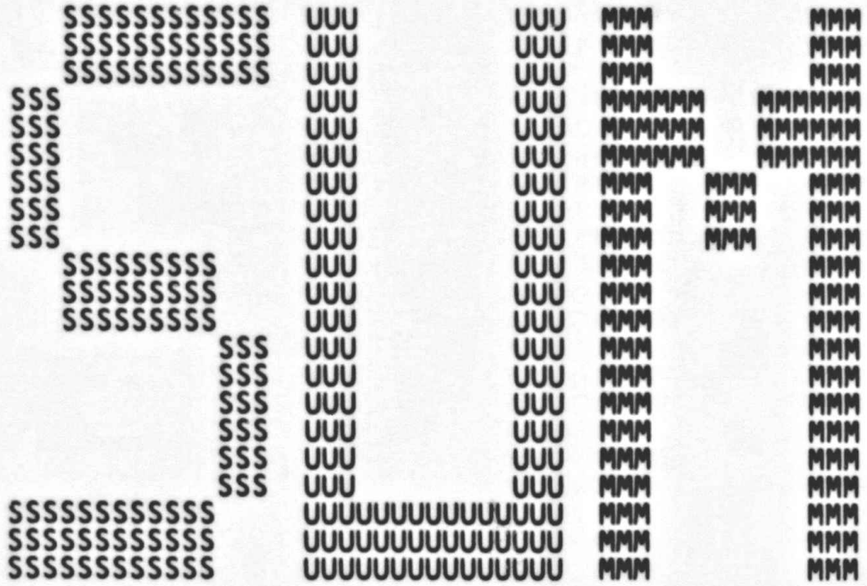
SSSSSSSSSSS UUU UUU MMM MMM
SSSSSSSSSSS UUU UUU MMM MMM
SSSSSSSSSSS UUU UUU MMM MMM
SSS UUU UUU MMMMMM MMMMMM
SSS UUU UUU MMMMMM MMMMMM
SSS UUU UUU MMMMMM MMMMMM
SSS UUU UUU MMM MMM MMM
SSS UUU UUU MMM MMM MMM
SSS UUU UUU MMM MMM
SSSSSSSSS UUU UUU MMM MMM
SSSSSSSSS UUU UUU MMM MMM
SSSSSSSSS UUU UUU MMM 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
SSSSSSSSSSS UUUUUUUUUUUUUUU MMM MMM
SSSSSSSSSSS UUUUUUUUUUUUUUU MMM MMM
SSSSSSSSSSS UUUUUUUUUUUUUUU MMM MMM

