```
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
```

```
RRRRRRRR    MM      MM    SSSSSSSS    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
RRRRRRRR    MM      MM    SSSSSSSS    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
RR      RR  MMMM  MMMM  SS              II      NN      NN  PP      PP  UU      UU      TT
RR      RR  MMMM  MMMM  SS              II      NN      NN  PP      PP  UU      UU      TT
RR      RR  MM  MM  MM  SS              II      NNNN    NN  PP      PP  UU      UU      TT
RR      RR  MM  MM  MM  SS              II      NNNN    NN  PP      PP  UU      UU      TT
RRRRRRRR    MM      MM    SSSSSS        II      NN  NN  NN  PPPPPPPP    UU      UU      TT
RRRRRRRR    MM      MM    SSSSSS        II      NN  NN  NN  PPPPPPPP    UU      UU      TT
RR  RR      MM      MM          SS      II      NN    NNNN  PP          UU      UU      TT
RR  RR      MM      MM          SS      II      NN    NNNN  PP          UU      UU      TT
RR    RR    MM      MM          SS      II      NN      NN  PP          UU      UU      TT          ....
RR    RR    MM      MM          SS      II      NN      NN  PP          UU      UU      TT          ....
RR      RR  MM      MM  SSSSSSSS    IIIIII    NN      NN  PP          UUUUUUUUUU      TT          ....
RR      RR  MM      MM  SSSSSSSS    IIIIII    NN      NN  PP          UUUUUUUUUU      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
```

```
;     1          0001  0 %title 'RMSINPUT - Handle RMS File Input'
;     2          0002  0         module rmsinput (
;     3          0003  1                         ident='V04-000') = begin
;     4          0004  1
;     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:     VAX/VMS Analyze Facility, Handle RMS File Input
;    32          0032  1 !
;    33          0033  1 ! Abstract:     This module is responsible for handling file specs from
;    34          0034  1 !               the ANALYZE/RMS_FILE command line, and reading data from
;    35          0035  1 !               file headers and RMS files.
;    36          0036  1 !
;    37          0037  1 !
;    38          0038  1 ! Environment:
;    39          0039  1 !
;    40          0040  1 ! Author: Paul C. Anagnostopoulos, Creation Date: 16 February 1981
;    41          0041  1 !
;    42          0042  1 ! Modified By:
;    43          0043  1 !
;    44          0044  1 !       V03-009 BLS0286         Benn Schreiber          20-MAR-1984
;    45          0045  1 !               Correct changes made in 007.
;    46          0046  1 !
;    47          0047  1 !       V03-008 JWT0164         Jim Teague              10-Mar-1984
;    48          0048  1 !               Change arguments to signal when status is not rms$_nmf.
;    49          0049  1 !
;    50          0050  1 !       V03-007 LJA0114         Laurie J. Anderson      24-Feb-1984
;    51          0051  1 !               Add new related file parsing arguments to LIB$FIND_FILE
;    52          0052  1 !               to make search lists behave properly.
;    53          0053  1 !
;    54          0054  1 !       V03-006 RRB0002         Rowland R. Bradley
;    55          0055  1 !               Add support for the display of journaling information.
;    56          0056  1 !
;    57          0057  1 !       V03-005 PCA1019         Paul C. Anagnostopoulos 23-May-1983
```

```
;    58          0058  1 !                 Remove READ_CHECK and WRITE_CHECK FDL secondaries, as they
;    59          0059  1 !                 have always been bogus.
;    60          0060  1 !
;    61          0061  1 !       V03-004 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
;    62          0062  1 !                 Change the message prefix to ANLRMS$_ to ensure that
;    63          0063  1 !                 message symbols are unique across all ANALYZEs.  This
;    64          0064  1 !                 is necessitated by the new merged message files.
;    65          0065  1 !
;    66          0066  1 !       V03-003 PCA1002         Paul C. Anagnostopoulos 26-Oct-1982
;    67          0067  1 !                 Change the code that generates the FDL FILE primary
;    68          0068  1 !                 so that it uses the new routine ANL$PREPARE_QUOTED_STRING
;    69          0069  1 !                 to put out lines with quoted strings.
;    70          0070  1 !                 Put out the GLOBAL_BUFFER_COUNT secondary for all files,
;    71          0071  1 !                 not just relative and indexed ones.
;    72          0072  1 !
;    73          0073  1 !       V03-002 PCA0030         Paul Anagnostopoulos    24-Mar-1982
;    74          0074  1 !                 Fix error messages so they use the correct STV value.
;    75          0075  1 !
;    76          0076  1 !       V03-001 PCA0009         Paul Anagnostopoulos    16-Mar-1982
;    77          0077  1 !                 If LIB$FIND_FILE returns a bad status, used the resultant
;    78          0078  1 !                 spec in the error message, rather than the wildcard spec.
;    79          0079  1 !--
```

```
;    81          0080  1 %sbttl 'Module Declarations'
;    82          0081  1 !
;    83          0082  1 ! Libraries and Requires:
;    84          0083  1 !
;    85          0084  1
;    86          0085  1 library 'lib';
;    87          0086  1 require 'rmsreq';
;    88          0595  1
;    89          0596  1 !
;    90          0597  1 ! Table of Contents:
;    91          0598  1 !
;    92          0599  1
;    93          0600  1 forward routine
;    94          0601  1         anl$open_next_rms_file,
;    95          0602  1         anl$prolog_info: novalue,
;    96          0603  1         anl$bucket: novalue,
;    97          0604  1         anl$format_file_header: novalue,
;    98          0605  1         anl$fdl_file: novalue;
;    99          0606  1
;   100          0607  1 !
;   101          0608  1 ! External References:
;   102          0609  1 !
;   103          0610  1
;   104          0611  1 external routine
;   105          0612  1         anl$format_error,
;   106          0613  1         anl$format_line,
;   107          0614  1         anl$format_protection_mask,
;   108          0615  1         anl$format_skip,
;   109          0616  1         anl$prepare_quoted_string,
;   110          0617  1         cli$get_value: addressing_mode(general),
;   111          0618  1         lib$find_file: addressing_mode(general),
;   112          0619  1         lib$free_vm: addressing_mode(general),
;   113          0620  1         lib$get_vm: addressing_mode(general),
;   114          0621  1         str$trim: addressing_mode(general);
;   115          0622  1
;   116          0623  1 !
