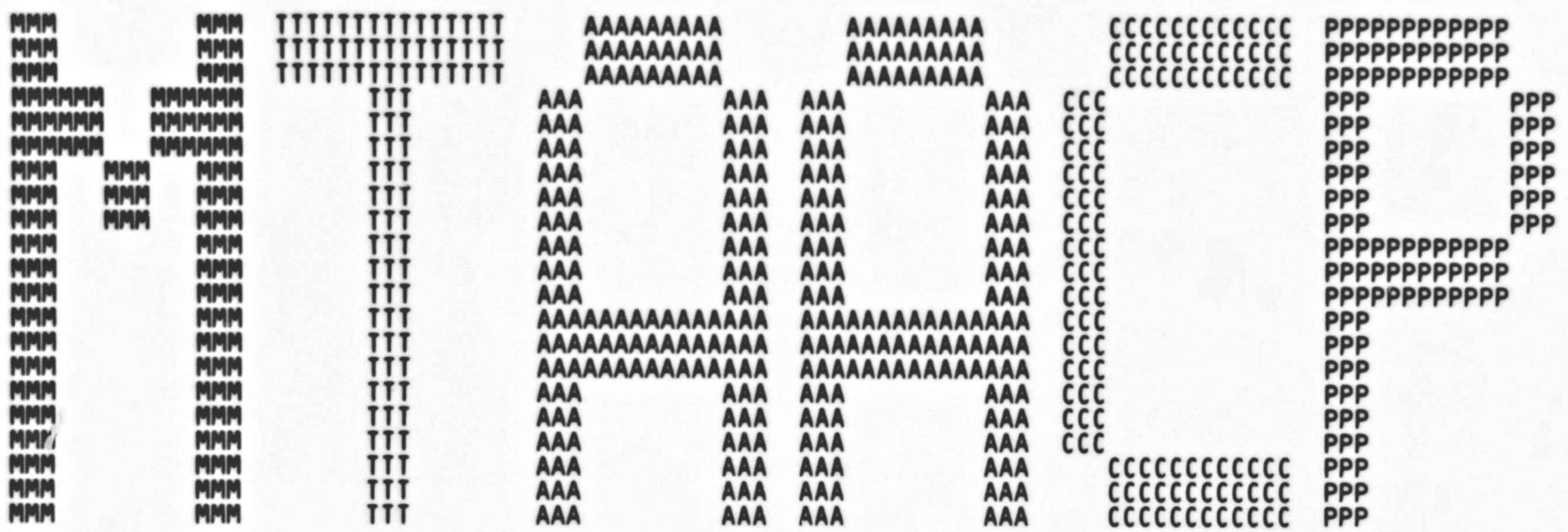
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP

FILEID**COMLABPRC

```
  CCCCCCCC    OOOOOO    MM      MM  LL            AAAAAA    BBBBBBBB    PPPPPPPP    RRRRRRRR      CCCCCCCC
  CCCCCCCC    OOOOOO    MM      MM  LL            AAAAAA    BBBBBBBB    PPPPPPPP    RRRRRRRR      CCCCCCCC
CC          OO      OO  MMMM  MMMM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MMMM  MMMM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM  MM  MM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM  MM  MM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM      MM  LL          AA      AA  BBBBBBBB    PPPPPPPP    RRRRRRRR    CC
CC          OO      OO  MM      MM  LL          AA      AA  BBBBBBBB    PPPPPPPP    RRRRRRRR    CC
CC          OO      OO  MM      MM  LL          AAAAAAAAAA  BB      BB  PP          RR  RR      CC
CC          OO      OO  MM      MM  LL          AAAAAAAAAA  BB      BB  PP          RR  RR      CC
CC          OO      OO  MM      MM  LL          AA      AA  BB      BB  PP          RR    RR    CC                ....
  CCCCCCCC    OOOOOO    MM      MM  LLLLLLLLLL  AA      AA  BBBBBBBB    PP          RR      RR    CCCCCCCC        ....
  CCCCCCCC    OOOOOO    MM      MM  LLLLLLLLLL  AA      AA  BBBBBBBB    PP          RR      RR    CCCCCCCC        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII    SSSSSSSS
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
```

```
;     1          0001  0 MODULE COMLABPROC (LANGUAGE (BLISS32) ,
;     2          0002  0                     IDENT = 'V04-000'
;     3          0003  0                     ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY: INITIALIZE, MOUNT, MTAACP
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !       This module contains routines that are shared amoung the
;    35          0035  1 !       MOUNT, INIT, and MTAACP. These routines deal with the
;    36          0036  1 !       processing of the various labels that the MTAACP supports.
;    37          0037  1 !
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       VMS operating system, including privileged system services
;    42          0042  1 !       and internal exec routines.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Meg Dumont,                    CREATION DATE:  21-Feb-1983
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-005 HH0041          Hai Huang       24-Jul-1984
;    53          0053  1 !               Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
;    54          0054  1 !
;    55          0055  1 !       V03-004 MMD0272         Meg Dumont,     23-Mar-1984  9:41
;    56          0056  1 !               Add the common routine GET_RECORD part of support for $MTACCESS
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-003 MMD0175         Meg Dumont,     26-May-1983  15:10
;    59          0059  1 !               Change VOL1 to indicate ANSI level 4 when writing system
;    60          0060  1 !               code in VOL1
;    61          0061  1 !
;    62          0062  1 !       V03-002 MMD0137         Meg Dumont,     12-Apr-1983  17:30
;    63          0063  1 !               In TAPE_OWNER_PROT, added a check for a nonVMS nonblank
;    64          0064  1 !               VOL1 OWNER IDENTIFIER field.
;    65          0065  1 !
;    66          0066  1 !       V03-001 MMD0122         Meg Dumont,     29-Mar-1983  0:46
;    67          0067  1 !               This module is does the common ANSI label processing for
;    68          0068  1 !               thE MTAACP, MOUNT and INIT.
;    69          0069  1 !
;    70          0070  1 !
;    71          0071  1 !**
;    72          0072  1
;    73          0073  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    74          0074  1
;    75          0075  1 REQUIRE 'SRC$:MTADEF.B32';
;    76          0459  1
;    77          0460  1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.B32';
;    78          0592  1
;    79          0593  1 FORWARD ROUTINE
;    80          0594  1     GET_RECORD,                             ! routine to get record tape is reading
;    81          0595  1     CHECK_PROT,                             ! check VMS protection on tape
;    82          0596  1     FORMAT_VOLOWNER     : NOVALUE,          ! format the volume onwer field
;    83          0597  1     PROCESS_VOL2_LABEL,                     ! intepret the VOL2 label
;    84          0598  1     TAPE_OWN_PROT;                          ! determine the VMS owner and
;    85          0599  1                                             ! protection of a tape
;    86          0600  1     EXTERNAL ROUTINE
;    87          0601  1         LIB$CVT_OTB     : ADDRESSING_MODE (GENERAL);
;    88          0602  1
;    89          0603  1
```

```
;    91          0604  1 GLOBAL ROUTINE GET_RECORD(UCB) =
;    92          0605  1
;    93          0606  1 !++
;    94          0607  1 !
;    95          0608  1 ! FUNCTIONAL DESCRIPTION:
;    96          0609  1 !       This routine is called before and after the call to $MTACCESS to return
;    97          0610  1 !       the record that the tape drive is currently processing
;    98          0611  1 !
;    99          0612  1 ! CALLING SEQUENCE:
;   100          0613  1 !       KERNEL_CALL (GET_RECORD, ARG1)
;   101          0614  1 !
;   102          0615  1 ! INPUT PARAMETERS:
;   103          0616  1 !       ARG1 -  Address of tapes UCB
;   104          0617  1 !
;   105          0618  1 ! IMPLICIT INPUTS:
;   106          0619  1 !       NONE
;   107          0620  1 !
;   108          0621  1 ! OUTPUT PARAMETERS:
;   109          0622  1 !       NONE
;   110          0623  1 !
;   111          0624  1 ! IMPLICIT OUTPUTS:                                                              ;
;   112          0625  1 !
;   113          0626  1 ! ROUTINE VALUE:
;   114          0627  1 !       Current record the tape drive is processing.                             ;
;   115          0628  1 !
;   116          0629  1 ! SIDE EFFECTS:
;   117          0630  1 !       NONE
;   118          0631  1 !
;   119          0632  1 ! USER ERRORS:
;   120          0633  1 !       NONE
;   121          0634  1 !
;   122          0635  1 !--
;   123          0636  1
;   124          0637  2     BEGIN
;   125          0638  2         MAP UCB : REF BBLOCK;
;   126          0639  2         RETURN .UCB[UCB$L_RECORD];
;   127          0640  1     END;

                                                        .TITLE  COMLABPROC
                                                        .IDENT  \V04-000\

                                                        .EXTRN  LIB$CVT_OTB

                                                        .PSECT  $CODE$,NOWRT,2

                                        0000 00000      .ENTRY  GET_RECORD, Save nothing                ; 0604
                              50    04  AC   D0 00002   MOVL    UCB, R0                                 ; 0639
                              50  00B0  C0   D0 00006   MOVL    176(R0), R0                             ;
                                        04   0000B      RET                                             ; 0640

; Routine Size:  12 bytes,    Routine Base:  $CODE$ + 0000

;   128          0641  1
;   129          0642  1 GLOBAL ROUTINE TAPE_OWN_PROT ( VOLUIC, VOLUME_PROT : REF BITVECTOR[%BPVAL],
;   130          0643  1                                PROCESS_UIC, VOL1 ) =
```

```
;  131     0644  1
;  132     0645  1 !++
;  133     0646  1 !
;  134     0647  1 ! FUNCTIONAL DESCRIPTION:
;  135     0648  1 !       This routine determines the tape owner and protection of the volume.
;  136     0649  1 !       It uses the OWNER IDENTIFIER field of the VOL1 label. If this
;  137     0650  1 !       field contians a value that VMS does not interpret then, the
;  138     0651  1 !       user is required to have privlegdes to mount the tape. Unless
;  139     0652  1 !       that user is the VMS owner of the tape determined from the VOL2