FILEID**SUMEDIT

```
  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
```

```
0000     1          .TITLE  SUM$EDIT
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:     SUMSHR shareable library
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;
0000    34  ;
0000    35  ; ENVIRONMENT:  USER MODE
0000    36  ;
0000    37  ; AUTHOR:       R. Newland
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-002 MTR0002         Mike Rhodes             18-May-1983
0000    42  ;               Correct handling of file access switching in READ_UPD_LINE
0000    43  ;               when an error occurs.  Also, make the RAB globally available
0000    44  ;               to the TPARSE action routines.
0000    45  ;
0000    46  ;       V03-001 MTR0001         Mike Rhodes             19-Jan-1983
0000    47  ;               Create and a local UBF for use in SUM$INIT and SUM$LINE.
0000    48  ;               The local UBF precludes ACCVIOs resulting from the caller's
0000    49  ;               RAB ROP=LOC, when processing SUMSHR's escape character "<".
0000    50  ;
0000    51  ;       V02-001                 B. Schreiber            21-Mar-1980
0000    52  ;               Make totally position independent.
0000    53  ;
0000    54  ;--
```

```
                     0000     56          .SBTTL  DECLARATIONS
                     0000     57 ;
                     0000     58 ;
                     0000     59 ; Macro definitions
                     0000     60 ;
                     0000     61          DEFUPFBLK                       ; Source update merge offsets
                     0000     62          DEFEDBLK                        ; Edit block offsets
                     0000     63          DEFISBLK                        ; Input stream block offsets
                     0000     64          DEFCMDTYPE                      ; Command line type
                     0000     65          DEFSUMCBL                       ; SUM control block
                     0000     66          $FABDEF                         ; FAB
                     0000     67          $RABDEF                         ; RAB
                     0000     68          $NAMDEF                         ; NAM block
                     0000     69          $TPADEF                         ; TPARSE definitions
                     0000     70          $RMSDEF                         ; RMS definitions
                     0000     71 ;
                     0000     72 ;
                     0000     73 ; state definitions
                     0000     74 ;
                     0000     75 $EQULST SUM_ST_,,0,,< -
                     0000     76          SET , -                         ; Set up for source or update
                     0000     77          NUP , -                         ; No more updates to process
                     0000     78          SRC , -                         ; Next line from source file
                     0000     79          UPD , -                         ; Next line from update file
                     0000     80          UPE , -                         ; Report update errors
                     0000     81          UPR , -                         ; Update ready
                     0000     82          BLK , -                         ; Process next edit block of update
                     0000     83          GET , -                         ; Get next update line
                     0000     84          EOF >                           ; End of file
                     0000     85 ;
                     0000     86 ;
                     0000     87 ; Procedure flag byte definitions
                     0000     88 ;
                     0000     89 $VIELD  PRC,0,< -
                     0000     90          <EXPED,,M> -                    ; Expected edit command
                     0000     91          <DELINE,,M> -                   ; Deleted lines information pending
                     0000     92          <ERRORS,,M> -                   ; Clash errors to report
                     0000     93          <HIEDIT,,M> -                   ; Highest precedence edit overides others
                     0000     94          <NODATA,,M> -                   ; Data from edit being ignored
                     0000     95          >
                     0000     96 ;
                     0000     97 ;
                     0000     98 ;
                     0000     99 ; Local storage
                     0000    100 ;
                     0000    101 ;
                00000000     102          .PSECT  SUM$RW_DATA,NOEXE,LONG
                     0000    103 ;
                     0000    104 ;
                     0000    105 SUM_CUR_RAB:                            ; Address of the currently active RAB.
00000000             0000    106          .LONG   0
                     0004    107
                     0004    108 SUM_UBF_ADDR:                           ; Address of local UBF.  The size of
00000000             0004    109          .LONG   0                       ; the UBF is established by the size
                     0008    110                                          ; of the main program's (caller's) RAB.
                     0008    111
                00000000     112          .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   121          .SBTTL  TPARSE
                       0008   122 ;
                       0008   123          .SAVE
                       0008   124
                   00000008   125          .PSECT  SUM$RW_DATA,NOEXE,LONG
                       0008   126 ;
                       0008   127 ;
                       0008   128 TPARSE_BLOCK:
                   00000008   0008   129          .LONG   TPA$K_COUNTO
                   0000002C   000C   130          .BLKB   TPA$K_LENGTHO-4
                       002C   131 ;
                       002C   132 ; Continue Tparse parameter block with own data
                       002C   133 ;
                       002C   134 SUM_TPARSE:
                       002C   135 ;
                   00000024   002C   136 TPA_W_LOC1 = .-TPARSE_BLOCK
                   0000002E   002C   137          .BLKW   1
                   00000026   002E   138 TPA_W_LOC2 = .-TPARSE_BLOCK
                   00000030   002E   139          .BLKW   1
                   00000028   0030   140 TPA_B_ISFLAGS = .-TPARSE_BLOCK
                   00000031   0030   141          .BLKB   1
                   00000029   0031   142 TPA_B_EDFLAGS = .-TPARSE_BLOCK
                   00000032   0031   143          .BLKB   1
                   0000002A   0032   144 TPA_W_DOT = .-TPARSE_BLOCK
                   00000034   0032   145          .BLKW   1
                   0000002C   0034   146 TPA_W_LOC = .-TPARSE_BLOCK
                   00000036   0034   147          .BLKW   1
                   0000002E   0036   148 TPA_W_LINTYP = .-TPARSE_BLOCK
                   00000038   0036   149          .BLKW   1
                   00000030   0038   150 TPA_Q_AUDDS = .-TPARSE_BLOCK
                   00000040   0038   151          .BLKQ   1
                   00000038   0040   152 TPA_Q_CMNT = .-TPARSE_BLOCK
                   00000048   0040   153          .BLKQ   1
                   00000040   0048   154 TPA_Q_LINEDS = .-TPARSE_BLOCK
                   00000050   0048   155          .BLKQ   1
                       0050   156 ;
                       0050   157 ;
                   00000008   158          .PSECT  SUM$RO_DATA
                       0008   159 ;
                   0000002C   0008   160          COMMA = ^X2C
                   0000003B   0008   161          SEMICOLON = ^X3B
                   0000003C   0008   162          LESSTHAN = ^X3C
                       0008   163 ;
                       0008   164          $INIT_STATE     MER_STATE,MER_KEY
                       0008   165 ;
                       0008   166 ; Get 1st character of line
                       0008   167 ;
                       0008   168          $STATE
                       0008   169          $TRAN   TPA$_LAMBDA,,ACT_BLANKS_SIG
                       0008   170          $STATE
                       0008   171          $TRAN   '-',EDIT
                       0008   172          $TRAN   '%',CMND,ACT_PERCENT
                       0008   173          $TRAN   '/',TERM
                       0008   174          $TRAN   LESSTHAN,DATA,ACT_ESC
                       0008   175          $TRAN   '@',TPA$_FAIL
                       0008   176          $TRAN   '\',CMND,ACT_BACKSLASH
                       0008   177          $TRAN   TPA$_EOS,DATA
```

```
0008    178          $TRAN   TPA$_ANY,DATA
0008    179 ;
0008    180 ; End data line
0008    181 ;
0008    182          $STATE  DATA
0008    183          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,,0
0008    184 ;
0008    185 ; End normal command line
0008    186 ;
0008    187          $STATE  CMND
0008    188          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,,CMD_M_CMND
0008    189 ;
0008    190 ; End data terminating command
0008    191 ;
0008    192          $STATE  TERM
0008    193          $TRAN   TPA$_LAMBDA,TPA$_EXIT,ACT_EXIT,,, -
0008    194                          <CMD_M_CMND!CMD_M_EDTRM!CMD_M_EDEND>
0008    195 ;
0008    196 ;
0008    197 ; Edit command
0008    198 ;
0008    199 ; Read locator-1
0008    200 ;
0008    201          $STATE  EDIT
0008    202          $TRAN   '-',,ACT_SUPPRESS
0008    203          $TRAN   TPA$_LAMBDA
0008    204          $STATE
0008    205          $TRAN   TPA$_LAMBDA,,ACT_BLANKS_NSIG
0008    206          $STATE
0008    207          $TRAN   !LOCATOR,,ACT_LOC1
0008    208 ;
0008    209 ; Read Locator-2
0008    210 ;
0008    211          $STATE
0008    212          $TRAN   TPA$_EOS,TPA$_EXIT
0008    213          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008    214          $TRAN   COMMA
0008    215          $STATE
0008    216          $TRAN   !LOCATOR,,ACT_LOC2
0008    217          $TRAN   TPA$_EOS,TPA$_EXIT
0008    218 ; Read audit string
0008    219 ;
0008    220          $STATE
0008    221          $TRAN   TPA$_EOS,TPA$_EXIT
0008    222          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008    223          $TRAN   COMMA
0008    224          $STATE
0008    225          $TRAN   '/',,ACT_AUDIT
0008    226          $TRAN   TPA$_EOS,TPA$_EXIT
0008    227          $TRAN   SEMICOLON,CMNT,ACT_CMNT
0008    228          $STATE  AUDCH
0008    229          $TRAN   '/',,ACT_AUDEND
0008    230          $TRAN   TPA$_ANY,AUDCH,ACT_AUDCH
0008    231 ;
0008    232 ; Read comment line
0008    233 ;
0008    234          $STATE
```

```
                     0008    235          $TRAN   TPA$_EOS,TPA$_EXIT
                     0008    236          $TRAN   SEMICOLON,CMNT,ACT_CMNT
                     0008    237          $STATE  CMNT
                     0008    238          $TRAN   TPA$_LAMBDA,TPA$_EXIT
                     0008    239 ;
                     0008    240 ;
                     0008    241 ; Subexpression to parse locator
                     0008    242 ;
                     0008    243          $STATE  LOCATOR
                     0008    244          $TRAN   '.',,ACT_DOT
                     0008    245          $TRAN   TPA$_DECIMAL,,ACT_LOCNUM
                     0008    246          $TRAN   TPA$_LAMBDA,TPA$_EXIT
                     0008    247          $STATE
                     0008    248          $TRAN   '+'
                     0008    249          $TRAN   TPA$_LAMBDA,TPA$_EXIT
                     0008    250          $STATE
                     0008    251          $TRAN   TPA$_DECIMAL,,ACT_PLUS
                     0008    252          $STATE
                     0008    253          $TRAN   TPA$_LAMBDA,TPA$_EXIT
                     0008    254 ;
                     0008    255          $END_STATE
                     0008    256 ;
                     0008    257 ;
                 00000008    258          .RESTORE
```

```
                     0008   260          .SBTTL  SUM$INIT
                     0008   261 ;
                     0008   262 ;++
                     0008   263 ; Functional description:
                     0008   264 ;
                     0008   265 ;       This procedure is called to initialise the update files.
                     0008   266 ;
                     0008   267 ;
                     0008   268 ; Input parameters:
                     0008   269 ;
                     0008   270 ;        4(AP) = Address of input stream control block
                     0008   271 ;        8(AP) = Address of update files list
                     0008   272 ;       12(AP) = Address of main program RAB
                     0008   273 ;
                     0008   274 ;
                     0008   275 ; Outputs:
                     0008   276 ;
                     0008   277 ;       IS_L_MAIN_FAB(R9) = FAB address of source file