;   117          0624  1 ! Global Variables:
;   118          0625  1 !
;   119          0626  1 ! The following variable is set to point at the file attribute (FAT)
;   120          0627  1 ! structure for the file being analyzed.
;   121          0628  1
;   122          0629  1 global
;   123          0630  1         anl$gl_fat: ref block[,byte],
;   124          0631  1
;   125          0632  1 ! The following variable contains the prolog version number.
;   126          0633  1
;   127          0634  1         anl$gw_prolog: word;
;   128          0635  1
;   129          0636  1 !
;   130          0637  1 ! Own Variables:
;   131          0638  1 !
;   132          0639  1
;   133          0640  1 ! The following data structures are used to access and read records from
;   134          0641  1 ! a file we are to analyze.
;   135          0642  1
;   136          0643  1 own
;   137          0644  1         own_described_buffer(resultant_spec,nam$c_maxrss);
```

```
;  138          0645  1 own
;  139          0646  1         rms_bia: block[xab$c_maxjnlnam,byte],
;  140          0647  1         rms_ata: block[xab$c_maxjnlnam,byte],
;  141          0648  1         rms_aia: block[xab$c_maxjnlnam,byte],
;  142        P 0649  1         rms_xabjnl: $xabjnl(aia=rms_aia,
;  143        P 0650  1                            ais=xab$c_maxjnlnam,
;  144        P 0651  1                            ata=rms_ata,
;  145        P 0652  1                            ats=xab$c_maxjnlnam,
;  146        P 0653  1                            bia=rms_bia,
;  147        P 0654  1                            bis=xab$c_maxjnlnam,
;  148          0655  1                            nxt=0),
;  149          0656  1
;  150          0657  1         rms_xabfhc: $xabfhc(nxt=rms_xabjnl),
;  151          0658  1
;  152          0659  1         rms_xabpro: $xabpro(nxt=rms_xabfhc),
;  153          0660  1
;  154          0661  1         rms_xabdat: $xabdat(nxt=rms_xabpro),
;  155          0662  1
;  156          0663  1         rms_rsa: block[nam$c_maxrss,byte],
;  157        P 0664  1         rms_nam: $nam(esa=rms_rsa,
;  158          0665  1                       ess=nam$c_maxrss),
;  159          0666  1
;  160        P 0667  1         rms_fab: $fab(fac=<bio,get>,
;  161        P 0668  1                       nam=rms_nam,
;  162        P 0669  1                       shr=get,
;  163          0670  1                       xab=rms_xabdat),
;  164          0671  1
;  165          0672  1         rms_rab: $rab(fab=rms_fab);
```

```
;  167          0673  1
;  168          0674  1 %sbttl 'ANL$OPEN_NEXT_RMS_FILE - Right'
;  169          0675  1 !++
;  170          0676  1 ! Functional Description:
;  171          0677  1 !       This routine is called to open the next RMS file we are to analyze.
;  172          0678  1 !       It handles multiple file specs and wildcarding.
;  173          0679  1 !
;  174          0680  1 ! Formal Parameters:
;  175          0681  1 !       opened_spec     Address of descriptor of buffer in which to return
;  176          0682  1 !                       the spec of the file we open.  We set the length.
;  177          0683  1 !
;  178          0684  1 ! Implicit Inputs:
;  179          0685  1 !       global data
;  180          0686  1 !
;  181          0687  1 ! Implicit Outputs:
;  182          0688  1 !       global data
;  183          0689  1 !
;  184          0690  1 ! Returned Value:
;  185          0691  1 !       True if there is another file, false otherwise.
;  186          0692  1 !
;  187          0693  1 ! Side Effects:
;  188          0694  1 !
;  189          0695  1 !--
;  190          0696  1
;  191          0697  1
;  192          0698  2 global routine anl$open_next_rms_file(opened_spec) = begin
;  193          0699  2
;  194          0700  2 bind
;  195          0701  2         opened_spec_dsc = .opened_spec: descriptor;
;  196          0702  2
;  197          0703  2 own
;  198          0704  2         own_described_buffer(wildcard_spec,nam$c_maxrss),
;  199          0705  2         wildcard_context: long initial(0),
;  200          0706  2         get_new_spec : long initial(true);
;  201          0707  2
;  202          0708  2 local
;  203          0709  2         stv: long,
;  204          0710  2         status: long;
;  205          0711  2
;  206          0712  2
;  207          0713  2 ! If the wildcard context is zero, it means this is the first call, or
;  208          0714  2 ! we finished with a file spec on the previous call.  So we must obtain
;  209          0715  2 ! the next file spec from the command line.
;  210          0716  2
;  211          0717  3 if .get_new_spec then (
;  212          0718  3         wildcard_spec[len] = nam$c_maxrss;
;  213          0719  3         status = cli$get_value(describe('file_specs'),wildcard_spec);
;  214          0720  3
;  215          0721  3         ! If there are no more specs, we are all done.
;  216          0722  3
;  217          0723  3         if not .status then
;  218          0724  3                 return false;
;  219          0725  3         str$trim(wildcard_spec,wildcard_spec,wildcard_spec);
;  220          0726  2 );
;  221          0727  2
;  222          0728  2 ! On the other hand, if the previous call is done, we may have just
;  223          0729  2 ! finished processing a file.  Better close it.
```

```
;  224          0730  2
;  225          0731  3 if .rms_fab[fab$w_ifi] nequ 0 then (
;  226          0732  3         status = $close(fab=rms_fab);
;  227          0733  3         check (.status, anlrms$_closein,1,resultant_spec,.status,.rms_fab[fab$l_stv]);
;  228          0734  3
;  229          0735  3         ! We also better flush the bucket cache, in case we cancelled the
;  230          0736  3         ! analysis of this file in the middle.
;  231          0737  3
;  232          0738  3         anl$bucket();
;  233          0739  2 );
```

```
;  235          0740  2 ! We have obtained a wildcard spec from the file parameter.
;  236          0741  2
;  237          0742  2 ! Now we need to find the next file that matches the current wildcard spec.