;  140     0653  1 !       label.
;  141     0654  1 !
;  142     0655  1 ! CALLING SEQUENCE:
;  143     0656  1 !       TAPE_OWN_PROT ( ARG1, ARG2, ARG3, ARG4 )
;  144     0657  1 !
;  145     0658  1 ! INPUT PARAMETERS:
;  146     0659  1 !       ARG1 -  Address of area to store the volume uic
;  147     0660  1 !       ARG2 -  Address of area to store the volume protection
;  148     0661  1 !       ARG3 -  Process UIC
;  149     0662  1 !       ARG4 -  Address of ANSI VOL1 label
;  150     0663  1 !
;  151     0664  1 ! IMPLICIT INPUTS:
;  152     0665  1 !       NONE
;  153     0666  1 !
;  154     0667  1 ! OUTPUT PARAMETERS:
;  155     0668  1 !       NONE
;  156     0669  1 !
;  157     0670  1 ! IMPLICIT OUTPUTS:
;  158     0671  1 !       VOLUME_UIC - owning uic of tape
;  159     0672  1 !       VOLUME_PROT -  tape protection
;  160     0673  1 !
;  161     0674  1 ! ROUTINE VALUE:
;  162     0675  1 !       TRUE - Field was blank or was specied in VMS format
;  163     0676  1 !       FALSE - Field was not VMS format, but was pre ANSI Label Standard
;  164     0677  1 !               version 4 and the tape was created on another DEC operating
;  165     0678  1 !               system that is it has D% information.
;  166     0679  1 !
;  167     0680  1 ! SIDE EFFECTS:
;  168     0681  1 !       NONE
;  169     0682  1 !
;  170     0683  1 ! USER ERRORS:
;  171     0684  1 !       NONE
;  172     0685  1 !
;  173     0686  1 !--
;  174     0687  1
;  175     0688  2     BEGIN
;  176     0689  2
;  177     0690  2     BIND
;  178     0691  2         VOLUME_UIC = .VOLUIC;               ! Address of volume uic
;  179     0692  2
;  180     0693  2     MAP
;  181     0694  2         VOL1    : REF BBLOCK;               ! Address of VOL1 label
;  182     0695  2
;  183     0696  2     LOCAL
;  184     0697  2         CONV_BUF        : VECTOR [6, BYTE], ! buffer used for converting UIC
;  185     0698  2         VALUE,                              ! used to hold parital UIC's
;  186     0699  2         P;                                  ! ptr into VOL1 tape owner field
;  187     0700  2
```

```
;  188          0701  2     ! bit numbers for different protections
;  189          0702  2     !
;  190          0703  2     LITERAL
;  191          0704  2         WORLD_WRITE = 13,
;  192          0705  2         WORLD_READ  = 12,
;  193          0706  2         GROUP_WRITE = 9,
;  194          0707  2         GROUP_READ  = 8;
;  195          0708  2
;  196          0709  2     ! If the LABEL STANDARD VERSION of the VOL1 label (CP 80) is a 4 then
;  197          0710  2     ! do not process the VOL1 OWNER IDENTIFIER field.
;  198          0711  2
;  199          0712  2     IF .VOL1[VL1$B_LBLSTDVER] EQL '4'
;  200          0713  2         THEN RETURN TRUE;
;  201          0714  2
;  202          0715  2     ! if ANSI tape produced by VAX system, decode tape owner field
;  203          0716  2
;  204          0717  2     IF .(VOL1[VL1$T_VOLOWNER])<0, 24> EQL 'D%C'
;  205          0718  2       THEN
;  206          0719  2         BEGIN
;  207          0720  2
;  208          0721  2         ! set up the pointer to begining of tape owner field
;  209          0722  2
;  210          0723  2         P = VOL1[VL1$T_VOLOWNER] + 3;
;  211          0724  2
;  212          0725  2         ! test for encoding
;  213          0726  2
;  214          0727  2         IF .(.P)<0, 8> NEQ ' '
;  215          0728  3         THEN
;  216          0729  4             BEGIN
;  217          0730  4
;  218          0731  4             ! move the UIC group field from the VOL1 label to the buffer
;  219          0732  4
;  220          0733  4             CH$MOVE(5, .P, CONV_BUF);
;  221          0734  4
;  222          0735  4             ! remove overlay encoding
;  223          0736  4
;  224          0737  4             IF .(.P)<0, 8> GEQ 'A'
;  225          0738  4             THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
;  226          0739  4
;  227          0740  4             ! convert to ASCII to binary exit with failure not a VMS tape
;  228          0741  4
;  229          0742  4             IF NOT LIB$CVT_OTB(5, CONV_BUF, VALUE) THEN RETURN FALSE;
;  230          0743  4
;  231          0744  4             ! fill in the UIC group field
;  232          0745  4
;  233          0746  4             VOLUME_UIC<16, 16> = .VALUE<0, 16>;
;  234          0747  3             END;
;  235          0748  3
;  236          0749  3         ! point to UIC member field
;  237          0750  3
;  238          0751  3         P = .P + 5;
;  239          0752  3
;  240          0753  3         ! test for encoding
;  241          0754  3
;  242          0755  3         IF .(.P)<0, 8> NEQ ' '
;  243          0756  3         THEN
;  244          0757  4             BEGIN
```

```
;  245          0758  4
;  246          0759  4              ! move member number into convert buffer
;  247          0760  4
;  248          0761  4              CH$MOVE(5, .P, CONV_BUF);
;  249          0762  4
;  250          0763  4              ! remove overlay encoding
;  251          0764  4
;  252          0765  4              IF .(.P)<0, 8> GEQ 'A'
;  253          0766  4              THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
;  254          0767  4
;  255          0768  4              ! convert to ASCII to binary exit when failure not a VAX tape
;  256          0769  4
;  257          0770  4              IF NOT LIB$CVT_OTB(5, CONV_BUF, VALUE)
;  258          0771  4              THEN
;  259          0772  5                  BEGIN
;  260          0773  5
;  261          0774  5                  ! patch up UIC before returning
;  262          0775  5
;  263          0776  5                  VOLUME_UIC = .PROCESS_UIC;
;  264          0777  5                  RETURN FALSE;
;  265          0778  4                  END;
;  266          0779  4
;  267          0780  4              ! fill in the UIC member field
;  268          0781  4
;  269          0782  4              VOLUME_UIC <0, 16> = .VALUE<0, 16>;
;  270          0783  3              END;
;  271          0784  3
;  272          0785  3          ! Now tape_prot must be decoded if both group and member are blank then
;  273          0786  3          ! all privileges granted
;  274          0787  3          !
;  275          0788  3          ! pointer to group uic
;  276          0789  3
;  277          0790  3          P = .P - 5;
;  278          0791  3
;  279          0792  3          ! if field is not blank, then there is a protection mask
;  280          0793  3
;  281          0794  3          IF NOT CH$FAIL(CH$FIND_NOT_CH(10, .P, ' '))
;  282          0795  3          THEN
;  283          0796  4              BEGIN
;  284          0797  4
;  285          0798  4              ! any mask means no world write
;  286          0799  4
;  287          0800  4              VOLUME_PROT[WORLD_WRITE] = 1;
;  288          0801  4
;  289          0802  4              ! if the 1st char is a digit then no world access
;  290          0803  4
;  291          0804  4              IF .(.P)<0, 8> LSS 'A'
;  292          0805  4              THEN VOLUME_PROT[WORLD_READ] = 1;
;  293          0806  4
;  294          0807  4              ! pointer to member field
;  295          0808  4
;  296          0809  4              P = .P + 5;
;  297          0810  4
;  298          0811  4              ! test for group rights.  all spaces means both read and write
;  299          0812  4
;  300          0813  4              IF NOT CH$FAIL(CH$FIND_NOT_CH(5, .P, ' '))
;  301          0814  4              THEN
```

```
;  302          0815  5                BEGIN
;  303          0816  5
;  304          0817  5                ! write protection against group if non-blank
;  305          0818  5
;  306          0819  5                VOLUME_PROT[GROUP_WRITE] = 1;
;  307          0820  5
