```
   SSSSSSSSSSSS  UUU         UUU  MMM         MMM
   SSSSSSSSSSSS  UUU         UUU  MMM         MMM
   SSSSSSSSSSSS  UUU         UUU  MMM         MMM
SSS              UUU         UUU  MMMMMM   MMMMMM
SSS              UUU         UUU  MMMMMM   MMMMMM
SSS              UUU         UUU  MMMMMM   MMMMMM
SSS              UUU         UUU  MMM   MMM   MMM
SSS              UUU         UUU  MMM   MMM   MMM
SSS              UUU         UUU  MMM   MMM   MMM
   SSSSSSSSS     UUU         UUU  MMM         MMM
   SSSSSSSSS     UUU         UUU  MMM         MMM
   SSSSSSSSS     UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
            SSS  UUU         UUU  MMM         MMM
SSSSSSSSSSSS     UUUUUUUUUUUUUUU  MMM         MMM
SSSSSSSSSSSS     UUUUUUUUUUUUUUU  MMM         MMM
SSSSSSSSSSSS     UUUUUUUUUUUUUUU  MMM         MMM
```

```
**FILE**ID**SUMEDIT                                                  M 16




  SSSSSSSS  UU      UU  MM      MM  EEEEEEEEEE  DDDDDDDD      IIIIII    TTTTTTTTTT
  SSSSSSSS  UU      UU  MM      MM  EEEEEEEEEE  DDDDDDDD      IIIIII    TTTTTTTTTT
SS          UU      UU  MMMM  MMMM  EE          DD      DD      II          TT
SS          UU      UU  MMMM  MMMM  EE          DD      DD      II          TT
SS          UU      UU  MM  MM  MM  EE          DD      DD      II          TT
SS          UU      UU  MM  MM  MM  EE          DD      DD      II          TT
  SSSSSS    UU      UU  MM      MM  EEEEEEEE    DD      DD      II          TT
  SSSSSS    UU      UU  MM      MM  EEEEEEEE    DD      DD      II          TT
        SS  UU      UU  MM      MM  EE          DD      DD      II          TT
        SS  UU      UU  MM      MM  EE          DD      DD      II          TT
        SS  UU      UU  MM      MM  EE          DD      DD      II          TT          ....
        SS  UU      UU  MM      MM  EE          DD      DD      II          TT          ....
SSSSSSSS    UUUUUUUUUU  MM      MM  EEEEEEEEEE  DDDDDDDD      IIIIII      TT          ....
SSSSSSSS    UUUUUUUUUU  MM      MM  EEEEEEEEEE  DDDDDDDD      IIIIII      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
```

```
     (2)        56        DECLARATIONS
     (3)       121        TPARSE
     (4)       260        SUM$INIT
     (5)       351        GET_IS_BLK
     (6)       385        PROCESS_FILE
     (7)       433        SET_UP_NODES
     (8)       498        INSERT_NODE
     (9)       540        READ_UPD_LINE
     (10)      592        SUM$LINE
     (11)      658        LINE_SET
     (12)      693        LINE_NUP
     (13)      713        LINE_SRC
     (14)      739        LINE_UPD
     (15)      830        LINE_UPE
     (16)      858        LINE_UPR
     (17)      887        LINE_BLK
     (18)      918        LINE_GET
     (19)      980        LINE_EOF
     (20)     1000        ACCESS_SRC
     (21)     1035        SAVE_SRC_RFA
     (22)     1056        RESTORE_SRC_RFA
     (23)     1096        ACCESS_UPDATE
     (24)     1166        READ_SRC_LINE
     (25)     1214        SKIP_SRC_LINES
     (26)     1246        COMMAND_CHECK
     (28)     1450        SUM$CLOSE
```