;  238          0743  2
;  239          0744  2 resultant_spec[len] = nam$c_maxrss;
;  240          0745  2 status = lib$find_file(wildcard_spec,resultant_spec,wildcard_context,describe('.DAT'),
;  241          0746  2                       0, stv, %ref(2) );
;  242          0747  2 str$trim(resultant_spec,resultant_spec,resultant_spec);
;  243          0748  2
;  244          0749  2 ! If we failed to find a file, then reset the wildcard context and call
;  245          0750  2 ! ourselves recursively to process the next file spec.  Also give an
;  246          0751  2 ! error, unless we just plain ran out of files.
;  247          0752  2
;  248          0753  3 if not .status then (
;  249          0754  3         if .status nequ rms$_nmf then
;  250          0755  3                 signal (anlrms$_openin,1,resultant_spec,.status,.stv);
;  251          0756  3         get_new_spec = true;
;  252          0757  3         return anl$open_next_rms_file(opened_spec_dsc);
;  253          0758  2 );
;  254          0759  2
;  255          0760  2 ! Hey, we got a file spec.  Open the file and connect the RAB.
;  256          0761  2
;  257          0762  2 get_new_spec = false;
;  258          0763  2
;  259          0764  2 rms_fab[fab$b_fns] = .resultant_spec[len];
;  260          0765  2 rms_fab[fab$l_fna] = .resultant_spec[ptr];
;  261          0766  2 status = $open(fab=rms_fab);
;  262          0767  2 check (.status, anlrms$_openin,1,resultant_spec,.status,.rms_fab[fab$l_stv]);
;  263          0768  3 if .status then (
;  264          0769  3         status = $connect(rab=rms_rab);
;  265          0770  3         check (.status, anlrms$_openin,1,resultant_spec,.status,.rms_rab[rab$l_stv]);
;  266          0771  2 );
;  267          0772  2
;  268          0773  2 ! If the open failed, then we need to recurse to try the next file.
;  269          0774  2
;  270          0775  2 if not .status then
;  271          0776  2         return anl$open_next_rms_file(opened_spec_dsc);
```

```
;  273          0777  2 ! There are some ODS-1 crocks in the RMS attribute structure returned in the
;  274          0778  2 ! File Header Characteristics XAB.  Let's fix them here so that everyone else
;  275          0779  2 ! can just assume they're OK.  They are the following:
;  276          0780  2 !       A VFC header size of zero is the same as two.
;  277          0781  2 !       A bucket size of zero is the same as one.
;  278          0782  2
;  279          0783  2 if .rms_fab[fab$b_rfm] eqlu fab$c_vfc and .rms_xabfhc[xab$b_hsz] eqlu 0 then
;  280          0784  2         rms_xabfhc[xab$b_hsz] = 2;
;  281          0785  2 rms_xabfhc[xab$b_bkz] = maxu(.rms_xabfhc[xab$b_bkz],1);
;  282          0786  2
;  283          0787  2 ! Set up a global pointer to the attribute structure.  We can now use the
;  284          0788  2 ! FAT$ symbols to reference the information.  This assumes that the
;  285          0789  2 ! information in the XAB is in the same order as it is in the file header.
;  286          0790  2
;  287          0791  2 anl$gl_fat = rms_xabfhc[xab$b_rfo];
;  288          0792  2
;  289          0793  2 ! There is some information in the prolog that should be global because
;  290          0794  2 ! it is used a lot.  Set up that information.
;  291          0795  2
;  292          0796  2 anl$prolog_info();
;  293          0797  2
;  294          0798  2 ! The preceeding hackery has set up the XAB so that information can
;  295          0799  2 ! be easily retrieved.  This is a design flaw.  I should have designed
;  296          0800  2 ! my own structure to contain all this info, along with the prolog
;  297          0801  2 ! version and any other global info about the file.
;  298          0802  2
;  299          0803  2 ! Finally, we have to return the resultant file spec to the caller.
;  300          0804  2
;  301          0805  2 opened_spec_dsc[len] = .resultant_spec[len];
;  302          0806  2 ch$move(.resultant_spec[len],.resultant_spec[ptr], .opened_spec_dsc[ptr]);
;  303          0807  2
;  304          0808  2 return true;
;  305          0809  2
;  306          0810  1 end;

                                                                .TITLE  RMSINPUT RMSINPUT - Handle RMS File Input
                                                                .IDENT  \V04-000\

                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                73  63  65  70  73  5F  65  6C  69  66  00000 P.AAB:  .ASCII  \file_specs\                                ;
                                                        0000A           .BLKB   2
                                                  0000000A  0000C P.AAA:  .LONG   10                                    ;
                                                  00000000' 00010         .ADDRESS P.AAB                                ;
                                              54  41  44  2E  00014 P.AAD:  .ASCII  \.DAT\                               ;
                                                  00000004  00018 P.AAC:  .LONG   4                                     ;
                                                  00000000' 0001C         .ADDRESS P.AAD                                ;

                                                                .PSECT  $OWN$,NOEXE,2

                                                  000000FF  00000 RESULTANT_SPEC:
                                                                .LONG   255