;  308          0821  5                ! if the 1st char is a digit then no group access
;  309          0822  5
;  310          0823  5                IF .(.P)<0, 8> LSS 'A'
;  311          0824  5                THEN VOLUME_PROT[GROUP_READ] = 1;
;  312          0825  5
;  313          0826  4                END;
;  314          0827  4
;  315          0828  3            END;
;  316          0829  3
;  317          0830  3        END
;  318          0831  3
;  319          0832  3        ! If there is no VMS protection but was pre ANSI Label Standard
;  320          0833  3        ! version 4 and the tape was created on another DEC operating
;  321          0834  3        ! system that is it has D% information. Then require priviledges
;  322          0835  3        ! to mount the tape.
;  323          0836  3
;  324          0837  2        ELSE
;  325          0838  2        BEGIN
;  326          0839  3           IF .(VOL1[VL1$T_VOLOWNER])<0,16> NEQ 'D%'
;  327          0840  3                THEN RETURN TRUE
;  328          0841  3                ELSE RETURN FALSE;
;  329          0842  2        END;
;  330          0843  2
;  331          0844  2    RETURN TRUE;
;  332          0845  1    END;                                    ! end of routine TAPE_OWN_PRO


                                    00FC 00000          .ENTRY  TAPE_OWN_PROT, Save R2,R3,R4,R5,R6,R7       ; 0642
                          57 00000000G  00   9E 00002   MOVAB   LIB$CVT_OTB, R7                             ;
                          5E            0C   C2 00009   SUBL2   #12, SP                                     ;
                          50      10    AC   D0 0000C   MOVL    VOL1, R0                                    ; 0712
                          34      4F    A0   91 00010   CMPB    79(R0), #52                                 ;
                                        78   13 00014   BEQL    9$                                          ;
00432544    8F      25    A0          18    00   ED 00016   CMPZV   #0, #24, 37(R0), #4400452               ; 0717
                                        03   13 00020   BEQL    1$                                          ;
                                  00A0  31   00022   BRW     12$                                         ;
                          56      28    A0   9E 00025 1$:   MOVAB   40(R0), P                              ; 0723
                          20            66   91 00029   CMPB    (P), #32                                    ; 0727
                                        23   13 0002C   BEQL    3$                                          ;
                    04    AE          66   05   28 0002E   MOVC3   #5, (P), CONV_BUF                       ; 0733
                                 41   8F   66   91 00033   CMPB    (P), #65                                    ; 0737
                                        05   1F 00037   BLSSU   2$                                          ;
                    04    AE          66   11   83 00039   SUBB3   #17, (P), CONV_BUF                       ; 0738
                                        5E   DD 0003E 2$:   PUSHL   SP                                      ; 0742
                                  08    AE   9F 00040   PUSHAB  CONV_BUF                                    ;
                                        05   DD 00043   PUSHL   #5                                         ;
                          67            03   FB 00045   CALLS   #3, LIB$CVT_OTB                           ;
                          30            50   E9 00048   BLBC    R0, 5$                                  ;
```

```
  04   BC        10          10    6E F0 0004B      INSV    VALUE, #16, #16, @VOLUIC          ; 0746
                             56    05 C0 00051 3$:  ADDL2   #5, P                             ; 0751
                             20    66 91 00054      CMPB    (P), #32                          ; 0755
                                   28 13 00057      BEQL    7$
            04   AE          66    05 28 00059      MOVC3   #5, (P), CONV_BUF                 ; 0761
                        41   8F    66 91 0005E      CMPB    (P), #65                          ; 0765
                                   05 1F 00062      BLSSU   4$
            04   AE          66    11 83 00064      SUBB3   #17, (P), CONV_BUF                ; 0766
                                   5E DD 00069 4$:  PUSHL   SP                                ; 0770
                              08   AE 9F 0006B      PUSHAB  CONV_BUF
                                   05 DD 0006E      PUSHL   #5
                             67    03 FB 00070      CALLS   #3, LIB$CVT_OTB
                             07    50 E8 00073      BLBS    R0, 6$
                        04   BC        0C    AC D0 00076      MOVL    PROCESS_UIC, @VOLUIC    ; 0776
                                   54 11 0007B 5$:  BRB     14$                               ; 0777
                        04   BC    6E B0 0007D 6$:  MOVW    VALUE, @VOLUIC                    ; 0782
                             56    04 C2 00081 7$:  SUBL2   #4, P                             ; 0790
                 76          0A    20 3B 00084      SKPC    #32, #10, -(P)                    ; 0794
                                   02 12 00088      BNEQ    8$
                                   51 D4 0008A      CLRL    R1
                                   51 D5 0008C 8$:  TSTL    R1
                                   3D 13 0008E 9$:  BEQL    13$
                        08   BC  2000    8F A8 00090      BISW2   #8192, @VOLUME_PROT         ; 0800
                        41   8F    66 91 00096      CMPB    (P), #65                          ; 0804
                                   06 1E 0009A      BGEQU   10$
                        08   BC  1000    8F A8 0009C      BISW2   #4096, @VOLUME_PROT         ; 0805
                             56    05 C0 000A2 10$: ADDL2   #5, P                             ; 0809
                 66          05    20 3B 000A5      SKPC    #32, #5, (P)                      ; 0813
                                   02 12 000A9      BNEQ    11$
                                   51 D4 000AB      CLRL    R1
                                   51 D5 000AD 11$: TSTL    R1
                                   1C 13 000AF      BEQL    13$
                        08   BC  0200    8F A8 000B1      BISW2   #512, @VOLUME_PROT          ; 0819
                        41   8F    66 91 000B7      CMPB    (P), #65                          ; 0823
                                   10 1E 000BB      BGEQU   13$
                        08   BC  0100    8F A8 000BD      BISW2   #256, @VOLUME_PROT          ; 0824
                                   08 11 000C3      BRB     13$                               ; 0717
                      2544   8F    25    A0 B1 000C5 12$: CMPW    37(R0), #9540               ; 0839
                                   04 13 000CB      BEQL    14$
                             50    01 D0 000CD 13$: MOVL    #1, R0                            ; 0844
                                      04 000D0      RET
                                   50 D4 000D1 14$: CLRL    R0                                ; 0845
                                      04 000D3      RET

; Routine Size:  212 bytes,    Routine Base:  $CODE$ + 000C

;  333         0846  1
```

```
;  335     0847  1 GLOBAL ROUTINE PROCESS_VOL2_LABEL ( VOLUIC, VOLUME_PROT : REF BITVECTOR[%BPVAL],
;  336     0848  1                                      PROCESS_UIC, VOL2 ) =
;  337     0849  1
;  338     0850  1 !++
;  339     0851  1 !
;  340     0852  1 ! FUNCTIONAL DESCRIPTION:
;  341     0853  1 !       This routine determines the tape_owner and protection of the volume.
;  342     0854  1 !       It uses the VOL2 label to intepret the VMS specified/formatted
;  343     0855  1 !       protection of this volume. This protection used to exist in the
;  344     0856  1 !       OWNER IDENTIFIER field of the VOL1 label. We have moved it into