                     0008   278 ;
                     0008   279 ; Implicit outputs:
                     0008   280 ;
                     0008   281 ;       The edit nodes list.
                     0008   282 ;
                     0008   283 ;       SUM_UBF_ADDR points to the local UBF which is allocated (if it has
                     0008   284 ;       not been previously).
                     0008   285 ;
                     0008   286 ;--
                     0008   287 ;
                00000000    288          .PSECT  SUM$CODE,NOWRT,LONG
                     0000   289 ;
                OFFC 0000   290          .ENTRY  SUM$INIT_EDIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
              02  11 0002   291          BRB     SUM$INIT
                     0004   292 ;
                    OFFC 0004   293          .ENTRY  SUM$INIT_CMND,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                         0006   294 ;
                         0006   295 SUM$INIT:
           50    01    D0 0006   296          MOVL    #1,R0                   ; Assume successful completion
        58    04 AC    D0 0009   297          MOVL    4(AP),R8                ; Get address of SUM control block
        59    04 A8    D0 000D   298          MOVL    SUM_L_ISDATA(R8),R9     ; Get input stream data block address
              0A       12 0011   299          BNEQ    5$                      ; Branch if block has been allocated
            00A4       30 0013   300          BSBW    GET_IS_BLK              ; Get and initialise data block
          6D  50       E9 0016   301          BLBC    R0,7$                   ; Error of LBC
        04 A8    59    D0 0019   302          MOVL    R9,SUM_L_ISDATA(R8)     ; Save data block address
                         001D   303 5$:
           18 A8    B4 001D   304          CLRW    SUM_W_LINE_NO(R8)       ; Reset return line number
        04 A9    00    90 0020   305          MOVB    #SUM_ST_SET,IS_B_STATE(R9)      ; Initialise state to SET
        06 A9    01    B0 0024   306          MOVW    #1,IS_W_LINE_NO(R9)     ; and source file line number
        5A    08 AC    D0 0028   307          MOVL    8(AP),R10               ; Get file list address
        58    0C AC    D0 002C   308          MOVL    12(AP),R8               ; Get RAB address
     00000004'EF    D5 0030   309          TSTL    SUM_UBF_ADDR            ; Has a UBF been allocated?
              19    12 0036   310          BNEQ    6$                      ; If NEQ a UBF already exists.
           7E    20 A8 3C 0038   311          MOVZWL  RAB$W_USZ(R8),-(SP)     ; Set up the buffer size.
     00000004'EF    9F 003C   312          PUSHAB  SUM_UBF_ADDR            ; Stack arguments for LIB$GET_VM
              04 AE    DF 0042   313          PUSHAL  4(SP)                   ;
00000000'GF    02    FB 0045   314          CALLS   #2, G^LIB$GET_VM        ; Allocate a local UBF.
              37 50    E9 004C   315          BLBC    R0, 7$                  ; Error if LBC
                 8E    D5 004F   316          TSTL    (SP)+                   ; Clean up the stack.
```

```
                20 A9    58   D0   0051   317 6$:     MOVL    R8,IS_L_MAIN_RAB(R9)     ; Save RAB address
                   10 A8    D4   0055   318          CLRL    RAB$W_RFA+0(R8)          ; Clear RFA
                   14 A8    B4   0058   319          CLRW    RAB$W_RFA+4(R8)          ; (3 words)
                    048B    30   005B   320          BSBW    SAVE_SRC_RFA             ; and save it
            1C A9    3C A8    D0   005E   321          MOVL    RAB$L_FAB(R8),IS_L_MAIN_FAB(R9) ; Save FAB address
                69    5A    D0   0063   322          MOVL    R10,IS_L_FILELIST(R9)    ; Save file list address
                      51    13   0066   323          BEQL    40$                      ; If EQL there is no list so return
            40 08 AA    00    E2   0068   324          BBSS    #UPF_V_INIT, -           ; Branch if already initialised
                                0069   325                  UPF_B_FIFLAGS(R10),30$
            10 AA    10 AA    DE   006D   326          MOVAL   UPF_Q_EDITS(R10), -      ; Initialise edit list head in
                                0072   327                  UPF_Q_EDITS(R10)         ; first file block
            14 AA    10 AA    DE   0072   328          MOVAL   UPF_Q_EDITS(R10), -
                                0077   329                  UPF_Q_EDITS+4(R10)
                                0077   330          $DISCONNECT RAB=R8,ERR=SUM$CLOSE_ERR ; Disconnect RAB
                                0086   331 7$:
                   30 50    E9   0086   332          BLBC    R0,40$                   ; Error if LBC
                                0089   333 10$:
                      64    10   0089   334          BSB     PROCESS_FILE             ; Process update files
                   05 50    E9   008B   335          BLBC    R0,20$                   ; Error if LBC
                5A    6A    D0   008E   336          MOVL    (R10),R10                ; Get next file block address
                      F6    12   0091   337          BNEQ    10$                      ; End of list if EQL
                                0093   338 20$:
                5A    69    D0   0093   339          MOVL    IS_L_FILELIST(R9),R10    ; Reset file list pointer
            3C A8    1C A9    D0   0096   340          MOVL    IS_L_MAIN_FAB(R9),RAB$L_FAB(R8) ; Reset FAB address
                                009B   341          $CONNECT RAB=R8,ERR=SUM$OPEN_ERR
                     0447    30   00AA   342          BSBW    RESTORE_SRC_RFA          ; Restore source file RFA
                                00AD   343 30$:
            10 A9    10 AA    D0   00AD   344          MOVL    UPF_Q_EDITS(R10),IS_L_EDIT_BLK(R9) ; Reset edit block pointer
               2A A9    03    88   00B2   345          BISB2   #SUM_M_AUDIT!SUM_M_AUDITNEW, -  ; Switch on audit trail and
                                00B6   346                  IS_B_FLAGS(R9)           ; mark first audit as new
                   30 A9    B4   00B6   347          CLRW    IS_W_DELETES(R9)         ; Initialise number of deleted lines
                                00B9   348 40$:
                                04   00B9   349          RET
```

```
                                00BA   351          .SBTTL  GET_IS_BLK
                                00BA   352 ;
                                00BA   353 ;++
                                00BA   354 ;
                                00BA   355 ; Functional description:
                                00BA   356 ;
                                00BA   357 ;       This routine obtains a memory block for an input stream data
                                00BA   358 ;       block and if successful initialises the block.
                                00BA   359 ;
                                00BA   360 ; Inputs:
                                00BA   361 ;
                                00BA   362 ;       None
                                00BA   363 ;
                                00BA   364 ; Outputs:
                                00BA   365 ;
                                00BA   366 ;       R9  = Address of memory block
                                00BA   367 ;
                                00BA   368 ;--
                                00BA   369 ;
                                00BA   370 GET_IS_BLK:
            00000000'EF   9F    00BA   371          PUSHAB  SUM$VIRT_ADDR           ; Stack arguments for LIB$GET_VM
            00000000'EF   9F    00C0   372          PUSHAB  SUM_ISSZE               ; ...
         00000000'GF    02    FB   00C6   373          CALLS   #2,G^LIB$GET_VM         ; Get memory block
                1E   50    E9   00CD   374          BLBC    R0,10$                  ; Error if LBC
         59   00000000'EF    D0   00D0   375          MOVL    SUM$VIRT_ADDR,R9        ; Get block address
69   0082 8F   00   69    00    2C   00D7   376          MOVC5   #0,(R9),#0,#IS_K_BLN,(R9) ; Clear block
              51   32 A9    9E   00DF   377          MOVAB   IS_T_FAB(R9),R1         ; Set FAB block pointer
                                00E3   378          $FAB_STORE  FAB = R1, -         ; and initialise as a FAB
                                00E3   379                      BID = #FAB$C_BID, -
                                00E3   380                      BLN = #FAB$C_BLN
                50   01    D0   00EB   381          MOVL    #1,R0                   ; Set success status
                                00EE   382 10$:
                             05   00EE   383          RSB
```

```
                     00EF    385          .SBTTL  PROCESS_FILE
                     00EF    386 ;
                     00EF    387 ;++
                     00EF    388 ; Functional description:
                     00EF    389 ;
                     00EF    390 ;       This routine is called to process each update file
                     00EF    391 ;
                     00EF    392 ; Inputs:
                     00EF    393 ;       R8  = RAB address
                     00EF    394 ;       R9  = Input stream data block address
                     00EF    395 ;       R10 = File node address
                     00EF    396 ;
                     00EF    397 ; Outputs:
                     00EF    398 ;
                     00EF    399 ;       R0  = Success/error status
                     00EF    400 ;
                     00EF    401 ; Implicit outputs:
                     00EF    402 ;
                     00EF    403 ;       Edit blocks list
                     00EF    404 ;
                     00EF    405 ;--
                     00EF    406 PROCESS_FILE:
5A A9   38 AA   9E   00EF    407          MOVAB   UPF_T_NAM(R10), -       ; Set NAM block pointer
                     00F4    408                  IS_T_FAB+FAB$L_NAM(R9)
00 36 A9   18   E2   00F4    409          BBSS    #FAB$V_NAM, -           ; Set for open by NAM block
                     00F9    410                  IS_T_FAB+FAB$L_FOP(R9),5$
                     00F9    411 5$:
                     00F9    412          $OPEN   FAB=IS_T_FAB(R9),ERR=SUM$OPEN_ERR       ; Open input file
      4D 50    E9   0109    413          BLBC    R0,30$                  ; Error if LBC
3C A8 32 A9    DE   010C    414          MOVAL   IS_T_FAB(R9),RAB$L_FAB(R8)      ; Put FAB address into RAB
                     0111    415          $CONNECT RAB=R8,ERR=SUM$OPEN_ERR ; Connect RAB to FAB
      26 50    E9   0120    416          BLBC    R0,20$                  ; Error if LBC
                     0123    417          $FIND   RAB=R8,ERR=SUM$READ_ERR ; Initialise RFA
      05 50    E9   0132    418          BLBC    R0,10$                  ; Error if LBC
                     0135    419 ;
      10 A9    D4   0135    420          CLRL    IS_L_EDIT_BLK(R9)       ; Clear last edit node address
                     0138    421 ;
                     0138    422 ; Read update file and create edit nodes
                     0138    423 ;
         20    10   0138    424          BSBB    SET_UP_NODES            ; Read update file
                     013A    425 ;
                     013A    426 10$:
                     013A    427          $DISCONNECT RAB=R8,ERR=SUM$CLOSE_ERR
                     0149    428 20$:
                     0149    429          $CLOSE  FAB=IS_T_FAB(R9),ERR=SUM$CLOSE_ERR      ; Close input file
                     0159    430 30$:
               05    0159    431          RSB
```

```
                     015A   433          .SBTTL  SET_UP_NODES
                     015A   434 ;
                     015A   435 ; Subroutine to form all edit_nodes
                     015A   436 ;
                     015A   437 ; Inputs:
                     015A   438 ;       R8  = RAB address
                     015A   439 ;       R10 = file node address
                     015A   440 ;
                     015A   441 ; Outputs:
                     015A   442 ;       R0  = Success/error status
                     015A   443 ;
                     015A   444 ;
                     015A   445 SET_UP_NODES:
                     015A   446          ASSUME  UPF_W_LOC2 EQ <UPF_W_LOC1+2>
                     015A   447          ASSUME  ED_W_LOC2 EQ <ED_W_LOC1+2>
                     015A   448 10$:
     00000000'EF   9F 015A   449          PUSHAB  SUM$VIRT_ADDR            ; Stack arguments for LIB$GET_VM
     00000004'EF   9F 0160   450          PUSHAB  SUM_EDSZE                ; ...
00000000'GF    02   FB 0166   451          CALLS   #2,G^LIB$GET_VM          ; Get edit block
          7D   50   E9 016D   452          BLBC    R0,70$                   ; Error if LBC
5B   00000000'EF   D0 0170   453          MOVL    SUM$VIRT_ADDR,R11        ; Set block pointer
       14 AB   5A   D0 0177   454          MOVL    R10,ED_L_FILE(R11)       ;  Fill in file block address