SUM
V04-

```
0000          .TITLE  SUM$EDIT
0000          .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:     SUMSHR shareable library
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;
0000    ; ENVIRONMENT:  USER MODE
0000    ;
0000    ; AUTHOR:       R. Newland
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-002 MTR0002         Mike Rhodes             18-May-1983
0000    ;               Correct handling of file access switching in READ_UPD_LINE
0000    ;               when an error occurs.  Also, make the RAB globally available
0000    ;               to the TPARSE action routines.
0000    ;
0000    ;       V03-001 MTR0001         Mike Rhodes             19-Jan-1983
0000    ;               Create and a local UBF for use in SUM$INIT and SUM$LINE.
0000    ;               The local UBF precludes ACCVIOs resulting from the caller's
0000    ;               RAB ROP=LOC, when processing SUMSHR's escape character "<".
0000    ;
0000    ;       V02-001                         B. Schreiber            21-Mar-1980
0000    ;               Make totally position independent.
0000    ;
0000    ;--
```

```
                0000          .SBTTL  DECLARATIONS
                0000    ;
                0000    ;
                0000    ; Macro definitions
                0000    ;
                0000          DEFUPFBLK                          ; Source update merge offsets
                0000          DEFEDBLK                           ; Edit block offsets
                0000          DEFISBLK                           ; Input stream block offsets
                0000          DEFCMDTYPE                         ; Command line type
                0000          DEFSUMCBL                          ; SUM control block
                0000          $FABDEF                            ; FAB
                0000          $RABDEF                            ; RAB
                0000          $NAMDEF                            ; NAM block
                0000          $TPADEF                            ; TPARSE definitions
                0000          $RMSDEF                            ; RMS definitions
                0000    ;
                0000    ;
                0000    ; state definitions
                0000    ;
                0000    $EQULST SUM_ST_,,0,,< -
                0000          SET , -                            ; Set up for source or update
                0000          NUP , -                            ; No more updates to process
                0000          SRC , -                            ; Next line from source file
                0000          UPD , -                            ; Next line from update file
                0000          UPE , -                            ; Report update errors
                0000          UPR , -                            ; Update ready
                0000          BLK , -                            ; Process next edit block of update
                0000          GET , -                            ; Get next update line
                0000          EOF >                              ; End of file
                0000    ;
                0000    ;
                0000    ; Procedure flag byte definitions
                0000    ;
                0000    _VIELD  PRC,0,< -
                0000          <EXPED,,M> -                       ; Expected edit command
                0000          <DELINE,,M> -                      ; Deleted lines information pending
                0000          <ERRORS,,M> -                      ; Clash errors to report
                0000          <HIEDIT,,M> -                      ; Highest precedence edit overides others
                0000          <NODATA,,M> -                      ; Data from edit being ignored
                0000          >
                0000    ;
                0000    ;
                0000    ;
                0000    ; Local storage
                0000    ;
                0000    ;
        00000000              .PSECT  SUM$RW_DATA,NOEXE,LONG
                0000    ;
                0000    ;
                0000    SUM_CUR_RAB:                             ; Address of the currently active RAB.
 00000000       0000          .LONG   0
                0004
                0004    SUM_UBF_ADDR:                            ; Address of local UBF.  The size of
 00000000       0004          .LONG   0                          ; the UBF is established by the size
                0008                                             ; of the main program's (caller's) RAB.
                0008
        00000000              .PSECT  SUM$RO_DATA,NOEXE,NOWRT,LONG
```

```
                 0000   113 ;
                 0000   114 ;
                 0000   115 SUM_ISSZE:                                   ; Size of input stream block
        00000082 0000   116         .LONG   IS_K_BLN
                 0004   117 ;
                 0004   118 SUM_EDSZE:                                   ; Size of Edit block
        0000001A 0004   119         .LONG   ED_K_BLN
```

```
                  0008          .SBTTL  TPARSE
                  0008  ;
                  0008          .SAVE
                  0008
              00000008          .PSECT  SUM$RW_DATA,NOEXE,LONG
                  0008  ;
                  0008  ;
                  0008  TPARSE_BLOCK:
        00000008  0008          .LONG   TPA$K_COUNT0
        0000002C  000C          .BLKB   TPA$K_LENGTH0-4
                  002C  ;
                  002C  ; Continue Tparse parameter block with own data
                  002C  ;
                  002C  SUM_TPARSE:
                  002C  ;
        00000024  002C  TPA_W_LOC1 = .-TPARSE_BLOCK
        0000002E  002C          .BLKW   1
        00000026  002E  TPA_W_LOC2 = .-TPARSE_BLOCK
        00000030  002E          .BLKW   1
        00000028  0030  TPA_B_ISFLAGS = .-TPARSE_BLOCK
        00000031  0030          .BLKB   1
        00000029  0031  TPA_B_EDFLAGS = .-TPARSE_BLOCK
        00000032  0031          .BLKB   1
        0000002A  0032  TPA_W_DOT = .-TPARSE_BLOCK
        00000034  0032          .BLKW   1
        0000002C  0034  TPA_W_LOC = .-TPARSE_BLOCK
        00000036  0034          .BLKW   1
        0000002E  0036  TPA_W_LINTYP = .-TPARSE_BLOCK
        00000038  0036          .BLKW   1
        00000030  0038  TPA_Q_AUDDS = .-TPARSE_BLOCK
        00000040  0038          .BLKQ   1
        00000038  0040  TPA_Q_CMNT = .-TPARSE_BLOCK
        00000048  0040          .BLKQ   1
        00000040  0048  TPA_Q_LINEDS = .-TPARSE_BLOCK
        00000050  0048          .BLKQ   1
                  0050  ;
                  0050  ;
              00000008          .PSECT  SUM$RO_DATA
                  0008  ;
        0000002C  0008          COMMA = ^X2C
        0000003B  0008          SEMICOLON = ^X3B
        0000003C  0008          LESSTHAN = ^X3C
                  0008  ;
                  0008          $INIT_STATE     MER_STATE,MER_KEY
                  0008  ;
                  0008  ; Get 1st character of line
                  0008  ;
                  0008          $STATE
                  0008          $TRAN   TPA$_LAMBDA,,ACT_BLANKS_SIG
                  0008          $STATE
                  0008          $TRAN   '-',EDIT
                  0008          $TRAN   '%',CMND,ACT_PERCENT
                  0008          $TRAN   '/',TERM
                  0008          $TRAN   LESSTHAN,DATA,ACT_ESC
                  0008          $TRAN   '@',TPA$_FAIL
                  0008          $TRAN   '\',CMND,ACT_BACKSLASH
                  0008          $TRAN   TPA$_EOS,DATA
```

```
0008          $TRAN   TPA$_ANY,DATA
0008    ;
0008    ; End data line
0008    ;
0008          $STATE  DATA
0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,,0
0008    ;
0008    ; End normal command line
0008    ;
0008          $STATE  CMND
0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,,CMD_M_CMND
0008    ;
0008    ; End data terminating command
0008    ;
0008          $STATE  TERM
0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,, -
0008                          <CMD_M_CMND!CMD_M_EDTRM!CMD_M_EDEND>
0008    ;
0008    ;
0008    ; Edit command
0008    ;
0008    ; Read locator-1
0008    ;
0008          $STATE  EDIT
0008          $TRAN   '-',,ACT_SUPPRESS
0008          $TRAN   TPA$_LAMBDA
0008          $STATE
0008          $TRAN   TPA$_LAMBDA,,ACT_BLANKS_NSIG
0008          $STATE
0008          $TRAN   !LOCATOR,,ACT_LOC1
0008    ;
0008    ; Read Locator-2
0008    ;
0008          $STATE
0008          $TRAN   TPA$_EOS,TPA$_EXIT
0008          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008          $TRAN   COMMA
0008          $STATE
0008          $TRAN   !LOCATOR,,ACT_LOC2
0008          $TRAN   TPA$_EOS,TPA$_EXIT
0008    ; Read audit string
0008    ;
0008          $STATE
0008          $TRAN   TPA$_EOS,TPA$_EXIT
0008          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008          $TRAN   COMMA
0008          $STATE
0008          $TRAN   '/',,ACT_AUDIT
0008          $TRAN   TPA$_EOS,TPA$_EXIT
0008          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008          $STATE  AUDCH
0008          $TRAN   '/',,ACT_AUDEND
0008          $TRAN   TPA$_ANY,AUDCH,ACT_AUDCH
0008    ;
0008    ; Read comment line
0008    ;
0008          $STATE
```

```
    0008          $TRAN   TPA$_EOS,TPA$_EXIT
    0008          $TRAN   SEMICOLON,CMNT,ACT_CMNT
    0008          $STATE  CMNT
    0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT
    0008  ;
    0008  ;
    0008  ; Subexpression to parse locator
    0008  ;
    0008          $STATE  LOCATOR
    0008          $TRAN   '.',,ACT_DOT
    0008          $TRAN   TPA$_DECIMAL,,ACT_LOCNUM
    0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT
    0008          $STATE
    0008          $TRAN   '+'
    0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT
    0008          $STATE
    0008          $TRAN   TPA$_DECIMAL,,ACT_PLUS
    0008          $STATE
    0008          $TRAN   TPA$_LAMBDA,TPA$_EXIT
    0008  ;
    0008          $END_STATE
    0008  ;
    0008  ;
00000008          .RESTORE
```

```
                              0008            .SBTTL  SUM$INIT
                              0008    ;
                              0008    ;++
                              0008    ; Functional description:
                              0008    ;
                              0008    ;       This procedure is called to initialise the update files.
                              0008    ;
                              0008    ;
                              0008    ; Input parameters:
                              0008    ;
                              0008    ;        4(AP) = Address of input stream control block
                              0008    ;        8(AP) = Address of update files list
                              0008    ;       12(AP) = Address of main program RAB
                              0008    ;
                              0008    ;
                              0008    ; Outputs:
                              0008    ;
                              0008    ;       IS_L_MAIN_FAB(R9) = FAB address of source file
                              0008    ;
                              0008    ; Implicit outputs:
                              0008    ;
                              0008    ;       The edit nodes list.
                              0008    ;
                              0008    ;       SUM_UBF_ADDR points to the local UBF which is allocated (if it has
                              0008    ;       not been previously).
                              0008    ;
                              0008    ;--
                              0008    ;
                          00000000            .PSECT  SUM$CODE,NOWRT,LONG
                              0000    ;
                      0FFC    0000            .ENTRY  SUM$INIT_EDIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
               02       11    0002            BRB     SUM$INIT
                              0004    ;
                      0FFC    0004            .ENTRY  SUM$INIT_CMND,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              0006    ;
                              0006    SUM$INIT:
          50     01     D0    0006            MOVL    #1,R0                   ; Assume successful completion
       58    04  AC     D0    0009            MOVL    4(AP),R8                ; Get address of SUM control block
       59    04  A8     D0    000D            MOVL    SUM_L_ISDATA(R8),R9     ; Get input stream data block address
                 0A     12    0011            BNEQ    5$                      ; Branch if block has been allocated
               00A4     30    0013            BSBW    GET_IS_BLK              ; Get and initialise data block
              6D 50     E9    0016            BLBC    R0,7$                   ; Error of LBC
        04 A8    59     D0    0019            MOVL    R9,SUM_L_ISDATA(R8)     ; Save data block address
                              001D    5$:
              18 A8     B4    001D            CLRW    SUM_W_LINE_NO(R8)       ; Reset return line number
        04 A9    00     90    0020            MOVB    #SUM_ST_SET,IS_B_STATE(R9)      ; Initialise state to SET
        06 A9    01     B0    0024            MOVW    #1,IS_W_LINE_NO(R9)     ; and source file line number
       5A    08  AC     D0    0028            MOVL    8(AP),R10               ; Get file list address
       58    0C  AC     D0    002C            MOVL    12(AP),R8               ; Get RAB address
       00000004'EF      D5    0030            TSTL    SUM_UBF_ADDR            ; Has a UBF been allocated?
                 19     12    0036            BNEQ    6$                      ; If NEQ a UBF already exists.
       7E    20  A8     3C    0038            MOVZWL  RAB$W_USZ(R8),-(SP)     ; Set up the buffer size.
       00000004'EF      9F    003C            PUSHAB  SUM_UBF_ADDR            ; Stack arguments for LIB$GET_VM
              04 AE     DF    0042            PUSHAL  4(SP)                   ; ...
 00000000'GF     02     FB    0045            CALLS   #2, G^LIB$GET_VM        ; Allocate a local UBF.
              37 50     E9    004C            BLBC    R0, 7$                  ; Error if LBC
                 8E     D5    004F            TSTL    (SP)+                   ; Clean up the stack.
```

```
     20 A9    58   D0  0051  6$:    MOVL    R8,IS_L_MAIN_RAB(R9)        ; Save RAB address
          10 A8    D4  0055         CLRL    RAB$W_RFA+0(R8)             ; Clear RFA
          14 A8    B4  0058         CLRW    RAB$W_RFA+4(R8)             ; (3 words)
           048B    30  005B         BSBW    SAVE_SRC_RFA                ; and save it
  1C A9    3C A8   D0  005E         MOVL    RAB$L_FAB(R8),IS_L_MAIN_FAB(R9) ; Save FAB address
        69    5A   D0  0063         MOVL    R10,IS_L_FILELIST(R9)       ; Save file list address
              51   13  0066         BEQL    40$                         ; If EQL there is no list so return
  40 08 AA    00   E2  0068         BBSS    #UPF_V_INIT, -              ; Branch if already initialised
                       006D                 UPF_B_FIFLAGS(R10),30$
  10 AA    10 AA   DE  006D         MOVAL   UPF_Q_EDITS(R10), -         ; Initialise edit list head in
                       0072                 UPF_Q_EDITS(R10)            ; first file block
  14 AA    10 AA   DE  0072         MOVAL   UPF_Q_EDITS(R10), -
                       0077                 UPF_Q_EDITS+4(R10)
                       0077         $DISCONNECT RAB=R8,ERR=SUM$CLOSE_ERR ; Disconnect RAB
                       0086  7$:
           30 50   E9  0086         BLBC    R0,40$                      ; Error if LBC
                       0089  10$:
              64   10  0089         BSB     PROCESS_FILE                ; Process update files
           05 50   E9  008B         BLBC    R0,20$                      ; Error if LBC
        5A    6A   D0  008E         MOVL    (R10),R10                   ; Get next file block address
              F6   12  0091         BNEQ    10$                         ; End of list if EQL
                       0093  20$:
        5A    69   D0  0093         MOVL    IS_L_FILELIST(R9),R10       ; Reset file list pointer
  3C A8    1C A9   D0  0096         MOVL    IS_L_MAIN_FAB(R9),RAB$L_FAB(R8) ; Reset FAB address
                       009B         $CONNECT RAB=R8,ERR=SUM$OPEN_ERR
           0447    30  00AA         BSBW    RESTORE_SRC_RFA             ; Restore source file RFA
                       00AD  30$:
  10 A9    10 AA   D0  00AD         MOVL    UPF_Q_EDITS(R10),IS_L_EDIT_BLK(R9) ; Reset edit block pointer
     2A A9    03   88  00B2         BISB2   #SUM_M_AUDIT!SUM_M_AUDITNEW, -  ; Switch on audit trail and
                       00B6                 IS_B_FLAGS(R9)              ; mark first audit as new
           30 A9   B4  00B6         CLRW    IS_W_DELETES(R9)            ; Initialise number of deleted lines
                       00B9  40$:
                   04  00B9         RET
```

```
                                   00BA          .SBTTL  GET_IS_BLK
                                   00BA     ;
                                   00BA     ;++
                                   00BA     ;
                                   00BA     ; Functional description:
                                   00BA     ;
                                   00BA     ;        This routine obtains a memory block for an input stream data
                                   00BA     ;        block and if successful initialises the block.
                                   00BA     ;
                                   00BA     ; Inputs:
                                   00BA     ;
                                   00BA     ;        None
                                   00BA     ;
                                   00BA     ; Outputs:
                                   00BA     ;
                                   00BA     ;        R9  = Address of memory block
                                   00BA     ;
                                   00BA     ;--
                                   00BA     ;
                                   00BA     GET_IS_BLK:
            00000000'EF    9F      00BA          PUSHAB  SUM$VIRT_ADDR              ; Stack arguments for LIB$GET_VM
            00000000'EF    9F      00C0          PUSHAB  SUM_ISSZE                  ; ...
        00000000'GF    02  FB      00C6          CALLS   #2,G^LIB$GET_VM            ; Get memory block