;  345     0857  1 !       this label because of changes which will be adopted in the
;  346     0858  1 !       upcoming (version 4) ANSI MAGNETIC TAPE STANDARD
;  347     0859  1 !
;  348     0860  1 ! CALLING SEQUENCE:
;  349     0861  1 !       PROCESS_VOL2_LABEL ( ARG1, ARG2, ARG3, ARG4 )
;  350     0862  1 !
;  351     0863  1 ! INPUT PARAMETERS:
;  352     0864  1 !       ARG1 -  Address of area to store the volume uic
;  353     0865  1 !       ARG2 -  Address of area to store the volume protection
;  354     0866  1 !       ARG3 -  Process UIC
;  355     0867  1 !       ARG4 -  Address of ANSI VOL1 label
;  356     0868  1 !
;  357     0869  1 ! IMPLICIT INPUTS:
;  358     0870  1 !       NONE
;  359     0871  1 !
;  360     0872  1 ! OUTPUT PARAMETERS:
;  361     0873  1 !       NONE
;  362     0874  1 !
;  363     0875  1 ! IMPLICIT OUTPUTS:
;  364     0876  1 !       VOLUME_UIC - owning uic of tape
;  365     0877  1 !       VOLUME_PROT -  tape protection
;  366     0878  1 !
;  367     0879  1 ! ROUTINE VALUE:
;  368     0880  1 !       TRUE - Field was blank or was specied in VMS format
;  369     0881  1 !       FALSE - Field was non-blank and not VMS format
;  370     0882  1 !
;  371     0883  1 ! SIDE EFFECTS:
;  372     0884  1 !       NONE
;  373     0885  1 !
;  374     0886  1 ! USER ERRORS:
;  375     0887  1 !       NONE
;  376     0888  1 !
;  377     0889  1 !--
;  378     0890  1
;  379     0891  2     BEGIN
;  380     0892  2
;  381     0893  2     BIND
;  382     0894  2         VOLUME_UIC = .VOLUIC;               ! Address of volume uic
;  383     0895  2
;  384     0896  2     MAP
;  385     0897  2         VOL2    : REF BBLOCK;               ! Address of VOL2 label
;  386     0898  2
;  387     0899  2     LOCAL
;  388     0900  2         CONV_BUF        : VECTOR [6, BYTE], ! buffer used for converting UIC
;  389     0901  2         VALUE,                              ! used to hold parital UIC's
;  390     0902  2         P;                                  ! ptr into VOL2 owner field
;  391     0903  2
```

```
;  392     0904  2     ! bit numbers for different protections
;  393     0905  2     !
;  394     0906  2     LITERAL
;  395     0907  2         WORLD_WRITE = 13,
;  396     0908  2         WORLD_READ  = 12,
;  397     0909  2         GROUP_WRITE = 9,
;  398     0910  2         GROUP_READ  = 8;
;  399     0911  2
;  400     0912  2     ! if ANSI tape produced by VAX system, decode tape owner field
;  401     0913  2
;  402     0914  2     IF .(VOL2[VL2$T_VOLOWNER])<0, 24> EQL 'D%C'
;  403     0915  2       THEN
;  404     0916  3         BEGIN
;  405     0917  3
;  406     0918  3         ! set up the pointer to begining of tape owner field
;  407     0919  3
;  408     0920  3         P = VOL2[VL2$T_VOLOWNER] + 3;
;  409     0921  3
;  410     0922  3         ! test for encoding
;  411     0923  3
;  412     0924  3         IF .(.P)<0, 8> NEQ ' '
;  413     0925  3         THEN
;  414     0926  4             BEGIN
;  415     0927  4
;  416     0928  4             ! move the UIC group field from the VOL2 label to the buffer
;  417     0929  4
;  418     0930  4             CH$MOVE(6, .P, CONV_BUF);
;  419     0931  4
;  420     0932  4             ! remove overlay encoding
;  421     0933  4
;  422     0934  4             IF .(.P)<0, 8> GEQ 'A'
;  423     0935  4             THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
;  424     0936  4
;  425     0937  4             ! convert to ASCII to binary exit with failure not a VMS tape
;  426     0938  4
;  427     0939  4             IF NOT LIB$CVT_OTB(6, CONV_BUF, VALUE) THEN RETURN FALSE;
;  428     0940  4
;  429     0941  4             ! fill in the UIC group field
;  430     0942  4
;  431     0943  4             VOLUME_UIC<16, 16> = .VALUE<0, 16>;
;  432     0944  3             END;
;  433     0945  3
;  434     0946  3         ! point to UIC member field
;  435     0947  3
;  436     0948  3         P = .P + 6;
;  437     0949  3
;  438     0950  3         ! test for encoding
;  439     0951  3
;  440     0952  3         IF .(.P)<0, 8> NEQ ' '
;  441     0953  3         THEN
;  442     0954  4             BEGIN
;  443     0955  4
;  444     0956  4             ! move member number into convert buffer
;  445     0957  4
;  446     0958  4             CH$MOVE(6, .P, CONV_BUF);
;  447     0959  4
;  448     0960  4             ! remove overlay encoding
```

```
;  449          0961  4
;  450          0962  4              IF .(.P)<0, 8> GEQ 'A'
;  451          0963  4              THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
;  452          0964  4
;  453          0965  4              ! convert to ASCII to binary exit when failure not a VAX tape
;  454          0966  4
;  455          0967  4              IF NOT LIB$CVT_OTB(6, CONV_BUF, VALUE)
;  456          0968  4              THEN
;  457          0969  5                  BEGIN
;  458          0970  5
;  459          0971  5                  ! patch up UIC before returning
;  460          0972  5
;  461          0973  5                  VOLUME_UIC = .PROCESS_UIC;
;  462          0974  5                  RETURN FALSE;
;  463          0975  4                  END;
;  464          0976  4
;  465          0977  4              ! fill in the UIC member field
;  466          0978  4
;  467          0979  4              VOLUME_UIC <0, 16> = .VALUE<0, 16>;
;  468          0980  3              END;
;  469          0981  3
;  470          0982  3          ! Now tape_prot must be decoded if both group and member are blank then
;  471          0983  3          ! all privileges granted
;  472          0984  3          !
;  473          0985  3          ! pointer to group uic
;  474          0986  3
;  475          0987  3          P = .P - 6;
;  476          0988  3
;  477          0989  3          ! if field is not blank, then there is a protection mask
;  478          0990  3
;  479          0991  3          IF NOT CH$FAIL(CH$FIND_NOT_CH(12, .P, ' '))
;  480          0992  3          THEN
;  481          0993  4              BEGIN
;  482          0994  4
;  483          0995  4              ! any mask means no world write
;  484          0996  4
;  485          0997  4              VOLUME_PROT[WORLD_WRITE] = 1;
;  486          0998  4
;  487          0999  4              ! if the 1st char is a digit then no world access
;  488          1000  4
;  489          1001  4              IF .(.P)<0, 8> LSS 'A'
;  490          1002  4              THEN VOLUME_PROT[WORLD_READ] = 1;
;  491          1003  4
;  492          1004  4              ! pointer to member field
;  493          1005  4
;  494          1006  4              P = .P + 6;
;  495          1007  4
;  496          1008  4              ! test for group rights.  all spaces means both read and write
;  497          1009  4
;  498          1010  4              IF NOT CH$FAIL(CH$FIND_NOT_CH(6, .P, ' '))
;  499          1011  4              THEN
;  500          1012  5                  BEGIN
;  501          1013  5
;  502          1014  5                  ! write protection against group if non-blank
;  503          1015  5
;  504          1016  5                  VOLUME_PROT[GROUP_WRITE] = 1;
;  505          1017  5
```

```
;  506      1018  5                     ! if the 1st char is a digit then no group access
;  507      1019  5
;  508      1020  5                     IF .(.P)<0, 8> LSS 'A'
;  509      1021  5                     THEN VOLUME_PROT[GROUP_READ] = 1;
;  510      1022  5
;  511      1023  4                     END;
;  512      1024  4
;  513      1025  3                 END;
;  514      1026  3
;  515      1027  2             END;
;  516      1028  2
;  517      1029  2         RETURN TRUE;
;  518      1030  1         END;                                        ! end of routine TAPE_OWN_PRO



                                            00FC 00000          .ENTRY  PROCESS_VOL2_LABEL, Save R2,R3,R4,R5,R6,R7  ; 0847