                                                  00000000' 00004         .ADDRESS RESULTANT_SPEC+8                     ;
                                                            00008         .BLKB   255
                                                            00107         .BLKB   1
                                                            00108 RMS_BIA:.BLKB   16
```

```
                    00118 RMS_ATA:.BLKB   16
                    00128 RMS_AIA:.BLKB   16
          22        00138 RMS_XABJNL:
                          .BYTE   34
          3C        00139           .BYTE   60
        0000        0013A           .WORD   0
    00000000        0013C           .LONG   0
        0000        00140           .WORD   0
        0000        00142           .WORD   0
          10        00144           .BYTE   16
          00        00145           .BYTE   0
        0000        00146           .WORD   0
    00000000'       00148           .ADDRESS RMS_BIA
          10        0014C           .BYTE   16
          00        0014D           .BYTE   0
        0000        0014E           .WORD   0
    00000000'       00150           .ADDRESS RMS_AIA
          10        00154           .BYTE   16
          00        00155           .BYTE   0
        0000        00156           .WORD   0
    00000000'       00158           .ADDRESS RMS_ATA
                    0015C           .BLKB   24
          1D        00174 RMS_XABFHC:
                          .BYTE   29
          2C        00175           .BYTE   44
        0000        00176           .WORD   0
    00000000'       00178           .ADDRESS RMS_XABJNL
    00000000#       0017C           .LONG   0[9]
          13        001A0 RMS_XABPRO:
                          .BYTE   19
          58        001A1           .BYTE   88
        0000        001A2           .WORD   0
    00000000'       001A4           .ADDRESS RMS_XABFHC
        FFFF        001A8           .WORD   -1
          00        001AA           .BYTE   0
          00        001AB           .BYTE   0
    0000  0000      001AC           .WORD   0, 0
          00        001B0           .BYTE   0
          00        001B1           .BYTE   0
        0000        001B2           .WORD   0
    00000000        001B4           .LONG   0
    00000000        001B8           .LONG   0
        0000        001BC           .WORD   0
        0000        001BE           .WORD   0
    00000000        001C0           .LONG   0
    00000000        001C4           .LONG   0
                    001C8           .BLKB   48
          12        001F8 RMS_XABDAT:
                          .BYTE   18
          2C        001F9           .BYTE   44
        0000        001FA           .WORD   0
    00000000'       001FC           .ADDRESS RMS_XABPRO
        0000        00200           .WORD   0
        0000        00202           .WORD   0
    00000000#       00204           .LONG   0[2]
    00000000#       0020C           .LONG   0[2]
    00000000        00214           .LONG   0
```

```
00000000  00218          .LONG   0
00000000# 0021C          .LONG   0[2]
          00224 RMS_RSA: .BLKB   255
          00323          .BLKB   1
      02  00324 RMS_NAM: .BYTE   2
      60  00325          .BYTE   96
      00  00326          .BYTE   0
      00  00327          .BYTE   0
00000000  00328          .LONG   0
      00  0032C          .BYTE   0
      00  0032D          .BYTE   0
      FF  0032E          .BYTE   -1
      00  0032F          .BYTE   0
00000000' 00330          .ADDRESS RMS_RSA
00000000  00334          .LONG   0
    0000# 00338          .WORD   0[8]
    0000# 00348          .WORD   0[3]
    0000# 0034E          .WORD   0[3]
00000000  00354          .LONG   0
00000000  00358          .LONG   0
      00  0035C          .BYTE   0
      00  0035D          .BYTE   0
      00  0035E          .BYTE   0
      00  0035F          .BYTE   0
      00  00360          .BYTE   0
      00  00361          .BYTE   0
      00# 00362          .BYTE   0[2]
00000000  00364          .LONG   0
00000000  00368          .LONG   0
00000000  0036C          .LONG   0
00000000  00370          .LONG   0
00000000  00374          .LONG   0
00000000  00378          .LONG   0
00000000# 0037C          .LONG   0[2]
      03  00384 RMS_FAB: .BYTE   3
      50  00385          .BYTE   80
    0000  00386          .WORD   0
00000000  00388          .LONG   0
00000000  0038C          .LONG   0
00000000  00390          .LONG   0
00000000  00394          .LONG   0
    0000  00398          .WORD   0
      22  0039A          .BYTE   34
      02  0039B          .BYTE   2
00000000  0039C          .LONG   0
      00  003A0          .BYTE   0
      00  003A1          .BYTE   0
      00  003A2          .BYTE   C
      02  003A3          .BYTE   2
00000000  003A4          .LONG   0
00000000' 003A8          .ADDRESS RMS_XABDAT
00000000' 003AC          .ADDRESS RMS_NAM
00000000  003B0          .LONG   0
00000000  003B4          .LONG   0
      00  003B8          .BYTE   0
      00  003B9          .BYTE   0
    0000  003BA          .WORD   0
```

```
00000000  003BC          .LONG    0                        ;
    0000  003C0          .WORD    0                        ;
      00  003C2          .BYTE    0                        ;
      00  003C3          .BYTE    0                        ;
00000000  003C4          .LONG    0                        ;
00000000  003C8          .LONG    0                        ;
    0000  003CC          .WORD    0                        ;