                   1E  50  E9      00CD          BLBC    R0,10$                     ; Error if LBC
         59 00000000'EF    D0      00D0          MOVL    SUM$VIRT_ADDR,R9           ; Get block address
69  0082 8F  00  69  00    2C      00D7          MOVC5   #0,(R9),#0,#IS_K_BLN,(R9)  ; Clear block
          51   32  A9      9E      00DF          MOVAB   IS_T_FAB(R9),R1            ; Set FAB block pointer
                                   00E3          $FAB_STORE  FAB = R1, -            ; and initialise as a FAB
                                   00E3                  BID = #FAB$C_BID, -
                                   00E3                  BLN = #FAB$C_BLN
               50  01      D0      00EB          MOVL    #1,R0                      ; Set success status
                                   00EE     10$:
                           05      00EE          RSB
```

```
                          00EF            .SBTTL  PROCESS_FILE
                          00EF     ;
                          00EF     ;++
                          00EF     ; Functional description:
                          00EF     ;
                          00EF     ;       This routine is called to process each update file
                          00EF     ;
                          00EF     ; Inputs:
                          00EF     ;       R8  = RAB address
                          00EF     ;       R9  = Input stream data block address
                          00EF     ;       R10 = File node address
                          00EF     ;
                          00EF     ; Outputs:
                          00EF     ;
                          00EF     ;       R0  = Success/error status
                          00EF     ;
                          00EF     ; Implicit outputs:
                          00EF     ;
                          00EF     ;       Edit blocks list
                          00EF     ;
                          00EF     ;--
                          00EF     PROCESS_FILE:
  5A A9   38 AA     9F    00EF             MOVAB   UPF_T_NAM(R10), -         ; Set NAM block pointer
                          00F4                     IS_T_FAB+FAB$L_NAM(R9)
  00 36 A9   18     E2    00F4             BBSS    #FAB$V_NAM, -             ; Set for open by NAM block
                          00F9                     IS_T_FAB+FAB$L_FOP(R9),5$
                          00F9     5$:
                          00F9             $OPEN   FAB=IS_T_FAB(R9),ERR=SUM$OPEN_ERR       ; Open input file
        4D 50       E9    0109             BLBC    R0,30$                    ; Error if LBC
  3C A8   32 A9     DE    010C             MOVAL   IS_T_FAB(R9),RAB$L_FAB(R8)       ; Put FAB address into RAB
                          0111             $CONNECT RAB=R8,ERR=SUM$OPEN_ERR ; Connect RAB to FAB
        26 50       E9    0120             BLBC    R0,20$                    ; Error if LBC
                          0123             $FIND   RAB=R8,ERR=SUM$READ_ERR ; Initialise RFA
        05 50       E9    0132             BLBC    R0,10$                    ; Error if LBC
                          0135     ;
        10 A9       D4    0135             CLRL    IS_L_EDIT_BLK(R9)         ; Clear last edit node address
                          0138     ;
                          0138     ; Read update file and create edit nodes
                          0138     ;
           20       10    0138             BSBB    SET_UP_NODES              ; Read update file
                          013A     ;
                          013A     10$:
                          013A             $DISCONNECT RAB=R8,ERR=SUM$CLOSE_ERR
                          0149     20$:
                          0149             $CLOSE  FAB=IS_T_FAB(R9),ERR=SUM$CLOSE_ERR      ; Close input file
                          0159     30$:
                    05    0159             RSB
```

```
                                015A          .SBTTL  SET_UP_NODES
                                015A  ;
                                015A  ; Subroutine to form all edit_nodes
                                015A  ;
                                015A  ; Inputs:
                                015A  ;       R8  = RAB address
                                015A  ;       R10 = file node address
                                015A  ;
                                015A  ; Outputs:
                                015A  ;       R0  = Success/error status
                                015A  ;
                                015A  ;
                                015A  SET_UP_NODES:
                                015A          ASSUME  UPF_W_LOC2 EQ <UPF_W_LOC1+2>
                                015A          ASSUME  ED_W_LOC2 EQ <ED_W_LOC1+2>
                                015A  10$:
        00000000'EF     9F      015A          PUSHAB  SUM$VIRT_ADDR               ; Stack arguments for LIB$GET_VM
        00000004'EF     9F      0160          PUSHAB  SUM_EDSZE                   ; ...
     00000000'GF  02    FB      0166          CALLS   #2,G^LIB$GET_VM             ; Get edit block
              7D  50    E9      016D          BLBC    R0,70$                      ; Error if LBC
  5B    00000000'EF     D0      0170          MOVL    SUM$VIRT_ADDR,R11           ; Set block pointer
        14 AB     5A    D0      0177          MOVL    R10,ED_L_FILE(R11)          ;  Fill in file block address
     19 AB     0C AA    90      017B          MOVB    UPF_B_FILENO(R10), -        ; and file number
                                0180                  ED_B_FILENO(R11)
     0E AB     10 A8    D0      0180          MOVL    RAB$Q_RFA+0(R8),ED_W_RFA+0(R11) ; Record file address (3 words)
     12 AB     14 A8    B0      0185          MOVW    RAB$W_RFA+4(R8),ED_W_RFA+4(R11)
               0C AB    B4      018A          CLRW    ED_W_LINES(R11)
     08 AB     04 AA    D0      018D          MOVL    UPF_Q_LOC1(R10),ED_W_LOC1(R11) ; Move both locator numbers
     18 AB     09 AA    90      0192          MOVB    UPF_B_EDFLAGS(R10),ED_B_FLAGS(R11) ; and flags to edit node
                                0197  30$:
              009C      30      0197          BSBW    READ_UPD_LINEA              ; Read line from input file
              0E  50    E8      019A          BLBS    R0,40$                      ; OK if LBS
    0001827A 8F   50    D1      019D          CMPL    R0,#RMS$_EOF                ; Is error end-of-file?
                  4E    12      01A4          BNEQ    80$                         ; No if NEQ
            54    07    D0      01A6          MOVL    #CMD_M_ALL,R4               ; Fake an end-of-edit command
                  13    11      01A9          BRB     50$                         ; Error will be reported on next pass
                                01AB  40$:
              04AB      30      01AB          BSBW    COMMAND_CHECK               ; Check for command
              E6  50    E9      01AE          BLBC    R0,30$                      ; Syntax error if LBC
        09 54     01    E0      01B1          BBS     #CMD_V_EDTRM,R4,50$         ; Branch if data terminating command
        DE 54     00    E0      01B5          BBS     #CMD_V_CMND,R4,30$          ; Branch if normal command
              0C  AB    B6      01B9          INCW    ED_W_LINES(R11)             ; Increment number of insert lines for
                  D9    11      01BC          BRB     30$                         ; this edit
                                01BE  50$:
              08  AB    D5      01BE          TSTL    ED_W_LOC1(R11)              ; If Loc-1 and Loc-2 = 0 and Lines <> 0
                  21    12      01C1          BNEQ    60$                         ; there is an insert in front of
                                01C3                                      ; the file, otherwise throw this
                                01C3                                      ; Edit node away
              0C  AB    B5      01C3          TSTW    ED_W_LINES(R11)
                  1C    12      01C6          BNEQ    60$
        18 54     02    E1      01C8          BBC     #CMD_V_EDEND,R4,60$         ; Branch if not end of edits
        00000000'EF     9F      01CC          PUSHAB  SUM$VIRT_ADDR               ; Stack arguments for LIB$FREE_VM
        00000004'EF     9F      01D2          PUSHAB  SUM_EDSZE                   ; ...
     00000000'GF  02    FB      01D8          CALLS   #2,G^LIB$FREE_VM            ; Return unused memory block
              0B  50    E9      01DF          BLBC    R0,70$                      ; Error if LBC
                  10    11      01E2          BRB     80$
                                01E4  60$:
```

```
                     0F   10  01E4    490         BSB     INSERT_NODE              ; Insert block into edits list
              0A 54  02   E0  01E6    491         BBS     #CMD_V_EDEND,R4,80$      ; Branch if edit terminating command
                   FF6D   31  01EA    492         BRW     10$                      ; Go back for next edit command
                              01ED    493 70$:
        00000000'EF   00   FB  01ED    494         CALLS   #0,SUM$LIB_ERR           ; Report error
                              01F4    495 80$:
                          05  01F4    496         RSB
```

```
                               01F5            .SBTTL  INSERT_NODE
                               01F5    ;
                               01F5    ; Subroutine to insert block into edit list
                               01F5    ;
                               01F5    ; This routine checks that the edit node is in sequence with any other nodes
                               01F5    ; from the same update file.  If not, the edit node is marked so that a
                               01F5    ; warning can be produced later.  However, the node is placed in the correct
                               01F5    ; position.
                               01F5    ;
                               01F5    ; Inputs:
                               01F5    ;       R11  = address of block to insert
                               01F5    ;       IS_L_EDIT_BLK(R9) = Last edit node inserted from current update file
                               01F5    ;
                               01F5    ; Outputs:
                               01F5    ;       None
                               01F5    ;
                               01F5    ;
                               01F5   INSERT_NODE:
          50    08 AC    D0    01F5            MOVL    8(AP),R0                 ; Get address of first file block
          50    10 A0    DE    01F9            MOVAL   UPF_Q_EDITS(R0),R0       ; and form edit list head address
          51    10 A9    D0    01FD            MOVL    IS_L_EDIT_BLK(R9),R1     ; Get address of last node inserted
                   05    12    0201            BNEQ    10$                      ; If NEQ there is one
             51    50    D0    0203            MOVL    R0,R1                    ; This is first node so scan list
                   10    11    0206            BRB     20$                      ; from list head
                               0208   10$:
    08 A1    08 AB    B1    0208            CMPW    ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is edit out of sequence?
                   09    14    020D            BGTR    20$                      ; No if GTR
       18 AB    02    88    020F            BISB    #ED_M_SEQERR,ED_B_FLAGS(R11) ; Mark edit node
             51    50    D0    0213            MOVL    R0,R1                    ; Scan list from list head to find
                   04    11    0216            BRB     30$                      ; correct position
                               0218   20$:
       10 A9    5B    D0    0218            MOVL    R11,IS_L_EDIT_BLK(R9)    ; Set new 'last edit' address
                               021C   30$:
             51    61    D0    021C            MOVL    (R1),R1                  ; Get next block
             50    51    D1    021F            CMPL    R1,R0                    ; At end of list?
                   07    13    0222            BEQL    40$                      ; Yes if EQL
    08 A1    08 AB    B1    0224            CMPW    ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is new LOC-1 <= current LOC-1
                   F1    14    0229            BGTR    30$                      ; No if GTR
                               022B   40$:
       04 B1    6B    0E    022B            INSQUE  (R11),@ED_L_BWD(R1)      ; Insert new node into list
                   05    022F            RSB
```

```
                              0230   540            .SBTTL  READ_UPD_LINE
                              0230   541   ;
                              0230   542   ; Subroutine to read line sequentially from current update file
                              0230   543   ;
                              0230   544   ; There are two entry points:
                              0230   545   ;
                              0230   546   ;        READ_UPD_LINE   to access the file and read line
                              0230   547   ;
                              0230   548   ;        READ_UPD_LINEA  if update file is already accessed and ready
                              0230   549   ;                        for next line to be read
                              0230   550   ;
                              0230   551   ;
                              0230   552   ; Inputs:
                              0230   553   ;        R8 = RAB address for reading file
                              0230   554   ;
                              0230   555   ; Implicit Inputs:
                              0230   556   ;        SUM_UBF_ADDR    address of local UBF, to aviod access conflicts.
                              0230   557   ;
                              0230   558   ; Outputs:
                              0230   559   ;        R0 = success/error status
                              0230   560   ;        R6 = Line size
                              0230   561   ;        R7 = Line buffer address
                              0230   562   ;
                              0230   563            .ENABL  LSB
                              0230   564   ;
                              0230   565   ;
                              0230   566   READ_UPD_LINE:
              030D     30     0230   567            BSBW    ACCESS_UPDATE                   ; Access update file
             4F 50     E9     0233   568            BLBC    R0,10$                          ; Error if LBC
                              0236   569   ;
                              0236   570   READ_UPD_LINEA:
             24 A8     DD     0236   571            PUSHL   RAB$L_UBF(R8)                   ; Save the old UBF address.
             04 A8     DD     0239   572            PUSHL   RAB$L_ROP(R8)                   ; Save the old ROP field.
04 A8   00010000 8F    CA     023C   573            BICL2   #RAB$M_LOC, RAB$L_ROP(R8)       ; Set MOVE mode for $GET.
24 A8   00000004'EF    D0     0244   574            MOVL    SUM_UBF_ADDR, RAB$L_UBF(R8)     ; Use local buffer.
                              024C   575            $GET    RAB = R8, ERR = SUM$READ_ERR    ; Read line
          04 A8  8E    D0     025B   576            MOVL    (SP)+, RAB$L_ROP(R8)            ; Restore old ROP
          24 A8  8E    D0     025F   577            MOVL    (SP)+, RAB$L_UBF(R8)            ; and UBF.
             1F 50     E9     0263   578            BLBC    R0,10$                          ; If error, don't copy string.
       56   22 A8      3C     0266   579            MOVZWL  RAB$W_RSZ(R8),R6                ; Set line size
       57   28 A8      D0     026A   580            MOVL    RAB$L_RBF(R8),R7                ; and buffer address
       2A A9    04     88     026E   581            BISB2   #SUM_M_SRCUPD,IS_B_FLAGS(R9)    ; Mark as update line
    0E 04 A8    10     E0     0272   582            BBS     #RAB$V_LOC, RAB$L_ROP(R8), 10$  ; Should we copy string to UBF?
                3F     BB     0277   583            PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save registers across MOVC3
             22 A8     28     0279   584            MOVC3   RAB$W_RSZ(R8),-                 ; String length
        00000004'FF           027C   585                    @SUM_UBF_ADDR,-                 ; Source buffer
             24 B8            0281   586                    @RAB$L_UBF(R8)                  ; Destination buffer
                3F     BA     0283   587            POPR    #^M<R0,R1,R2,R3,R4,R5>          ; Restore registers
                       05     0285   588   10$:     RSB
                              0286   589   ;
                              0286   590            .DSABL  LSB
```

```
                          0286          .SBTTL  SUM$LINE
                          0286  ;
                          0286  ; This procedure is called from the main program to get the next
                          0286  ; input line.  This line may come from either the source file or
                          0286  ; an update file.
                          0286  ;
                          0286  ; Inputs:
                          0286  ;
                          0286  ;       4(AP) = Address of control block
                          0286  ;
                          0286  ; Ouputs:
                          0286  ;
                          0286  ;       Next line
                          0286  ;
                          0286  ;
                    0FFC  0286          .ENTRY  SUM$LINE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
         51    04 AC  D0  0288          MOVL    4(AP),R1                ; Get address of control block
         59    04 A1  D0  028C          MOVL    SUM_L_ISDATA(R1),R9     ; Set input stream data block address
         58    20 A9  D0  0290          MOVL    IS_L_MAIN_RAB(R9),R8    ; Get main program RAB address
         5B    10 A9  D0  0294          MOVL    IS_L_EDIT_BLK(R9),R11   ; Get current edit block address
                          0298  SUM_DISPATCH:
   08    00    04 A9  8F  0298          CASEB   IS_B_STATE(R9),#SUM_ST_SET,#SUM_ST_EOF ; Branch to service routine
                   0063'  029D  10$:    .SIGNED_WORD    LINE_SET-10$
                   008A'  029F          .SIGNED_WORD    LINE_NUP-10$
                   0090'  02A1          .SIGNED_WORD    LINE_SRC-10$
                   00A1'  02A3          .SIGNED_WORD    LINE_UPD-10$
                   0127'  02A5          .SIGNED_WORD    LINE_UPE-10$