                          57 00000000G  00    9E 00002          MOVAB   LIB$CVT_OTB, R7                             ;
                          5E            0C    C2 00009          SUBL2   #12, SP                                     ;
                          56        10  AC    D0 0000C          MOVL    VOL2, R6                                    ; 0914
00432544   8F     04   A6     18            00    ED 00010          CMPZV   #0, #24, 4(R6), #4400452                    ;
                          03            13 0001A          BEQL    1$                                          ;
                                     009D    31 0001C          BRW     10$                                         ;
                          56            07    C0 0001F 1$:      ADDL2   #7, P                                       ; 0920
                          20            66    91 00022          CMPB    (P), #32                                    ; 0924
                          23            13 00025          BEQL    3$                                          ;
                 04   AE        66            06    28 00027          MOVC3   #6, (P), CONV_BUF                           ; 0930
                          41   8F       66    91 0002C          CMPB    (P), #65                                    ; 0934
                          05            1F 00030          BLSSU   2$                                          ;
                 04   AE        66            11    83 00032          SUBB3   #17, (P), CONV_BUF                          ; 0935
                                        5E    DD 00037 2$:      PUSHL   SP                                          ; 0939
                                    08  AE    9F 00039          PUSHAB  CONV_BUF                                    ;
                                        06    DD 0003C          PUSHL   #6                                          ;
                          67            03    FB 0003E          CALLS   #3, LIB$CVT_OTB                             ;
                          7C            50    E9 00041          BLBC    R0, 11$                                     ;
     04   BC          10        10            6E    F0 00044          INSV    VALUE, #16, #16, @VOLUIC                    ; 0943
                          56            06    C0 0004A 3$:      ADDL2   #6, P                                       ; 0948
                          20            66    91 0004D          CMPB    (P), #32                                    ; 0952
                          28            13 00050          BEQL    6$                                          ;
                 04   AE        66            06    28 00052          MOVC3   #6, (P), CONV_BUF                           ; 0958
                          41   8F       66    91 00057          CMPB    (P), #65                                    ; 0962
                          05            1F 0005B          BLSSU   4$                                          ;
                 04   AE        66            11    83 0005D          SUBB3   #17, (P), CONV_BUF                          ; 0963
                                        5E    DD 00062 4$:      PUSHL   SP                                          ; 0967
                                    08  AE    9F 00064          PUSHAB  CONV_BUF                                    ;
                                        06    DD 00067          PUSHL   #6                                          ;
                          67            03    FB 00069          CALLS   #3, LIB$CVT_OTB                             ;
                          07            50    E8 0006C          BLBS    R0, 5$                                      ;
                      04   BC       0C  AC    D0 0006F          MOVL    PROCESS_UIC, @VOLUIC                        ; 0973
                          4A            11 00074          BRB     11$                                         ; 0974
                      04   BC           6E    B0 00076 5$:      MOVW    VALUE, @VOLUIC                              ; 0979
                          56            05    C2 0007A 6$:      SUBL2   #5, P                                       ; 0987
                 76           0C            20    3B 0007D          SKPC    #32, #12, -(P)                              ; 0991
                          02            12 00081          BNEQ    7$                                          ;
                          51            D4 00083          CLRL    R1                                          ;
```

```
                                  51  D5 00085 7$:     TSTL    R1                           ;
                                  33  13 00087         BEQL    10$                          ;
           08    BC        2000   8F  A8 00089         BISW2   #8192, @VOLUME_PROT          ; 0997
           41    8F               66  91 0008F         CMPB    (P), #65                     ; 1001
                                  06  1E 00093         BGEQU   8$                           ;
           08    BC        1000   8F  A8 00095         BISW2   #4096, @VOLUME_PROT          ; 1002
                 56               06  C0 0009B 8$:     ADDL2   #6, P                        ; 1006
     66          06               20  3B 0009E         SKPC    #32, #6, (P)                 ; 1010
                                  02  12 000A2         BNEQ    9$                           ;
                                  51  D4 000A4         CLRL    R1                           ;
                                  51  D5 000A6 9$:     TSTL    R1                           ;
                                  12  13 000A8         BEQL    10$                          ;
           08    BC        0200   8F  A8 000AA         BISW2   #512, @VOLUME_PROT           ; 1016
           41    8F               66  91 000B0         CMPB    (P), #65                     ; 1020
                                  06  1E 000B4         BGEQU   10$                          ;
           08    BC        0100   8F  A8 000B6         BISW2   #256, @VOLUME_PROT           ; 1021
                 50               01  D0 000BC 10$:    MOVL    #1, R0                       ; 1029
                                      04 000BF         RET                                  ;
                                  50  D4 000C0 11$:    CLRL    R0                           ; 1030
                                      04 000C2         RET                                  ;

; Routine Size:  195 bytes,    Routine Base:  $CODE$ + 00E0
```

```
;  520          1031  1 GLOBAL ROUTINE CHECK_PROT(VOL_PROT,VOL_UIC, PROCUIC,WRT_RING) =
;  521          1032  1
;  522          1033  1 !++
;  523          1034  1 !
;  524          1035  1 ! FUNCTIONAL DESCRIPTION:
;  525          1036  1 !       this routine check VMS volume protection
;  526          1037  1 !
;  527          1038  1 ! CALLING SEQUENCE:
;  528          1039  1 !       CHECK_PROT(ARG1,ARG2,ARG3,ARG4)
;  529          1040  1 !
;  530          1041  1 ! INPUT PARAMETERS:
;  531          1042  1 !       ARG1 - volume protection
;  532          1043  1 !       ARG2 - volume owner UIC
;  533          1044  1 !       ARG3 - Process UIC
;  534          1045  1 !       ARG4 - Write ring status
;  535          1046  1 !
;  536          1047  1 ! IMPLICIT INPUTS:
;  537          1048  1 !       NONE
;  538          1049  1 !
;  539          1050  1 ! OUTPUT PARAMETERS:
;  540          1051  1 !       NONE
;  541          1052  1 !
;  542          1053  1 ! IMPLICIT OUTPUTS:
;  543          1054  1 !       NONE
;  544          1055  1 !
;  545          1056  1 ! ROUTINE VALUE:
;  546          1057  1 !       TRUE  - if passes protection
;  547          1058  1 !       FALSE - if does not pass protection
;  548          1059  1 !
;  549          1060  1 ! SIDE EFFECTS:
;  550          1061  1 !       NONE
