
61  20  68  63  74  61  70  20  74  6E  65  72  72  75  63   000DD         .ASCII  \current patch area size:\<9>\!XL\  ;
        4C  58  21  09  3A  65  7A  69  73  20  61  65  72   000EC                                                     ;
                                                    1F       000F9 P.ABJ:  .BYTE   31                                  ;
61  20  68  63  74  61  70  20  74  6E  65  72  72  75  63   000FA         .ASCII  \current patch area address:\<9>\!XL\ ;
58  21  09  3A  73  73  65  72  64  64  61  20  61  65  72   00109                                                     ;
                                                    4C       00118                                                     ;


                                                                           .PSECT  _PAT$CODE,NOWRT,2

                                                 0FFC 00000                .ENTRY  PAT$PERFORM_CMD, Save R2,R3,R4,R5,R6,R7,R8,-; 3557
                                                                                   R9,R10,R11                          ;
                            5B 00000000'    EF   9E 00002                  MOVAB   P.ABG, R11                          ;
                            5A 00000000G    00   9E 00009                  MOVAB   LIB$SIGNAL, R10                     ;
                            59 00000000G    00   9E 00010                  MOVAB   PAT$GL_PATAREA, R9                  ;
                            58 00000000G    00   9E 00017                  MOVAB   PAT$GL_FLAGS, R8                    ;
                            57 00000000G    00   9E 0001E                  MOVAB   PAT$GL_CONTEXT, R7                  ;
                            5E        FF74  CE   9E 00025                  MOVAB   -140(SP), SP                        ;
                            13 00000000G    00   E8 0002A                  BLBS    PAT$GB_EXEC_CMD, 1$                 ; 3615
              0E       02   A7              02   E0 00031                  BBS     #2, PAT$GL_CONTEXT+2, 1$            ; 3616
                            50       04     AC   D0 00036                  MOVL    SEMSP, R0                           ; 3617
```

```
                                 0A 00000000G0040   D1 0003A          CMPL    PAT$GL_SEMAN1[R0], #10                ;
                                               75   12 00042          BNEQ    7$                                    ;
                                 54        04  AC   D0 00044 1$:      MOVL    SEMSP, R4                             ; 3620
                                 50 00000000G0044   D0 00048          MOVL    PAT$GL_SEMAN1[R4], R0                 ;
                      10                       01                                                                   ;
                                               50   CF 00050          CASEL   R0, #1, #16                           ;
    007A          0070          003B         002E      00054 2$:      .WORD   3$-2$,-                               ;
    00BB          00A5          00A0         0085      0005C                  4$-2$,-                               ;
    0134          010E          00D7         00AE      00064                  9$-2$,-                               ;
    02B8          025C          0150         02C5      0006C                  11$-2$,-                              ;
                                             02C1      00074                  13$-2$,-                              ;
                                                                              15$-2$,-                              ;
                                                                              16$-2$,-                              ;
                                                                              19$-2$,-                              ;
                                                                              17$-2$,-                              ;
                                                                              21$-2$,-                              ;
                                                                              25$-2$,-                              ;
                                                                              27$-2$,-                              ;
                                                                              54$-2$,-                              ;
                                                                              29$-2$,-                              ;
                                                                              46$-2$,-                              ;
                                                                              52$-2$,-                              ;
                                                                              53$-2$                                ;
                  00000063      8F             50   D1 00076          CMPL    R0, #99                               ; 3887
                                               58   12 0007D          BNEQ    12$                                   ;
                                             01CB   31 0007F          BRW     39$                                   ; 3890
                      3E        68             06   E0 00082 3$:      BBS     #6, PAT$GL_FLAGS, 9$                  ; 3625
                  00000000G     00             00   FB 00086          CALLS   #0, PAT$ALIGN_CMD                     ; 3629
                                               48   11 0008D          BRB     12$                                   ; 3625
                                09             67   E9 0008F 4$:      BLBC    PAT$GL_CONTEXT, 5$                    ; 3632
                  00000000G     00             00   FB 00092          CALLS   #0, PAT$RESET_DEF                     ; 3634
                                               72   11 00099          BRB     18$                                   ;
                      25        68             06   E0 0009B 5$:      BBS     #6, PAT$GL_FLAGS, 9$                  ; 3637
                      0A        02  A7         03   E1 0009F          BBC     #3, PAT$GL_CONTEXT+2, 6$              ; 3645
                      69 00000000G  00         10   C1 000A4          ADDL3   #16, PAT$GL_IHPPTR, PAT$GL_PATAREA    ; 3646
                                               5F   11 000AC          BRB     18$                                   ;
                                               67   95 000AE 6$:      TSTB    PAT$GL_CONTEXT                        ; 3648
                                               09   18 000B0          BGEQ    8$                                    ;
                  00000000G     00             00   FB 000B2          CALLS   #0, PAT$CANC_MODULE                   ; 3649
                                               52   11 000B9 7$:      BRB     18$                                   ;
                                4E         02  A7   E9 000BB 8$:      BLBC    PAT$GL_CONTEXT+2, 18$                 ; 3651
                                               7E   D4 000BF          CLRL    -(SP)                                 ; 3652
                                             010F   31 000C1          BRW     32$                                   ;
                      03        68             06   E1 000C4 9$:      BBC     #6, PAT$GL_FLAGS, 10$                 ; 3656
                                             01F5   31 000C8          BRW     48$                                   ;
                                             0113   31 000CB 10$:     BRW     35$                                   ;
                                               7E   7C 000CE 11$:     CLRQ    -(SP)                                 ; 3663
                  00000000G     00             02   FB 000D0          CALLS   #2, PAT$OPEN_COMFIL                   ;
                                               34   11 000D7 12$:     BRB     18$                                   ;
                                52 00000000G   00   D0 000D9 13$:     MOVL    PAT$GL_HEAD_LST, POINTER              ; 3671
                                               7E   13 000E0 14$:     BEQL    24$                                   ; 3672
                                               01   DD 000E2          PUSHL   #1                                    ; 3675
                                7E         04  A2   7D 000E4          MOVQ    4(POINTER), -(SP)                     ;
                  00000000G     00             03   FB 000E8          CALLS   #3, PAT$DEFINE_SYM                    ;
                                52             62   D0 000EF          MOVL    (POINTER), POINTER                    ; 3676
                                               EC   11 000F2          BRB     14$                                   ; 3672
                      02        A7         40  8F   88 000F4 15$:     BISB2   #64, PAT$GL_CONTEXT+2                 ; 3682
                  00000000G     00             00   FB 000F9 16$:     CALLS   #0, PAT$DEPOSIT_CMD                   ; 3687
```

```
PATACT                                                    H 15
V04-000                                         16-Sep-1984 00:23:16    VAX-11 Bliss-32 V4.0-742                 Page  26
                                                14-Sep-1984 12:52:23    DISK$VMSMASTER:[PATCH.SRC]PATACT.B32;1    (5)