    19 AB   0C AA   90 017B   455          MOVB    UPF_B_FILENO(R10), -     ; and file number
                    0180   456                  ED_B_FILENO(R11)
    0E AB   10 A8   D0 0180   457          MOVL    RAB$W_RFA+0(R8),ED_W_RFA+0(R11) ; Record file address (3 words)
    12 AB   14 A8   B0 0185   458          MOVW    RAB$W_RFA+4(R8),ED_W_RFA+4(R11)
            0C AB   B4 018A   459          CLRW    ED_W_LINES(R11)
    08 AB   04 AA   D0 018D   460          MOVL    UPF_W_LOC1(R10),ED_W_LOC1(R11) ; Move both locator numbers
    18 AB   09 AA   90 0192   461          MOVB    UPF_B_EDFLAGS(R10),ED_B_FLAGS(R11) ; and flags to edit node
                    0197   462 30$:
          009C   30 0197   463          BSBW    READ_UPD_LINEA           ; Read line from input file
        0E 50   E8 019A   464          BLBS    R0,40$                   ; OK if LBS
0001827A 8F    50   D1 019D   465          CMPL    R0,#RMS$_EOF             ; Is error end-of-file?
          4E   12 01A4   466          BNEQ    80$                      ; No if NEQ
       54    07   D0 01A6   467          MOVL    #CMD_M_ALL,R4            ; Fake an end-of-edit command
          13   11 01A9   468          BRB     50$                      ; Error will be reported on next pass
                    01AB   469 40$:
        04AB   30 01AB   470          BSBW    COMMAND_CHECK            ; Check for command
      E6 50   E9 01AE   471          BLBC    R0,30$                   ; Syntax error if LBC
   09 54    01   E0 01B1   472          BBS     #CMD_V_EDTRM,R4,50$      ; Branch if data terminating command
   DE 54    00   E0 01B5   473          BBS     #CMD_V_CMND,R4,30$       ; Branch if normal command
      0C AB   B6 01B9   474          INCW    ED_W_LINES(R11)          ; Increment number of insert lines for
          D9   11 01BC   475          BRB     30$                      ; this edit
                    01BE   476 50$:
      08 AB   D5 01BE   477          TSTL    ED_W_LOC1(R11)           ; If Loc-1 and Loc-2 = 0 and Lines <> 0
          21   12 01C1   478          BNEQ    60$                      ; there is an insert in front of
                    01C3   479                                           ; the file, otherwise throw this
                    01C3   480                                           ; Edit node away
      0C AB   B5 01C3   481          TSTW    ED_W_LINES(R11)
          1C   12 01C6   482          BNEQ    60$
    18 54    02   E1 01C8   483          BBC     #CMD_V_EDEND,R4,60$      ; Branch if not end of edits
 00000000'EF   9F 01CC   484          PUSHAB  SUM$VIRT_ADDR            ; Stack arguments for LIB$FREE_VM
 00000004'EF   9F 01D2   485          PUSHAB  SUM_EDSZE                ; ...
00000000'GF    02   FB 01D8   486          CALLS   #2,G^LIB$FREE_VM         ; Return unused memory block
          0B   50   E9 01DF   487          BLBC    R0,70$                   ; Error if LBC
          10   11 01E2   488          BRB     80$
                    01E4   489 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   498          .SBTTL  INSERT_NODE
                     01F5   499 ;
                     01F5   500 ; Subroutine to insert block into edit list
                     01F5   501 ;
                     01F5   502 ; This routine checks that the edit node is in sequence with any other nodes
                     01F5   503 ; from the same update file.  If not, the edit node is marked so that a
                     01F5   504 ; warning can be produced later.  However, the node is placed in the correct
                     01F5   505 ; position.
                     01F5   506 ;
                     01F5   507 ; Inputs:
                     01F5   508 ;       R11  = address of block to insert
                     01F5   509 ;       IS_L_EDIT_BLK(R9) = Last edit node inserted from current update file
                     01F5   510 ;
                     01F5   511 ; Outputs:
                     01F5   512 ;       None
                     01F5   513 ;
                     01F5   514 ;
                     01F5   515 INSERT_NODE:
     50   08 AC   D0  01F5   516          MOVL    8(AP),R0                        ; Get address of first file block
     50   10 A0   DE  01F9   517          MOVAL   UPF_Q_EDITS(R0),R0              ; and form edit list head address
     51   10 A9   D0  01FD   518          MOVL    IS_L_EDIT_BLK(R9),R1            ; Get address of last node inserted
          05   12    0201   519          BNEQ    10$                             ; If NEQ there is one
        51   50   D0  0203   520          MOVL    R0,R1                           ; This is first node so scan list
          10   11    0206   521          BRB     20$                             ; from list head
               0208   522 10$:
 08 A1   08 AB   B1  0208   523          CMPW    ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is edit out of sequence?
          09   14    020D   524          BGTR    20$                             ; No if GTR
    18 AB   02   88  020F   525          BISB    #ED_M_SEQERR,ED_B_FLAGS(R11) ; Mark edit node
        51   50   D0  0213   526          MOVL    R0,R1                           ; Scan list from list head to find
          04   11    0216   527          BRB     30$                             ; correct position
               0218   528 20$:
    10 A9   5B   D0  0218   529          MOVL    R11,IS_L_EDIT_BLK(R9)           ; Set new 'last edit' address
               021C   530 30$:
        51   61   D0  021C   531          MOVL    (R1),R1                         ; Get next block
        50   51   D1  021F   532          CMPL    R1,R0                           ; At end of list?
          07   13    0222   533          BEQL    40$                             ; Yes if EQL
 08 A1   08 AB   B1  0224   534          CMPW    ED_W_LOC1(R11),ED_W_LOC1(R1) ; Is new LOC-1 <= current LOC-1
          F1   14    0229   535          BGTR    30$                             ; No if GTR
               022B   536 40$:
    04 B1   6B   0E  022B   537          INSQUE  (R11),@ED_L_BWD(R1)             ; Insert new node into list
               05    022F   538          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   592          .SBTTL  SUM$LINE
                                0286   593 ;
                                0286   594 ; This procedure is called from the main program to get the next
                                0286   595 ; input line.  This line may come from either the source file or
                                0286   596 ; an update file.
                                0286   597 ;
                                0286   598 ; Inputs:
                                0286   599 ;
                                0286   600 ;       4(AP) = Address of control block
                                0286   601 ;
                                0286   602 ; Ouputs:
                                0286   603 ;
                                0286   604 ;       Next line
                                0286   605 ;
                                0286   606 ;
                           0FFC 0286   607          .ENTRY  SUM$LINE,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                 51  04 AC    D0 0288   608          MOVL    4(AP),R1                ; Get address of control block
                 59  04 A1    D0 028C   609          MOVL    SUM_L_ISDATA(R1),R9     ; Set input stream data block address
                 58  20 A9    D0 0290   610          MOVL    IS_L_MAIN_RAB(R9),R8    ; Get main program RAB address
                 5B  10 A9    D0 0294   611          MOVL    IS_L_EDIT_BLK(R9),R11   ; Get current edit block address
                                0298   612 SUM_DISPATCH:
           08    00  04 A9    8F 0298   613          CASEB   IS_B_STATE(R9),#SUM_ST_SET,#SUM_ST_EOF ; Branch to service routine
                           0063' 029D   614 10$:     .SIGNED_WORD    LINE_SET-10$
                           008A' 029F   615          .SIGNED_WORD    LINE_NUP-10$
                           0090' 02A1   616          .SIGNED_WORD    LINE_SRC-10$
                           00A1' 02A3   617          .SIGNED_WORD    LINE_UPD-10$
                           0127' 02A5   618          .SIGNED_WORD    LINE_UPE-10$
                           014F' 02A7   619          .SIGNED_WORD    LINE_UPR-10$
                           016E' 02A9   620          .SIGNED_WORD    LINE_BLK-10$
                           0191' 02AB   621          .SIGNED_WORD    LINE_GET-10$
                           0204' 02AD   622          .SIGNED_WORD    LINE_EOF-10$
                                02AF   623 ;
                                02AF   624 SUM_RETURN:
              10 A9    5B    D0 02AF   625          MOVL    R11,IS_L_EDIT_BLK(R9)   ; Preserve edit block address
              51  04 AC    D0 02B3   626          MOVL    4(AP),R1                ; Get address of control block
                 61    50   D0 02B7   627          MOVL    R0,SUM_L_STS(R1)        ; Return status
           1C A1    2A A9    90 02BA   628          MOVB    IS_B_FLAGS(R9), -       ; Edit flags
                                02BF   629                  SUM_B_FLAGS(R1)
           20 1C A1    02    E1 02BF   630          BBC     #SUM_V_SRCUPD, -        ; Branch if source line
                                02C4   631                  SUM_B_FLAGS(R1),5$
           1A A1    2E A9    B0 02C4   632          MOVW    IS_W_INSERT_NO(R9),SUM_W_INSERT_NO(R1) ; Inserts
           08 A1    18 AA    7D 02C9   633          MOVQ    UPF_Q_AUDDS(R10), -     ; Supply audit string descriptor
                                02CE   634                  SUM_Q_AUDDS(R1)
              5A    38 AA    DE 02CE   635          MOVAL   UPF_T_NAM(R10),R10      ; Form NAM block address
           10 A1    03 AA    9A 02D2   636          MOVZBL  NAM$B_RSL(R10), -       ; Get file spec size
                                02D7   637                  SUM_Q_FILESP+0(R1)
           14 A1    04 AA    D0 02D7   638          MOVL    NAM$L_RSA(R10), -       ; and address
                                02DC   639                  SUM_Q_FILESP+4(R1)
                    20 50    E8 02DC   640          BLBS    R0,10$                  ; If error line
           1B 1C A1    02    E4 02DF   641          BBSC    #SUM_V_SRCUPD,SUM_B_FLAGS(R1),10$ ; don't mark as update line
                                02E4   642 ;
                                02E4   643 ; Source file line
                                02E4   644 ;
                                02E4   645 5$:
     18 A1    06 A9    01    A3 02E4   646          SUBW3   #1,IS_W_LINE_NO(R9),SUM_W_LINE_NO(R1) ; Number of line being returne
                       12 50    E9 02EA   647          BLBC    R0,10$                  ; If error save deleted line information
                                02ED   648                                          ; 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   658          .SBTTL  LINE_SET
                     0300   659 ;
                     0300   660 ; Routine to service SET state
                     0300   661 ; Determines if the next line is to come from the main source file
                     0300   662 ; or from an update file.  If there are no more updates to be processed
                     0300   663 ; the state is set to NUP; if there are updates but the next update is to
                     0300   664 ; be applied to a later source line the state is set to SRC; if the next
                     0300   665 ; line is to come from an update file the state is set to UPD.
                     0300   666 ;
                     0300   667 ;
                     0300   668 ; Inputs:
                     0300   669 ;
                     0300   670 ;       R11 = Current edit block address
                     0300   671 ;
                     0300   672 ; Outputs:
                     0300   673 ;
                     0300   674 ;       state changed
                     0300   675 ;
                     0300   676 LINE_SET:
    04 A9    01   90 0300   677          MOVB    #SUM_ST_NUP,IS_B_STATE(R9) ; Assume no more updates
       51    69   D0 0304   678          MOVL    IS_L_FILELIST(R9),R1       ; Get address of first file block
             18   13 0307   679          BEQL    10$                        ; If EQL there are no update files
 51      10 A1   DE 0309   680          MOVAL   UPF_Q_EDITS(R1),R1         ; Form edit block list head address
       5B    51   D1 030D   681          CMPL    R1,R11                     ; Any edits still in list?
             0F   13 0310   682          BEQL    10$                        ; No if EQL so must be source line
    04 A9    03   90 0312   683          MOVB    #SUM_ST_UPD,IS_B_STATE(R9) ; Assume next line is from update file
08 AB    06 A9   B1 0316   684          CMPW    IS_W_LINE_NO(R9), -        ; Is line number of source file less