;  551          1062  1 !
;  552          1063  1 ! USER ERRORS:
;  553          1064  1 !       NONE
;  554          1065  1 !
;  555          1066  1 !--
;  556          1067  1
;  557          1068  2     BEGIN
;  558          1069  2
;  559          1070  2     LOCAL
;  560          1071  2         PROCESS_UIC     : VECTOR [ 2, WORD ],   ! the process UIC
;  561          1072  2         WRITE_RING      : BITVECTOR [ 1 ];      ! is this a write mount
;  562          1073  2
;  563          1074  2     MAP
;  564          1075  2         VOL_PROT        : REF BITVECTOR,
;  565          1076  2         VOL_UIC         : REF VECTOR [ 2, WORD ],
;  566          1077  2         WRT_RING        : BITVECTOR [ 1 ];      ! is this a write mount
;  567          1078  2
;  568          1079  2     EXTERNAL
;  569          1080  2         EXE$GL_SYSUIC   : REF BBLOCK ADDRESSING_MODE ( ABSOLUTE );
;  570          1081  2
;  571          1082  2     LITERAL
;  572          1083  2         NOT_GROUP_READ  = 8,  ! the group read disable bit
;  573          1084  2         NOT_GROUP_WRITE = 9,  ! the group write disable bit
;  574          1085  2         NOT_WORLD_READ  = 12, ! the world read disable bit
;  575          1086  2         NOT_WORLD_WRITE = 13; ! the world write disable bit
;  576          1087  2
```

```
;  577        1088  2      ! get the process UIC
;  578        1089  2
;  579        1090  2      PROCESS_UIC <0,32> = .PROCUIC;
;  580        1091  2
;  581        1092  2      ! get the write protectio of teh tape
;  582        1093  2
;  583        1094  2      WRITE_RING [0] = NOT .WRT_RING [0];
;  584        1095  2
;  585        1096  2      !  check if the user has write access to the tape
;  586        1097  2
;  587        1098  2      IF ( .PROCESS_UIC [ 1 ] LEQ .EXE$GL_SYSUIC ) OR        ! the user's UIC has a
;  588        1099  2                                                             ! system group number
;  589        1100  2
;  590        1101  2         ( NOT .VOL_PROT [ NOT_WORLD_WRITE ] ) OR            ! the tape is world write
;  591        1102  2
;  592        1103  3         (( NOT .VOL_PROT [ NOT_WORLD_READ] ) AND            ! tape is world read and
;  593        1104  2          ( NOT .WRITE_RING [ 0 ]         )) OR              !  read only mount
;  594        1105  2
;  595        1106  3         (( .PROCESS_UIC [ 1 ] EQL .VOL_UIC [ 1 ] ) AND    ! (tape's and user's
;  596        1107  4          (( NOT .VOL_PROT [ NOT_GROUP_WRITE ] ) OR         ! group match) and
;  597        1108  5           (( NOT .VOL_PROT [ NOT_GROUP_READ ] ) AND        ! ((tape is group write)
;  598        1109  4            ( NOT .WRITE_RING [ 0 ]               ))OR      ! or (tape is group read
;  599        1110  4                                                           !   and read only mount)
;  600        1111  3            ( .PROCESS_UIC [ 0 ] EQL .VOL_UIC [ 0 ] )))   ! or (member UIC match))
;  601        1112  3
;  602        1113  2         THEN RETURN TRUE;
;  603        1114  2
;  604        1115  3      IF (( .VOL_PROT [ NOT_WORLD_WRITE] ) AND                ! user does not have write
;  605        1116  2          ( NOT .VOL_PROT [ NOT_WORLD_READ])) OR              ! acess but does have read
;  606        1117  2
;  607        1118  3         (( .VOL_PROT [ NOT_GROUP_WRITE]) AND                ! or the same for group
;  608        1119  3          ( NOT .VOL_PROT [ NOT_GROUP_READ]))                ! they have read access
;  609        1120  3
;  610        1121  2          THEN                                               ! Then allow mount but
;  611        1122  3             BEGIN                                           ! set the tape write lock
;  612        1123  3             WRT_RING [ 0 ] = 0;
;  613        1124  3             RETURN TRUE;
;  614        1125  2             END;
;  615        1126  2
;  616        1127  2      !  user does not have needed priviledges return error
;  617        1128  2      !
;  618        1129  2      RETURN FALSE;
;  619        1130  2
;  620        1131  1      END;                                    ! end of Routine CHECK_PROT


                                                                  .EXTRN  EXE$GL_SYSUIC

                                                    0000 00000    .ENTRY  CHECK_PROT, Save nothing                    ; 1031
                                           0C    AC   DD 00002    PUSHL   PROCUIC                                      ; 1090
          50        10  AC              01    00   EF 00005    EXTZV   #0, #1, WRT_RING, R0                         ; 1094
                                        50    50   92 0000B    MCOMB   R0, R0                                       ;
          51            01              00    50   F0 0000E    INSV    R0, #0, #1, WRITE_RING                       ;
00000000G 9F        02  AE              10    00   ED 00013    CMPZV   #0, #16, PROCESS_UIC+2, @#EXE$GL_SYSUIC     ; 1098
                                              42   15 0001D    BLEQ    6$                                           ;
                        3D        04    BC    0D   E1 0001F    BBC     #13, @VOL_PROT, 6$                           ; 1101
```

```
                03      04  BC              0C  E0 00024          BBS     #12, @VOL_PROT, 1$                      ; 1103
                                            51  E9 00029          BLBC    WRITE_RING, 6$                          ; 1104
                        35  50          08  AC  D0 0002C  1$:     MOVL    VOL_UIC, R0                             ; 1106
                    02  A0          02  AE  B1 00030          CMPW    PROCESS_UIC+2, 2(R0)                    ;
                                            12  12 00035          BNEQ    3$                                      ;
                25      04  BC              09  E1 00037          BBC     #9, @VOL_PROT, 6$                       ; 1107
                03      04  BC              08  E0 0003C          BBS     #8, @VOL_PROT, 2$                       ; 1108
                        1D  51              51  E9 00041          BLBC    WRITE_RING, 6$                          ; 1109
                        60  6E              6E  B1 00044  2$:     CMPW    PROCESS_UIC, (R0)                       ; 1111
                                            18  13 00047          BEQL    6$                                      ;
                05      04  BC              0D  E1 00049  3$:     BBC     #13, @VOL_PROT, 4$                      ; 1115
                0A      04  BC              0C  E1 0004E          BBC     #12, @VOL_PROT, 5$                      ; 1116
                0D      04  BC              09  E1 00053  4$:     BBC     #9, @VOL_PROT, 7$                       ; 1118
                08      04  BC              08  E0 00058          BBS     #8, @VOL_PROT, 7$                       ; 1119
                    10  AC              01  8A 0005D  5$:     BICB2   #1, WRT_RING                            ; 1123
                        50              01  D0 00061  6$:     MOVL    #1, R0                                  ; 1124
                                            04  00064          RET                                             ;
                                        50  D4 00065  7$:     CLRL    R0                                      ; 1129
                                            04  00067          RET                                             ; 1131

; Routine Size:  104 bytes,    Routine Base:  $CODE$ + 01A3

;  621          1132  1
```

```
;  623          1133  1 GLOBAL ROUTINE FORMAT_VOLOWNER(VOL_LABEL,OWNER,PROTECTION) : NOVALUE  =
;  624          1134  1
;  625          1135  1 !++
;  626          1136  1 !
;  627          1137  1 ! FUNCTIONAL DESCRIPTION:
;  628          1138  1 !       This routine formats the volume owner field in the VOL2 label
;  629          1139  1 !