                                   5E  11 00100        BRB     24$                                              ;
         01   A7                   01  88 00102 17$:   BISB2   #1, PAT$GL_CONTEXT+1                            ; 3691
   00000000G  00                   00  FB 00106        CALLS   #0, PAT$EXAMINE_CMD                             ; 3692
                                   77  11 0010D 18$:   BRB     26$                                              ; 3620
              52 00000000G         00  D0 0010F 19$:   MOVL    PAT$GL_HEAD_LST, POINTER                        ; 3699
                                   6E  13 00116 20$:   BEQL    26$                                              ; 3700
                                   01  DD 00118        PUSHL   #1                                               ; 3703
                                   7E  D4 0011A        CLRL    -(SP)                                            ;
                        04         A2  9F 0011C        PUSHAB  4(POINTER)                                       ;
   00000000G  00                   03  FB 0011F        CALLS   #3, PAT$OUT_MEM_LOC                             ;
              52                   62  D0 00126        MOVL    (POINTER), POINTER                              ; 3704
                                   EB  11 00129        BRB     20$                                              ; 3700
                 00000000G         00  94 0012B 21$:   CLRB    PAT$GB_TAKE_CMD                                 ; 3710
   51         68                   04  E1 00131        BBC     #4, PAT$GL_FLAGS, 26$                           ; 3711
              53 00000000G         00  D0 00135        MOVL    PAT$GL_IHPPTR, ECOLVL_PTR                       ; 3714
                                   52  D4 0013C        CLRL    BIT_NUMBER                                      ; 3715
   12 00000000G  00                52  E1 0013E 22$:   BBC     BIT_NUMBER, PAT$GL_ECO_UPD, 23$                 ; 3717
   0E         63                   52  E0 00146        BBS     BIT_NUMBER, (ECOLVL_PTR), 23$                   ; 3719
                        01         A2  9F 0014A        PUSHAB  1(BIT_NUMBER)                                   ; 3721
                                   01  DD 0014D        PUSHL   #1                                               ;
                 006D801B          8F  DD 0014F        PUSHL   #7176219                                         ;
              6A                   03  FB 00155        CALLS   #3, LIB$SIGNAL                                  ;
   DE         52 0000007F          8F  F3 00158 23$:   AOBLEQ  #127, BIT_NUMBER, 22$                           ; 3717
                                   78  11 00160 24$:   BRB     33$                                              ; 3620
                 00000000G         00  9F 00162 25$:   PUSHAB  LIB$GET_INPUT                                   ; 3726
         04   AE                   0E  D0 00168        MOVL    #14, 4(SP)                                       ;
                        04         AE  9F 0016C        PUSHAB  4(SP)                                            ;
                                   5B  DD 0016F        PUSHL   R11                                              ;
                 00000000G         00  9F 00171        PUSHAB  PAT$GL_HELP_LIN                                 ;
                                   7E  D4 00177        CLRL    -(SP)                                            ;
                 00000000G         00  9F 00179        PUSHAB  LIB$PUT_OUTPUT                                  ;
   00000000G  00                   06  FB 0017F        CALLS   #6, LBR$OUTPUT_HELP                             ;
                                   60  11 00186 26$:   BRB     36$                                              ;
              50 00000000G         00  D0 00188 27$:   MOVL    PAT$GB_MOD_PTR, R0                              ; 3730
              09         03        A0  E8 0018F        BLBS    3(R0), 28$                                       ;
                 006DBE82          8F  DD 00193        PUSHL   #7192194                                         ; 3732
              6A                   01  FB 00199        CALLS   #1, LIB$SIGNAL                                  ;
         02   A7         80        8F  88 0019C 28$:   BISB2   #128, PAT$GL_CONTEXT+2                          ; 3733
                               0175  31 001A1        BRW     54$                                              ; 3738
              1F                   67  E9 001A4 29$:   BLBC    PAT$GL_CONTEXT, 30$                             ; 3741
                 00000000G         00  9F 001A7        PUSHAB  PAT$GL_COMRAB                                   ; 3751
                        99         AB  9F 001AD        PUSHAB  SET_MODE_CMD+1                                  ;
              7E        98         AB  9A 001B0        MOVZBL  SET_MODE_CMD, -(SP)                             ;
   00000000G  00                   03  FB 001B4        CALLS   #3, PAT$WRITEFILE                               ;
                                   01  DD 001BB        PUSHL   #1                                               ; 3752
   00000000G  00                   01  FB 001BD        CALLS   #1, PAT$SET_MOD_LST                             ;
                                   22  11 001C4        BRB     36$                                              ; 3741
   03         68                   06  E1 001C6 30$:   BBC     #6, PAT$GL_FLAGS, 31$                           ; 3756
                               00F3  31 001CA        BRW     48$                                              ;
              0B         02        A7  E9 001CD 31$:   BLBC    PAT$GL_CONTEXT+2, 34$                           ; 3764
                                   01  DD 001D1        PUSHL   #1                                               ; 3765
   00000000G  00                   01  FB 001D3 32$:   CALLS   #1, PAT$SAVE_SCOPE                              ;
                                   0C  11 001DA 33$:   BRB     36$                                              ;
   0A         02  A7               02  E1 001DC 34$:   BBC     #2, PAT$GL_CONTEXT+2, 37$                       ; 3767
   00000000G  00                   00  FB 001E1 35$:   CALLS   #0, PAT$ECO_CMDS                                ; 3768
                               0135  31 001E8 36$:   BRW     55$                                              ;
   03         02  A7               03  E0 001EB 37$:   BBS     #3, PAT$GL_CONTEXT+2, 38$                       ; 3770
```

```
                                  00AE   31 001F0          BRW      45$                                          ;
                          04       AE   9F 001F3  38$:     PUSHAB   ISE_PTR                                      ; 3773
                                   59   DD 001F6           PUSHL    R9                                           ;
                   50 00000000G    00   D0 001F8           MOVL     PAT$GL_HEAD_LST, R0                          ;
                          04       A0   DD 001FF           PUSHL    4(R0)                                        ;
          00000000G 00              03   FB 00202           CALLS    #3, PAT$MAP_ADDR                             ;
     7B         02  A7              01   E1 00209           BBC      #1, PAT$GL_CONTEXT+2, 44$                    ; 3799
                    52              69   D0 0020E           MOVL     PAT$GL_PATAREA, R2                           ; 3801
                    50 00000000G    00   D0 00211           MOVL     PAT$GL_HEAD_LST, R0                          ; 3802
                    56         04   A0   9E 00218           MOVAB    4(R0), R6                                    ;
                    55         08   A0   9E 0021C           MOVAB    8(R0), R5                                    ; 3803
     50         04  AE              14   C1 00220           ADDL3    #20, ISE_PTR, ISD_PTR                        ; 3809
                    51         02   A0   3C 00225           MOVZWL   2(ISD_PTR), R1                               ; 3810
     51             51              09   78 00229           ASHL     #9, R1, R1                                   ;
     50         04  A0              09   78 0022D           ASHL     #9, 4(ISD_PTR), R0                           ; 3811
                    50              66   C2 00232           SUBL2    (R6), R0                                     ;
     53             51              50   C1 00235           ADDL3    R0, R1, AVAIL_BYTE_CNT                       ;
                    0C              53   D1 00239           CMPL     AVAIL_BYTE_CNT, #12                          ; 3813
                                    14   18 0023C           BGEQ     40$                                          ;
                                    53   DD 0023E           PUSHL    AVAIL_BYTE_CNT                               ; 3815
                                    66   DD 00240           PUSHL    (R6)                                         ;
                                    02   DD 00242           PUSHL    #2                                           ;
                006D811A            8F   DD 00244           PUSHL    #7176474                                     ;
                    6A              04   FB 0024A           CALLS    #4, LIB$SIGNAL                               ;
                                    54   DD 0024D  39$:     PUSHL    R4                                           ; 3816
                             00ED   31 0024F               BRW      56$                                          ;
                                    65   D5 00252  40$:     TSTL     (R5)                                         ; 3820
                                    05   15 00254           BLEQ     41$                                          ;
                    53              65   D1 00256           CMPL     (R5), AVAIL_BYTE_CNT                         ;
                                    16   15 00259           BLEQ     42$                                          ;
                    65              53   D0 0025B  41$:     MOVL     AVAIL_BYTE_CNT, (R5)                         ; 3822
                                    62   D5 0025E           TSTL     (R2)                                         ; 3823
                                    0F   14 00260           BGTR     42$                                          ;
     7E             65              08   C3 00262           SUBL3    #8, (R5), -(SP)                              ; 3824
                                    01   DD 00266           PUSHL    #1                                           ;
                006D8053            8F   DD 00268           PUSHL    #7176275                                     ;
                    6A              03   FB 0026E           CALLS    #3, LIB$SIGNAL                               ;
                                    62   D5 00271  42$:     TSTL     (R2)                                         ; 3828
                                    0B   14 00273           BGTR     43$                                          ;
     62             65              08   C3 00275           SUBL3    #8, (R5), (R2)                               ; 3830
04   A2             66              08   C1 00279           ADDL3    #8, (R6), 4(R2)                              ; 3831
                                    09   11 0027E           BRB      44$                                          ; 3828
                006D805B            8F   DD 00280  43$:     PUSHL    #7176283                                     ; 3834
                    6A              01   FB 00286           CALLS    #1, LIB$SIGNAL                               ;
                                    7E   D4 00289  44$:     CLRL     -(SP)                                        ; 3838
                    50              69   D0 0028B           MOVL     PAT$GL_PATAREA, R0                           ; 3839
                    51              60   3C 0028E           MOVZWL   (R0), R1                                     ;
                          04 B041   9F 00291               PUSHAB   @4(R0)[R1]                                   ;
                          04       A0   DD 00295           PUSHL    4(R0)                                        ; 3838
          00000000G 00              03   FB 00298           CALLS    #3, PAT$ADD_PAL                              ;
                                    7F   11 0029F           BRB      55$                                          ; 3761
                                    67   95 002A1  45$:     TSTB     PAT$GL_CONTEXT                               ; 3843
                                    7B   18 002A3           BGEQ     55$                                          ;
                                    7E   D4 002A5           CLRL     -(SP)                                        ; 3844
          00000000G 00              01   FB 002A7           CALLS    #1, PAT$SET_MODULE                           ;
                                    70   11 002AE           BRB      55$                                          ; 3741
                    09              67   E9 002B0  46$:     BLBC     PAT$GL_CONTEXT, 47$                          ; 3849
```