                   014F'  02A7          .SIGNED_WORD    LINE_UPR-10$
                   016E'  02A9          .SIGNED_WORD    LINE_BLK-10$
                   0191'  02AB          .SIGNED_WORD    LINE_GET-10$
                   0204'  02AD          .SIGNED_WORD    LINE_EOF-10$
                          02AF  ;
                          02AF  SUM_RETURN:
      10 A9    5B     D0  02AF          MOVL    R11,IS_L_EDIT_BLK(R9)   ; Preserve edit block address
         51    04 AC  D0  02B3          MOVL    4(AP),R1                ; Get address of control block
            61    50  D0  02B7          MOVL    R0,SUM_L_STS(R1)        ; Return status
   1C A1    2A A9     90  02BA          MOVB    IS_B_FLAGS(R9), -       ; Edit flags
                          02BF                  SUM_B_FLAGS(R1)
   20 1C A1    02     E1  02BF          BBC     #SUM_V_SRCUPD, -        ; Branch if source line
                          02C4                  SUM_B_FLAGS(R1),5$
   1A A1    2E A9     B0  02C4          MOVW    IS_W_INSERT_NO(R9),SUM_W_INSERT_NO(R1) ; Inserts
   08 A1    18 AA     7D  02C9          MOVQ    UPF_Q_AUDDS(R10), -     ; Supply audit string descriptor
                          02CE                  SUM_Q_AUDDS(R1)
         5A    38 AA  DE  02CE          MOVAL   UPF_T_NAM(R10),R10      ; Form NAM block address
   10 A1    03 AA     9A  02D2          MOVZBL  NAM$B_RSL(R10), -       ; Get file spec size
                          02D7                  SUM_Q_FILESP+0(R1)
   14 A1    04 AA     D0  02D7          MOVL    NAM$L_RSA(R10), -       ; and address
                          02DC                  SUM_Q_FILESP+4(R1)
            20 50     E8  02DC          BLBS    R0,10$                  ; If error line
   1B 1C A1    02     E4  02DF          BBSC    #SUM_V_SRCUPD,SUM_B_FLAGS(R1),10$ ; don't mark as update line
                          02E4  ;
                          02E4  ; Source file line
                          02E4  ;
                          02E4  5$:
 18 A1    06 A9    01     A3  02E4          SUBW3   #1,IS_W_LINE_NO(R9),SUM_W_LINE_NO(R1) ; Number of line being returne
            12 50     E9  02EA          BLBC    R0,10$                  ; If error save deleted line information
                          02ED                                          ; until first good line
   0D 05 A9    01   E5  02ED   649          BBCC    #PRC_V_DELINE, -          ; Branch if no pending deleted info
                        02F2   650                  IS_B_PROCFLAGS(R9),10$
   1A A1    30 A9   B0  02F2   651          MOVW    IS_W_DELETES(R9),SUM_W_INSERT_NO(R1) ; Return number of lines delete
   03 1C A1    04   E2  02F7   652          BBSS    #SUM_V_DELETE, -          ; Set deleted lines information flag
                        02FC   653                  SUM_B_FLAGS(R1),10$
            30 A9   B4  02FC   654          CLRW    IS_W_DELETES(R9)          ; Reset number of deleted lines
                        02FF   655 10$:
                    04  02FF   656          RET
```

```
                       0300             .SBTTL  LINE_SET
                       0300      ;
                       0300      ; Routine to service SET state
                       0300      ; Determines if the next line is to come from the main source file
                       0300      ; or from an update file.  If there are no more updates to be processed
                       0300      ; the state is set to NUP; if there are updates but the next update is to
                       0300      ; be applied to a later source line the state is set to SRC; if the next
                       0300      ; line is to come from an update file the state is set to UPD.
                       0300      ;
                       0300      ;
                       0300      ; Inputs:
                       0300      ;
                       0300             R11 = Current edit block address
                       0300      ;
                       0300      ; Outputs:
                       0300      ;
                       0300             state changed
                       0300      ;
                       0300      LINE_SET:
   04 A9    01    90   0300             MOVB    #SUM_ST_NUP,IS_B_STATE(R9) ; Assume no more updates
      51    69    D0   0304             MOVL    IS_L_FILELIST(R9),R1    ; Get address of first file block
            18    13   0307             BEQL    10$                     ; If EQL there are no update files
   51    10 A1    DE   0309             MOVAL   UPF_Q_EDITS(R1),R1      ; Form edit block list head address
      5B    51    D1   030D             CMPL    R1,R11                  ; Any edits still in list?
            0F    13   0310             BEQL    10$                     ; No if EQL so must be source line
   04 A9    03    90   0312             MOVB    #SUM_ST_UPD,IS_B_STATE(R9) ; Assume next line is from update file
08 AB    06 A9    B1   0316             CMPW    IS_W_LINE_NO(R9), -     ; Is line number of source file less
                       031B                     ED_W_LOC1(R11)          ; than locator-1 of next edit?
            07    18   031B             BGEQ    20$                     ; No if GEQ
   04 A9    02    90   031D             MOVB    #SUM_ST_SRC,IS_B_STATE(R9) ; Change state to source
                       0321      10$:
          0187    30   0321             BSBW    ACCESS_SRC              ; Access source file
                       0324      20$:
          FF71    31   0324             BRW     SUM_DISPATCH            ; and dispatch again
```

```
                    0327   693          .SBTTL  LINE_NUP
                    0327   694 ;
                    0327   695 ; There are no more updates to process so just read next source line
                    0327   696 ; and return to caller.  The source file is already accessed so
                    0327   697 ; READ_SRC_LINEA can be used.
                    0327   698 ;
                    0327   699 ;
                    0327   700 ; Inputs:
                    0327   701 ;
                    0327   702 ;        None
                    0327   703 ;
                    0327   704 ; Outputs:
                    0327   705 ;
                    0327   706 ;        None
                    0327   707 ;
                    0327   708 ;
                    0327   709 LINE_NUP:
      02CA    30    0327   710          BSBW    READ_SRC_LINEA          ; Get next source line
      FF82    31    032A   711          BRW     SUM_RETURN              ; and return
```

```
                                032D            .SBTTL  LINE_SRC
                                032D    ;
                                032D    ; The next source line is read from the main input file.  The line
                                032D    ; number is incremented and compared with the locator-1 value of the
                                032D    ; next edit.  If the line number remains lower the state remains at SRC.
                                032D    ; If the line number is equal or greater the state is changed to UPD.
                                032D    ; The next call to SUM$LINE will then get an update line.
                                032D    ;
                                032D    ; Inputs:
                                032D    ;
                                032D    ;       R11 = Current edit block address
                                032D    ;
                                032D    ; Outputs:
                                032D    ;
                                032D    ;       state
                                032D    ;
                                032D    ;
                                032D    LINE_SRC:
               02C4      30     032D            BSBW    READ_SRC_LINE           ; Get next line from source file
      08 AB    06 A9     B1     0330            CMPW    IS_W_LINE_NO(R9), -     ; Is source line number still lower
                                0335                    ED_W_LOC1(R11)          ; than next locator-1
                 04      19     0335            BLSS    10$                     ; Yes if LSS
         04 A9   03      90     0337            MOVB    #SUM_ST_UPD,IS_B_STATE(R9) ; Reset state to UPD
                                033B    10$:
               FF71      31     033B            BRW     SUM_RETURN              ; and return with line
```

```
                        033E         .SBTTL  LINE_UPD
                        033E ;
                        033E ; The next update operation is prepared by determining the range of
                        033E ; the edit, that is the number of edit operations which have clashed.
                        033E ;
                        033E ; Inputs:
                        033E ;
                        033E         R9  = Input stream data pointer
                        033E         R11 = Current edit block address
                        033E ;
                        033E ; Outputs:
                        033E ;
                        033E         IS_L_FIRST_EDIT(R9) = First edit block of update
                        033E         IS_L_LAST_EDIT(R9)  = Last edit block of update
                        033E         IS_W_HIGH_LOC2(R9)  = Highest loc-2 value of update
                        033E ;
                        033E ;
                        033E LINE_UPD:
14 A9     5B       D0   033E         MOVL    R11,IS_L_FIRST_EDIT(R9) ; Save address of first edit
54   0A AB         3C   0342         MOVZWL  ED_W_LOC2(R11),R4       ; Set highest loc-2 value
2C A9     54       B0   0346         MOVW    R4,IS_W_HIGH_LOC2(R9)   ; and supply as routine output
2A A9     08       8A   034A         BICB2   #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; May be first edit in clash
                   8A   034E         BICB2   #<PRC_M_ERRORS! -       ; Assume no clash errors,
                        034F                   PRC_M_HIEDIT! -       ; highest edit does not overide others,
                        034F                   PRC_M_NODATA>,-       ; and all data lines inserted
05 A9     1C            034F                 IS_B_PROCFLAGS(R9)
04 A9     05       90   0352         MOVB    #SUM_ST_UPR,IS_B_STATE(R9)
   55     69       D0   0356         MOVL    IS_L_FILELIST(R9),R5    ; Set files list
55   10 A5         DE   0359         MOVAL   UPF_Q_EDITS(R5),R5      ; list head address
                        035D 10$:
   52     6B       D0   035D         MOVL    (R11),R2                ; Point to next edit block
   55     52       D1   0360         CMPL    R2,R5                   ; At end of list?
          2F       13   0363         BEQL    40$                     ; Yes if EQL
   51     54       D0   0365         MOVL    R4,R1                   ; Set highest locator value of edit
          0A       12   0368         BNEQ    20$                     ; If zero set from loc-2 of current edit
51   0A AB         3C   036A         MOVZWL  ED_W_LOC2(R11),R1       ; Set highest locator value of edit
          04       12   036E         BNEQ    20$                     ; If zero set from loc-1 of current edit
51   08 AB         3C   0370         MOVZWL  ED_W_LOC1(R11),R1       ; Set highest locator value of edit
                        0374 20$:
08 A2     51       B1   0374         CMPW    R1,ED_W_LOC1(R2)        ; Does this edit overlap with next?
          1A       19   0378         BLSS    40$                     ; No if LSS
                        037A ;
                        037A ; This edit block clashes with next
                        037A ;
0A A2     54       B1   037A         CMPW    R4,ED_W_LOC2(R2)        ; Is its loc-2 higher than current loc-2
          04       18   037E         BGEQ    25$                     ; No if GEQ
54   0A A2         3C   0380         MOVZWL  ED_W_LOC2(R2),R4        ; Extend range of edit
                        0384 25$:
     0A A2         B5   0384         TSTW    ED_W_LOC2(R2)           ; Is edit all inserts?
          04       13   0387         BEQL    30$                     ; Yes if EQL
05 A9     08       88   0389         BISB    #PRC_M_HIEDIT,IS_B_PROCFLAGS(R9) ; Highest edit overides others
                        038D 30$:                                    ; therefore replace later)
          21       10   038D         BSBB    CHECK_ERR               ; See if error should be reported
   5B     52       D0   038F         MOVL    R2,R11                  ; Point to next edit block
          C9       11   0392         BRB     10$
                        0394 40$:
18 A9     5B       D0   0394         MOVL    R11,IS_L_LAST_EDIT(R9)  ; Set address of last edit block
```

```
      14 A9    5B    D1  0398          CMPL    R11,IS_L_FIRST_EDIT(R9) ; If first block then single non-clashing
               0F    13  039C          BEQL    50$                     ; edit else last block of clashing edits
               10    10  039E          BSBB    CHECK_ERR               ; See if error should be reported
   08 05 A9    02    E1  03A0          BBC     #PRC_V_ERRORS, -        ; Branch if no errors to report
                         03A5                  IS_B_PROCFLAGS(R9),50$
      04 A9    04    90  03A5          MOVB    #SUM_ST_UPE,IS_B_STATE(R9) ; Set state to report errors
      5B    14 A9    D0  03A9          MOVL    IS_L_FIRST_EDIT(R9),R11 ; Reset edit block pointer to first
                         03AD  50$:
             FEE8    31  03AD          BRW     SUM_DISPATCH
                         03B0  ;
                         03B0  ;
                         03B0  ;
                         03B0  ; Local subroutine to check if clashing edit should be reported
                         03B0  ;
                         03B0  ; Inputs:
                         03B0  ;
                         03B0  ;       R11 = Edit block address
                         03B0  ;
                         03B0  ; Outputs:
                         03B0  ;
                         03B0  ;       None
                         03B0  ;
                         03B0  CHECK_ERR:
   0E 18 AB    00    E0  03B0          BBS     #ED_V_SUPPRESS, -       ; Branch if suppress bit set
                         03B5                  ED_B_FLAGS(R11),20$
         08 AB       D5  03B5          TSTL    ED_W_LOC1(R11)          ; If Loc-1, Loc-2 and lines = 0
            05       12  03B8          BNEQ    10$                     ; then do not report as error
         0C AB       B5  03BA          TSTW    ED_W_LINES(R11)
            04       13  03BD          BEQL    20$
                         03BF  10$:
      05 A9    04    88  03BF          BISB    #PRC_M_ERRORS,IS_B_PROCFLAGS(R9) ; Set error report bit
                         03C3  20$:
                     05  03C3          RSB
```

```
                               03C4          .SBTTL  LINE_UPE
                               03C4    ;
                               03C4    ; The update operation contains clashing edits which must be reported
                               03C4    ;
                               03C4    ; Inputs:
                               03C4    ;
                               03C4    ;        R11 = Address of next clashing edit
                               03C4    ;
                               03C4    ; Outputs:
                               03C4    ;
                               03C4    ;        R11 = Edit block pointer advanced
                               03C4    ;
                               03C4    ;
                               03C4    LINE_UPE:
        5A   14 AB     D0      03C4          MOVL    ED_L_FILE(R11),R10      ; Get file block address of clashing edit
                FE65   30      03C8          BSBW    READ_UPD_LINE           ; Read update file to get edit line
   50    00848800 8F   D0      03CB          MOVL    #SUM$_EDITSCLSH,R0      ; Set return status
        14 A9     5B   D1      03D2          CMPL    R11,IS_L_FIRST_EDIT(R9) ; First report of this set of clashes
                  04   13      03D6          BEQL    10$                     ; Yes if EQL
        2A A9     08   88      03D8          BISB    #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; Set 2nd or later flag
                               03DC    10$:
        18 A9     5B   D1      03DC          CMPL    R11,IS_L_LAST_EDIT(R9)  ; At last edit?
                  04   12      03E0          BNEQ    20$                     ; No if NEQ
        04 A9     05   90      03E2          MOVB    #SUM_ST_UPR,IS_B_STATE(R9) ; Set state to Update Ready
                               03E6    20$:
             5B   6B   D0      03E6          MOVL    (R11),R11               ; Advance to next edit block
                FEC3   31      03E9          BRW     SUM_RETURN
```

```
                        03EC          .SBTTL  LINE_UPR
                        03EC    ;
                        03EC    ; The next update operation is ready.  Any errors have been reported
                        03EC    ; to the caller.
                        03EC    ;
                        03EC    ;
                        03EC    ; Inputs:
                        03EC    ;
                        03EC    ;       R11 = Current edit block address
                        03EC    ;
                        03EC    ; Outputs:
                        03EC    ;
                        03EC    ;       None
                        03EC    ;
                        03EC    ;
                        03EC    LINE_UPR:
5B    14 A9     D0      03EC            MOVL    IS_L_FIRST_EDIT(R9),R11 ; Reset pointer to first edit block
04 A9    06     90      03F0            MOVB    #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
54    2C A9     3C      03F4            MOVZWL  IS_W_HIGH_LOC2(R9),R4   ; Is edit operation an insert?
         0B     12      03F8            BNEQ    50$                     ; No if NEQ
      08 AB     B5      03FA            TSTW    ED_W_LOC1(R11)          ; Is insert to front of file?
         09     13      03FD            BEQL    60$                     ; Yes if EQL
       01E9     30      03FF            BSBW    READ_SRC_LINE           ; Read one more line from source
       FEAA     31      0402            BRW     SUM_RETURN
                        0405    50$:
       0220     30      0405            BSBW    SKIP_SRC_LINES          ; Skip over source lines to be deleted
                        0408    60$:
       FE8D     31      0408            BRW     SUM_DISPATCH            ; and dispatch
```

```