;  630          1140  1 ! CALLING SEQUENCE:
;  631          1141  1 !       FORMAT_VOLOWNER(ARG1,ARG2,ARG3)
;  632          1142  1 !
;  633          1143  1 ! INPUT PARAMETERS:
;  634          1144  1 !       ARG1 - address of VOL2 label
;  635          1145  1 !       ARG2 - owner of tape
;  636          1146  1 !       ARG3 - tape protection
;  637          1147  1 !
;  638          1148  1 ! IMPLICIT INPUTS:
;  639          1149  1 !       D%C preinitialized
;  640          1150  1 !
;  641          1151  1 ! OUTPUT PARAMETERS:
;  642          1152  1 !       none
;  643          1153  1 !
;  644          1154  1 ! IMPLICIT OUTPUTS:
;  645          1155  1 !       none
;  646          1156  1 !
;  647          1157  1 ! ROUTINE VALUE:
;  648          1158  1 !       none
;  649          1159  1 !
;  650          1160  1 ! SIDE EFFECTS:
;  651          1161  1 !       none
;  652          1162  1 !
;  653          1163  1 ! USER ERRORS:
;  654          1164  1 !       none
;  655          1165  1 !
;  656          1166  1 !--
;  657          1167  1
;  658          1168  2 BEGIN
;  659          1169  2
;  660          1170  2 MAP
;  661          1171  2         VOL_LABEL       : REF BBLOCK,   ! address of VOL1 label
;  662          1172  2         PROTECTION      : BITVECTOR;    ! protection to be encoded on tape
;  663          1173  2
;  664          1174  2 LOCAL
;  665          1175  2         DESCR           : VECTOR [2],   ! descriptor
;  666          1176  2         P;                              ! pointer
;  667          1177  2
;  668          1178  2 LITERAL
;  669          1179  2         WORLD_WRITE = 13,
;  670          1180  2         WORLD_READ = 12,
;  671          1181  2         GROUP_WRITE = 9,
;  672          1182  2         GROUP_READ = 8;
;  673          1183  2
;  674          1184  2
;  675          1185  2 ! first convert binary owner to ASCII
;  676          1186  2 !
;  677          1187  2 DESCR[0] = 12;
;  678          1188  2 DESCR[1] = VOL_LABEL[VL2$T_VOLOWNER] + 3;
;  679        P 1189  2 $FAO(
```

```
;  680        P 1190  2          DESCRIPTOR('!6OW!6OW'), 0,
;  681        P 1191  2          DESCR[0],
;  682          1192  2          .OWNER<16,16>,.OWNER<0,16>);
;  683          1193  2
;  684          1194  2  ! now format protection
;  685          1195  2  !
;  686          1196  2  IF NOT .PROTECTION[GROUP_READ] OR NOT .PROTECTION[WORLD_READ] THEN
;  687          1197  3      BEGIN
;  688          1198  3      P = VOL_LABEL[VL2$T_VOLOWNER] + 9;
;  689          1199  3      (.P)<0,8> = .(.P)<0,8> + ('A' - '0');
;  690          1200  2      END;
;  691          1201  2
;  692          1202  2  ! now if group can also write, blank fill member field
;  693          1203  2  !
;  694          1204  2  IF NOT .PROTECTION[GROUP_WRITE] THEN CH$FILL(' ',6,VOL_LABEL[VL2$T_VOLOWNER] + 9);
;  695          1205  2
;  696          1206  2  IF NOT .PROTECTION[WORLD_READ] THEN
;  697          1207  3      BEGIN
;  698          1208  3      P = VOL_LABEL[VL2$T_VOLOWNER] + 3;
;  699          1209  3      (.P)<0,8> = .(.P)<0,8> + ('A' - '0');
;  700          1210  2      END;
;  701          1211  2
;  702          1212  2  IF NOT .PROTECTION[WORLD_WRITE] THEN CH$FILL(' ',12,VOL_LABEL[VL2$T_VOLOWNER] + 3);
;  703          1213  1  END;                                    !end of routine FORMAT_VOLOWNER


                        57  4F  36  21  57  4F  36  21  0020B P.AAB:  .ASCII  \!6OW!6OW\                          ;
                                                        00213        .BLKB   1
                                                00000008  00214 P.AAA:  .LONG   8                                  ;
                                                00000000' 00218        .ADDRESS P.AAB                             ;

                                                                       .EXTRN  SYS$FAO

                                                    00FC  00000        .ENTRY  FORMAT_VOLOWNER, Save R2,R3,R4,R5,R6,R7 ; 1133
                                5E                  04   C2  00002     SUBL2   #4, SP                             ;
                                                    0C   DD  00005     PUSHL   #12                                ; 1187
                                57                  04   AC  D0  00007 MOVL    VOL_LABEL, R7                      ; 1188
                             04 AE                  07   A7  9E  0000B MOVAB   7(R7), DESCR+4                     ;
                                7E                  08   AC  3C  00010 MOVZWL  OWNER, -(SP)                      ; 1192
                                7E                  0A   AC  3C  00014 MOVZWL  OWNER+2, -(SP)                     ;
                                                    08   AE  9F  00018 PUSHAB  DESCR                               ;
                                                    7E   D4  0001B     CLRL    -(SP)                              ;
                                                    D8   AF  9F  0001D PUSHAB  P.AAA                              ;
                      00000000G 00                  05   FB  00020     CALLS   #5, SYS$FAO                        ;
                                05                  0D   AC  E9  00027 BLBC    PROTECTION+1, 1$                   ; 1196
                          07    0D AC               04   E0  0002B     BBS     #4, PROTECTION+1, 2$               ;
                                56                  0D   A7  9E  00030 1$:     MOVAB   13(R7), P                  ; 1198
                                66                  11   80  00034     ADDB2   #17, (P)                           ; 1199
                          07    0D AC               01   E0  00037 2$:     BBS     #1, PROTECTION+1, 3$               ; 1204
                    06    20    6E                  00   2C  0003C     MOVC5   #0, (SP), #32, #6, 13(R7)          ;
                                                    0D   A7      00041
                          07    0D AC               04   E0  00043 3$:     BBS     #4, PROTECTION+1, 4$               ; 1206
                                56                  07   A7  9E  00048 MOVAB   7(R7), P                          ; 1208
                                66                  11   80  0004C     ADDB2   #17, (P)                           ; 1209
                          07    0D AC               05   E0  0004F 4$:     BBS     #5, PROTECTION+1, 5$               ; 1212
                    0C    20    6E                  00   2C  00054     MOVC5   #0, (SP), #32, #12, 7(R7)          ;
```

```
                         07    A7      00059                                   ;
                                  04 0005B 5$:     RET                         ; 1213

; Routine Size:  92 bytes,    Routine Base:  $CODE$ + 021C

;   704          1214  1
;   705          1215  1 END
;   706          1216  0 ELUDOM



;                               PSECT SUMMARY
;
;       Name                     Bytes                    Attributes
;
;   $CODE$                           632  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                          Library Statistics
;
;                                         -------- Symbols --------       Pages       Processing
;       File                              Total    Loaded   Percent       Mapped      Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1         18619       13         0        1000         00:01.9



;                              COMMAND QUALIFIERS

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

; Size:          615 code + 17 data bytes
; Run Time:         00:18.2
; Elapsed Time:     00:58.1
; Lines/CPU Min:      4017
; Lexemes/CPU-Min: 26593
; Memory Used:   128 pages
; Compilation Complete
```

FREEPG
LIS

DEACCS
LIS

FIND
LIS

DATECN
LIS

FRMOD1
LIS

CREATE
LIS

GETREQ
LIS

EXPIRE
LIS

CNTRL
LIS

FRMHDR
LIS

HEADER
LIS

COMLABPRC
LIS

ENDVOL
LIS

GETFIB
LIS