```
             00000000G  00            00  FB 002B3        CALLS   #0, PAT$SHOW_DEFAL                          ; 3851
                                      64  11 002BA        BRB     55$                                         ;
        0B              68            06  E1 002BC 47$:   BBC     #6, PAT$GL_FLAGS, 49$                       ; 3854
                          006DBE82    8F  DD 002C0 48$:   PUSHL   #7192194                                    ; 3856
                        6A            01  FB 002C6        CALLS   #1, LIB$SIGNAL                              ;
                                      55  11 002C9        BRB     55$                                         ;
                        09        02  A7  E9 002CB 49$:   BLBC    PAT$GL_CONTEXT+2, 50$                       ; 3862
             00000000G  00            00  FB 002CF        CALLS   #0, PAT$SHOW_SCOPE                          ; 3863
                                      48  11 002D6        BRB     55$                                         ;
                                      67  95 002D8 50$:   TSTB    PAT$GL_CONTEXT                              ; 3865
                                      09  18 002DA        BGEQ    51$                                         ;
             00000000G  00            00  FB 002DC        CALLS   #0, PAT$SHOW_MODULE                         ; 3866
                                      3B  11 002E3        BRB     55$                                         ;
        36         02   A7            03  E1 002E5 51$:   BBC     #3, PAT$GL_CONTEXT+2, 55$                   ; 3868
                        50            69  D0 002EA        MOVL    PAT$GL_PATAREA, R0                          ; 3871
                        7E            60  3C 002ED        MOVZWL  (R0), -(SP)                                 ;
                                  08  AB  9F 002F0        PUSHAB  P.ABI                                       ;
             00000000G  00            02  FB 002F3        CALLS   #2, PAT$FAO_OUT                             ;
                        50            69  D0 002FA        MOVL    PAT$GL_PATAREA, R0                          ; 3873
                                  04  A0  DD 002FD        PUSHL   4(R0)                                       ;
                                  25  AB  9F 00300        PUSHAB  P.ABJ                                       ;
             00000000G  00            02  FB 00303        CALLS   #2, PAT$FAO_OUT                             ;
                                      14  11 0030A        BRB     55$                                         ; 3849
             00000000G  00            00  FB 0030C 52$:   CALLS   #0, PAT$WRTIMG                              ; 3878
                                      0B  11 00313        BRB     55$                                         ;
                   02   A7            20  88 00315 53$:   BISB2   #32, PAT$GL_CONTEXT+2                       ; 3882
             00000000G  00            00  FB 00319 54$:   CALLS   #0, PAT$REPLACE_CMD                         ; 3883
                                  04  AC  DD 00320 55$:   PUSHL   SEMSP                                       ; 3904
             00000000V  EF            01  FB 00323        CALLS   #1, WRITE_CMD                               ;
                        50        04  AC  D0 0032A        MOVL    SEMSP, R0                                   ; 3911
             00000063   8F 00000000G0040  D1 0032E        CMPL    PAT$GL_SEMAN1+8[R0], #99                    ;
                                      0A  12 0033A        BNEQ    57$                                         ;
                                  04  AC  DD 0033C        PUSHL   SEMSP                                       ; 3914
                 FC72   CF            01  FB 0033F 56$:   CALLS   #1, PAT$END_OF_LINE                         ;
                                      0C  11 00344        BRB     58$                                         ; 3915
                                  04  AC  DD 00346 57$:   PUSHL   SEMSP                                       ; 3918
                 FADB   CF            01  FB 00349        CALLS   #1, PAT$END_OF_CMD                          ;
                        50            01  D0 0034E        MOVL    #1, R0                                      ; 3920
                                          04 00351        RET                                                 ;
                                      50  D4 00352 58$:   CLRL    R0                                          ; 3921
                                          04 00354        RET                                                 ;

; Routine Size:  853 bytes,    Routine Base:  _PAT$CODE + 01D7
```

```
GLOBAL ROUTINE WRITE_CMD (SEMSP) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine builds the command lines for the output command file
!       and the appended patch command text.  The command has already been
!       executed successfully, the command verb is on the stack, and the
!       parameters are in the parameter list.  The routine PAT$WRITEFILE does
!       all the output to the command file and to the text buffers.  If a
!       command file is not being created, then the commands are only entered
!       in the text buffers.
!
! CALLING SEQUENCE:
!
!       WRITE_CMD (SEMSP)
!
! INPUTS:
!
!       SEMSP - Offset in parse stack which holds current verb token
!
! IMPLICIT INPUTS:
!
!       none
!
! OUTPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       NONE
!
! SIDE EFFECTS:
!
!       A PATCH command is entered into the appended command text buffers
!       and written to the output command file, if one is being created.
!
!--

BEGIN

LITERAL
        BLANK_FILL = %X'20';                            ! Ascii value for space

LOCAL
        ALIGN_QUAL_OFF,                                 ! Offset into ALIGN qualifier table
        COMMAND_BUF : VECTOR[NO_OF_INP_CHARS,BYTE],     ! Buffer for output of command line to file
        COUNT;                                          ! Counter for scope name loop

!++
! Execute the complete command.
!--
```

```
2 IF .PAT$GB_EXEC_CMD
2 THEN
2         CASE .PAT$GL_SEMAN1 [.SEMSP] FROM ALIGN_TOKEN TO VERIFY_TOKEN OF
2 
2         SET
2 
2         [ALIGN_TOKEN]:
3                 BEGIN
3                 CH$COPY(.ALIGN_CMD[0], ALIGN_CMD[1], BLANK_FILL,
3                         .ALIGN_CMD[0], CH$PTR(COMMAND_BUF, 0));
3                 IF .PAT$GL_CONTEXT[ALIGN_BYTE]
3                 THEN
3                         ALIGN_QUAL_OFF = 0
3                 ELSE
3                         IF .PAT$GL_CONTEXT[ALIGN_WORD]
3                         THEN
3                                 ALIGN_QUAL_OFF = ALIGN_QUAL_LNG
3                         ELSE
3                                 IF .PAT$GL_CONTEXT[ALIGN_LONG]
3                                 THEN
3                                         ALIGN_QUAL_OFF = ALIGN_QUAL_LNG*2
3                                 ELSE
3                                         IF .PAT$GL_CONTEXT[ALIGN_QUAD]
3                                         THEN
3                                                 ALIGN_QUAL_OFF = ALIGN_QUAL_LNG*3
3                                         ELSE
3                                                 ALIGN_QUAL_OFF = ALIGN_QUAL_LNG*4;
3                 CH$COPY(ALIGN_QUAL_LNG, ALIGN_QUAL_TB[.ALIGN_QUAL_OFF],
3                         BLANK_FILL, ALIGN_QUAL_LNG,
3                         CH$PTR(COMMAND_BUF, .ALIGN_CMD[0]));
3                 PAT$WRITEFILE(.ALIGN_CMD[0]+ALIGN_QUAL_LNG,
3                                 CH$PTR(COMMAND_BUF, 0), PAT$GL_COMRAB);
3                 PAT$WRITE_NAME(.SEMSP);
2                 END;
2 
2         [CANCEL_TOKEN]:
2 
2                 SELECTONE TRUE OF
2                 SET
2 
2                 [.PAT$GL_CONTEXT[PAT_AREA_BIT]]:
3                         BEGIN
3                         PAT$WRITEFILE(.CANCEL_PAT_CMD[0], CANCEL_PAT_CMD[1], PAT$GL_COMRAB);
2                         END;
2 
2                 [.PAT$GL_CONTEXT[MODE_BIT]]:
3                         BEGIN
3                         PAT$WRITEFILE(.CANCEL_MODE_CMD[0], CANCEL_MODE_CMD[1], PAT$GL_COMRAB);
2                         END;
2 
2                 [.PAT$GL_CONTEXT[MODULE_BIT]]:
3                         BEGIN
4                         IF (.PAT$GL_HEAD_LST NEQU 0)
3                         THEN
4                                 BEGIN
4                                 PAT$WRITEFILE(.CANCEL_MODU_CMD[0], CANCEL_MODU_CMD[1], PAT$GL_COMRAB);
4                                 PAT$WRITE_NAME(.SEMSP);
```

```
4                               PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
4                               END
3                       ELSE
3                               PAT$WRITEFILE(.CAN_MOD_ALL_CMD[0], CAN_MOD_ALL_CMD[1], PAT$GL_COMRAB);
2                       END;
2
2               [.PAT$GL_CONTEXT[SCOPE_BIT]]:
3                       BEGIN
3                       PAT$WRITEFILE(.CANCEL_SCO_CMD[0], CANCEL_SCO_CMD[1], PAT$GL_COMRAB);
2                       END;
2               TES;
2
2       [CHECK_TOKEN]:
3               BEGIN
3               IF .PAT$GL_CONTEXT[SET_NOT_ECO]
3               THEN
3                       PAT$WRITEFILE(.CHECK_N_ECO_CMD[0], CHECK_N_ECO_CMD[1],
3                                       PAT$GL_COMRAB)
3               ELSE
3                       PAT$WRITEFILE(.CHECK_ECO_CMD[0], CHECK_ECO_CMD[1],
3                                       PAT$GL_COMRAB);
3               PAT$WRITE_EXP1(.SEMSP);
3               PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2               END;
2
2       [CREATE_TOKEN]:
2               0;
2
2       [DEFINE_TOKEN]:
3               BEGIN
3               PAT$WRITEFILE(.DEFINE_CMD[0], DEFINE_CMD[1], PAT$GL_COMRAB);
3               PAT$WRITE_NAME(.SEMSP);
3               PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2               END;
2
2       [DELETE_TOKEN]:
3               BEGIN
3               CH$COPY(.DELETE_CMD[0], DELETE_CMD[1], BLANK_FILL,
3                       .DELETE_CMD[0], CH$PTR(COMMAND_BUF, 0));
3               PAT$GET_COMQUAL( COMMAND_BUF, .DELETE_CMD[0], .SEMSP);
3               PAT$WRITE_INS(.SEMSP);
3               PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2               END;
2
2       [DEPOSIT_TOKEN]:
3               BEGIN
3               CH$COPY(.DEPOSIT_CMD[0], DEPOSIT_CMD[1], BLANK_FILL,
3                       .DEPOSIT_CMD[0], CH$PTR(COMMAND_BUF, 0));
3               PAT$GET_COMQUAL( COMMAND_BUF, .DEPOSIT_CMD[0], .SEMSP);
3               PAT$WRITE_INS(.SEMSP);
3               PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2               END;
2
2       [EXAMINE_TOKEN]:
2               0;
2
2       [EVALUATE_TOKEN]:
```