      00  003CE          .BYTE    0                        ;
      00  003CF          .BYTE    0                        ;
00000000  003D0          .LONG    0                        ;
      01  003D4 RMS_RAB:.BYTE    1                        ;
      44  003D5          .BYTE    68                       ;
    0000  003D6          .WORD    0                        ;
00000000  003D8          .LONG    0                        ;
00000000  003DC          .LONG    0                        ;
00000000  003E0          .LONG    0                        ;
   0000#  003E4          .WORD    0[3]                     ;
    0000  003EA          .WORD    0                        ;
00000000  003EC          .LONG    0                        ;
    0000  003F0          .WORD    0                        ;
      00  003F2          .BYTE    0                        ;
      00  003F3          .BYTE    0                        ;
    0000  003F4          .WORD    0                        ;
    0000  003F6          .WORD    0                        ;
00000000  003F8          .LONG    0                        ;
00000000  003FC          .LONG    0                        ;
00000000  00400          .LONG    0                        ;
00000000  00404          .LONG    0                        ;
      00  00408          .BYTE    0                        ;
      00  00409          .BYTE    0                        ;
      00  0040A          .BYTE    0                        ;
      00  0040B          .BYTE    0                        ;
00000000  0040C          .LONG    0                        ;
00000000' 00410          .ADDRESS RMS_FAB                  ;
00000000  00414          .LONG    0                        ;
000000FF  00418 WILDCARD_SPEC:
                         .LONG    255                      ;
00000000' 0041C          .ADDRESS WILDCARD_SPEC+8          ;
          00420          .BLKB    255
          0051F          .BLKB    1
00000000  00520 WILDCARD_CONTEXT:
                         .LONG    0                        ;
00000001  00524 GET_NEW_SPEC:
                         .LONG    1                        ;

                         .PSECT   $GLOBAL$,NOEXE,2

          00000 ANL$GL_FAT::
                         .BLKB    4
          00004 ANL$GW_PROLOG::
                         .BLKB    2

                         .EXTRN   ANLRMS$_OK, ANLRMS$_ALLOC
                         .EXTRN   ANLRMS$_ANYTHING
                         .EXTRN   ANLRMS$_BACKUP, ANLRMS$_BKT
                         .EXTRN   ANLRMS$_BKTAREA
                         .EXTRN   ANLRMS$_BKTCHECK
```

```
.EXTRN  ANLRMS$_BKTFLAGS
.EXTRN  ANLRMS$_BKTFREE
.EXTRN  ANLRMS$_BKTKEY, ANLRMS$_BKTLEVEL
.EXTRN  ANLRMS$_BKTNEXT
.EXTRN  ANLRMS$_BKTPTRSIZE
.EXTRN  ANLRMS$_BKTRECID
.EXTRN  ANLRMS$_BKTRECID3
.EXTRN  ANLRMS$_BKTSAMPLE
.EXTRN  ANLRMS$_BKTVBNFREE
.EXTRN  ANLRMS$_BUCKETSIZE
.EXTRN  ANLRMS$_CELL, ANLRMS$_CELLDATA
.EXTRN  ANLRMS$_CELLFLAGS
.EXTRN  ANLRMS$_CHECKHDG
.EXTRN  ANLRMS$_CONTIG, ANLRMS$_CREATION
.EXTRN  ANLRMS$_CTLSIZE
.EXTRN  ANLRMS$_DATAREC
.EXTRN  ANLRMS$_DATABKTVBN
.EXTRN  ANLRMS$_DUMPHEADING
.EXTRN  ANLRMS$_EOF, ANLRMS$_ERRORCOUNT
.EXTRN  ANLRMS$_ERRORNONE
.EXTRN  ANLRMS$_ERRORS, ANLRMS$_EXPIRATION
.EXTRN  ANLRMS$_FILEATTR
.EXTRN  ANLRMS$_FILEHDR
.EXTRN  ANLRMS$_FILEID, ANLRMS$_FILEORG
.EXTRN  ANLRMS$_FILESPEC
.EXTRN  ANLRMS$_FLAG, ANLRMS$_GLOBALBUFS
.EXTRN  ANLRMS$_HEXDATA
.EXTRN  ANLRMS$_HEXHEADING1
.EXTRN  ANLRMS$_HEXHEADING2
.EXTRN  ANLRMS$_IDXAREA
.EXTRN  ANLRMS$_IDXAREAALLOC
.EXTRN  ANLRMS$_IDXAREABKTSZ
.EXTRN  ANLRMS$_IDXAREANEXT
.EXTRN  ANLRMS$_IDXAREANOALLOC
.EXTRN  ANLRMS$_IDXAREAQTY
.EXTRN  ANLRMS$_IDXARERECL
.EXTRN  ANLRMS$_IDXAREAUSED
.EXTRN  ANLRMS$_IDXKEY, ANLRMS$_IDXKEYAREAS
.EXTRN  ANLRMS$_IDXKEYBKTSZ
.EXTRN  ANLRMS$_IDXKEYBYTES
.EXTRN  ANLRMS$_IDXKEY1TYPE
.EXTRN  ANLRMS$_IDXKEYDATAVBN
.EXTRN  ANLRMS$_IDXKEYFILL
.EXTRN  ANLRMS$_IDXKEYFLAGS
.EXTRN  ANLRMS$_IDXKEYKEYSZ
.EXTRN  ANLRMS$_IDXKEYNAME
.EXTRN  ANLRMS$_IDXKEYNEXT
.EXTRN  ANLRMS$_IDXKEYMINREC
.EXTRN  ANLRMS$_IDXKEYNULL
.EXTRN  ANLRMS$_IDXKEYPOSS
.EXTRN  ANLRMS$_IDXKEYROOTLVL
.EXTRN  ANLRMS$_IDXKEYROOTVBN
.EXTRN  ANLRMS$_IDXKEYSEGS
.EXTRN  ANLRMS$_IDXKEYSIZES
.EXTRN  ANLRMS$_IDXPRIMREC
.EXTRN  ANLRMS$_IDXPRIMRECFLAGS
.EXTRN  ANLRMS$_IDXPRIMRECID
```

```
                .EXTRN  ANLRMS$_IDXPRIMRECLEN
                .EXTRN  ANLRMS$_IDXPRIMRECRRV
                .EXTRN  ANLRMS$_IDXPROAREAS
                .EXTRN  ANLRMS$_IDXPROLOG
                .EXTRN  ANLRMS$_IDXREC, ANLRMS$_IDXRECPTR
                .EXTRN  ANLRMS$_IDXSIDR
                .EXTRN  ANLRMS$_IDXSIDRDUPCNT
                .EXTRN  ANLRMS$_IDXSIDRFLAGS
                .EXTRN  ANLRMS$_IDXSIDRRECID
                .EXTRN  ANLRMS$_IDXSIDRPTRFLAGS
                .EXTRN  ANLRMS$_IDXSIDRPTRREF
                .EXTRN  ANLRMS$_INTERCOMMAND
                .EXTRN  ANLRMS$_INTERHDG
                .EXTRN  ANLRMS$_LONGREC
                .EXTRN  ANLRMS$_MAXRECSIZE
                .EXTRN  ANLRMS$_NOBACKUP
                .EXTRN  ANLRMS$_NOEXPIRATION
                .EXTRN  ANLRMS$_NOSPANFILLER
                .EXTRN  ANLRMS$_PERFORM
                .EXTRN  ANLRMS$_PROLOGFLAGS
                .EXTRN  ANLRMS$_PROLOGVER
                .EXTRN  ANLRMS$_PROT, ANLRMS$_RECATTR
                .EXTRN  ANLRMS$_RECFMT, ANLRMS$_RECLAIMBKT
                .EXTRN  ANLRMS$_RELBUCKET
                .EXTRN  ANLRMS$_RELEOFVBN
                .EXTRN  ANLRMS$_RELMAXREC
                .EXTRN  ANLRMS$_RELPROLOG