                    040B          .SBTTL  LINE_BLK
                    040B  ;
                    040B  ; This routine is called to begin processing of the next edit block
                    040B  ; The file from which edit lines will come is prepared for access.  The
                    040B  ; state is reset to GET.
                    040B  ;
                    040B  ;
                    040B  ; Inputs:
                    040B  ;
                    040B  ;       R11 = Current edit block address
                    040B  ;
                    040B  ; Outputs:
                    040B  ;
                    040B  ;       None
                    040B  ;
                    040B  ;
                    040B  LINE_BLK:
       5A    14 AB  D0  040B          MOVL    ED_L_FILE(R11),R10      ; Get file block address of file
              012E  30  040F          BSBW    ACCESS_UPDATE           ; Prepare for reading file
             16 50  E9  0412          BLBC    R0,20$                  ; Error if LBC
   00 05 A9     00  E5  0415          BBCC    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),5$ ; Clear expected edit flag
                    041A  5$:
             08 AB  B5  041A          TSTW    ED_W_LOC1(R11)          ; Is this insert in front of file?
                05  12  041D          BNEQ    10$                     ; No if NEQ
   00 05 A9     00  E2  041F          BBSS    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),10$ ; Set expected edit flag
                    0424  10$:
       04 A9    07  90  0424          MOVB    #SUM_ST_GET,IS_B_STATE(R9) ; Reset state to GET
              FE6D  31  0428          BRW     SUM_DISPATCH            ; and dispatch again
                    042B  20$:
              FE81  31  042B          BRW     SUM_RETURN              ; Return to caller with error
```

```
                               042E           .SBTTL  LINE_GET
                               042E   ;
                               042E   ; Routine to get next line from update file
                               042E   ;
                               042E   ;
                               042E   ; Inputs:
                               042E   ;
                               042E   ;       R11 = Current edit block address
                               042E   ;
                               042E   ; Outputs:
                               042E   ;
                               042E   ;       R11 = Next edit block address
                               042E   ;
                               042E   ;
                               042E   LINE_GET:
          5A   14 AB     D0    042E           MOVL    ED_L_FILE(R11),R10      ; Set file block pointer
                               0432   10$:
                 FE01    30    0432           BSBW    READ_UPD_LINEA          ; Get next line from update file
                 12 50   E8    0435           BLBS    R0,20$                  ; OK if LBS
      0001827A 8F   50   D1    0438           CMPL    R0,#RMS$_EOF            ; Is error end-of-file?
                   2F    12    043F           BNEQ    35$                     ; No if NEQ
   50   00848810 8F      D0    0441           MOVL    #SUM$_PRMEOF,R0         ; Set premature end-of-file status
                   28    11    0448           BRB     40$
                               044A   20$:
                 020C    30    044A           BSBW    COMMAND_CHECK           ; Check for syntax and type
                 44 50   E9    044D           BLBC    R0,80$                  ; Syntax error if LBC
             07 54   00  E0    0450           BBS     #CMD_V_CMND,R4,30$      ; Branch if command line
          D9 05 A9   04  E0    0454           BBS     #PRC_V_NODATA, -        ; Ignore data line if higher precedence
                               0459                   IS_B_PROCFLAGS(R9),10$  ; edit is overiding others
                   40    11    0459           BRB     90$                     ; Return to caller with line
                               045B   30$:
             D3 54   01  E1    045B           BBC     #CMD_V_EDTRM,R4,10$     ; Branch if not edit terminating command
          0E 05 A9   00  E2    045F           BBSS    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),40$ ; If expecting edit get next lin
          C9 18 AB   01  E1    0464           BBC     #ED_V_SEQERR,ED_B_FLAGS(R11),10$ ; Was edit out of sequence?
   50   00848818 8F      D0    0469           MOVL    #SUM$_EDOUTSEQ,R0       ; Yes: report error now
                               0470   35$:
                   2C    11    0470           BRB     100$
                               0472   ;
                               0472   ; Found end of this set of lines
                               0472   ;
                               0472   40$:
          18 A9    5B    D1    0472           CMPL    R11,IS_L_LAST_EDIT(R9)  ; Last edit block in range?
                   0F    13    0476           BEQL    60$                     ; Yes if EQL
       04 05 A9    03    E1    0478           BBC     #PRC_V_HIEDIT, -        ; Branch if concatenating inserts
                               047D                   IS_B_PROCFLAGS(R9),50$  ;
          05 A9    10    88    047D           BISB    #PRC_M_NODATA,IS_B_PROCFLAGS(R9) ; Ignore data from other edits
                               0481   50$:
          04 A9    06    90    0481           MOVB    #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
                   04    11    0485           BRB     70$
                               0487   60$:
          04 A9    00    90    0487           MOVB    #SUM_ST_SET,IS_B_STATE(R9) ; Reset state to SET
                               048B   70$:
             5B    6B    D0    048B           MOVL    (R11),R11               ; Point to next edit block
                 0D 50   E9    048E           BLBC    R0,100$                 ; If error return to caller first
                 FE04    31    0491           BRW     SUM_DISPATCH            ; or dispatch again
                               0494   80$:
   50   00848808 8F      D0    0494           MOVL    #SUM$_SLPSYNERR,R0      ; Set SLP syntax error status
```

```
                049B   975 90$:
2E A9     B6    049B   976          INCW    IS_W_INSERT_NO(R9)       ; Increment number of new/replace lines
                049E   977 100$:
 FE0E     31    049E   978          BRW     SUM_RETURN               ; Return to caller
```

```
                        04A1   980          .SBTTL  LINE_EOF
                        04A1   981  ;
                        04A1   982  ; Routine to service EOF state.  An RMS end-of-file state is
                        04A1   983  ; returned to the caller
                        04A1   984  ;
                        04A1   985  ;
                        04A1   986  ; Inputs:
                        04A1   987  ;
                        04A1   988  ;       None
                        04A1   989  ;
                        04A1   990  ;
                        04A1   991  ; Outputs:
                        04A1   992  ;
                        04A1   993  ;       None
                        04A1   994  ;
                        04A1   995  ;
                        04A1   996 LINE_EOF:
 50   0001827A 8F   D0  04A1   997          MOVL    #RMS$_EOF,R0            ; Set R0 to eof state
            FE04   31  04A8   998          BRW     SUM_RETURN              , and return to caller
```

```
                          04AB          .SBTTL  ACCESS_SRC
                          04AB    ;
                          04AB    ; Routine to access main source file.  The RAB is connected to
                          04AB    ; the main file FAB if it is not already connected.
                          04AB    ;
                          04AB    ; Inputs:
                          04AB    ;
                          04AB    ;       R8  = Main program RAB address
                          04AB    ;
                          04AB    ;
                          04AB    ; Outputs:
                          04AB    ;
                          04AB    ;       None
                          04AB    ;
                          04AB    ;
                          04AB   ACCESS_SRC:
             02 A8   B5   04AB          TSTW    RAB$W_ISI(R8)           ; Is it connected to a FAB?
                1F   13   04AE          BEQL    10$                     ; No if EQL
     51      32 A9   DE   04B0          MOVAL   IS_T_FAB(R9),R1         ; Set input stream FAB address
     51      3C A8   D1   04B4          CMPL    RAB$L_FAB(R8),R1        ; Is it connected to SUM FAB?
                2E   12   04B8          BNEQ    20$                     ; No if NEQ, it's connected to main FAB
                          04BA          $DISCONNECT RAB = R8, -         ; Disconnect RAB from SUM FAB
                          04BA                  ERR = SUM$CLOSE_ERR
             0C A9   D4   04C9          CLRL    IS_L_CONN_FILE(R9)      ; Clear file connected flag
             19 50   E9   04CC          BLBC    R0,20$                  ; Error if LBC
                          04CF   10$:
  3C A8      1C A9   D0   04CF          MOVL    IS_L_MAIN_FAB(R9), -    ; Put main program FAB into RAB
                          04D4                  RAB$L_FAB(R8)
                          04D4          $CONNECT RAB = R8, -            ; Connect main program FAB to RAB
                          04D4                  ERR = SUM$OPEN_ERR
             02 50   E9   04E3          BLBC    R0,20$                  ; Error if LBC
                0C   10   04E6          BSB     RESTORE_SRC_RFA         ; Restore source file RFA
                          04E8   20$:
                     05   04E8          RSB
```

```
SUM
Sym

$$$
$$$
$$$
$$$
$$$
$$.
$$.
$$K
..A
..F
..M
..T
.LE
ACC
ACC
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
ACT
AUD
BIT
CHE
CMD
CMD
CMD
CMD
CMD
CMD
CMD
CMN
CMN
COM
COM
DAT
EDI
ED_
ED_
ED_
ED_
ED_
ED_
ED_
ED_
ED_
ED_
```

```
                         04E9  1035          .SBTTL  SAVE_SRC_RFA
                         04E9  1036  ;
                         04E9  1037  ;
                         04E9  1038  ; Routine to save source file record file address
                         04E9  1039  ;
                         04E9  1040  ; Inputs:
                         04E9  1041  ;
                         04E9  1042  ;       R8  = RAB address
                         04E9  1043  ;
                         04E9  1044  ; Outputs:
                         04E9  1045  ;
                         04E9  1046  ;       None
                         04E9  1047  ;
                         04E9  1048  ;
                         04E9  1049  SAVE_SRC_RFA:
  24 A9    10 A8    D0   04E9  1050          MOVL    RAB$W_RFA+0(R8), -        ; Move RFA to save buffer
                         04EE  1051                  IS_W_MAIN_RFA+0(R9)
  28 A9    14 A8    B0   04EE  1052          MOVW    RAB$W_RFA+4(R8), -
                         04F3  1053                  IS_W_MAIN_RFA+4(R9)
                    05   04F3  1054          RSB
```

```
                             04F4               .SBTTL  RESTORE_SRC_RFA
                             04F4       ;
                             04F4       ;
                             04F4       ; Routine to restore source file record file address and
                             04F4       ; reset record pointers.  If RFA is zero a rewind is performed,
                             04F4       ; if non-zero the record is located b; a find.
                             04F4       ;
                             04F4       ;
                             04F4       ; Inputs:
                             04F4       ;
                             04F4       ;       R8  = RAB address
                             04F4       ;
                             04F4       ;
                             04F4       ; Outputs:
                             04F4       ;
                             04F4       ;       R0  = Success/error status
                             04F4       ;
                             04F4       ;
                             04F4      RESTORE_SRC_RFA:
    10 A8    24 A9    D0     04F4               MOVL    IS_W_MAIN_RFA+0(R9), -   ; Move RFA back to RAB
                             04F9                       RAB$Q_RFA+0(R8)          ; (3 words)
    14 A8    28 A9    B0     04F9               MOVW    IS_W_MAIN_RFA+4(R9), -
                             04FE                       RAB$Q_RFA+4(R8)
                16    12     04FE               BNEQ    10$                      ; If NEQ then do find
             10 A8    D5     0500               TSTL    RAB$W_RFA+0(R8)          ; Test other part of RFA
                11    12     0503               BNEQ    10$                      ; If NEQ then do find
                             0505               $REWIND RAB = R8, -              ; Rewind to start of file
                             0505                       ERR = SUM$READ_ERR
                29    11     0514               BRB     20$
                             0516      10$:
       1E A8    02    90     0516               MOVB    #RAB$C_RFA,RAB$B_RAC(R8); Put into RFA access mode
                             051A               $FIND   RAB = R8, -              ; Reset record pointers
                             051A                       ERR = SUM$READ_ERR
       1E A8    00    90     0529               MOVB    #RAB$C_SEQ,RAB$B_RAC(R8); Reset to sequential access mode
          0F    50    E9     052D               BLBC    R0,20$                   ; Error if LBC
                             0530               $GET    RAB = R8, -              ; Advance past this record which has
                             0530                       ERR = SUM$READ_ERR       ; read before.
                             053F      20$:
                      05     053F               RSB
```

```
SUM
Symb
TPA_
TPA_
TPA_
TPA_
TPA_
TPA_
TPA_
TPA_
TPA_
TPA_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_
UPF_

PSEC
----
.  A
$ABS
SUM
SUM
_LIE
_LIE
_LIE
SUM

Phas
----
Init
Comm
Pass
Symb
Pass
Symb
Psec
Cros
Asse

The
```

```
                    0540         .SBTTL  ACCESS_UPDATE
                    0540 ;
                    0540 ; Routine to access update file
                    0540 ;
                    0540 ;
                    0540 ; Inputs:
                    0540 ;
                    0540 ;       R8  = Main program RAB address
                    0540 ;       R10 = File block address of required update file
                    0540 ;       R11 = Edit block address of next edit
                    0540 ;
                    0540 ;
                    0540 ; Ouputs:
                    0540 ;
                    0540 ;       R9  = FAB address
                    0540 ;
                    0540 ;
                    0540 ACCESS_UPDATE:
           1C   BB  0540         PUSHR   #^M<R2,R3,R4>
    52  08 A9   9E  0542         MOVAB   IS_L_OPEN_FILE(R9),R2   ; Set pointer to file open
    53  0C A9   9E  0546         MOVAB   IS_L_CONN_FILE(R9),R3   ; and file connected markers
    54  32 A9   DE  054A         MOVAL   IS_T_FAB(R9),R4         ; Set pointer to SUM's FAB
        02 A8   B5  054E         TSTW    RAB$W_ISI(R8)           ; Is RAB connected to a FAB?
           5B   13  0551         BEQL    30$                     ; No if EQL
    54  3C A8   D1  0553         CMPL    RAB$L_FAB(R8),R4        ; Is it connected to SUM's FAB?
           07   12  0557         BNEQ    10$                     ; No if NEQ
       63  5A   D1  0559         CMPL    R10,(R3)                ; Is it connected to required file?
           69   13  055C         BEQL    40$                     ; Yes if EQL
           03   11  055E         BRB     20$
                    0560 10$:
         FF86   30  0560         BSBW    SAVE_SRC_RFA            ; Save source file RFA
                    0563 20$:
                    0563         $DISCONNECT RAB = R8, -         ; Disconnect RAB from FAB
                    0563                 ERR = SUM$CLOSE_ERR
        73 50   E9  0572         BLBC    R0,50$                  ; Error if LBC
           63   D4  0575         CLRL    (R3)                    ; Mark that no file is connected
       62  5A   D1  0577         CMPL    R10,(R2)                ; Is required file already open?
           32   13  057A         BEQL    30$                     ; Yes if EQL
           62   D5  057C         TSTL    (R2)                    ; Is any file open on this FAB?
           14   13  057E         BEQL    25$                     ; No if EQL
                    0580         $CLOSE  FAB = R4, -             ; Close currently open update file
                    0580                 ERR = SUM$CLOSE_ERR
        56 50   E9  058F         BLBC    R0,50$                  ; Error if LBC
           62   D4  0592         CLRL    (R2)                    ; Mark that no file is open
                    0594 25$:
 28 A4  38 AA   DE  0594         MOVAL   UPF_T_NAM(R10), -       ; Put NAM block into FAB
                    0599                 FAB$L_NAM(R4)
                    0599         $OPEN   FAB = R4, -             ; Open required update file
                    0599                 ERR = SUM$OPEN_ERR
        3D 50   E9  05A8         BLBC    R0,50$                  ; Error if LBC
       62  5A   D0  05AB         MOVL    R10,(R2)                ; Mark which file is open
                    05AE 30$:
    3C A8  54   D0  05AE         MOVL    R4,RAB$L_FAB(R8)        ; Put FAB address in RAB
                    05B2         $CONNECT RAB = R8, -            ; Connect RAB to FAB
                    05B2                 ERR = SUM$OPEN_ERR
        24 50   E9  05C1         BLBC    R0,50$                  ; Error if LBC
       63  5A   D0  05C4         MOVL    R10,(R3)                ; Mark which file is connected
```

```
                       05C7       40$:
10 A8   0E AB    D0    05C7                 MOVL    ED_W_RFA+0(R11), -      ; Reset RFA (3 words)
                       05CC                         RAB$Q_RFA+0(R8)
14 A8   12 AB    B0    05CC                 MOVW    ED_W_RFA+4(R11), -
                       05D1                         RAB$Q_RFA+4(R8)
   1E A8    02   90    05D1                 MOVB    #RAB$C_RFA,RAB$B_RAC(R8); Put into RFA access mode
                       05D5                 $FIND   RAB = R8, -             ; Position file
                       05D5                         ERR = SUM$READ_ERR