                    031B   685                  ED_W_LOC1(R11)             ; than locator-1 of next edit?
             07   18 031B   686          BGEQ    20$                        ; No if GEQ
    04 A9    02   90 031D   687          MOVB    #SUM_ST_SRC,IS_B_STATE(R9) ; Change state to source
                    0321   688 10$:
          0187   30 0321   689          BSBW    ACCESS_SRC                 ; Access source file
                    0324   690 20$:
          FF71   31 0324   691          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   713          .SBTTL  LINE_SRC
                     032D   714 ;
                     032D   715 ; The next source line is read from the main input file.  The line
                     032D   716 ; number is incremented and compared with the locator-1 value of the
                     032D   717 ; next edit.  If the line number remains lower the state remains at SRC.
                     032D   718 ; If the line number is equal or greater the state is changed to UPD.
                     032D   719 ; The next call to SUM$LINE will then get an update line.
                     032D   720 ;
                     032D   721 ; Inputs:
                     032D   722 ;
                     032D   723 ;       R11 = Current edit block address
                     032D   724 ;
                     032D   725 ; Outputs:
                     032D   726 ;
                     032D   727 ;       state
                     032D   728 ;
                     032D   729 ;
                     032D   730 LINE_SRC:
          02C4   30  032D   731          BSBW    READ_SRC_LINEA          ; Get next line from source file
08 AB   06 A9   B1  0330   732          CMPW    IS_W_LINE_NO(R9), -     ; Is source line number still lower
                    0335   733                  ED_W_LOC1(R11)          ; than next locator-1
             04   19  0335   734          BLSS    10$                     ; Yes if LSS
04 A9        03   90  0337   735          MOVB    #SUM_ST_UPD,IS_B_STATE(R9) ; Reset state to UPD
                    033B   736 10$:
         FF71   31  033B   737          BRW     SUM_RETURN              ; and return with line
```

```
                     033E   739          .SBTTL  LINE_UPD
                     033E   740 ;
                     033E   741 ; The next update operation is prepared by determining the range of
                     033E   742 ; the edit, that is the number of edit operations which have clashed.
                     033E   743 ;
                     033E   744 ; Inputs:
                     033E   745 ;
                     033E   746 ;       R9  = Input stream data pointer
                     033E   747 ;       R11 = Current edit block address
                     033E   748 ;
                     033E   749 ; Outputs:
                     033E   750 ;
                     033E   751 ;       IS_L_FIRST_EDIT(R9) = First edit block of update
                     033E   752 ;       IS_L_LAST_EDIT(R9)  = Last edit block of update
                     033E   753 ;       IS_W_HIGH_LOC2(R9)  = Highest loc-2 value of update
                     033E   754 ;
                     033E   755 ;
                     033E   756 LINE_UPD:
14 A9    5B    D0    033E   757          MOVL    R11,IS_L_FIRST_EDIT(R9) ; Save address of first edit
54    0A AB    3C    0342   758          MOVZWL  ED_W_LOC2(R11),R4       ; Set highest loc-2 value
2C A9    54    B0    0346   759          MOVW    R4,IS_W_HIGH_LOC2(R9)   ; and supply as routine output
2A A9    08    8A    034A   760          BICB2   #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; May be first edit in clash
                     8A    034E   761          BICB2   #<PRC_M_ERRORS! -       ; Assume no clash errors,
                     034F   762                    PRC_M_HIEDIT! -       ; highest edit does not overide others,
                     034F   763                    PRC_M_NODATA>,-       ; and all data lines inserted
05 A9    1C          034F   764                  IS_B_PROCFLAGS(R9)
04 A9    05    90    0352   765          MOVB    #SUM_ST_UPR,IS_B_STATE(R9)
   55    69    D0    0356   766          MOVL    IS_L_FILELIST(R9),R5    ; Set files list
55    10 A5    DE    0359   767          MOVAL   UPF_Q_EDITS(R5),R5      ; list head address
                     035D   768 10$:
   52    6B    D0    035D   769          MOVL    (R11),R2                ; Point to next edit block
   55    52    D1    0360   770          CMPL    R2,R5                   ; At end of list?
         2F    13    0363   771          BEQL    40$                     ; Yes if EQL
   51    54    D0    0365   772          MOVL    R4,R1                   ; Set highest locator value of edit
         0A    12    0368   773          BNEQ    20$                     ; If zero set from loc-2 of current edit
51    0A AB    3C    036A   774          MOVZWL  ED_W_LOC2(R11),R1       ; Set highest locator value of edit
         04    12    036E   775          BNEQ    20$                     ; If zero set from loc-1 of current edit
51    08 AB    3C    0370   776          MOVZWL  ED_W_LOC1(R11),R1       ; Set highest locator value of edit
                     0374   777 20$:
08 A2    51    B1    0374   778          CMPW    R1,ED_W_LOC1(R2)        ; Does this edit overlap with next?
         1A    19    0378   779          BLSS    40$                     ; No if LSS
                     037A   780 ;
                     037A   781 ; This edit block clashes with next
                     037A   782 ;
0A A2    54    B1    037A   783          CMPW    R4,ED_W_LOC2(R2)        ; Is its loc-2 higher than current loc-2
         04    18    037E   784          BGEQ    25$                     ; No if GEQ
54    0A A2    3C    0380   785          MOVZWL  ED_W_LOC2(R2),R4        ; Extend range of edit
                     0384   786 25$:
      0A A2    B5    0384   787          TSTW    ED_W_LOC2(R2)           ; Is edit all inserts?
         04    13    0387   788          BEQL    30$                     ; Yes if EQL
05 A9    08    88    0389   789          BISB    #PRC_M_HIEDIT,IS_B_PROCFLAGS(R9) ; Highest edit overides others
                     038D   790 30$:                                     ; therefore replace later)
         21    10    038D   791          BSBB    CHECK_ERR               ; See if error should be reported
   5B    52    D0    038F   792          MOVL    R2,R11                  ; Point to next edit block
         C9    11    0392   793          BRB     10$
                     0394   794 40$:
18 A9    5B    D0    0394   795          MOVL    R11,IS_L_LAST_EDIT(R9)  ; Set address of last edit block
```

```
                14 A9   5B   D1   0398   796          CMPL    R11,IS_L_FIRST_EDIT(R9) ; If first block then single non-clashing
                        0F   13   039C   797          BEQL    50$                     ; edit else last block of clashing edits
                        10   10   039E   798          BSBB    CHECK_ERR               ; See if error should be reported
          08 05 A9      02   E1   03A0   799          BBC     #PRC_V_ERRORS, -        ; Branch if no errors to report
                                03A5   800                  IS_B_PROCFLAGS(R9),50$
                04 A9   04   90   03A5   801          MOVB    #SUM_ST_UPE,IS_B_STATE(R9) ; Set state to report errors
             5B    14 A9   D0   03A9   802          MOVL    IS_L_FIRST_EDIT(R9),R11 ; Reset edit block pointer to first
                                03AD   803 50$:
                     FEE8   31   03AD   804          BRW     SUM_DISPATCH
                                03B0   805 ;
                                03B0   806 ;
                                03B0   807 ;
                                03B0   808 ; Local subroutine to check if clashing edit should be reported
                                03B0   809 ;
                                03B0   810 ; Inputs:
                                03B0   811 ;
                                03B0   812 ;       R11 = Edit block address
                                03B0   813 ;
                                03B0   814 ; Outputs:
                                03B0   815 ;
                                03B0   816 ;       None
                                03B0   817 ;
                                03B0   818 CHECK_ERR:
          0E 18 AB      00   E0   03B0   819          BBS     #ED_V_SUPPRESS, -       ; Branch if suppress bit set
                                03B5   820                  ED_B_FLAGS(R11),20$
                     08 AB   D5   03B5   821          TSTL    ED_W_LOC1(R11)          ; If Loc-1, Loc-2 and lines = 0
                        05   12   03B8   822          BNEQ    10$                     ; then do not report as error
                     0C AB   B5   03BA   823          TSTW    ED_W_LINES(R11)
                        04   13   03BD   824          BEQL    20$
                                03BF   825 10$:
                05 A9   04   88   03BF   826          BISB    #PRC_M_ERRORS,IS_B_PROCFLAGS(R9) ; Set error report bit
                                03C3   827 20$:
                        05   03C3   828          RSB
```

```
                     03C4    830          .SBTTL  LINE_UPE
                     03C4    831 ;
                     03C4    832 ; The update operation contains clashing edits which must be reported
                     03C4    833 ;
                     03C4    834 ; Inputs:
                     03C4    835 ;
                     03C4    836 ;       R11 = Address of next clashing edit
                     03C4    837 ;
                     03C4    838 ; Outputs:
                     03C4    839 ;
                     03C4    840 ;       R11 = Edit block pointer advanced
                     03C4    841 ;
                     03C4    842 ;
                     03C4    843 LINE_UPE:
      5A   14 AB     D0      03C4    844          MOVL    ED_L_FILE(R11),R10      ; Get file block address of clashing edit
          FE65       30      03C8    845          BSBW    READ_UPD_LINE           ; Read update file to get edit line
50    00848800 8F    D0      03CB    846          MOVL    #SUM$_EDITSCLSH,R0      ; Set return status
      14 A9    5B    D1      03D2    847          CMPL    R11,IS_L_FIRST_EDIT(R9) ; First report of this set of clashes
               04    13      03D6    848          BEQL    10$                     ; Yes if EQL
      2A A9    08    88      03D8    849          BISB    #SUM_M_SUBCLSH,IS_B_FLAGS(R9) ; Set 2nd or later flag
                     03DC    850 10$:
      18 A9    5B    D1      03DC    851          CMPL    R11,IS_L_LAST_EDIT(R9)  ; At last edit?
               04    12      03E0    852          BNEQ    20$                     ; No if NEQ
      04 A9    05    90      03E2    853          MOVB    #SUM_ST_UPR,IS_B_STATE(R9) ; Set state to Update Ready
                     03E6    854 20$:
         5B    6B    D0      03E6    855          MOVL    (R11),R11               ; Advance to next edit block
             FEC3    31      03E9    856          BRW     SUM_RETURN
```

```
                     03EC   858          .SBTTL  LINE_UPR
                     03EC   859 ;
                     03EC   860 ; The next update operation is ready.  Any errors have been reported
                     03EC   861 ; to the caller.
                     03EC   862 ;
                     03EC   863 ;
                     03EC   864 ; Inputs:
                     03EC   865 ;
                     03EC   866 ;       R11 = Current edit block address
                     03EC   867 ;
                     03EC   868 ; Outputs:
                     03EC   869 ;
                     03EC   870 ;       None
                     03EC   871 ;
                     03EC   872 ;
                     03EC   873 LINE_UPR:
5B    14 A9    D0    03EC   874          MOVL    IS_L_FIRST_EDIT(R9),R11 ; Reset pointer to first edit block
04 A9    06    90    03F0   875          MOVB    #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
54    2C A9    3C    03F4   876          MOVZWL  IS_W_HIGH_LOC2(R9),R4   ; Is edit operation an insert?
         0B    12    03F8   877          BNEQ    50$                     ; No if NEQ
      08 AB    B5    03FA   878          TSTW    ED_W_LOC1(R11)          ; Is insert to front of file?
         09    13    03FD   879          BEQL    60$                     ; Yes if EQL
       01E9    30    03FF   880          BSBW    READ_SRC_LINE           ; Read one more line from source
       FEAA    31    0402   881          BRW     SUM_RETURN
                     0405   882 50$:
       0220    30    0405   883          BSBW    SKIP_SRC_LINES          ; Skip over source lines to be deleted
                     0408   884 60$:
       FE8D    31    0408   885          BRW     SUM_DISPATCH            ; and dispatch
```