                .EXTRN  ANLRMS$_RELIAB, ANLRMS$_REVISION
                .EXTRN  ANLRMS$_STATHDG
                .EXTRN  ANLRMS$_SUMMARYHDG
                .EXTRN  ANLRMS$_OWNERUIC
                .EXTRN  ANLRMS$_JNL, ANLRMS$_AIJNL
                .EXTRN  ANLRMS$_BIJNL, ANLRMS$_ATJNL
                .EXTRN  ANLRMS$_ATTOP, ANLRMS$_BADCMD
                .EXTRN  ANLRMS$_BADPATH
                .EXTRN  ANLRMS$_BADVBN, ANLRMS$_DOWNHELP
                .EXTRN  ANLRMS$_DOWNPATH
                .EXTRN  ANLRMS$_EMPTYBKT
                .EXTRN  ANLRMS$_NODATA, ANLRMS$_NODOWN
                .EXTRN  ANLRMS$_NONEXT, ANLRMS$_NORECLAIMED
                .EXTRN  ANLRMS$_NORECS, ANLRMS$_NORRV
                .EXTRN  ANLRMS$_RESTDONE
                .EXTRN  ANLRMS$_STACKFULL
                .EXTRN  ANLRMS$_UNINITINDEX
                .EXTRN  ANLRMS$_FDLIDENT
                .EXTRN  ANLRMS$_FDLSYSTEM
                .EXTRN  ANLRMS$_FDLSOURCE
                .EXTRN  ANLRMS$_FDLFILE
                .EXTRN  ANLRMS$_FDLALLOC
                .EXTRN  ANLRMS$_FDLNOALLOC
                .EXTRN  ANLRMS$_FDLBESTTRY
                .EXTRN  ANLRMS$_FDLBUCKETSIZE
                .EXTRN  ANLRMS$_FDLCLUSTERSIZE
                .EXTRN  ANLRMS$_FDLCONTIG
                .EXTRN  ANLRMS$_FDLEXTENSION
                .EXTRN  ANLRMS$_FDLGLOBALBUFS
                .EXTRN  ANLRMS$_FDLMAXRECORD
```

```
                .EXTRN  ANLRMS$_FDLFILENAME
                .EXTRN  ANLRMS$_FDLORG, ANLRMS$_FDLOWNER
                .EXTRN  ANLRMS$_FDLPROTECTION
                .EXTRN  ANLRMS$_FDLRECORD
                .EXTRN  ANLRMS$_FDLSPAN
                .EXTRN  ANLRMS$_FDLCC, ANLRMS$_FDLVFCSIZE
                .EXTRN  ANLRMS$_FDLFORMAT
                .EXTRN  ANLRMS$_FDLSIZE
                .EXTRN  ANLRMS$_FDLAREA
                .EXTRN  ANLRMS$_FDLKEY, ANLRMS$_FDLCHANGES
                .EXTRN  ANLRMS$_FDLDATAAREA
                .EXTRN  ANLRMS$_FDLDATAFILL
                .EXTRN  ANLRMS$_FDLDATAKEYCOMPB
                .EXTRN  ANLRMS$_FDLDATARECCOMPB
                .EXTRN  ANLRMS$_FDLDUPS
                .EXTRN  ANLRMS$_FDLINDEXAREA
                .EXTRN  ANLRMS$_FDLINDEXCOMPB
                .EXTRN  ANLRMS$_FDLINDEXFILL
                .EXTRN  ANLRMS$_FDLL1INDEXAREA
                .EXTRN  ANLRMS$_FDLKEYNAME
                .EXTRN  ANLRMS$_FDLNORECS
                .EXTRN  ANLRMS$_FDLNULLKEY
                .EXTRN  ANLRMS$_FDLNULLVALUE
                .EXTRN  ANLRMS$_FDLPROLOG
                .EXTRN  ANLRMS$_FDLSEGLENGTH
                .EXTRN  ANLRMS$_FDLSEGPOS
                .EXTRN  ANLRMS$_FDLSEGTYPE
                .EXTRN  ANLRMS$_FDLANALAREA
                .EXTRN  ANLRMS$_FDLRECL
                .EXTRN  ANLRMS$_FDLANALKEY
                .EXTRN  ANLRMS$_FDLDATAKEYCOMP
                .EXTRN  ANLRMS$_FDLDATARECCOMP
                .EXTRN  ANLRMS$_FDLDATARECS
                .EXTRN  ANLRMS$_FDLDATASPACE
                .EXTRN  ANLRMS$_FDLDEPTH
                .EXTRN  ANLRMS$_FDLDUPSPER
                .EXTRN  ANLRMS$_FDLIDXCOMP
                .EXTRN  ANLRMS$_FDLIDXFILL
                .EXTRN  ANLRMS$_FDLIDXSPACE
                .EXTRN  ANLRMS$_FDLIDXL1RECS
                .EXTRN  ANLRMS$_FDLDATALENMEAN
                .EXTRN  ANLRMS$_FDLIDXLENMEAN
                .EXTRN  ANLRMS$_STATAREA
                .EXTRN  ANLRMS$_STATRECL
                .EXTRN  ANLRMS$_STATKEY
                .EXTRN  ANLRMS$_STATDEPTH
                .EXTRN  ANLRMS$_STATIDXL1RECS
                .EXTRN  ANLRMS$_STATIDXLENMEAN
                .EXTRN  ANLRMS$_STATIDXSPACE
                .EXTRN  ANLRMS$_STATIDXFILL
                .EXTRN  ANLRMS$_STATIDXCOMP
                .EXTRN  ANLRMS$_STATDATARECS
                .EXTRN  ANLRMS$_STATDUPSPER
                .EXTRN  ANLRMS$_STATDATALENMEAN
                .EXTRN  ANLRMS$_STATDATASPACE
                .EXTRN  ANLRMS$_STATDATAFILL
                .EXTRN  ANLRMS$_STATDATAKEYCOMP
```

```
.EXTRN  ANLRMS$_STATDATARECCOMP
.EXTRN  ANLRMS$_STATEFFICIENCY
.EXTRN  ANLRMS$_BADAREA1ST2
.EXTRN  ANLRMS$_BADAREABKTSIZE
.EXTRN  ANLRMS$_BADAREAFIT
.EXTRN  ANLRMS$_BADAREAID
.EXTRN  ANLRMS$_BADAREANEXT
.EXTRN  ANLRMS$_BADAREAROOT
.EXTRN  ANLRMS$_BADAREAUSED
.EXTRN  ANLRMS$_BADBKTAREAID
.EXTRN  ANLRMS$_BADBKTCHECK
.EXTRN  ANLRMS$_BADBKTFREE
.EXTRN  ANLRMS$_BADBKTKEYID
.EXTRN  ANLRMS$_BADBKTLEVEL
.EXTRN  ANLRMS$_BADBKTROOTBIT
.EXTRN  ANLRMS$_BADBKTSAMPLE
.EXTRN  ANLRMS$_BADCELLFIT
.EXTRN  ANLRMS$_BADCHECKSUM
.EXTRN  ANLRMS$_BADDATARECBITS
.EXTRN  ANLRMS$_BADDATARECFIT
.EXTRN  ANLRMS$_BADDATARECPS
.EXTRN  ANLRMS$_BAD3IDXKEYFIT
.EXTRN  ANLRMS$_BADIDXLASTKEY
.EXTRN  ANLRMS$_BADIDXORDER
.EXTRN  ANLRMS$_BADIDXRECBITS
.EXTRN  ANLRMS$_BADIDXRECFIT
.EXTRN  ANLRMS$_BADIDXRECPS
.EXTRN  ANLRMS$_BADKEYAREAID
.EXTRN  ANLRMS$_BADKEYDATABKT
.EXTRN  ANLRMS$_BADKEYDATAFIT
.EXTRN  ANLRMS$_BADKEYDATATYPE
.EXTRN  ANLRMS$_BADKEYIDXBKT
.EXTRN  ANLRMS$_BADKEYFILL
.EXTRN  ANLRMS$_BADKEYFIT
.EXTRN  ANLRMS$_BADKEYREFID
.EXTRN  ANLRMS$_BADKEYROOTLEVEL
.EXTRN  ANLRMS$_BADKEYSEGCOUNT
.EXTRN  ANLRMS$_BADKEYSEGVEC
.EXTRN  ANLRMS$_BADKEYSUMMARY
.EXTRN  ANLRMS$_BADREADNOPAR
.EXTRN  ANLRMS$_BADREADPAR
.EXTRN  ANLRMS$_BADSIDRDUPCT
.EXTRN  ANLRMS$_BADSIDRPTRFIT
.EXTRN  ANLRMS$_BADSIDRPTRSZ
.EXTRN  ANLRMS$_BADSIDRSIZE
.EXTRN  ANLRMS$_BADSTREAMEOF
.EXTRN  ANLRMS$_BADVBNFREE
.EXTRN  ANLRMS$_BKTLOOP
.EXTRN  ANLRMS$_EXTENDERR
.EXTRN  ANLRMS$_FLAGERROR
.EXTRN  ANLRMS$_MISSINGBKT
.EXTRN  ANLRMS$_NOTOK, ANLRMS$_SPANERROR
.EXTRN  ANLRMS$_TOOMANYRECS
.EXTRN  ANLRMS$_UNWIND, ANLRMS$_VFCTOOSHORT