   1E A8    00   90    05E4                 MOVB    #RAB$C_SEQ,RAB$B_RAC(R8); Reset to sequential access mode
                       05E8       50$:
            1C   BA    05E8                 POPR    #^M<R2,R3,R4>
                 05    05EA                 RSB
```

```
                           05EB         .SBTTL  READ_SRC_LINE
                           05EB   ;
                           05EB   ; Routine to read one line from source file
                           05EB   ;
                           05EB   ; There are two entry points:
                           05EB   ;
                           05EB   ;     READ_SRC_LINE   to access file and read line
                           05EB   ;
                           05EB   ;     READ_SRC_LINEA  if file is already accessed and ready to
                           05EB   ;                     read next line
                           05EB   ;
                           05EB   ; Inputs:
                           05EB   ;
                           05EB   ;     R8  = RAB address
                           05EB   ;
                           05EB   ; Outputs:
                           05EB   ;
                           05EB   ;     R0  = Success/error status
                           05EB   ;     R6  = Line size
                           05EB   ;     R7  = Line buffer address
                           05EB   ;     IS_W_LINE_NO(R9) = line number
                           05EB   ;
                           05EB         .ENABL  LSB
                           05EB   ;
                           05EB   READ_SRC_LINE:
          50    01    D0   05EB         MOVL    #1,R0                   ; Assume success
              FEBA    30   05EE         BSBW    ACCESS_SRC              ; Access source file
             33 50    E9   05F1         BLBC    R0,20$                  ; Error if LBC
                           05F4   ;
                           05F4   READ_SRC_LINEA:
                           05F4         $GET    RAB = R8, -             ; Get next line from source file
                           05F4                 ERR = SUM$READ_ERR
             0F 50    E8   0603         BLBS    R0,10$                  ; OK if LBS
 0001827A 8F    50    D1   0606         CMPL    R0,#RMS$_EOF            ; Was error end-of-file?
                18    12   060D         BNEQ    20$                     ; No if NEQ
       04 A9    08    90   060F         MOVB    #SUM_ST_EOF,IS_B_STATE(R9) ; Set into EOF state
                12    11   0613         BRB     20$
                           0615   10$:
       56    22 A8    3C   0615         MOVZWL  RAB$W_RSZ(R8),R6        ; Set record size
       57    28 A8    D0   0619         MOVL    RAB$L_RBF(R8),R7        ; and buffer address
             06 A9    B6   061D         INCW    IS_W_LINE_NO(R9)                ; Increment line number
       2A A9    04    8A   0620         BICB2   #SUM_M_SRCUPD,IS_B_FLAGS(R9)    ; Mark as source line
             2E A9    B4   0624         CLRW    IS_W_INSERT_NO(R9)              ; Reset new/replacement lines count
                           0627   20$:
                      05   0627         RSB
                           0628   ;
                           0628         .DSABL  LSB
```

```
                          0628         .SBTTL  SKIP_SRC_LINES
                          0628   ;
                          0628   ; Routine to skip over source file lines
                          0628   ;
                          0628   ; Inputs:
                          0628   ;
                          0628   ;       R4  = Last line number to skip
                          0628   ;       R8  = RAB address
                          0628   ;
                          0628   ; Outputs:
                          0628   ;
                          0628   ;       IS_W_LINE_NO(R9) = Last line number
                          0628   ;
                          0628   ;
                          0628   SKIP_SRC_LINES:
           50    01   D0  0628          MOVL    #1,R0                          ; Assume success
        06 A9    54   B1  062B          CMPW    R4,IS_W_LINE_NO(R9)                    ; Need to skip any?
                 27   19  062F          BLSS    20$                            ; No if LSS
               FE77   30  0631          BSBW    ACCESS_SRC                     ; Access source file
           21 50      E9  0634          BLBC    R0,20$                         ; Error if LBC
                          0637   10$:
                          0637          $FIND   RAB = R8, -                    ; Skip one line
                          0637                  ERR = SUM$READ_ERR
           0F 50      E9  0646          BLBC    R0,20$                         ; Error if LBC
           30 A9      B6  0649          INCW    IS_W_DELETES(R9)                       ; Increment deleted lines count
  FFE4 06 A9   01  54  3D  064C          ACBW    R4,#1,IS_W_LINE_NO(R9),10$             ; Increment line number and branch b
                          0653                                                 ; if more lines to skip
     00 05 A9    01   E2  0653          BBSS    #PRC_V_DELINE, -               ; Set deleted lines information
                          0658                  IS_B_PROCFLAGS(R9),20$         ; pending flag
                          0658   20$:
                      05  0658          RSB
```

```
                              0659          .SBTTL  COMMAND_CHECK
                              0659  ;
                              0659  ; Subroutine to check if line is a command
                              0659  ;
                              0659  ; Inputs:
                              0659  ;
                              0659  ;       R6 = Size of line
                              0659  ;       R7 = Address of line
                              0659  ;       R8 = RAB address
                              0659  ;       R9 = Input stream control block
                              0659  ;       R10= File block address
                              0659  ;
                              0659  ; Outputs:
                              0659  ;       R4[CMND] = 0:Data  1:Command
                              0659  ;       R4[EDTRM] = 0:Normal command  1:Data terminator command
                              0659  ;       R4[EDEND] = 0:Data terminator  1:End of edit
                              0659  ;       R6 = Size of line
                              0659  ;       R7 = Address of line
                              0659  ;
                              0659 COMMAND_CHECK:
                              0659          ASSUME  UPF_W_LOC2 EQ <UPF_W_LOC1+2>
00000000'EF    58       D0    0659          MOVL    R8, SUM_CUR_RAB            ; Make the currently active RAB available
                              0660                                             ; to the TPARSE action routines.
51   00000008'EF        DE    0660          MOVAL   TPARSE_BLOCK,R1            ; Set pointer to Tparse parameter block
   28 A1   2A A9        90    0667          MOVB    IS_B_FLAGS(R9), -          ; Get current input stream flags byte
                              066C                  TPA_B_ISFLAGS(R1)
   00 28 A1   01        E5    066C          BBCC    #SUM_V_AUDITNEW, -         ; but clear new audit trail flag
                              0671                  TPA_B_ISFLAGS(R1),5$
                              0671 5$:
          29 A1         94    0671          CLRB    TPA_B_EDFLAGS(R1)          ; Clear all edit flags
   2A A1   0A AA        B0    0674          MOVW    UPF_W_DOT(R10), -          ; Get current dot value
                              0679                  TPA_W_DOT(R1)
          2C A1         D4    0679          CLRL    TPA_W_LOC(R1)              ; Clear locator value and line type
          24 A1         D4    067C          CLRL    TPA_W_LOC1(R1)             ; Clear loc-1 and loc-2
          30 A1         7C    067F          CLRQ    TPA_Q_AUDDS(R1)            ; Clear audit descriptor
          38 A1         7C    0682          CLRQ    TPA_Q_CMNT(R1)             ; Comment descriptor
     40 A1     56       7D    0685          MOVQ    R6,TPA_Q_LINEDS(R1)        ; Save line size and address
     08 A1     56       7D    0689          MOVQ    R6,TPA$L_STRINGCNT(R1)     ; Set TPARSE input descriptor
       00000000'EF      DF    068D          PUSHAL  MER_KEY
       00000000'EF      DF    0693          PUSHAL  MER_STATE
                51      DD    0699          PUSHL   R1
00000000'GF    03       FB    069B          CALLS   #3,G^LIB$TPARSE
             51 50      E9    06A2          BLBC    R0,20$                     ; Error if LBC
51   00000008'EF        DE    06A5          MOVAL   TPARSE_BLOCK,R1            ; Set pointer to Tparse paramter block
             26 A1      B5    06AC          TSTW    TPA_W_LOC2(R1)             ; Were two locators in command?
                0B      13    06AF          BEQL    8$                         ; No if EQL, so don't compare them
   26 A1     24 A1      B1    06B1          CMPW    TPA_W_LOC1(R1),TPA_W_LOC2(R1) ; Is loc-1 <= loc-2?
                04      15    06B6          BLEQ    8$                         ; Yes if LEQ
                50      D4    06B8          CLRL    R0                         ; Set error status
                3A      11    06BA          BRB     20$                        ; and return
                              06BC 8$:
     50    40 A1        7D    06BC          MOVQ    TPA_Q_LINEDS(R1),R6        ; Reset line size and address,
     22 A8     56       B0    06C0          MOVW    R6,RAB$W_RSZ(R8)           ; Reset RAB block record size
   2A A9    28 A1       90    06C4          MOVB    TPA_B_ISFLAGS(R1), -       ; input stream flags byte,
                              06C9                  IS_B_FLAGS(R9)
   09 AA    29 A1       90    06C9          MOVB    TPA_B_EDFLAGS(R1), -       ; edit flags byte,
                              06CE                  UPF_B_EDFLAGS(R10)
          0A AA    2A A1   B0  06CE          MOVW    TPA_W_DOT(R1),UPF_W_DOT(R10) ; dot value,
          04 AA    24 A1   D0  06D3          MOVL    TPA_W_LOC1(R1), -       ; locator 1,
                               06D8                  UPF_W_LOC1(R10)         ; and locator 2
          20 AA    38 A1   7D  06D8          MOVQ    TPA_Q_CMNT(R1), -       ; Comment descriptor
                               06DD                  UPF_Q_CMNT(R10)
          10 2A A9    01   E1  06DD          BBC     #SUM_V_AUDITNEW, -      ; If new audit trail
                               06E2                  IS_B_FLAGS(R9),10$
          18 AA    30 A1   D0  06E2          MOVL    TPA_Q_AUDDS(R1), -      ; Copy size of string
                               06E7                  UPF_Q_AUDDS(R10)
                      3F   BB  06E7          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
    28 AA    34 B1    30 A1   28  06E9          MOVC3   TPA_Q_AUDDS(R1), -      ; Copy audit string
                               06F0                  @TPA_Q_AUDDS+4(R1),UPF_T_AUDST(R10)
                      3F   BA  06F0          POPR    #^M<R0,R1,R2,R3,R4,R5>
                               06F2     10$:
             54    2E A1   3C  06F2          MOVZWL  TPA_W_LINTYP(R1),R4     ; Set line type flags
                               06F6     20$:
                           05  06F6          RSB
                           06F7  ;
                           06F7  ; Tparse action routines
                           06F7  ;
                           06F7  ;
                           06F7  ACT_BLANKS_SIG:
                     0000  06F7          .WORD   0
    00 04 AC    00     E2  06F9          BBSS    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$
                           06FE  10$:
                       04  06FE          RET
                           06FF  ;
                           06FF  ;
                           06FF  ACT_BLANKS_NSIG:
                     0000  06FF          .WORD   0
    00 04 AC    00     E5  0701          BBCC    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$
                           0706  10$:
                       04  0706          RET
                           0707  ;
                           0707  ;
                           0707  ACT_PERCENT:
                     0000  0707          .WORD   0
          28 AC    01  88  0709          BISB    #SUM_M_AUDIT, -           ; Switch on audit trail
                           070D                  TPA_B_ISFLAGS(AP)
                       04  070D          RET
                           070E  ;
                           070E  ;
                           070E  ACT_BACKSLASH:
                     0000  070E          .WORD   0
          28 AC    01  8A  0710          BICB    #SUM_M_AUDIT, -           ; Switch off audit trail
                           0714                  TPA_B_ISFLAGS(AP)
                       04  0714          RET
                           0715  ;
                           0715  ;
                           0715  ACT_ESC:
                     0040  0715          .WORD   ^M<R6>
           51      01  D0  0717          MOVL    #1,R1                     ; Set index
          56    40 AC  9E  071A          MOVAB   TPA_Q_LINEDS(AP),R6       ; Point to buffer descriptor
                   66  B7  071E          DECW    (R6)                      ; Reduce line length by one
                   3F  BB  0720          PUSHR   #^M<R0,R1,R2,R3,R4,R5>    ; Save registers across MOVC3s.
 04 B6   04 B641   66  28  0722          MOVC3   (R6),@4(R6)[R1],@4(R6)    ; Move up line.
   51   00000000'EF    D0  0729          MOVL    SUM_CUR_RAB, R1           ; Get the current RAB,
                   10  E0  0730          BBS     #RAB$V_COC,-              ; check to see if we should
            06 04 A1       0732                  RAB$L_ROP(R1), 10$        ; propagate the shifted string
   24 B1     04 B6  66  28  0735         MOVC3   (R6),@4(R6),@RAB$L_UBF(R1) ; to the UBF.
                   3F  BA  073B  10$:    POPR    #^M<R0,R1,R2,R3,R4,R5>    ; Restore registers.
                       04  073D          RET
                           073E  ;
                           073E  ;
                           073E  ACT_EXIT:
                     0000  073E          .WORD   0
    2E AC       20 AC  B0  0740          MOVW    TPA$L_PARAM(AP), -        ; Set return type
                           0745                  TPA_W_LINTYP(AP)
                       04  0745          RET
                           0746  ;
                           0746  ;
                           0746  ACT_LOC1:
                     0000  0746          .WORD   0
          2E AC    01  B0  0748          MOVW    #CMD_M_CMND,TPA_W_LINTYP(AP)    ; Assume normal command
```

```
24 AC    2C AC   B0  074C               MOVW    TPA_W_LOC(AP),TPA_W_LOC1(AP)
            07   13  0751               BEQL    10$                     ; If EQL is a normal command
   2E AC    03   B0  0753               MOVW    #CMD_M_CMND!CMD_M_EDTRM,TPA_W_LINTYP(AP) ; Set as data terminator co
         2C AC   B4  0757               CLRW    TPA_W_LOC(AP)
                     075A      10$:
                 04  075A               RET
                     075B      ;
                     075B      ;
                     075B      ACT_LOC2:
               0000  075B               .WORD   0
26 AC    2C AC   B0  075D               MOVW    TPA_W_LOC(AP),TPA_W_LOC2(AP)
                 04  0762               RET
                     0763      ;
                     0763      ;
                     0763      ACT_DOT:
               0000  0763               .WORD   0
2C AC    2A AC   B0  0765               MOVW    TPA_W_DOT(AP),TPA_W_LOC(AP)
                 04  076A               RET
                     076B      ;
                     076B      ;
                     076B      ACT_LOCNUM:
               0000  076B               .WORD   0
2C AC    1C AC   B0  076D               MOVW    TPA$L_NUMBER(AP),TPA_W_LOC(AP)
2A AC    2C AC   B0  0772               MOVW    TPA_W_LOC(AP),TPA_W_DOT(AP)
                 04  0777               RET
                     0778      ;
                     0778      ;
                     0778      ACT_PLUS:
               0000  0778               .WORD   0
2C AC    1C AC   A0  077A               ADDW2   TPA$L_NUMBER(AP),TPA_W_LOC(AP)
2A AC    2C AC   B0  077F               MOVW    TPA_W_LOC(AP),TPA_W_DOT(AP)
                 04  0784               RET
                     0785      ;
                     0785      ;
                     0785      ACT_AUDIT:
               0000  0785               .WORD   0
34 AC    0C AC   D0  0787               MOVL    TPA$L_STRINGPTR(AP),TPA_Q_AUDDS+4(AP)
   28 AC    02   88  078C               BISB    #SUM_M_AUDITNEW, -      ; Set new audit trail flag
                     0790                       TPA_B_TSFLAGS(AP)
00 04 AC    00   E2  0790               BBSS    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$ ; Make blanks significant
                     0795      10$:
                 04  0795               RET
                     0796      ;
                     0796      ;
                     0796      ACT_AUDCH:
               0000  0796               .WORD   0
   10    30 AC   D1  0798               CMPL    TPA_Q_AUDDS(AP),#16     ; Is audit trail at maximum size?
            03   18  079C               BGEQ    10$                     ; Yes if GEQ
         30 AC   D6  079E               INCL    TPA_Q_AUDDS(AP)         ; Increment audit trail size
                     07A1      10$:
                 04  07A1               RET
                     07A2      ;
                     07A2      ;
                     07A2      ACT_AUDEND:
               0000  07A2               .WORD   0
00 04 AC    00   E5  07A4               BBCC    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$ ; Switch off blank processing
                     07A9      10$:
                          04  07A9  1435          RET
                              07AA  1436 ;
                              07AA  1437 ;
                              07AA  1438 ACT_CMNT:
                        0000  07AA  1439          .WORD   0
     38 AC     08 AC      7D  07AC  1440          MOVQ    TPA$L_STRINGCNT(AP),TPA_Q_CMNT(AP)
                          04  07B1  1441          RET
                              07B2  1442 ;
                              07B2  1443 ;
                              07B2  1444 ACT_SUPPRESS:
                        0000  07B2  1445          .WORD   0