```
2                          0;
2
2                  [EXIT_TOKEN]:
2                          PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2
2                  [HELP_TOKEN]:
2                          0;
2
2                  [INSERT_TOKEN]:
3                          BEGIN
3                          CH$COPY(.INSERT_CMD[0], INSERT_CMD[1], BLANK_FILL,
3                                  .INSERT_CMD[0], CH$PTR(COMMAND_BUF, 0));
3                          PAT$GET_COMQUAL( COMMAND_BUF, .INSERT_CMD[0], .SEMSP);
3                          PAT$WRITE_INS(.SEMSP);
3                          PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2                          END;
2
2                  [REPLACE_TOKEN]:
3                          BEGIN
3                          CH$COPY(.REPLACE_CMD[0], REPLACE_CMD[1], BLANK_FILL,
3                                  .REPLACE_CMD[0], CH$PTR(COMMAND_BUF, 0));
3                          PAT$GET_COMQUAL( COMMAND_BUF, .REPLACE_CMD[0], .SEMSP);
3                          PAT$WRITE_INS(.SEMSP);
3                          PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2                          END;
2
2                  [SET_TOKEN]:
2
2                          SELECTONE TRUE OF
2                          SET
2
2                          [.PAT$GL_CONTEXT[SCOPE_BIT]]:
3                                  BEGIN
3                                  PAT$WRITEFILE(.SET_SCO_CMD[0], SET_SCO_CMD[1], PAT$GL_COMRAB);
3                                  PAT$GL_BUF_SIZ = 0;
3                                  PAT$CP_OUT_STR = CH$PTR(COMMAND_BUF, 0);
3                                  COUNT = 0;
3                                  WHILE .PAT$GL_CSP_PTR[ .COUNT ] NEQA 0
3                                  DO
4                                          BEGIN
4                                          PAT$FAO_PUT(SCO_NAM_CMD, .PAT$GL_CSP_PTR[.COUNT]);
4                                          COUNT = .COUNT + 1;
3                                          END;
3                                  PAT$WRITEFILE(.PAT$GL_BUF_SIZ, COMMAND_BUF, PAT$GL_COMRAB);
2                                  END;
2
2                          [.PAT$GL_CONTEXT[SET_ECO]]:
3                                  BEGIN
3                                  PAT$WRITEFILE(.SET_ECO_CMD[0], SET_ECO_CMD[1], PAT$GL_COMRAB);
3                                  PAT$WRITE_EXP1(.SEMSP);
2                                  END;
2
2                          [.PAT$GL_CONTEXT[MODE_BIT]]:
3                                  BEGIN
3                                  PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
2                                  END;
2
```

```
                [.PAT$GL_CONTEXT[PAT_AREA_BIT]]:
                        BEGIN
                        IF (.PAT$GL_CONTEXT[INIT_PAT_BIT]) THEN
                                BEGIN
                                LOCAL
                                        OUTPUT_BUFFER : BLOCK [132, BYTE];

                                PAT$CP_OUT_STR = CH$PTR (OUTPUT_BUFFER, 0);
                                CH$COPY (.SET_PAT_CMD[0], SET_PAT_CMD[1], BLANK_FILL,
                                        .SET_PAT_CMD[0], CH$PTR (COMMAND_BUF, 0));
                                PAT$GET_COMQUAL (COMMAND_BUF, .SET_PAT_CMD[0], .SEMSP);
                                PAT$GL_BUF_SIZ = 0;
                                PAT$CP_OUT_STR = CH$PTR (OUTPUT_BUFFER, 0);
                                PAT$OUT_PAC_EXP (.LIST_ELEM_EXP2 (.PAT$GL_HEAD_LST), 0);
                                PAT$WRITEFILE (.PAT$GL_BUF_SIZ, OUTPUT_BUFFER, PAT$GL_COMRAB);
                                PAT$GL_BUF_SIZ = 0;
                                PAT$CP_OUT_STR = CH$PTR (OUTPUT_BUFFER, 0);
                                PAT$OUT_PAC_EXP (.LIST_ELEM_EXPT (.PAT$GL_HEAD_LST), 0);
                                PAT$WRITEFILE (.PAT$GL_BUF_SIZ, OUTPUT_BUFFER, PAT$GL_COMRAB);
                                END
                        ELSE
                                BEGIN
                                PAT$WRITEFILE(.SET_PAT_CMD[0], SET_PAT_CMD[1], PAT$GL_COMRAB);
                                PAT$WRITE_EXP1(.SEMSP);
                                END;
                        END;

                [.PAT$GL_CONTEXT[MODULE_BIT]]:
                        BEGIN
                        IF (.PAT$GL_HEAD_LST NEQU 0)
                        THEN
                                BEGIN
                                PAT$WRITEFILE(.SET_MODU_CMD[0], SET_MODU_CMD[1], PAT$GL_COMRAB);
                                PAT$WRITE_NAME(.SEMSP);
                                PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
                                END
                        ELSE
                                PAT$WRITEFILE(.SET_MOD_ALL_CMD[0], SET_MOD_ALL_CMD[1], PAT$GL_COMRAB);
                        END;
                TES;

        [SHOW_TOKEN]:
                0;

        [UPDATE_TOKEN]:
                PAT$WRITEFILE(.UPDATE_CMD[0], UPDATE_CMD[1], PAT$GL_COMRAB);

        [VERIFY_TOKEN]:
                BEGIN
                CH$COPY(.VERIFY_CMD[0], VERIFY_CMD[1], BLANK_FILL,
                        .VERIFY_CMD[0], CH$PTR(COMMAND_BUF, 0));
                PAT$GET_COMQUAL( COMMAND_BUF, .VERIFY_CMD[0], .SEMSP);
                PAT$WRITE_INS(.SEMSP);
                PAT$WRITEFILE(.EXIT_CMD[0], EXIT_CMD[1], PAT$GL_COMRAB);
                END;

        [OUTRANGE]:
```

```
; 1168          4207  2           0;
; 1169          4208  2
; 1170          4209  2       TES;
; 1171          4210  2 RETURN
; 1172          4211  1 END;




                                                  0FFC 00000            .ENTRY  WRITE_CMD, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 3922
                                                                                R10,R11                                        :
                           5B 00000000G  00   9E 00002            MOVAB   PAT$WRITEFILE, R11                              :
                           5A 00000000G  00   9E 00009            MOVAB   PAT$GL_CONTEXT, R10                             :
                           59 00000000G  0D   9E 00010            MOVAB   PAT$GL_COMRAB, R9                               :
                           58 00000000'  EF   9E 00017            MOVAB   SET_PAT_CMD+1, R8                               :
                           5E      FEF8  CE   9E 0001E            MOVAB   -264(SP), SP                                    :
                           01 00000000G  00   E8 00023            BLBS    PAT$GB_EXEC_CMD, 1$                             ; 3979
                                              04 0002A            RET                                                     :
                           57        04  AC   D0 0002B 1$:        MOVL    SEMSP, R7                                       ; 3981
                    10     01 00000000G0047   CF 0002F            CASEL   PAT$GL_SEMAN1[R7], #1, #16                      :
  0290          00CB     0078          0023      00038 2$:        .WORD   3$-2$,-                                         :
  0290          010B     00FE          00F2      00040                    9$-2$,-                                         :
  0118          0290     0284          0290      00048                    16$-2$,-                                        :
  0257          0290     0133          0125      00050                    44$-2$,-                                        :
                                       0262      00058                    19$-2$,-                                        :
                                                                          21$-2$,-                                        :
                                                                          22$-2$,-                                        :
                                                                          44$-2$,-                                        :
                                                                          44$-2$,-                                        :
                                                                          42$-2$,-                                        :
                                                                          44$-2$,-                                        :
                                                                          23$-2$,-                                        :
                                                                          24$-2$,-                                        :
                                                                          26$-2$,-                                        :
                                                                          44$-2$,-                                        :
                                                                          39$-2$,-                                        :
                                                                          40$-2$                                          :
                                              04 0005A            RET                                                     :
                           56      FF79  C8   9A 0005B 3$:        MOVZBL  ALIGN_CMD, R6                                   ; 3987
         FF7C   CD    FF7A   C8          56   28 00060            MOVC3   R6, ALIGN_CMD+1, COMMAND_BUF                    ; 3988
                04           6A          06   E1 00068            BBC     #6, PAT$GL_CONTEXT, 4$                          ; 3989
                                         50   D4 0006C            CLRL    ALIGN_QUAL_OFF                                  ; 3991
                                         1E   11 0006E            BRB     8$                                              :
                05           6A          04   E1 00070 4$:        BBC     #4, PAT$GL_CONTEXT, 5$                          ; 3993
                             50          04   D0 00074            MOVL    #4, ALIGN_QUAL_OFF                              ; 3995
                                         15   11 00077            BRB     8$                                              :
                05           6A          02   E1 00079 5$:        BBC     #2, PAT$GL_CONTEXT, 6$                          ; 3997
                             50          08   D0 0007D            MOVL    #8, ALIGN_QUAL_OFF                              ; 3999
                                         0C   11 00080            BRB     8$                                              :
                05           6A          03   E1 00082 6$:        BBC     #3, PAT$GL_CONTEXT, 7$                          ; 4001
                             50          0C   D0 00086            MOVL    #12, ALIGN_QUAL_OFF                             ; 4003
                                         03   11 00089            BRB     8$                                              :
                             50          10   D0 0008B 7$:        MOVL    #16, ALIGN_QUAL_OFF                             ; 4005
                                  FF7C CD46   9F 0008E 8$:        PUSHAB  COMMAND_BUF[R6]                                 ; 4008
                                    2A A840   9F 00093            PUSHAB  ALIGN_QUAL_TBL[ALIGN_QUAL_OFF]                  :
                             9E          9E   D0 00097            MOVL    @(SP)+, @(SP)+                                  :
```