.EXTRN  ANLRMS$_CACHEFULL
.EXTRN  ANLRMS$_CACHERELFAIL
.EXTRN  ANLRMS$_FACILITY
```

```
                                        .EXTRN  ANL$FORMAT_ERROR
                                        .EXTRN  ANL$FORMAT_LINE
                                        .EXTRN  ANL$FORMAT_PROTECTION_MASK
                                        .EXTRN  ANL$FORMAT_SKIP
                                        .EXTRN  ANL$PREPARE_QUOTED_STRING
                                        .EXTRN  CLI$GET_VALUE, LIB$FIND_FILE
                                        .EXTRN  LIB$FREE_VM, LIB$GET_VM
                                        .EXTRN  STR$TRIM, SYS$CLOSE
                                        .EXTRN  SYS$OPEN, SYS$CONNECT

                                        .PSECT  $CODE$,NOWRT,2

                          01FC 00000    .ENTRY  ANL$OPEN_NEXT_RMS_FILE, Save R2,R3,R4,R5,-      ; 0698
                                                R6,R7,R8
                5E 00000000G   00  9E 00002    MOVAB   STR$TRIM, R8                            ;
                57 00000000G   00  9E 00009    MOVAB   LIB$SIGNAL, R7                          ;
                56      0000'  CF  9E 00010    MOVAB   RESULTANT_SPEC, R6                      ;
                5E             08  C2 00015    SUBL2   #8, SP                                  ;
                53        04   AC  D0 00018    MOVL    OPENED_SPEC, R3                         ; 0701
                2D      0524   C6  E9 0001C    BLBC    GET_NEW_SPEC, 2$                        ; 0717
          0418  C6        FF   8F  9B 00021    MOVZBW  #255, WILDCARD_SPEC                     ; 0718
                        0418   C6  9F 00027    PUSHAB  WILDCARD_SPEC                           ; 0719
                        0000'  CF  9F 0002B    PUSHAB  P.AAA                                   ;
 00000000G      00             02  FB 0002F    CALLS   #2, CLI$GET_VALUE                       ;
                52             50  D0 00036    MOVL    R0, STATUS                              ;
                03             52  E8 00039    BLBS    STATUS, 1$                              ; 0723
                             0131  31 0003C    BRW     12$                                     ;
                        0418   C6  9F 0003F 1$: PUSHAB  WILDCARD_SPEC                          ; 0725
                        0418   C6  9F 00043    PUSHAB  WILDCARD_SPEC                           ;
                        0418   C6  9F 00047    PUSHAB  WILDCARD_SPEC                           ;
                68             03  FB 0004B    CALLS   #3, STR$TRIM                            ;
                        0386   C6  B5 0004E 2$: TSTW    RMS_FAB+2                               ; 0731
                               29  13 00052    BEQL    4$                                      ;
                        0384   C6  9F 00054    PUSHAB  RMS_FAB                                 ; 0732
 00000000G      00             01  FB 00058    CALLS   #1, SYS$CLOSE                           ;
                52             50  D0 0005F    MOVL    R0, STATUS                              ;
                13             52  E8 00062    BLBS    STATUS, 3$                              ; 0733
                        0390   C6  DD 00065    PUSHL   RMS_FAB+12                              ;
                               52  DD 00069    PUSHL   STATUS                                  ;
                               56  DD 0006B    PUSHL   R6                                      ;
                               01  DD 0006D    PUSHL   #1                                      ;
                    00B11052   8F  DD 0006F    PUSHL   #11604050                               ;
                67             05  FB 00075    CALLS   #5, LIB$SIGNAL                          ;
          0000V CF             00  FB 00078 3$: CALLS   #0, ANL$BUCKET                         ; 0738
                66        FF   8F  9B 0007D 4$: MOVZBW  #255, RESULTANT_SPEC                    ; 0744
                6E             02  D0 00081    MOVL    #2, (SP)                                ; 0746
                               5E  DD 00084    PUSHL   SP                                      ;
                          08   AE  9F 00086    PUSHAB  STV                                     ; 0745
                               7E  D4 00089    CLRL    -(SP)                                   ;
                        0000'  CF  9F 0008B    PUSHAB  P.AAC                                   ;
                        0520   C6  9F 0008F    PUSHAB  WILDCARD_CONTEXT                        ;
                               56  DD 00093    PUSHL   R6                                      ;
                        0418   C6  9F 00095    PUSHAB  WILDCARD_SPEC                           ;
 00000000G      00             07  FB 00099    CALLS   #7, LIB$FIND_FILE                       ;
                52             50  D0 000A0    MOVL    R0, STATUS                              ;
                               56  DD 000A3    PUSHL   R6                                      ; 0747
                               56  DD 000A5    PUSHL   R6                                      ;
```

```
                                  56  DD 000A7          PUSHL   R6                                       ;
                          68      03  FB 000A9          CALLS   #3, STR$TRIM                             ;
                          22      52  E8 000AC          BLBS    STATUS, 6$                               ; 0753