        29 AC      01     88  07B4  1446          BISB    #ED_M_SUPPRESS, -          ; Set clash messages suppressed flag
                              07B8  1447                  TPA_B_EDFLAGS(AP)
                          04  07B8  1448          RET
                          07B9             .SBTTL  SUM$CLOSE
                          07B9      ;
                          07B9      ;
                          07B9      ; This procedure is called from the main program prior to closing
                          07B9      ; the input file.  It ensures that the main program source file
                          07B9      ; is connected to the RAB.
                          07B9      ;
                          07B9      ;
                          07B9      ; Inputs:
                          07B9      ;
                          07B9      ;       4(AP) = Address of SUM control block
                          07B9      ;
                          07B9      ; Outputs:
                          07B9      ;
                          07B9      ;       None
                          07B9      ;
                          07B9      ;
                    0300  07B9             .ENTRY  SUM$CLOSE,^M<R8,R9>
                          07BB      ;
   51    04 AC     D0     07BB             MOVL    4(AP),R1                 ; Get control block address
   59    04 A1     D0     07BF             MOVL    SUM_L_ISDATA(R1),R9      ; and set data block pointer
            2C     13     07C3             BEQL    20$
            69     D5     07C5             TSTL    IS_L_FILELIST(R9)        ; Is there an update list?
            28     13     07C7             BEQL    20$                      ; No if EQL, file already accessed
   51    1C A9     D0     07C9             MOVL    IS_L_MAIN_FAB(R9),R1     ; Get main program FAB address
         02 A1     B5     07CD             TSTW    FAB$W_IFI(R1)            ; Is source file open?
            07     13     07D0             BEQL    10$                      ; No if EQL
   58    20 A9     D0     07D2             MOVL    IS_L_MAIN_RAB(R9),R8     ; Set RAB pointer
          FCD2     30     07D6             BSBW    ACCESS_SRC               ; Access source file
                          07D9      10$:
         08 A9     D5     07D9             TSTL    IS_L_OPEN_FILE(R9)       ; Is an update file open?
            13     13     07DC             BEQL    20$                      ; No if EQL
                          07DE             $CLOSE  FAB = IS_T_FAB(R9), -
                          07DE                     ERR = SUM$CLOSE_ERR      ; Close update file
         08 A9     D4     07EE             CLRL    IS_L_OPEN_FILE(R9)       ; and clear marker
                          07F1      20$:
                    04    07F1             RET
                          07F2      ;
                          07F2      ;
                          07F2             .END
```

```
$$$CNT                          = 00000003               ED_W_LINES                        0000000C
$$$FLG                          = FFFFFFFF               ED_W_LOC1                         00000008
$$$KEY                          = FFFFFFFF               ED_W_LOC2                         0000000A
$$$KFG                          = FFFFFFFF               ED_W_RFA                          0000000E
$$$MOD                          = 00000000               FAB$B_BID                       = 00000000
$$.TMP1                         = 00000002               FAB$B_BLN                       = 00000001
$$.TMP2                         = 000000A9               FAB$C_BID                       = 00000003
$$KEYTAB                        = 00000000 R     05      FAB$C_BLN                       = 00000050
..AFLG                          = 00000000               FAB$K_BLN                       = 00000050
..FLG                           = 00000002               FAB$L_FOP                       = 00000004
..MOD                           = 00000000               FAB$L_NAM                       = 00000028
..TYP                           = 0000001F               FAB$V_NAM                       = 00000018
.LEN                            = 00000001               FAB$W_IFI                       = 00000002
ACCESS_SRC                        000004AB R     07      GET_IS_BLK                        000000BA R     07
ACCESS_UPDATE                     00000540 R     07      INSERT_NODE                       000001F5 R     07
ACT_AUDCH                         00000796 R     07      IS_B_FLAGS                        0000002A
ACT_AUDEND                        000007A2 R     07      IS_B_PROCFLAGS                    00000005
ACT_AUDIT                         00000785 R     07      IS_B_STATE                        00000004
ACT_BACKSLASH                     0000070E R     07      IS_K_BLN                          00000082
ACT_BLANKS_NSIG                   000006FF R     07      IS_L_CONN_FILE                    0000000C
ACT_BLANKS_SIG                    000006F7 R     07      IS_L_EDIT_BLK                     00000010
ACT_CMNT                          000007AA R     07      IS_L_FILELIST                     00000000
ACT_DOT                           00000763 R     07      IS_L_FIRST_EDIT                   00000014
ACT_ESC                           00000715 R     07      IS_L_LAST_EDIT                    00000018
ACT_EXIT                          0000073E R     07      IS_L_MAIN_FAB                     0000001C
ACT_LOC1                          00000746 R     07      IS_L_MAIN_RAB                     00000020
ACT_LOC2                          0000075B R     07      IS_L_OPEN_FILE                    00000008
ACT_LOCNUM                        0000076B R     07      IS_T_FAB                          00000032
ACT_PERCENT                       00000707 R     07      IS_W_DELETES                      00000030                00
ACT_PLUS                          00000778 R     07      IS_W_HIGH_LOC2                    0000002C
ACT_SUPPRESS                      000007B2 R     07      IS_W_INSERT_NO                    0000002E
AUDCH                             000000A9 R     04      IS_W_LINE_NO                      00000006
BIT...                          = 00000005               IS_W_MAIN_RFA                     00000024
CHECK_ERR                         000003B0 R     07      LESSTHAN                        = 0000003C
CMD_M_ALL                       = 00000007               LIB$FREE_VM                       ********   X     07
CMD_M_CMND                      = 00000001               LIB$GET_VM                        ********   X     07
CMD_M_EDEND                     = 00000004               LIB$TPARSE                        ********   X     07
CMD_M_EDTRM                     = 00000002               LINE_BLK                          0000040B R       07
CMD_V_CMND                      = 00000000               LINE_EOF                          000004A1 R       07
CMD_V_EDEND                     = 00000002               LINE_GET                          0000042E R       07
CMD_V_EDTRM                     = 00000001               LINE_NUP                          00000327 R       07
CMND                              0000003F R     04      LINE_SET                          0000030C R       07
CMNT                              000000C3 R     04      LINE_SRC                          0000032D R       07
COMMA                           = 0000002C               LINE_UPD                          0000033E R       07
COMMAND_CHECK                     00000659 R     07      LINE_UPE                          000003C4 R       07
DATA                              00000032 R     04      LINE_UPR                          000003EC R       07
EDIT                              00000059 R     04      LOCATOR                           000000C7 R       04
ED_B_FILENO                       00000019               MER_KEY                           00000000 RG      05
ED_B_FLAGS                        00000018               MER_STATE                         00000000 RG      04
ED_K_BLN                          0000001A               NAM$B_RSL                       = 00000003
ED_L_BWD                          00000004               NAM$K_BLN                       = 00000060
ED_L_FILE                         00000014               NAM$L_RSA                       = 00000004
ED_L_FWD                          00000000               PRC_M_DELINE                    = 00000002
ED_M_SEQERR                     = 00000002               PRC_M_ERRORS                    = 00000004
ED_M_SUPPRESS                   = 00000001               PRC_M_EXPED                     = 00000001
ED_V_SEQERR                     = 00000001               PRC_M_HIEDIT                    = 00000008
ED_V_SUPPRESS                   = 00000000               PRC_M_NODATA                    = 00000010
```

```
PRC_V_DELINE                    = 00000001                    SUM_Q_AUDDS                      00000008
PRC_V_ERRORS                    = 00000002                    SUM_Q_FILESP                     00000010
PRC_V_EXPED                     = 00000000                    SUM_RETURN                       000002AF R       07
PRC_V_HIEDIT                    = 00000003                    SUM_ST_BLK                     = 00000006
PRC_V_NODATA                    = 00000004                    SUM_ST_EOF                     = 00000008
PROCESS_FILE                      000000EF R       07         SUM_ST_GET                     = 00000007
RAB$B_RAC                       = 0000001E                    SUM_ST_NUP                     = 00000001
RAB$C_RFA                       = 00000002                    SUM_ST_SET                     = 00000000
RAB$C_SEQ                       = 00000000                    SUM_ST_SRC                     = 00000002
RAB$L_FAB                       = 0000003C                    SUM_ST_UPD                     = 00000003
RAB$L_RBF                       = 00000028                    SUM_ST_UPE                     = 00000004
RAB$L_ROP                       = 00000004                    SUM_ST_UPR                     = 00000005
RAB$L_UBF                       = 00000024                    SUM_TPARSE                       0000002C R       02
RAB$M_LOC                       = 00010000                    SUM_UBF_ADDR                     00000004 R       02
RAB$V_LOC                       = 00000010                    SUM_V_AUDIT                    = 00000000
RAB$W_ISI                       = 00000002                    SUM_V_AUDITNEW                 = 00000001
RAB$W_RFA                       = 00000010                    SUM_V_DELETE                   = 00000004
RAB$W_RSZ                       = 00000022                    SUM_V_SRCUPD                   = 00000002
RAB$W_USZ                       = 00000020                    SUM_V_SUBCLSH                  = 00000003
READ_SRC_LINE                     000005EB R       07         SUM_W_INSERT_NO                  0000001A
READ_SRC_LINEA                    000005F4 R       07         SUM_W_LINE_NO                    00000018
READ_UPD_LINE                     00000230 R       07         SYS$CLOSE                        ********  GX    07
READ_UPD_LINEA                    00000236 R       07         SYS$CONNECT                      ********  GX    07
RESTORE_SRC_RFA                   000004F4 R       07         SYS$DISCONNECT                   ********  GX    07
RMS$_EOF                        = 0001827A                    SYS$FIND                         ********  GX    07
SAVE_SRC_RFA                      000004E9 R       07         SYS$GET                          ********  GX    07
SEMICOLON                       = 0000003B                    SYS$OPEN                         ********  GX    07
SET_UP_NODES                      0000015A R       07         SYS$REWIND                       ********  GX    07
SIZ...                          = 00000001                    TERM                             0000004C R       04
SKIP_SRC_LINES                    00000628 R       07         TPA$K_COUNT0                   = 00000008
SUM$CLOSE                         00000789 RG      07         TPA$K_LENGTH0                  = 00000024
SUM$CLOSE_ERR                     ********   X     07         TPA$L_NUMBER                   = 0000001C
SUM$INIT                          00000006 R       07         TPA$L_OPTIONS                  = 00000004
SUM$INIT_CMND                     00000004 RG      07         TPA$L_PARAM                    = 00000020
SUM$INIT_EDIT                     00000000 RG      07         TPA$L_STRINGCNT                = 00000008
SUM$LIB_ERR                       ********   X     07         TPA$L_STRINGPTR                = 0000000C
SUM$LINE                          00000286 RG      07         TPA$V_BLANKS                   = 00000000
SUM$OPEN_ERR                      ********   X     07         TPA$_ALPHA                     = 000001EE
SUM$READ_ERR                      ********   X     07         TPA$_ANY                       = 000001ED
SUM$VIRT_ADDR                     ********   X     07         TPA$_BLANK                     = 000001F2
SUM$_EDITSCLSH                  = 00848800                    TPA$_DECIMAL                   = 000001F3
SUM$_EDOUTSEQ                   = 00848818                    TPA$_DIGIT                     = 000001EF
SUM$_PRMEOF                     = 00848810                    TPA$_EOS                       = 000001F7
SUM$_SLPSYNERR                  = 00848808                    TPA$_EXIT                      = FFFFFFFF
SUM_B_FLAGS                       0000001C                    TPA$_FAIL                      = FFFFFFFE
SUM_COR_RAB                       00000000 R       02         TPA$_FILESPEC                  = 000001EA
SUM_DISPATCH                      00000298 R       07         TPA$_HEX                       = 000001F5
SUM_EDSZE                         00000004 R       03         TPA$_IDENT                     = 000001EC
SUM_ISSZE                         00000000 R       03         TPA$_KEYWORD                   = 00000100
SUM_K_BLN                         0000001D                    TPA$_LAMBDA                    = 000001F6
SUM_L_ISDATA                      00000004                    TPA$_MAXKEY                    = 000000DC
SUM_L_STS                         00000000                    TPA$_OCTAL                     = 000001F4
SUM_M_AUDIT                     = 00000001                    TPA$_STRING                    = 000001F0
SUM_M_AUDITNEW                  = 00000002                    TPA$_SUBXPR                    = 000001F8
SUM_M_DELETE                    = 00000010                    TPA$_SYMBOL                    = 000001F1
SUM_M_SRCUPD                    = 00000004                    TPA$_UIC                       = 000001EB
SUM_M_SUBCLSH                   = 00000008                    TPARSE_BLOCK                     00000008 R       02
```

```
TPA_B_EDFLAGS                    = 00000029
TPA_B_ISFLAGS                    = 00000028
TPA_Q_AUDDS                      = 00000030
TPA_Q_CMNT                       = 00000038
TPA_Q_LINEDS                     = 00000040
TPA_W_DOT                        = 0000002A
TPA_W_LINTYP                     = 0000002E
TPA_W_LOC                        = 0000002C
TPA_W_LOC1                       = 00000024
TPA_W_LOC2                       = 00000026
UPF_B_EDFLAGS                      00000009
UPF_B_FIFLAGS                      00000008
UPF_B_FILENO                       0000000C
UPF_K_BLN                          00000098
UPF_L_PTR                          00000000
UPF_Q_AUDDS                        00000018
UPF_Q_CMNT                         00000020
UPF_Q_EDITS                        00000010
UPF_T_AUDST                        00000028
UPF_T_NAM                          00000038
UPF_V_INIT                       = 00000000
UPF_W_DOT                          0000000A
UPF_W_LOC1                         00000004
UPF_W_LOC2                         00000006

                                                  +----------------+
                                                  ! Psect synopsis !
                                                  +----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                             00000098  (   152.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
SUM$RW_DATA                       00000050  (    80.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC LONG
SUM$RO_DATA                       00000008  (     8.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC LONG
_LIB$STATE$                       000000E7  (   231.)  04 (  4.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC BYTE
_LIB$KEY0$                        00000000  (     0.)  05 (  5.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC WORD
_LIB$KEY1$                        00000000  (     0.)  06 (  6.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC WORD
SUM$CODE                          000007F2  (  2034.)  07 (  7.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC LONG

                                               +------------------------+
                                               ! Performance indicators !
                                               +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                      38     00:00:00.08    00:00:00.60
Command processing                 147     00:00:00.51    00:00:01.72
Pass 1                             476     00:00:23.82    00:00:49.76
Symbol table sort                    0     00:00:01.18    00:00:01.99
Pass 2                             254     00:00:05.73    00:00:11.81
Symbol table output                 30     00:00:00.23    00:00:00.39
Psect synopsis output                3     00:00:00.04    00:00:00.06
Cross-reference output               0     00:00:00.00    00:00:00.00
Assembler run totals               950     00:00:31.60    00:01:06.56

The working set limit was 1950 pages.
```

```
SUM$
VAX-

Sym
Pse
Cro
Ass

The
332
The
320
14 p

Mac
---
_$2
-$2
TOT

688

The

MAC
```

```
121870 bytes (239 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 921 non-local and 83 local symbols.
1489 source lines were read in Pass 1, producing 43 object records in Pass 2.
65 pages of virtual memory were used to define 52 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[SUM.OBJ]SUM.MLB;1                        7
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    29
TOTALS (all libraries)                               36

1413 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SUMEDIT/OBJ=OBJ$:SUMEDIT MSRC$:SUMEDIT/UPDATE=(ENH$:SUMEDIT)+LIB$:SUM/LIB
```