```
                                          59  DD 0009A          PUSHL   R9                                       ; 4009
                              FF7C        CD  9F 0009C          PUSHAB  COMMAND_BUF                              ; 4010
                                04        A6  9F 000A0          PUSHAB  4(R6)                                    ; 4009
                        6B                03  FB 000A3          CALLS   #3, PAT$WRITEFILE                        ;
                                          57  DD 000A6          PUSHL   R7                                       ; 4011
            00000000G   00                01  FB 000A8          CALLS   #1, PAT$WRITE_NAME                       ;
                                              04 000AF          RET                                              ; 3981
     0B           02    AA                03  E1 000B0 9$:      BBC     #3, PAT$GL_CONTEXT+2, 10$                ; 4019
                                          59  DD 000B5          PUSHL   R9                                       ; 4021
                                9E        A8  9F 000B7          PUSHAB  CANCEL_PAT_CMD+1                         ;
                        7E      9D        A8  9A 000BA          MOVZBL  CANCEL_PAT_CMD, -(SP)                    ;
                                          40  11 000BE          BRB     15$                                      ;
                        0D                6A  E9 000C0 10$:     BLBC    PAT$GL_CONTEXT, 11$                      ; 4024
                                          59  DD 000C3          PUSHL   R9                                       ; 4026
                              FF7E        C8  9F 000C5          PUSHAB  CANCEL_MODE_CMD+1                        ;
                        7E    FF7D        C8  9A 000C9          MOVZBL  CANCEL_MODE_CMD, -(SP)                   ;
                                          30  11 000CE          BRB     15$                                      ;
                                          6A  95 000D0 11$:     TSTB    PAT$GL_CONTEXT                           ; 4029
                                          1E  18 000D2          BGEQ    13$                                      ;
                         00000000G        00  D5 000D4          TSTL    PAT$GL_HEAD_LST                          ; 4031
                                          0B  13 000DA          BEQL    12$                                      ;
                                          59  DD 000DC          PUSHL   R9                                       ; 4034
                                83        A8  9F 000DE          PUSHAB  CANCEL_MODU_CMD+1                        ;
                        7E      82        A8  9A 000E1          MOVZBL  CANCEL_MODU_CMD, -(SP)                   ;
                                          4C  11 000E5          BRB     20$                                      ;
                                          59  DD 000E7 12$:     PUSHL   R9                                       ; 4039
                                8B        A8  9F 000E9          PUSHAB  CAN_MOD_ALL_CMD+1                        ;
                        7E      8A        A8  9A 000EC          MOVZBL  CAN_MOD_ALL_CMD, -(SP)                   ;
                                          0E  11 000F0          BRB     15$                                      ;
                        01      02        AA  E8 000F2 13$:     BLBS    PAT$GL_CONTEXT+2, 14$                    ; 4042
                                              04 000F6          RET                                              ;
                                          59  DD 000F7 14$:     PUSHL   R9                                       ; 4044
                                98        A8  9F 000F9          PUSHAB  CANCEL_SCO_CMD+1                         ;
                        7E      97        A8  9A 000FC          MOVZBL  CANCEL_SCO_CMD, -(SP)                    ;
                                        01C2  31 00100 15$:     BRW     43$                                      ;
     0B                 6A                01  E1 00103 16$:     BBC     #1, PAT$GL_CONTEXT, 17$                  ; 4050
                                          59  DD 00107          PUSHL   R9                                       ; 4052
                                A5        A8  9F 00109          PUSHAB  CHECK_N_ECO_CMD+1                        ;
                        7E      A4        A8  9A 0010C          MOVZBL  CHECK_N_ECO_CMD, -(SP)                   ;
                                          09  11 00110          BRB     18$                                      ;
                                          59  DD 00112 17$:     PUSHL   R9                                       ; 4055
                                AF        A8  9F 00114          PUSHAB  CHECK_ECO_CMD+1                          ;
                        7E      AE        A8  9A 00117          MOVZBL  CHECK_ECO_CMD, -(SP)                     ;
                        6B                03  FB 0011B 18$:     CALLS   #3, PAT$WRITEFILE                        ;
                                          57  DD 0011E          PUSHL   R7                                       ; 4057
            00000000G   00                01  FB 00120          CALLS   #1, PAT$WRITE_EXP1                       ;
                                        0192  31 00127          BRW     42$                                      ; 4058
                                          59  DD 0012A 19$:     PUSHL   R9                                       ; 4066
                                B5        A8  9F 0012C          PUSHAB  DEFINE_CMD+1                             ;
                        7E      B4        A8  9A 0012F          MOVZBL  DEFINE_CMD, -(SP)                        ;
                                        0140  31 00133 20$:     BRW     37$                                      ;
                        56      B8        A8  9A 00136 21$:     MOVZBL  DELETE_CMD, R6                           ; 4073
  FF7C       CD         B9      A8        56  28 0013A          MOVC3   R6, DELETE_CMD+1, COMMAND_BUF            ; 4074
                                          25  11 00141          BRB     25$                                      ; 4075
                        56      BD        A8  9A 00143 22$:     MOVZBL  DEPOSIT_CMD, R6                          ; 4082
  FF7C       CD         BE      A8        56  28 00147          MOVC3   R6, DEPOSIT_CMD+1, COMMAND_BUF           ; 4083
                                          18  11 0014E          BRB     25$                                      ; 4084
                         56        CD   A8  9A 00150 23$:    MOVZBL  INSERT_CMD, R6                                 ; 4103
     FF7C   CD     CE    A8              56  28 00154         MOVC3   R6, INSERT_CMD+1, COMMAND_BUF                  ; 4104
                                         0B  11 0015B         BRB     25$                                            ; 4105
                         56        D7   A8  9A 0015D 24$:    MOVZBL  REPLACE_CMD, R6                                ; 4112
     FF7C   CD     D8    A8              56  28 00161         MOVC3   R6, REPLACE_CMD+1, COMMAND_BUF                 ; 4113
                                       013A  31 00168 25$:    BRW     41$                                            ; 4114
                         43        02   AA  E9 0016B 26$:    BLBC    PAT$GL_CONTEXT+2, 29$                          ; 4124
                                         59  DD 0016F         PUSHL   R9                                             ; 4126
                                   07   A8  9F 00171         PUSHAB  SET_SCO_CMD+1                                  ;
                         7E        06   A8  9A 00174         MOVZBL  SET_SCO_CMD, -(SP)                             ;
                         6B              03  FB 00178         CALLS   #3, PAT$WRITEFILE                              ;
                              00000000G  00  D4 0017B         CLRL    PAT$GL_BUF_SIZ                                 ; 4127
            00000000G    00      FF7C   CD  9E 00181         MOVAB   COMMAND_BUF, PAT$CP_OUT_STR                    ; 4128
                                         52  D4 0018A         CLRL    COUNT                                          ; 4129
                         50 00000000G    00  D0 0018C 27$:    MOVL    PAT$GL_CSP_PTR, R0                             ; 4130
                                       6042  D5 00193         TSTL    (R0)[COUNT]                                    ;
                                         11  13 00196         BEQL    28$                                            ;
                                       6042  DD 00198         PUSHL   (R0)[COUNT]                                    ; 4133
                                   DB   A8  9F 0019B         PUSHAB  SCO_NAM_CMD                                    ;
            00000000G    00              02  FB 0019E         CALLS   #2, PAT$FAO_PUT                                ;
                                         52  D6 001A5         INCL    COUNT                                          ; 4134
                                         E3  11 001A7         BRB     27$                                            ; 4130
                                         59  DD 001A9 28$:    PUSHL   R9                                             ; 4136
                                 FF7C   CD  9F 001AB         PUSHAB  COMMAND_BUF                                    ;
                                       0096  31 001AF         BRW     33$                                            ;
            0C     02    AA              02  E1 001B2 29$:    BBC     #2, PAT$GL_CONTEXT+2, 30$                      ; 4139
                                         59  DD 001B7         PUSHL   R9                                             ; 4141
                                   E0   A8  9F 001B9         PUSHAB  SET_ECO_CMD+1                                  ;
                         7E        DF   A8  9A 001BC         MOVZBL  SET_ECO_CMD, -(SP)                             ;
                                       0091  31 001C0         BRW     35$                                            ;
                         03              6A  E9 001C3 30$:    BLBC    PAT$GL_CONTEXT, 31$                            ; 4145
                                       00F3  31 001C6         BRW     42$                                            ;
            03     02    AA              03  E0 001C9 31$:    BBS     #3, PAT$GL_CONTEXT+2, 32$                      ; 4150
                                       0090  31 001CE         BRW     36$                                            ;
                         56        FF   A8  9A 001D1 32$:    MOVZBL  SET_PAT_CMD, R6                                ; 4158
            76     02    AA              01  E1 001D5         BBC     #1, PAT$GL_CONTEXT+2, 34$                      ; 4152
            00000000G    00              6E  9E 001DA         MOVAB   OUTPUT_BUFFER, PAT$CP_OUT_STR                  ; 4157
     FF7C   CD           68              56  28 001E1         MOVC3   R6, SET_PAT_CMD+1, COMMAND_BUF                 ; 4159
                         7E              56  7D 001E7         MOVQ    R6, -(SP)                                      ; 4160
                                 FF7C   CD  9F 001EA         PUSHAB  COMMAND_BUF                                    ;
            00000000V    EF              03  FB 001EE         CALLS   #3, PAT$GET_COMQUAL                            ;
                              00000000G  00  D4 001F5         CLRL    PAT$GL_BUF_SIZ                                 ; 4161
            00000000G    00              6E  9E 001FB         MOVAB   OUTPUT_BUFFER, PAT$CP_OUT_STR                  ; 4162
                                         7E  D4 00202         CLRL    -(SP)                                          ; 4163
                         50 00000000G    00  D0 00204         MOVL    PAT$GL_HEAD_LST, R0                            ;
                                   08   A0  DD 0020B         PUSHL   8(R0)                                          ;
            00000000G    00              02  FB 0020E         CALLS   #2, PAT$OUT_PAL_EXP                            ;
                                         59  DD 00215         PUSHL   R9                                             ; 4164
                                   04   AE  9F 00217         PUSHAB  OUTPUT_BUFFER                                  ;
                              00000000G  00  DD 0021A         PUSHL   PAT$GL_BUF_SIZ                                 ;
                         6B              03  FB 00220         CALLS   #3, PAT$WRITEFILE                              ;
                              00000000G  00  D4 00223         CLRL    PAT$GL_BUF_SIZ                                 ; 4165
            00000000G    00              6E  9E 00229         MOVAB   OUTPUT_BUFFER, PAT$CP_OUT_STR                  ; 4166
                                         7E  D4 00230         CLRL    -(SP)                                          ; 4167
                         50 00000000G    00  D0 00232         MOVL    PAT$GL_HEAD_LST, R0                            ;
                                   04   A0  DD 00239         PUSHL   4(R0)                                          ;
            00000000G    00              02  FB 0023C         CALLS   #2, PAT$OUT_PAL_EXP                            ;
```