LINE_BLK

```
                        040B   887          .SBTTL  LINE_BLK
                        040B   888 ;
                        040B   889 ; This routine is called to begin processing of the next edit block
                        040B   890 ; The file from which edit lines will come is prepared for access.  The
                        040B   891 ; state is reset to GET.
                        040B   892 ;
                        040B   893 ;
                        040B   894 ; Inputs:
                        040B   895 ;
                        040B   896 ;       R11 = Current edit block address
                        040B   897 ;
                        040B   898 ; Outputs:
                        040B   899 ;
                        040B   900 ;       None
                        040B   901 ;
                        040B   902 ;
                        040B   903 LINE_BLK:
    5A    14 AB    D0   040B   904          MOVL    ED_L_FILE(R11),R10      ; Get file block address of file
          012E     30   040F   905          BSBW    ACCESS_UPDATE           ; Prepare for reading file
        16 50      E9   0412   906          BLBC    R0,20$                  ; Error if LBC
00 05 A9   00      E5   0415   907          BBCC    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),5$ ; Clear expected edit flag
                        041A   908 5$:
        08 AB      B5   041A   909          TSTW    ED_W_LOC1(R11)          ; Is this insert in front of file?
           05      12   041D   910          BNEQ    10$                     ; No if NEQ
00 05 A9   00      E2   041F   911          BBSS    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),10$ ; Set expected edit flag
                        0424   912 10$:
    04 A9   07     90   0424   913          MOVB    #SUM_ST_GET,IS_B_STATE(R9) ; Reset state to GET
          FE6D     31   0428   914          BRW     SUM_DISPATCH            ; and dispatch again
                        042B   915 20$:
          FE81     31   042B   916          BRW     SUM_RETURN              ; Return to caller with error
```

```
                     042E   918          .SBTTL  LINE_GET
                     042E   919 ;
                     042E   920 ; Routine to get next line from update file
                     042E   921 ;
                     042E   922 ;
                     042E   923 ; Inputs:
                     042E   924 ;
                     042E   925 ;       R11 = Current edit block address
                     042E   926 ;
                     042E   927 ; Outputs:
                     042E   928 ;
                     042E   929 ;       R11 = Next edit block address
                     042E   930 ;
                     042E   931 ;
                     042E   932 LINE_GET:
          5A  14 AB  D0   042E   933          MOVL    ED_L_FILE(R11),R10      ; Set file block pointer
                     0432   934 10$:
              FE01   30   0432   935          BSBW    READ_UPD_LINEA          ; Get next line from update file
            12 50    E8   0435   936          BLBS    R0,20$                  ; OK if LBS
0001827A 8F    50    D1   0438   937          CMPL    R0,#RMS$_EOF            ; Is error end-of-file?
                2F   12   043F   938          BNEQ    35$                     ; No if NEQ
50   00848810 8F    D0   0441   939          MOVL    #SUM$_PRMEOF,R0         ; Set premature end-of-file status
                28   11   0448   940          BRB     40$
                     C44A   941 20$:
              020C   30   044A   942          BSBW    COMMAND_CHECK           ; Check for syntax and type
            44 50    E9   044D   943          BLBC    R0,80$                  ; Syntax error if LBC
      07 54    00    E0   0450   944          BBS     #CMD_V_CMND,R4,30$      ; Branch if command line
   D9 05 A9    04    E0   0454   945          BBS     #PRC_V_NODATA, -        ; Ignore data line if higher precedence
                     0459   946                  IS_B_PROCFLAGS(R9),10$  ; edit is overiding others
                40   11   0459   947          BRB     90$                     ; Return to caller with line
                     045B   948 30$:
      D3 54    01    E1   045B   949          BBC     #CMD_V_EDTRM,R4,10$     ; Branch if not edit terminating command
   0E 05 A9    00    E2   045F   950          BBSS    #PRC_V_EXPED,IS_B_PROCFLAGS(R9),40$ ; If expecting edit get next lin
   C9 18 AB    01    E1   0464   951          BBC     #ED_V_SEQERR,ED_B_FLAGS(R11),10$ ; Was edit out of sequence?
50   00848818 8F    D0   0469   952          MOVL    #SUM$_EDOUTSEQ,R0       ; Yes: report error now
                     0470   953 35$:
                2C   11   0470   954          BRB     100$
                     0472   955 ;
                     0472   956 ; Found end of this set of lines
                     0472   957 ;
                     0472   958 40$:
      18 A9    5B   D1   0472   959          CMPL    R11,IS_L_LAST_EDIT(R9)  ; Last edit block in range?
                0F   13   0476   960          BEQL    60$                     ; Yes if EQL
   04 05 A9    03   E1   0478   961          BBC     #PRC_V_HIEDIT, -        ; Branch if concatenating inserts
                     047D   962                  IS_B_PROCFLAGS(R9),50$  ;
      05 A9    10   88   047D   963          BISB    #PRC_M_NODATA,IS_B_PROCFLAGS(R9) ; Ignore data from other edits
                     0481   964 50$:
      04 A9    06   90   0481   965          MOVB    #SUM_ST_BLK,IS_B_STATE(R9) ; Reset state to BLK
                04   11   0485   966          BRB     70$
                     0487   967 60$:
      04 A9    00   90   0487   968          MOVB    #SUM_ST_SET,IS_B_STATE(R9) ; Reset state to SET
                     048B   969 70$:
         5B    6B   D0   048B   970          MOVL    (R11),R11               ; Point to next edit block
            0D 50    E9   048E   971          BLBC    R0,100$                 ; If error return to caller first
              FE04   31   0491   972          BRW     SUM_DISPATCH            ; or dispatch again
                     0494   973 80$:
50   00848808 8F    D0   0494   974          MOVL    #SUM$_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  1000          .SBTTL  ACCESS_SRC
                     04AB  1001 ;
                     04AB  1002 ; Routine to access main source file.  The RAB is connected to
                     04AB  1003 ; the main file FAB if it is not already connected.
                     04AB  1004 ;
                     04AB  1005 ; Inputs:
                     04AB  1006 ;
                     04AB  1007 ;       R8  = Main program RAB address
                     04AB  1008 ;
                     04AB  1009 ;
                     04AB  1010 ; Outputs:
                     04AB  1011 ;
                     04AB  1012 ;       None
                     04AB  1013 ;
                     04AB  1014 ;
                     04AB  1015 ACCESS_SRC:
          02 A8  B5  04AB  1016          TSTW    RAB$W_ISI(R8)           ; Is it connected to a FAB?
             1F  13  04AE  1017          BEQL    10$                     ; No if EQL
       51    32 A9  DE  04B0  1018          MOVAL   IS_T_FAB(R9),R1         ; Set input stream FAB address
       51    3C A8  D1  04B4  1019          CMPL    RAB$L_FAB(R8),R1        ; Is it connected to SUM FAB?
             2E  12  04B8  1020          BNEQ    20$                     ; No if NEQ, it's connected to main FAB
                     04BA  1021          $DISCONNECT RAB = R8, -         ; Disconnect RAB from SUM FAB
                     04BA  1022                      ERR = SUM$CLOSE_ERR
          0C A9  D4  04C9  1023          CLRL    IS_L_CONN_FILE(R9)      ; Clear file connected flag
          19 50  E9  04CC  1024          BLBC    R0,20$                  ; Error if LBC
                     04CF  1025 10$:
    3C A8 1C A9  D0  04CF  1026          MOVL    IS_L_MAIN_FAB(R9), -    ; Put main program FAB into RAB
                     04D4  1027                  RAB$L_FAB(R8)
                     04D4  1028          $CONNECT RAB = R8, -            ; Connect main program FAB to RAB
                     04D4  1029                  ERR = SUM$OPEN_ERR
          02 50  E9  04E3  1030          BLBC    R0,20$                  ; Error if LBC
             0C  10  04E6  1031          BSB     RESTORE_SRC_RFA         ; Restore source file RFA
                     04E8  1032 20$:
                 05  04E8  1033          RSB
```

```
                     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  1056          .SBTTL  RESTORE_SRC_RFA
                     04F4  1057 ;
                     04F4  1058 ;
                     04F4  1059 ; Routine to restore source file record file address and
                     04F4  1060 ; reset record pointers.  If RFA is zero a rewind is performed,
                     04F4  1061 ; if non-zero the record is located by a find.
                     04F4  1062 ;
                     04F4  1063 ;
                     04F4  1064 ; Inputs:
                     04F4  1065 ;
                     04F4  1066 ;       R8  = RAB address
                     04F4  1067 ;
                     04F4  1068 ;
                     04F4  1069 ; Outputs:
                     04F4  1070 ;
                     04F4  1071 ;       R0  = Success/error status
                     04F4  1072 ;
                     04F4  1073 ;
                     04F4  1074 RESTORE_SRC_RFA:
10 A8   24 A9   D0   04F4  1075          MOVL    IS_W_MAIN_RFA+0(R9), -  ; Move RFA back to RAB
                     04F9  1076                  RAB$W_RFA+0(R8)         ; (3 words)
14 A8   28 A9   B0   04F9  1077          MOVW    IS_W_MAIN_RFA+4(R9), -
                     04FE  1078                  RAB$W_RFA+4(R8)
        16      12   04FE  1079          BNEQ    10$                     ; If NEQ then do find
     10 A8      D5   0500  1080          TSTL    RAB$W_RFA+0(R8)         ; Test other part of RFA
        11      12   0503  1081          BNEQ    10$                     ; If NEQ then do find
                     0505  1082          $REWIND RAB = R8, -             ; Rewind to start of file
                     0505  1083                  ERR = SUM$READ_ERR
        29      11   0514  1084          BRB     20$
                     0516  1085 10$:
   1E A8    02   90   0516  1086          MOVB    #RAB$C_RFA,RAB$B_RAC(R8); Put into RFA access mode
                     051A  1087          $FIND   RAB = R8, -             ; Reset record pointers
                     051A  1088                  ERR = SUM$READ_ERR
   1E A8    00   90   0529  1089          MOVB    #RAB$C_SEQ,RAB$B_RAC(R8); Reset to sequential access mode
      0F 50      E9   052D  1090          BLBC    R0,20$                  ; Error if LBC
                     0530  1091          $GET    RAB = R8, -             ; Advance past this record which has
                     0530  1092                  ERR = SUM$READ_ERR      ; read before.
                     053F  1093 20$:
                05   053F  1094          RSB
```

```
                     0540  1096          .SBTTL  ACCESS_UPDATE
                     0540  1097 ;
                     0540  1098 ; Routine to access update file
                     0540  1099 ;
                     0540  1100 ;
                     0540  1101 ; Inputs:
                     0540  1102 ;
                     0540  1103 ;       R8  = Main program RAB address
                     0540  1104 ;       R10 = File block address of required update file
                     0540  1105 ;       R11 = Edit block address of next edit
                     0540  1106 ;
                     0540  1107 ;
                     0540  1108 ; Ouputs:
                     0540  1109 ;
                     0540  1110 ;       R9  = FAB address
                     0540  1111 ;
                     0540  1112 ;
                     0540  1113 ACCESS_UPDATE:
          1C   BB    0540  1114          PUSHR   #^M<R2,R3,R4>
52   08 A9   9E    0542  1115          MOVAB   IS_L_OPEN_FILE(R9),R2   ; Set pointer to file open
53   0C A9   9E    0546  1116          MOVAB   IS_L_CONN_FILE(R9),R3   ; and file connected markers
54   32 A9   DE    054A  1117          MOVAL   IS_T_FAB(R9),R4         ; Set pointer to SUM's FAB
     02 A8   B5    054E  1118          TSTW    RAB$W_ISI(R8)           ; Is RAB connected to a FAB?
        5B   13    0551  1119          BEQL    30$                     ; No if EQL
54   3C A8   D1    0553  1120          CMPL    RAB$L_FAB(R8),R4        ; Is it connected to SUM's FAB?
        07   12    0557  1121          BNEQ    10$                     ; No if NEQ
   63   5A   D1    0559  1122          CMPL    R10,(R3)                ; Is it connected to required file?
        69   13    055C  1123          BEQL    40$                     ; Yes if EQL