0368 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SUMSLP
MAP

SRTTRNMSG
LIS

DATA
LIS

SUMSHR
MAP

SUMDATA
LIS

UTLCHKCLI
LIS

SUM

SUM
MAP

SUMDEF
MAP

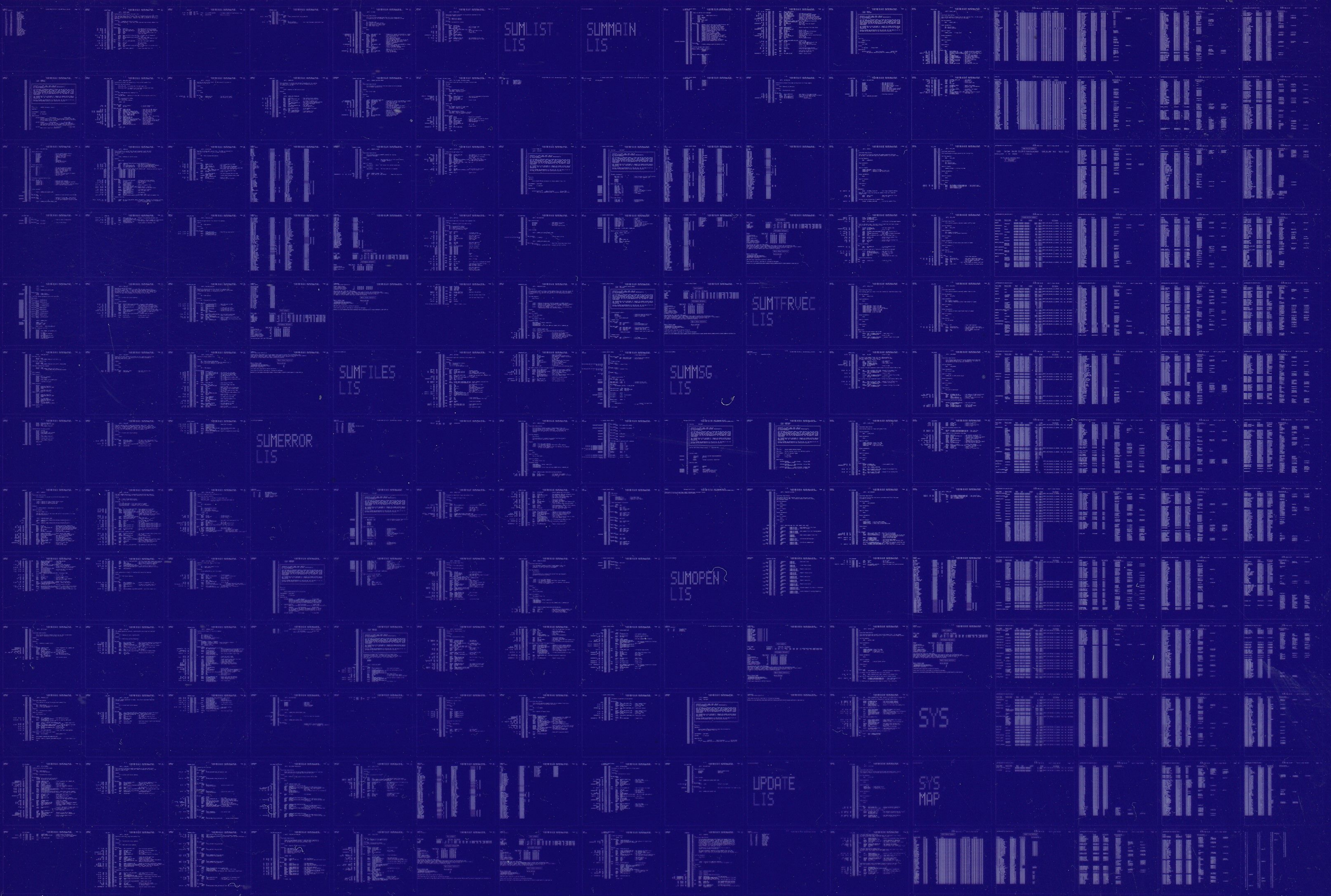
MAIN
LIS

SUMED
LIS

SUMEDIT
LIS

GETCMD
LIS

0369 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY



SUMLIST
LIS

SUMMAIN
LIS

SUMTFRVEC
LIS

SUMFILES
LIS

SUMMSG
LIS

SUMERROR
LIS

SUMOPEN
LIS

SYS

UPDATE
LIS

SYS
MAP