```
                                     59  DD 00243         PUSHL   R9                                          ; 4168
                          04         AE  9F 00245         PUSHAB  OUTPUT_BUFFER                               ;
                   00000000G         00  DD 00248 33$:    PUSHL   PAT$GL_BUF_SIZ                              ;
                                     75  11 0024E         BRB     43$                                         ;
                        0340         8F  BB 00250 34$:    PUSHR   #^M<R6,R8,R9>                               ; 4172
               6B                    03  FB 00254 35$:    CALLS   #3, PAT$WRITEFILE                           ;
                                     57  DD 00257         PUSHL   R7                                          ; 4173
      00000000G  00                  01  FB 00259         CALLS   #1, PAT$WRITE_EXP1                          ;
                                         04 00260         RET                                                 ; 4121
                                     6A  95 00261 36$:    TSTB    PAT$GL_CONTEXT                              ; 4177
                                     63  18 00263         BGEQ    44$                                         ;
                   00000000G         00  D5 00265         TSTL    PAT$GL_HEAD_LST                             ; 4179
                                     17  13 0026B         BEQL    38$                                         ;
                                     59  DD 0026D         PUSHL   R9                                          ; 4182
                          EB         A8  9F 0026F         PUSHAB  SET_MODU_CMD+1                              ;
               7E         EA         A8  9A 00272         MOVZBL  SET_MODU_CMD, -(SP)                         ;
               6B                    03  FB 00276 37$:    CALLS   #3, PAT$WRITEFILE                           ;
                                     57  DD 00279         PUSHL   R7                                          ; 4183
      00000000G  00                  01  FB 0027B         CALLS   #1, PAT$WRITE_NAME                          ;
                                     38  11 00282         BRB     42$                                         ; 4184
                                     59  DD 00284 38$:    PUSHL   R9                                          ; 4187
                          F3         A8  9F 00286         PUSHAB  SET_MOD_ALL_CMD+1                           ;
               7E         F2         A8  9A 00289         MOVZBL  SET_MOD_ALL_CMD, -(SP)                      ;
                                     36  11 0028D         BRB     43$                                         ;
                                     59  DD 0028F 39$:    PUSHL   R9                                          ; 4195
                          20         A8  9F 00291         PUSHAB  UPDATE_CMD+1                                ;
               7E         1F         A8  9A 00294         MOVZBL  UPDATE_CMD, -(SP)                           ;
                                     2B  11 00298         BRB     43$                                         ;
               56         27         A8  9A 0029A 40$:    MOVZBL  VERIFY_CMD, R6                              ; 4199
 FF7C    CD       28      A8         56  28 0029E         MOVC3   R6, VERIFY_CMD+1, COMMAND_BUF               ; 4200
               7E                    56  7D 002A5 41$:    MOVQ    R6, -(SP)                                   ; 4201
                        FF7C         CD  9F 002A8         PUSHAB  COMMAND_BUF                                 ;
      00000000V  EF                  03  FB 002AC         CALLS   #3, PAT$GET_COMQUAL                         ;
                                     57  DD 002B3         PUSHL   R7                                          ; 4202
      00000000G  00                  01  FB 002B5         CALLS   #1, PAT$WRITE_INS                           ;
                                     59  DD 002BC 42$:    PUSHL   R9                                          ; 4203
                          C7         A8  9F 002BE         PUSHAB  EXIT_CMD+1                                  ;
               7E         C6         A8  9A 002C1         MOVZBL  EXIT_CMD, -(SP)                             ;
               6B                    03  FB 002C5 43$:    CALLS   #3, PAT$WRITEFILE                           ;
                                         04 002C8 44$:    RET                                                 ; 4211

; Routine Size:  713 bytes,    Routine Base:  _PAT$CODE + 052C
; 1174          4212  1 GLOBAL ROUTINE PAT$SET_OVERS (LEVEL, TOKEN) : NOVALUE =
; 1175          4213  1
; 1176          4214  1 !++
; 1177          4215  1 ! FUNCTIONAL DESCRIPTION:
; 1178          4216  1 !
; 1179          4217  1 !       Sets OVERRIDE or LOCAL modes by setting the new mode level, and
; 1180          4218  1 !       then setting the mode itself.
; 1181          4219  1 !
; 1182          4220  1 ! CALLING SEQUENCE:
; 1183          4221  1 !
; 1184          4222  1 !       PAT$SET_OVERS ()
; 1185          4223  1 !
; 1186          4224  1 ! INPUTS:
; 1187          4225  1 !
; 1188          4226  1 !       LEVEL           - Level of modes to set
; 1189          4227  1 !       TOKEN           - Mode token to be set in the mode stack
; 1190          4228  1 !
; 1191          4229  1 ! IMPLICIT INPUTS:
; 1192          4230  1 !
; 1193          4231  1 !       none
; 1194          4232  1 !
; 1195          4233  1 ! OUTPUTS:
; 1196          4234  1 !
; 1197          4235  1 !       none
; 1198          4236  1 !
; 1199          4237  1 ! IMPLICIT OUTPUTS:
; 1200          4238  1 !
; 1201          4239  1 !       none
; 1202          4240  1 !
; 1203          4241  1 ! ROUTINE VALUE:
; 1204          4242  1 !
; 1205          4243  1 !       NOVALUE
; 1206          4244  1 !
; 1207          4245  1 ! SIDE EFFECTS:
; 1208          4246  1 !
; 1209          4247  1 !       The appropriate modes are set.
; 1210          4248  1 !--
; 1211          4249  1
; 1212          4250  2 BEGIN
; 1213          4251  2 PAT$SET_MOD_LVL (.LEVEL);
; 1214          4252  2 PAT$SET_NEW_MOD (.TOKEN);
; 1215          4253  1 END;



                                           0000 00000          .ENTRY  PAT$SET_OVERS, Save nothing                          ; 4212
                                 04    AC    DD 00002          PUSHL   LEVEL                                                ; 4251
                 00000000G  00         01    FB 00005          CALLS   #1, PAT$SET_MOD_LVL                                  ;
                                 08    AC    DD 0000C          PUSHL   TOKEN                                                ; 4252
                 00000000G  00         01    FB 0000F          CALLS   #1, PAT$SET_NEW_MOD                                  ;
                                             04 00016          RET                                                          ; 4253