        03   11    055E  1124          BRB     20$
                     0560  1125 10$:
          FF86   30  0560  1126          BSBW    SAVE_SRC_RFA            ; Save source file RFA
                     0563  1127 20$:
                     0563  1128          $DISCONNECT RAB = R8, -         ; Disconnect RAB from FAB
                     0563  1129                  ERR = SUM$CLOSE_ERR
        73 50   E9   0572  1130          BLBC    R0,50$                  ; Error if LBC
           63   D4   0575  1131          CLRL    (R3)                    ; Mark that no file is connected
     62    5A   D1   0577  1132          CMPL    R10,(R2)                ; Is required file already open?
           32   13   057A  1133          BEQL    30$                     ; Yes if EQL
           62   D5   057C  1134          TSTL    (R2)                    ; Is any file open on this FAB?
           14   13   057E  1135          BEQL    25$                     ; No if EQL
                     0580  1136          $CLOSE  FAB = R4, -             ; Close currently open update file
                     0580  1137                  ERR = SUM$CLOSE_ERR
        56 50   E9   058F  1138          BLBC    R0,50$                  ; Error if LBC
           62   D4   0592  1139          CLRL    (R2)                    ; Mark that no file is open
                     0594  1140 25$:
28 A4      38 AA   DE   0594  1141          MOVAL   UPF_T_NAM(R10), -       ; Put NAM block into FAB
                     0599  1142                  FAB$L_NAM(R4)
                     0599  1143          $OPEN   FAB = R4, -             ; Open required update file
                     0599  1144                  ERR = SUM$OPEN_ERR
        3D 50   E9   05A8  1145          BLBC    R0,50$                  ; Error if LBC
     62    5A   D0   05AB  1146          MOVL    R10,(R2)                ; Mark which file is open
                     05AE  1147 30$:
   3C A8    54   D0   05AE  1148          MOVL    R4,RAB$L_FAB(R8)        ; Put FAB address in RAB
                     05B2  1149          $CONNECT RAB = R8, -            ; Connect RAB to FAB
                     05B2  1150                  ERR = SUM$OPEN_ERR
        24 50   E9   05C1  1151          BLBC    R0,50$                  ; Error if LBC
     63    5A   D0   05C4  1152          MOVL    R10,(R3)                ; Mark which file is connected
```

```
                     05C7  1153 40$:
10 A8   0E AB   D0   05C7  1154          MOVL    ED_W_RFA+0(R11), -      ; Reset RFA (3 words)
                     05CC  1155                  RAB$W_RFA+0(R8)
14 A8   12 AB   B0   05CC  1156          MOVW    ED_W_RFA+4(R11), -
                     05D1  1157                  RAB$W_RFA+4(R8)
   1E A8    02   90   05D1  1158          MOVB    #RAB$C_RFA,RAB$B_RAC(R8); Put into RFA access mode
                     05D5  1159          $FIND   RAB = R8, -             ; Position file
                     05D5  1160                  ERR = SUM$READ_ERR
   1E A8    00   90   05E4  1161          MOVB    #RAB$C_SEQ,RAB$B_RAC(R8); Reset to sequential access mode
                     05E8  1162 50$:
            1C   BA   05E8  1163          POPR    #^M<R2,R3,R4>
                 05   05EA  1164          RSB
```

```
                        05EB  1166          .SBTTL  READ_SRC_LINE
                        05EB  1167 ;
                        05EB  1168 ; Routine to read one line from source file
                        05EB  1169 ;
                        05EB  1170 ; There are two entry points:
                        05EB  1171 ;
                        05EB  1172 ;       READ_SRC_LINE   to access file and read line
                        05EB  1173 ;
                        05EB  1174 ;       READ_SRC_LINEA  if file is already accessed and ready to
                        05EB  1175 ;                       read next line
                        05EB  1176 ;
                        05EB  1177 ; Inputs:
                        05EB  1178 ;
                        05EB  1179 ;       R8  = RAB address
                        05EB  1180 ;
                        05EB  1181 ; Outputs:
                        05EB  1182 ;
                        05EB  1183 ;       R0  = Success/error status
                        05EB  1184 ;       R6  = Line size
                        05EB  1185 ;       R7  = Line buffer address
                        05EB  1186 ;       IS_W_LINE_NO(R9) = line number
                        05EB  1187 ;
                        05EB  1188          .ENABL  LSB
                        05EB  1189 ;
                        05EB  1190 READ_SRC_LINE:
    50    01    D0      05EB  1191          MOVL    #1,R0                       ; Assume success
      FEBA    30      05EE  1192          BSBW    ACCESS_SRC                  ; Access source file
    33 50     E9      05F1  1193          BLBC    R0,20$                      ; Error if LBC
                        05F4  1194 ;
                        05F4  1195 READ_SRC_LINEA:
                        05F4  1196          $GET    RAB = R8, -                 ; Get next line from source file
                        05F4  1197                  ERR = SUM$READ_ERR
        0F  50    E8    0603  1198          BLBS    R0,10$                      ; OK if LBS
0001827A 8F     50    D1    0606  1199          CMPL    R0,#RMS$_EOF                ; Was error end-of-file?
              18    12    060D  1200          BNEQ    20$                         ; No if NEQ
      04 A9   08    90    060F  1201          MOVB    #SUM_ST_EOF,IS_B_STATE(R9) ; Set into EOF state
              12    11    0613  1202          BRB     20$
                        0615  1203 10$:
    56    22 A8   3C    0615  1204          MOVZWL  RAB$W_RSZ(R8),R6            ; Set record size
    57    28 A8   D0    0619  1205          MOVL    RAB$L_RBF(R8),R7            ; and buffer address
        06 A9   B6    061D  1206          INCW    IS_W_LINE_NO(R9)                ; Increment line number
    2A A9   04    8A    0620  1207          BICB2   #SUM_M_SRCUPD,IS_B_FLAGS(R9)    ; Mark as source line
        2E A9   B4    0624  1208          CLRW    IS_W_INSERT_NO(R9)              ; Reset new/replacement lines count
                        0627  1209 20$:
                05      0627  1210          RSB
                        0628  1211 ;
                        0628  1212          .DSABL  LSB
```

```
                     0628  1214          .SBTTL  SKIP_SRC_LINES
                     0628  1215 ;
                     0628  1216 ; Routine to skip over source file lines
                     0628  1217 ;
                     0628  1218 ; Inputs:
                     0628  1219 ;
                     0628  1220 ;       R4  = Last line number to skip
                     0628  1221 ;       R8  = RAB address
                     0628  1222 ;
                     0628  1223 ; Outputs:
                     0628  1224 ;
                     0628  1225 ;       IS_W_LINE_NO(R9) = Last line number
                     0628  1226 ;
                     0628  1227 ;
                     0628  1228 SKIP_SRC_LINES:
          50   01    D0  0628  1229          MOVL    #1,R0                   ; Assume success
      06 A9    54    B1  062B  1230          CMPW    R4,IS_W_LINE_NO(R9)             ; Need to skip any?
            27    19  062F  1231          BLSS    20$                     ; No if LSS
         FE77    30  0631  1232          BSBW    ACCESS_SRC              ; Access source file
        21 50    E9  0634  1233          BLBC    R0,20$                  ; Error if LBC
                     0637  1234 10$:
                     0637  1235          $FIND   RAB = R8, -             ; Skip one line
                     0637  1236                  ERR = SUM$READ_ERR
          0F 50    E9  0646  1237          BLBC    R0,20$                  ; Error if LBC
          30 A9    B6  0649  1238          INCW    IS_W_DELETES(R9)                ; Increment deleted lines count
FFE4 06 A9   01    54  3D  064C  1239          ACBW    R4,#1,IS_W_LINE_NO(R9),10$      ; Increment line number and branch b
                     0653  1240                                          ; if more lines to skip
      00 05 A9   01    E2  0653  1241          BBSS    #PRC_V_DELINE, -        ; Set deleted lines information
                     0658  1242                  IS_B_PROCFLAGS(R9),20$  ; pending flag
                     0658  1243 20$:
                05   0658  1244          RSB
```

```
                     0659  1246          .SBTTL  COMMAND_CHECK
                     0659  1247 ;
                     0659  1248 ; Subroutine to check if line is a command
                     0659  1249 ;
                     0659  1250 ; Inputs:
                     0659  1251 ;
                     0659  1252 ;       R6 = Size of line
                     0659  1253 ;       R7 = Address of line
                     0659  1254 ;       R8 = RAB address
                     0659  1255 ;       R9 = Input stream control block
                     0659  1256 ;       R10= File block address
                     0659  1257 ;
                     0659  1258 ; Outputs:
                     0659  1259 ;       R4[CMND] = 0:Data  1:Command
                     0659  1260 ;       R4[EDTRM] = 0:Normal command  1:Data terminator command
                     0659  1261 ;       R4[EDEND] = 0:Data terminator  1:End of edit
                     0659  1262 ;       R6 = Size of line
                     0659  1263 ;       R7 = Address of line
                     0659  1264 ;
                     0659  1265 COMMAND_CHECK:
                     0659  1266          ASSUME  UPF_W_LOC2 EQ <UPF_W_LOC1+2>
00000000'EF   58  D0 0659  1267          MOVL    R8, SUM_CUR_RAB         ; Make the currently active RAB available
                     0660  1268                                          ; to the TPARSE action routines.
51  00000008'EF   DE 0660  1269          MOVAL   TPARSE_BLOCK,R1         ; Set pointer to Tparse parameter block
   28 A1   2A A9  90 0667  1270          MOVB    IS_B_FLAGS(R9), -       ; Get current input stream flags byte
                     066C  1271                  TPA_B_ISFLAGS(R1)
   00 28 A1   01  E5 066C  1272          BBCC    #SUM_V_AUDITNEW, -      ; but clear new audit trail flag
                     0671  1273                  TPA_B_ISFLAGS(R1),5$
                     0671  1274 5$:
         29 A1    94 0671  1275          CLRB    TPA_B_EDFLAGS(R1)       ; Clear all edit flags
   2A A1   0A AA  B0 0674  1276          MOVW    UPF_W_DOT(R10), -       ; Get current dot value
                     0679  1277                  TPA_W_DOT(R1)
         2C A1    D4 0679  1278          CLRL    TPA_W_LOC(R1)           ; Clear locator value and line type
         24 A1    D4 067C  1279          CLRL    TPA_W_LOC1(R1)          ; Clear loc-1 and loc-2
         30 A1    7C 067F  1280          CLRQ    TPA_Q_AUDDS(R1)         ; Clear audit descriptor
         38 A1    7C 0682  1281          CLRQ    TPA_Q_CMNT(R1)          ; Comment descriptor
      40 A1       56 7D 0685  1282          MOVQ    R6,TPA_Q_LINEDS(R1)     ; Save line size and address
      08 A1       56 7D 0689  1283          MOVQ    R6,TPA$L_STRINGCNT(R1)  ; Set TPARSE input descriptor
   00000000'EF    DF 068D  1284          PUSHAL  MER_KEY
   00000000'EF    DF 0693  1285          PUSHAL  MER_STATE
                  51 DD 0699  1286          PUSHL   R1
00000000'GF       03 FB 069B  1287          CALLS   #3,G^LIB$TPARSE
            51    50 E9 06A2  1288          BLBC    R0,20$                  ; Error if LBC
51  00000008'EF   DE 06A5  1289          MOVAL   TPARSE_BLOCK,R1         ; Set pointer to Tparse paramter block
         26 A1    B5 06AC  1290          TSTW    TPA_W_LOC2(R1)          ; Were two locators in command?
                  0B 13 06AF  1291          BEQL    8$                      ; No if EQL, so don't compare them
   26 A1   24 A1  B1 06B1  1292          CMPW    TPA_W_LOC1(R1),TPA_W_LOC2(R1) ; Is loc-1 <= loc-2?
                  04 15 06B6  1293          BLEQ    8$                      ; Yes if LEQ
                  50 D4 06B8  1294          CLRL    R0                      ; Set error status
                  3A 11 06BA  1295          BRB     20$                     ; and return
                     06BC  1296 8$:
      56    40 A1   7D 06BC  1297          MOVQ    TPA_Q_LINEDS(R1),R6     ; Reset line size and address,
      22 A8    56   B0 06C0  1298          MOVW    R6,RAB$W_RSZ(R8)        ; Reset RAB block record size
   2A A9    28 A1   90 06C4  1299          MOVB    TPA_B_ISFLAGS(R1), -    ; input stream flags byte,
                     06C9  1300                  IS_B_FLAGS(R9)
   09 AA    29 A1   90 06C9  1301          MOVB    TPA_B_EDFLAGS(R1), -    ; edit flags byte,
                     06CE  1302                  UPF_B_EDFLAGS(R10)
```