; Routine Size:  23 bytes,    Routine Base:  _PAT$CODE + 07F5
```

```
1 GLOBAL ROUTINE PAT$SET_COMQUAL (QUAL_OFFSET) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Sets a bit in the command qualifier longword, PAT$GL_COMQUAL,
1 !       corresponding to the qualifier specified.  These bits are used to
1 !       reconstruct the command line for the output command file and the
1 !       appended patch text.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$SET_COMQUAL( QUAL_OFFSET)
1 !
1 ! INPUTS:
1 !
1 !       QUAL_OFFSET     - Offset to position in parse stack which contains
1 !                         the qualifier
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       none
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The appropriate bit is set.
1 !--
1
2 BEGIN
2
2 !++
2 ! The command qualifier table is a stream of bytes.  Each entry consists of two
2 ! bytes.  The first byte is the token value for the qualifier (which is the
2 ! value on the parse stack).  The second byte is the corresponding bit number
2 ! to be set in the command qualifier longword, PAT$GL_COMQUAL.
2 !--
2 BIND
2       COM_QUAL_TABLE = UPLIT BYTE (
2                                       INSTRUCTI_TOKEN, INSTR_QUAL,
2                                       DECIMAL_TOKEN, DECIMAL_QUAL,
2                                       WORD_TOKEN, WORD_QUAL,
2                                       BYTE_TOKEN, BYTE_QUAL,
2                                       PATCH_ARE_TOKEN, PATCH_QUAL,
2                                       NOINSTRUC_TOKEN, NOINSTR_QUAL,
2                                       LONG_TOKEN, LONG_QUAL,
; 1274          4311  2                                         HEXADECIM_TOKEN, HEX_QUAL,
; 1275          4312  2                                         ASCII_TOKEN, ASCII_QUAL,
; 1276          4313  2                                         NOASCII_TOKEN, NOASCII_QUAL,
; 1277          4314  2                                         OCTAL_TOKEN, OCTAL_QUAL,
; 1278          4315  2                                         LITERAL_TOKEN, LITER_QUAL,
; 1279          4316  2                                         INITIALIZ_TOKEN, INITIALIZE_QUAL
; 1280          4317  2                                                                 ) : VECTOR[,BYTE];
; 1281          4318  2
; 1282          4319  2 LOCAL
; 1283          4320  2         TOKEN_INDEX;                                            ! Index into command qualifier table
; 1284          4321  2
; 1285          4322  2 !++
; 1286          4323  2 ! Loop, searching the command table for a token matching the one in the
; 1287          4324  2 ! parse stack.  The corresponding command qualifier bit is set when a match
; 1288          4325  2 ! is found.
; 1289          4326  2 !--
; 1290          4327  2 INCR TOKEN_INDEX FROM MIN_QUAL TO MAX_QUAL*2 BY 2
; 1291          4328  2 DO
; 1292          4329  3         IF (.COM_QUAL_TABLE[.TOKEN_INDEX] EQL .PAT$GL_SEMAN1[.QUAL_OFFSET])
; 1293          4330  2         THEN
; 1294          4331  3                 BEGIN
; 1295          4332  3                 PAT$GL_COMQUAL [ .COM_QUAL_TABLE[.TOKEN_INDEX+1] ] = TRUE;
; 1296          4333  3                 EXITLOOP;
; 1297          4334  2                 END;
; 1298          4335  2 RETURN;
; 1299          4336  2
; 1300          4337  1 END;


                                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

1B  06  1F  05  26  04  2D  03  15  02  31  01  16  00  1C  00119 P.ABK:  .BYTE   28, 0, 22, 1, 49, 2, 21, 3, 45, 4, 38, 5, -  ;
                0C  32  0B  1E  0A  2A  09  24  08  14  07  00128               31, 6, 27, 7, 20, 8, 36, 9, 42, 10, 30, -    ;
                                                                        11, 50, 12                                   ;

                                                        COM_QUAL_TABLE=         P.ABK


                                                                .PSECT  _PAT$CODE,NOWRT,2

                                             0004 00000         .ENTRY  PAT$SET_COMQUAL, Save R2                     ; 4254
                              51        04  AC  D0 00002        MOVL    QUAL_OFFSET, R1                              ; 4329
                                            50  D4 00006        CLRL    TOKEN_INDEX                                  ;
00000000G0041 00000000'EF40   08            00  ED 00008 1$:    CMPZV   #0, #8, COM_QUAL_TABLE[TOKEN_INDEX], -       ;
                                                                        PAT$GL_SEMAN1[R1]                            ;
                                            11  12 00017        BNEQ    2$                                           ;
                              52 00000000'EF40  9A 00019        MOVZBL  COM_QUAL_TABLE+1[TOKEN_INDEX], R2            ; 4332
                    07 00000000G  00            52  E2 00021    BBSS    R2, PAT$GL_COMQUAL, 3$                       ;
                                            04 00029            RET                                                  ; 4331
        FFD8              50            02  18  F1 0002A 2$:    ACBL    #24, #2, TOKEN_INDEX, 1$                     ; 4329
                                            04 00030 3$:        RET                                                  ; 4337

; Routine Size:  49 bytes,    Routine Base:  _PAT$CODE + 080C
```

```
1 GLOBAL ROUTINE PAT$GET_COMQUAL (COMMAND_BUF, COMMAND_SIZE, SEMSP) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine enters the command qualifiers into the command line
1 !       buffer being constructed.  The qualifiers are indicated by bits
1 !       set in the command qualifier indicator longword, PAT$GL_COMQUAL.
1 !       The routine writes the command line to the output command file
1 !       after it enters the qualifiers.  Note that the command verb has
1 !       already been entered into the buffer.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$GET_COMQUAL (COMMAND_BUF, COMMAND_SIZE, SEMSP)
1 !
1 ! INPUTS:
1 !
1 !       COMMAND_BUF - Address of command line buffer
1 !       COMMAND_SIZE - Number of command bytes already entered in the buffer
1 !       SEMSP - Offset in parse stack to command token
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$GL_COMQUAL - Indicator for qualifiers specified in command
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NOVALUE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The command verb and qualifiers are written to the output command file.
1 !--
1
2 BEGIN
2
2 MAP
2       COMMAND_BUF : REF VECTOR[,BYTE];                          ! Command line buffer
2
2 LITERAL
2       HYPHEN = %X'2D',                                          ! Ascii continuation character (hyphen)
2       BLANK_FILL = %X'20';                                      ! Ascii fill character (space)
2
2 LOCAL
2       COM_SIZE,                                                 ! Number of bytes written into command line
2       QUALIFIER_BIT;                                            ! Number of qualifier bit
2
2 BIND
```

```
2               CQ_TABLE = UPLIT BYTE (
2                                               %ASCIC '/I',
2                                               %ASCIC '/DEC',
2                                               %ASCIC '/W',
2                                               %ASCIC '/B',
2                                               %ASCIC '/PAT',
2                                               %ASCIC '/NOI',
2                                               %ASCIC '/LO',
2                                               %ASCIC '/H',
2                                               %ASCIC '/AS',
2                                               %ASCIC '/NOAS',
2                                               %ASCIC '/OC',
2                                               %ASCIC '/LI',
2                                               %ASCIC '/INIT='
2                                                                               ) : VECTOR[,BYTE],
2               CQ_OFFSET_TBL = UPLIT BYTE (
2                                                       0,
2                                                       0+3,
2                                                       0+3+5,
2                                                       0+3+5+3,
2                                                       0+3+5+3+3,
2                                                       0+3+5+3+3+5,
2                                                       0+3+5+3+3+5+5,
2                                                       0+3+5+3+3+5+5+4,
2                                                       0+3+5+3+3+5+5+4+3,
2                                                       0+3+5+3+3+5+5+4+3+4,
2                                                       0+3+5+3+3+5+5+4+3+4+6,
2                                                       0+3+5+3+3+5+5+4+3+4+6+4,
2                                                       0+3+5+3+3+5+5+4+3+4+6+4+4
2                                                                               ) : VECTOR[,BYTE];
2
2
2       !++
2       ! Loop, testing each qualifier bit.  If it is set then write the qualifier
2       ! into the command buffer and update the size of the command line.
2       !--
2       COM_SIZE = .COMMAND_SIZE;
2       INCR QUALIFIER_BIT FROM MIN_QUAL TO MAX_QUAL BY 1
2       DO
2               IF .PAT$GL_COMQUAL [.QUALIFIER_BIT]
2               THEN
3                       BEGIN
3                       CH$COPY(.CQ_TABLE [ .CQ_OFFSET_TBL[.QUALIFIER_BIT] ],
3                               CH$PTR(CQ_TABLE[1], .CQ_OFFSET_TBL[.QUALIFIER_BIT]),
3                               BLANK_FIL[,
3                               .CQ_TABLE [ .CQ_OFFSET_TBL[.QUALIFIER_BIT] ],
3                               CH$PTR(COMMAND_BUF[0], .COM_SIZE));
3                       COM_SIZE = .COM_SIZE + .CQ_TABLE [ .CQ_OFFSET_TBL[.QUALIFIER_BIT] ];
2                       END;
2
2       !++
2       ! Check if this is an EXAMINE command.  If so, put a continuation character
2       ! on the end of the line.  This is due to the special syntax for the EXAMINE
2       ! command enabling one to examine sequential locations without specifying
2       ! the address.
2       !--
3       IF (.PAT$GL_SEMAN1[.SEMSP] EQL EXAMINE_TOKEN)
```

```
; 1416          4452  2 THEN
; 1417          4453  3         BEGIN
; 1418          4454  3 ! ****** THIS CH$PTR IS HERE TO GET AROUND A COMPILER BUG.
; 1419          4455  3 ! ****** IT SHOULD EVENTUALLY BE REMOVED AND BECOME:
; 1420          4456  3 !       COMMAND_BUF[.COM_SIZE] = BLANK_FILL;
; 1421          4457  3 !       COMMAND_BUF[.COM_SIZE + 1] = HYPHEN;
; 1422          4458  3         CH$PTR(COMMAND_BUF[.COM_SIZE], 0) = BLANK_FILL;
; 1423          4459  3         CH$PTR(COMMAND_BUF[.COM_SIZE], 1) = HYPHEN;
; 1424          4460  3         COM_SIZE = .COM_SIZE + 2;
; 1425          4461  2         END;
; 1426          4462  2
; 1427          4463  2 !++
; 1428          4464  2 ! Now write out the command verb and qualifiers to the command file.
; 1429          4465  2 !--
; 1430          4466  2 PAT$WRITEFILE(.COM_SIZE, COMMAND_BUF[0], PAT$GL_COMRAB);
; 1431          4467  2 RETURN;
; 1432          4468  1 END;



                                                                 .PSECT   _PAT$PLIT,NOWRT,NOEXE,0

                                      49  2F  02  00133 P.ABL:   .ASCII   <2>\/I\                                 ;
                              43  45  44  2F  04  00136          .ASCII   <4>\/DEC\                               ;
                                      57  2F  02  0013B          .ASCII   <2>\/W\                                 ;
                                      42  2F  02  0013E          .ASCII   <2>\/B\                                 ;
                              54  41  50  2F  04  00141          .ASCII   <4>\/PAT\                               ;
                              49  4F  4E  2F  04  00146          .ASCII   <4>\/NOI\                               ;
                                  4F  4C  2F  03  0014B          .ASCII   <3>\/LO\                                ;
                                      48  2F  02  0014F          .ASCII   <2>\/H\                                 ;
                                  53  41  2F  03  00152          .ASCII   <3>\/AS\                                ;
                          53  41  4F  4E  2F  05  00156          .ASCII   <5>\/NOAS\                              ;
                                  43  4F  2F  03  0015C          .ASCII   <3>\/OC\                                ;
                                  49  4C  2F  03  00160          .ASCII   <3>\/LI\                                ;
                      3D  54  49  4E  49  2F  06  00164          .ASCII   <6>\/INIT=\                             ;
  31  2D  29  23  1F  1C  18  13  0E  0B  08  03  00  0016B P.ABM:   .BYTE    0, 3, 8, 11, 14, 19, 24, 28, 31, 35, 41, -  ;
                                                                          45, 49                                  ;

                                                        CQ_TABLE=             P.ABL
                                                        CQ_OFFSET_TBL=        P.ABM


                                                                 .PSECT   _PAT$CODE,NOWRT,2

                                       03FC 00000                .ENTRY   PAT$GET_COMQUAL, Save R2,R3,R4,R5,R6,R7,R8,-; 4338
                                                                          R9                                      ;
                     59 00000000'  EF  9E 00002                  MOVAB    CQ_OFFSET_TBL, R9                       ;
                     58        08  AC  D0 00009                  MOVL     COMMAND_SIZE, COM_SIZE                  ; 4431
                                   56  D4 0000D                  CLRL     QUALIFIER_BIT                           ; 4432
       14 00000000G  00            56  E1 0000F 1$:              BBC      QUALIFIER_BIT, PAT$GL_COMQUAL, 2$       ; 4434
                     50          6946  9A 00017                  MOVZBL   CQ_OFFSET_TBL[QUALIFIER_BIT], R0        ; 4437
                     57       C8 A940  9A 0001B                  MOVZBL   CQ_TABLE[R0], R7                        ;
         04 BC48         C9 A940   57  28 00020                  MOVC3    R7, CQ_TABLE+1[R0], @COMMAND_BUF[COM_SIZE]  ; 4441
                     58            57  C0 00028                  ADDL2    R7, COM_SIZE                            ; 4442
               E0    56            0C  F3 0002B 2$:              AOBLEQ   #12, QUALIFIER_BIT, 1$                  ; 4434
                     50        0C  AC  D0 0002F                  MOVL     SEMSP, R0                               ; 4451
                     09 00000000G0040  D1 00033                  CMPL     PAT$GL_SEMAN1[R0], #9                   ;
```

```
                                           0F  12 0003B           BNEQ    3$                                        ;
             50             58      04     AC  C1 0003D           ADDL3   COMMAND_BUF, COM_SIZE, R0                 ; 4458
                            60             20  D0 00042           MOVL    #32, (R0)                                 ;
                     01     A0             2D  D0 00045           MOVL    #45, 1(R0)                                ; 4459
                            58             02  C0 00049           ADDL2   #2, COM_SIZE                              ; 4460
                                 00000000G 00  9F 0004C   3$:     PUSHAB  PAT$GL_COMRAB                             ; 4466
                                    04     AC  DD 00052           PUSHL   COMMAND_BUF                               ;
                                           58  DD 00055           PUSHL   COM_SIZE                                  ;
                00000000G  00              03  FB 00057           CALLS   #3, PAT$WRITEFILE                         ;
                                               04 0005E           RET                                               ; 4468

; Routine Size:  95 bytes,    Routine Base:  _PAT$CODE + 083D
; 1434          4469  1 END
; 1435          4470  0 ELUDOM



                                                                    .EXTRN  LIB$SIGNAL

;                                  PSECT SUMMARY
;
;        Name                   Bytes                     Attributes
;
;   _PAT$PLIT                      376  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;   _PAT$CODE                     2204  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   .  ABS  .                        0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)




;                            Library Statistics
;
;                                                 -------- Symbols --------        Pages        Processing
;        File                                     Total    Loaded   Percent       Mapped       Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1                 18619        13         0       1000           00:01.8



; Information:  1
; Warnings:     0
; Errors:       0



;                                 COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATACT/OBJ=OBJ$:PATACT MSRC$:PATACT/UPDATE=(ENH$:PATACT)

; Size:           2204 code + 376 data bytes
; Run Time:          01:03.4
; Elapsed Time:      03:18.5
; Lines/CPU Min:      4230
; Lexemes/CPU-Min:  33840
; Memory Used:    466 pages
; Compilation Complete
```

0299 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LISTEL
REQ

PATTAB
REQ

PREFIX
REQ

SYSSER
REQ

VXSMAC
REQ

DATBAS
REQ

PATACS
REQ

PATPCT
REQ

VAXERR
REQ

DYNMEM
LIS

PATPRE
REQ

PATTER
REQ

SCALIT
REQ

PATGEN
REQ

VAXOPS
REQ

PATACT
LIS

DLLNAM
REQ

SYMFMT
REQ

PATRTS
REQ

IMGDEF
REQ

PATTBL
REQ

PATTOK
REQ

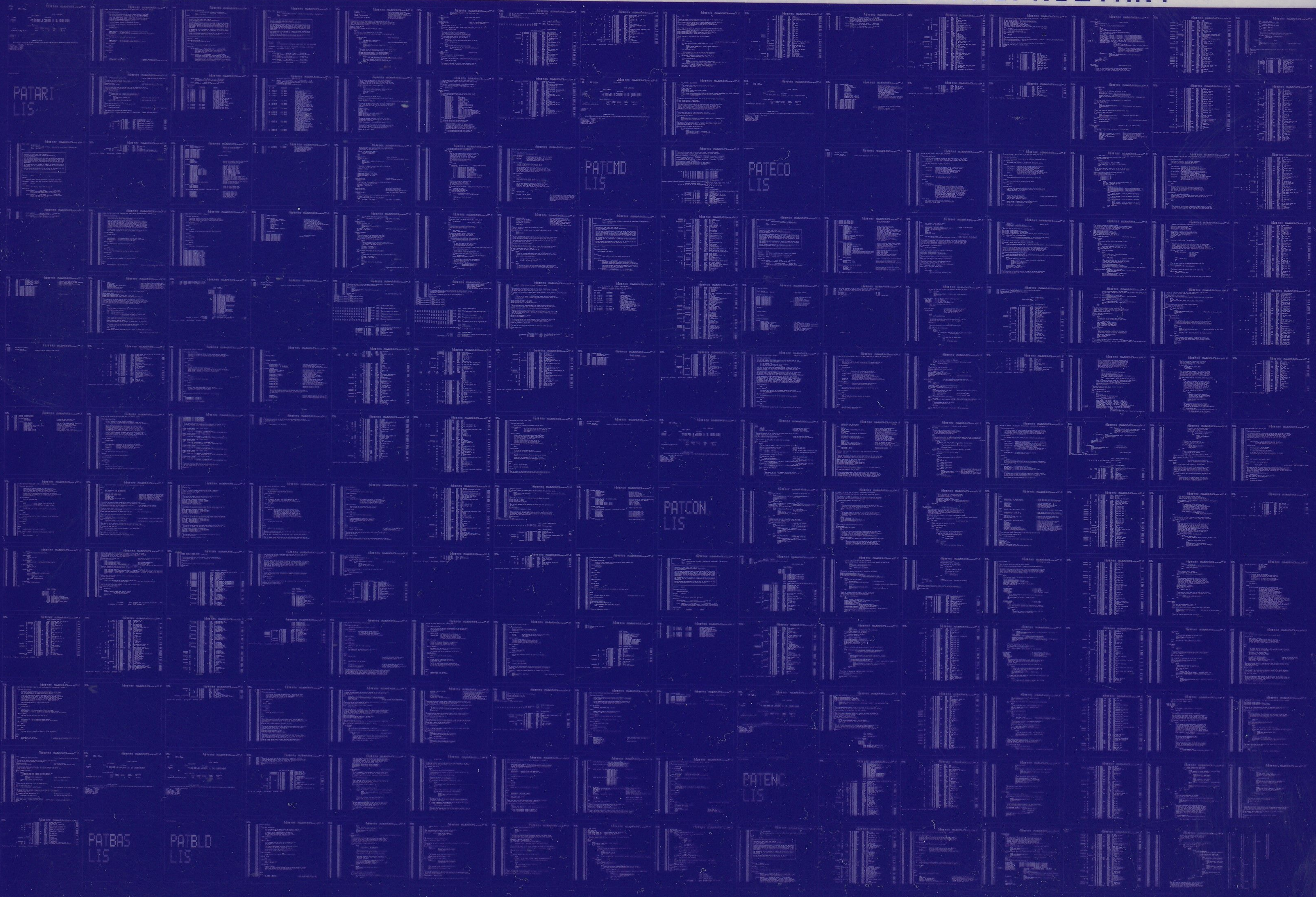
PATKEY
REQ

SYSLIT
REQ

VXPALT
REQ

0300 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



PATARI
LIS

PATCMD
LIS

PATECO
LIS

PATCON
LIS

PATENC
LIS

PATBAS
LIS

PATBLD
LIS