COMMAND_CHECK

```
                0A AA    2A A1    B0   06CE  1303          MOVW    TPA_W_DOT(R1),UPF_W_DOT(R10) ; dot value,
                04 AA    24 A1    D0   06D3  1304          MOVL    TPA_W_LOC1(R1), -            ; locator 1,
                                       06D8  1305                  UPF_W_LOC1(R10)             ; and locator 2
                20 AA    38 A1    7D   06D8  1306          MOVQ    TPA_Q_CMNT(R1), -            ; Comment descriptor
                                       06DD  1307                  UPF_Q_CMNT(R10)
                10 2A A9   01     E1   06DD  1308          BBC     #SUM_V_AUDITNEW, -           ; If new audit trail
                                       06E2  1309                  IS_B_FLAGS(R9),10$
                18 AA    30 A1    D0   06E2  1310          MOVL    TPA_Q_AUDDS(R1), -           ; Copy size of string
                                       06E7  1311                  UPF_Q_AUDDS(R10)
                         3F       BB   06E7  1312          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
            28 AA    34 B1    30 A1    28   06E9  1313          MOVC3   TPA_Q_AUDDS(R1), -           ; Copy audit string
                                       06F0  1314                  @TPA_Q_AUDDS+4(R1),UPF_T_AUDST(R10)
                         3F       BA   06F0  1315          POPR    #^M<R0,R1,R2,R3,R4,R5>
                                       06F2  1316 10$:
                   54    2E A1    3C   06F2  1317          MOVZWL  TPA_W_LINTYP(R1),R4          ; Set line type flags
                                       06F6  1318 20$:
                                  05   06F6  1319          RSB
```

```
                     06F7  1321 ;
                     06F7  1322 ; Tparse action routines
                     06F7  1323 ;
                     06F7  1324 ;
                     06F7  1325 ACT_BLANKS_SIG:
                0000 06F7  1326         .WORD   0
00 04 AC    00    E2 06F9  1327         BBSS    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$
                     06FE  1328 10$:
                  04 06FE  1329         RET
                     06FF  1330 ;
                     06FF  1331 ;
                     06FF  1332 ACT_BLANKS_NSIG:
                0000 06FF  1333         .WORD   0
00 04 AC    00    E5 0701  1334         BBCC    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$
                     0706  1335 10$:
                  04 0706  1336         RET
                     0707  1337 ;
                     0707  1338 ;
                     0707  1339 ACT_PERCENT:
                0000 0707  1340         .WORD   0
   28 AC    01    88 0709  1341         BISB    #SUM_M_AUDIT, -         ; Switch on audit trail
                     070D  1342                 TPA_B_ISFLAGS(AP)
                  04 070D  1343         RET
                     070E  1344 ;
                     070E  1345 ;
                     070E  1346 ACT_BACKSLASH:
                0000 070E  1347         .WORD   0
   28 AC    01    8A 0710  1348         BICB    #SUM_M_AUDIT, -         ; Switch off audit trail
                     0714  1349                 TPA_B_ISFLAGS(AP)
                  04 0714  1350         RET
                     0715  1351 ;
                     0715  1352 ;
                     0715  1353 ACT_ESC:
                0040 0715  1354         .WORD   ^M<R6>
      51    01    D0 0717  1355         MOVL    #1,R1                   ; Set index
   56    40 AC    9E 071A  1356         MOVAB   TPA_Q_LINEDS(AP),R6     ; Point to buffer descriptor
             66    B7 071E  1357         DECW    (R6)                    ; Reduce line length by one
             3F    BB 0720  1358         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers across MOVC3s.
04 B6   04 B641    66 0722  1359         MOVC3   (R6),@4(R6)[R1],@4(R6)  ; Move up line.
   51   00000000'EF D0 0729  1360         MOVL    SUM_CUR_RAB, R1         ; Get the current RAB,
             10    E0 0730  1361         BBS     #RAB$V_LOC,-            ; check to see if we should
          06 04 A1      0732  1362                 RAB$L_ROP(R1), 10$      ; propagate the shifted string
24 B1   04 B6    66    28 0735  1363         MOVC3   (R6),@4(R6),@RAB$L_UBF(R1) ; to the UBF.
             3F    BA 073B  1364 10$:      POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
             04    04 073D  1365         RET
                     073E  1366 ;
                     073E  1367 ;
                     073E  1368 ACT_EXIT:
                0000 073E  1369         .WORD   0
   2E AC    20 AC    B0 0740  1370         MOVW    TPA$L_PARAM(AP), -      ; Set return type
                     0745  1371                 TPA_W_LINTYP(AP)
                  04 0745  1372         RET
                     0746  1373 ;
                     0746  1374 ;
                     0746  1375 ACT_LOC1:
                0000 0746  1376         .WORD   0
   2E AC    01    B0 0748  1377         MOVW    #CMD_M_CMND,TPA_W_LINTYP(AP)    ; Assume normal command
```

```
24 AC   2C AC   B0   074C  1378          MOVW    TPA_W_LOC(AP),TPA_W_LOC1(AP)
                07   13   0751  1379          BEQL    10$                     ; If EQL is a normal command
   2E AC     03   B0   0753  1380          MOVW    #CMD_M_CMND!CMD_M_EDTRM,TPA_W_LINTYP(AP) ; Set as data terminator co
         2C AC    B4   0757  1381          CLRW    TPA_W_LOC(AP)
                       075A  1382 10$:
                04   075A  1383          RET
                       075B  1394 ;
                       075B  1385 ;
                       075B  1386 ACT_LOC2:
              0000     075B  1387          .WORD   0
26 AC   2C AC   B0   075D  1388          MOVW    TPA_W_LOC(AP),TPA_W_LOC2(AP)
                04   0762  1389          RET
                       0763  1390 ;
                       0763  1391 ;
                       0763  1392 ACT_DOT:
              0000     0763  1393          .WORD   0
2C AC   2A AC   B0   0765  1394          MOVW    TPA_W_DOT(AP),TPA_W_LOC(AP)
                04   076A  1395          RET
                       076B  1396 ;
                       076B  1397 ;
                       076B  1398 ACT_LOCNUM:
              0000     076B  1399          .WORD   0
2C AC   1C AC   B0   076D  1400          MOVW    TPA$L_NUMBER(AP),TPA_W_LOC(AP)
2A AC   2C AC   B0   0772  1401          MOVW    TPA_W_LOC(AP),TPA_W_DOT(AP)
                04   0777  1402          RET
                       0778  1403 ;
                       0778  1404 ;
                       0778  1405 ACT_PLUS:
              0000     0778  1406          .WORD   0
2C AC   1C AC   A0   077A  1407          ADDW2   TPA$L_NUMBER(AP),TPA_W_LOC(AP)
2A AC   2C AC   B0   077F  1408          MOVW    TPA_W_LOC(AP),TPA_W_DOT(AP)
                04   0784  1409          RET
                       0785  1410 ;
                       0785  1411 ;
                       0785  1412 ACT_AUDIT:
              0000     0785  1413          .WORD   0
34 AC     0C AC   D0   0787  1414          MOVL    TPA$L_STRINGPTR(AP),TPA_Q_AUDDS+4(AP)
   28 AC     02   88   078C  1415          BISB    #SUM_M_AUDITNEW, -      ; Set new audit trail flag
                       0790  1416                  TPA_B_ISFLAGS(AP)
00 04 AC    00   E2   0790  1417          BBSS    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$ ; Make blanks significant
                       0795  1418 10$:
                04   0795  1419          RET
                       0796  1420 ;
                       0796  1421 ;
                       0796  1422 ACT_AUDCH:
              0000     0796  1423          .WORD   0
   10   30 AC   D1   0798  1424          CMPL    TPA_Q_AUDDS(AP),#16     ; Is audit trail at maximum size?
                03   18   079C  1425          BGEQ    10$                     ; Yes if GEQ
         30 AC    D6   079E  1426          INCL    TPA_Q_AUDDS(AP)         ; Increment audit trail size
                       07A1  1427 10$:
                04   07A1  1428          RET
                       07A2  1429 ;
                       07A2  1430 ;
                       07A2  1431 ACT_AUDEND:
              0000     07A2  1432          .WORD   0
00 04 AC    00   E5   07A4  1433          BBCC    #TPA$V_BLANKS,TPA$L_OPTIONS(AP),10$ ; Switch off blank processing
                       07A9  1434 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  1450          .SBTTL  SUM$CLOSE
                        07B9  1451 ;
                        07B9  1452 ;
                        07B9  1453 ; This procedure is called from the main program prior to closing
                        07B9  1454 ; the input file.  It ensures that the main program source file
                        07B9  1455 ; is connected to the RAB.
                        07B9  1456 ;
                        07B9  1457 ;
                        07B9  1458 ; Inputs:
                        07B9  1459 ;
                        07B9  1460 ;       4(AP) = Address of SUM control block
                        07B9  1461 ;
                        07B9  1462 ; Outputs:
                        07B9  1463 ;
                        07B9  1464 ;       None
                        07B9  1465 ;
                        07B9  1466 ;
                0300    07B9  1467          .ENTRY  SUM$CLOSE,^M<R8,R9>
                        07BB  1468 ;
          51    04 AC   D0    07BB  1469          MOVL    4(AP),R1                ; Get control block address
          59    04 A1   D0    07BF  1470          MOVL    SUM_L_ISDATA(R1),R9     ; and set data block pointer
                2C      13    07C3  1471          BEQL    20$
                69      D5    07C5  1472          TSTL    IS_L_FILELIST(R9)       ; Is there an update list?
                28      13    07C7  1473          BEQL    20$                     ; No if EQL, file already accessed
          51    1C A9   D0    07C9  1474          MOVL    IS_L_MAIN_FAB(R9),R1    ; Get main program FAB address
                02 A1   B5    07CD  1475          TSTW    FAB$W_IFI(R1)           ; Is source file open?
                07      13    07D0  1476          BEQL    10$                     ; No if EQL
          58    20 A9   D0    07D2  1477          MOVL    IS_L_MAIN_RAB(R9),R8    ; Set RAB pointer
                FCD2    30    07D6  1478          BSBW    ACCESS_SRC              ; Access source file
                        07D9  1479 10$:
                08 A9   D5    07D9  1480          TSTL    IS_L_OPEN_FILE(R9)      ; Is an update file open?
                13      13    07DC  1481          BEQL    20$                     ; No if EQL
                        07DE  1482          $CLOSE  FAB = IS_T_FAB(R9), -
                        07DE  1483                  ERR = SUM$CLOSE_ERR         ; Close update file
                08 A9   D4    07EE  1484          CLRL    IS_L_OPEN_FILE(R9)      ; and clear marker
                        07F1  1485 20$:
                04      07F1  1486          RET
                        07F2  1487 ;
                        07F2  1488 ;
                        07F2  1489          .END
```

```
$$$CNT                  =  00000003
$$$FLG                  =  FFFFFFFF
$$$KEY                  =  FFFFFFFF
$$$KFG                  =  FFFFFFFF
$$$MOD                  =  00000000
$$.TMP1                 =  00000002
$$.TMP2                 =  000000A9
$$KEYTAB                =  00000000 R      05
..AFLG                  =  00000000
..FLG                   =  00000002
..MOD                   =  00000000
..TYP                   =  0000001F
.LEN                    =  00000001
ACCESS_SRC                 00000C4AB R      07
ACCESS_UPDATE              00000540 R      07
ACT_AUDCH                  00000796 R      07
ACT_AUDEND                 000007A2 R      07
ACT_AUDIT                  00000785 R      07
ACT_BACKSLASH              0000070E R      07
ACT_BLANKS_NSIG            000006FF R      07
ACT_BLANKS_SIG             000006F7 R      07
ACT_CMNT                   000007AA R      07
ACT_DOT                    00000763 R      07
ACT_ESC                    00000715 R      07
ACT_EXIT                   0000073E R      07
ACT_LOC1                   00000746 R      07
ACT_LOC2                   0000075B R      07
ACT_LOCNUM                 0000076B R      07
ACT_PERCENT                00000707 R      07
ACT_PLUS                   00000778 R      07
ACT_SUPPRESS               000007B2 R      07
AUDCH                      000000A9 R      04
BIT...                  =  00000005
CHECK_ERR                  000003B0 R      07
CMD_M_ALL               =  00000007
CMD_M_CMND              =  00000001
CMD_M_EDEND             =  00000004
CMD_M_EDTRM             =  00000002
CMD_V_CMND              =  00000000
CMD_V_EDEND             =  00000002
CMD_V_EDTRM             =  00000001
CMND                       0000003F R      04
CMNT                       000000C3 R      04
COMMA                   =  0000002C
COMMAND_CHECK              00000659 R      07
DATA                       00000032 R      04
EDIT                       00000059 R      04
ED_B_FILENO                00000019
ED_B_FLAGS                 00000018
ED_K_BLN                   0000001A
ED_L_BWD                   00000004
ED_L_FILE                  00000014
ED_L_FWD                   00000000
ED_M_SEQERR             =  00000002
ED_M_SUPPRESS           =  00000001
ED_V_SEQERR             =  00000001
ED_V_SUPPRESS           =  00000000
ED_W_LINES                                  0000000C
ED_W_LOC1                                   00000008
ED_W_LOC2                                   0000000A
ED_W_RFA                                    0000000E
FAB$B_BID                               =   00000000
FAB$B_BLN                               =   00000001
FAB$C_BID                               =   00000003
FAB$C_BLN                               =   00000050
FAB$K_BLN                               =   00000050
FAB$L_FOP                               =   00000004
FAB$L_NAM                               =   00000028
FAB$V_NAM                               =   00000018
FAB$W_IFI                               =   00000002
GET_IS_BLK                                  000000BA R      07
INSERT_NODE                                 000001F5 R      07
IS_B_FLAGS                                  0000002A
IS_B_PROCFLAGS                              00000005
IS_B_STATE                                  00000004
IS_K_BLN                                    00000082
IS_L_CONN_FILE                              0000000C
IS_L_EDIT_BLK                               00000010
IS_L_FILELIST                               00000000
IS_L_FIRST_EDIT                             00000014
IS_L_LAST_EDIT                              00000018
IS_L_MAIN_FAB                               0000001C
IS_L_MAIN_RAB                               00000020
IS_L_OPEN_FILE                              00000008
IS_T_FAB                                    00000032
IS_W_DELETES                                00000030
IS_W_HIGH_LOC2                              0000002C
IS_W_INSERT_NO                              0000002E
IS_W_LINE_NO                                00000006
IS_W_MAIN_RFA                               00000024
LESSTHAN                                =   0000003C
LIB$FREE_VM                                 ********    X   07
LIB$GET_VM                                  ********    X   07
LIB$TPARSE                                  ********    X   07
LINE_BLK                                    0000040B R      07
LINE_EOF                                    000004A1 R      07
LINE_GET                                    0000042E R      07
LINE_NUP                                    00000327 R      07
LINE_SET                                    00000300 R      07
LINE_SRC                                    0000032D R      07
LINE_UPD                                    0000033E R      07
LINE_UPE                                    000003C4 R      07
LINE_UPR                                    000003EC R      07
LOCATOR                                     000000C7 R      04
MER_KEY                                     00000000 RG     05
MER_STATE                                   00000000 RG     04
NAM$B_RSL                               =   00000003
NAM$K_BLN                               =   00000060
NAM$L_RSA                               =   00000004
PRC_M_DELINE                            =   00000002
PRC_M_ERRORS                            =   00000004
PRC_M_EXPED                             =   00000001
PRC_M_HIEDIT                            =   00000008
PRC_M_NODATA                            =   00000010
```

```
PRC_V_DELINE                =  00000001
PRC_V_ERRORS                =  00000002
PRC_V_EXPED                 =  00000000
PRC_V_HIEDIT                =  00000003
PRC_V_NODATA                =  00000004
PROCESS_FILE                   000000EF  R      07
RAB$B_RAC                   =  0000001E
RAB$C_RFA                   =  00000002
RAB$C_SEQ                   =  00000000
RAB$L_FAB                   =  0000003C
RAB$L_RBF                   =  00000028
RAB$L_ROP                   =  00000004
RAB$L_UBF                   =  00000024
RAB$M_LOC                   =  00010000
RAB$V_LOC                   =  00000010
RAB$W_ISI                   =  00000002
RAB$W_RFA                   =  00000010
RAB$W_RSZ                   =  00000022
RAB$W_USZ                   =  00000020
READ_SRC_LINE                  000005EB  R      07
READ_SRC_LINEA                 000005F4  R      07
READ_UPD_LINE                  00000230  R      07
READ_UPD_LINEA                 00000236  R      07
RESTORE_SRC_RFA                000004F4  R      07
RMS$_EOF                    =  0001827A
SAVE_SRC_RFA                   000004E9  R      07
SEMICOLON                   =  0000003B
SET_UP_NODES                   0000015A  R      07
SIZ...                      =  00000001
SKIP_SRC_LINES                 00000628  R      07
SUM$CLOSE                      000007B9  RG     07
SUM$CLOSE_ERR                  ********    X    07
SUM$INIT                       00000006  R      07
SUM$INIT_CMND                  00000004  RG     07
SUM$INIT_EDIT                  00000000  RG     07
SUM$LIB_ERR                    ********    X    07
SUM$LINE                       00000286  RG     07
SUM$OPEN_ERR                   ********    X    07
SUM$READ_ERR                   ********    X    07
SUM$VIRT_ADDR                  ********    X    07
SUM$_EDITSCLSH              =  00848800
SUM$_EDOUTSEQ               =  00848818
SUM$_PRMEOF                 =  00848810
SUM$_SLPSYNERR              =  00848808
SUM_B_FLAGS                    0000001C
SUM_CDR_RAB                    00000000  R      02
SUM_DISPATCH                   00000298  R      07
SUM_EDSZE                      00000004  R      03
SUM_ISSZE                      00000000  R      03
SUM_K_BLN                      0000001D
SUM_L_ISDATA                   00000004
SUM_L_STS                      00000000
SUM_M_AUDIT                 =  00000001
SUM_M_AUDITNEW              =  00000002
SUM_M_DELETE                =  00000010
SUM_M_SRCUPD                =  00000004
SUM_M_SUBCLSH               =  00000008
SUM_Q_AUDDS                    00000008
SUM_Q_FILESP                   00000010
SUM_RETURN                     000002AF  R      07
SUM_ST_BLK                  =  00000006
SUM_ST_EOF                  =  00000008
SUM_ST_GET                  =  00000007
SUM_ST_NUP                  =  00000001
SUM_ST_SET                  =  00000000
SUM_ST_SRC                  =  00000002
SUM_ST_UPD                  =  00000003
SUM_ST_UPE                  =  00000004
SUM_ST_UPR                  =  00000005
SUM_TPARSE                     0000002C  R      02
SUM_UBF_ADDR                   00000004  R      02
SUM_V_AUDIT                 =  00000000
SUM_V_AUDITNEW              =  00000001
SUM_V_DELETE                =  00000004
SUM_V_SRCUPD                =  00000002
SUM_V_SUBCLSH               =  00000003
SUM_W_INSERT_NO                0000001A
SUM_W_LINE_NO                  00000018
SYS$CLOSE                      ********    GX   07
SYS$CONNECT                    ********    GX   07
SYS$DISCONNECT                 ********    GX   07
SYS$FIND                       ********    GX   07
SYS$GET                        ********    GX   07
SYS$OPEN                       ********    GX   07
SYS$REWIND                     ********    GX   07
TERM                           0000004C  R      04
TPA$K_COUNTO                =  00000008
TPA$K_LENGTHO               =  00000024
TPA$L_NUMBER                =  0000001C
TPA$L_OPTIONS               =  00000004
TPA$L_PARAM                 =  00000020
TPA$L_STRINGCNT             =  00000008
TPA$L_STRINGPTR             =  0000000C
TPA$V_BLANKS                =  00000000
TPA$_ALPHA                  =  000001EE
TPA$_ANY                    =  000001ED
TPA$_BLANK                  =  000001F2
TPA$_DECIMAL                =  000001F3
TPA$_DIGIT                  =  000001EF
TPA$_EOS                    =  000001F7
TPA$_EXIT                   =  FFFFFFFF
TPA$_FAIL                   =  FFFFFFFE
TPA$_FILESPEC               =  000001EA
TPA$_HEX                    =  000001F5
TPA$_IDENT                  =  000001EC
TPA$_KEYWORD                =  00000100
TPA$_LAMBDA                 =  000001F6
TPA$_MAXKEY                 =  000000DC
TPA$_OCTAL                  =  000001F4
TPA$_STRING                 =  000001F0
TPA$_SUBXPR                 =  000001F8
TPA$_SYMBOL                 =  000001F1
TPA$_UIC                    =  000001EB
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.

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
```

SUMSLP
MAP

SRTTRNMSG
LIS

DATA
LIS

SUMSHR
MAP

SUMDATA
LIS

UTLCHKCLI
LIS

SUM

SUM
MAP

SUMDEF
MAR

MAIN
LIS

SUMED
LIS

SUMEDIT
LIS

GETCMD
LIS

SUMLIST
LIS

SUMMAIN
LIS

SUMTFRVEC
LIS

SUMFILES
LIS

SUMMSG
LIS

SUMERROR
LIS

SUMOPEN
LIS

SYS

UPDATE
LIS

SYS
MAP