            000182CA      8F      52  D1 000AF          CMPL    STATUS, #99018                           ; 0754
                                  12  13 000B6          BEQL    5$                                       ;
                              04  AE  DD 000B8          PUSHL   STV                                      ; 0755
                                  52  DD 000BB          PUSHL   STATUS                                   ;
                                  56  DD 000BD          PUSHL   R6                                       ;
                                  01  DD 000BF          PUSHL   #1                                       ;
                        00B1109A  8F  DD 000C1          PUSHL   #11604122                                ;
                          67      05  FB 000C7          CALLS   #5, LIB$SIGNAL                           ;
                0524      C6      01  D0 000CA  5$:     MOVL    #1, GET_NEW_SPEC                         ; 0756
                                  5D  11 000CF          BRB     8$                                       ; 0757
                            0524  C6  D4 000D1  6$:     CLRL    GET_NEW_SPEC                             ; 0762
                03B8      C6      66  90 000D5          MOVB    RESULTANT_SPEC, RMS_FAB+52               ; 0764
                03B0      C6  04  A6  D0 000DA          MOVL    RESULTANT_SPEC+4, RMS_FAB+44             ; 0765
                            0384  C6  9F 000E0          PUSHAB  RMS_FAB                                  ; 0766
            00000000G     00      01  FB 000E4          CALLS   #1, SYS$OPEN                             ;
                          52      50  D0 000EB          MOVL    R0, STATUS                               ;
                          16      52  E8 000EE          BLBS    STATUS, 7$                               ; 0767
                            0390  C6  DD 000F1          PUSHL   RMS_FAB+12                               ;
                                  52  DD 000F3          PUSHL   STATUS                                   ;
                                  56  DD 000F7          PUSHL   R6                                       ;
                                  01  DD 000F9          PUSHL   #1                                       ;
                        00B1109A  8F  DD 000FB          PUSHL   #11604122                                ;
                          67      05  FB 00101          CALLS   #5, LIB$SIGNAL                           ;
                          27      52  E9 00104          BLBC    STATUS, 8$                               ; 0768
                            03D4  C6  9F 00107  7$:     PUSHAB  RMS_RAB                                  ; 0769
            00000000G     00      01  FB 0010B          CALLS   #1, SYS$CONNECT                          ;
                          52      50  D0 00112          MOVL    R0, STATUS                               ;
                          1E      52  E8 00115          BLBS    STATUS, 9$                               ; 0770
                            03E0  C6  DD 00118          PUSHL   RMS_RAB+12                               ;
                                  52  DD 0011C          PUSHL   STATUS                                   ;
                                  56  DD 0011E          PUSHL   R6                                       ;
                                  01  DD 00120          PUSHL   #1                                       ;
                        00B1109A  8F  DD 00122          PUSHL   #11604122                                ;
                          67      05  FB 00128          CALLS   #5, LIB$SIGNAL                           ;
                          08      52  E8 0012B          BLBS    STATUS, 9$                               ; 0775
                                  53  DD 0012E  8$:     PUSHL   R3                                       ; 0776
                FECB      CF      01  FB 00130          CALLS   #1, ANL$OPEN_NEXT_RMS_FILE               ;
                                      04 00135          RET                                              ;
                          03  03A3  C6  91 00136  9$:   CMPB    RMS_FAB+31, #3                           ; 0783
                                  0B  12 0013B          BNEQ    10$                                      ;
                            018B  C6  95 0013D          TSTB    RMS_XABFHC+23                            ;
                                  05  12 00141          BNEQ    10$                                      ;
                018B      C6      02  90 00143          MOVB    #2, RMS_XABFHC+23                        ; 0784
                          50  018A  C6  9A 00148  10$:  MOVZBL  RMS_XABFHC+22, R0                        ; 0785
                                  03  12 0014D          BNEQ    11$                                      ;
                          50      01  D0 0014F          MOVL    #1, R0                                   ;
                018A      C6      50  90 00152  11$:    MOVB    R0, RMS_XABFHC+22                        ;
                0000'     CF  017C  C6  9E 00157        MOVAB   RMS_XABFHC+8, ANL$GL_FAT                 ; 0791
                0000V     CF      00  FB 0015E          CALLS   #0, ANL$PROLOG_INFO                      ; 0796
                          63      66  B0 00163          MOVW    RESULTANT_SPEC, (R3)                     ; 0805
      04    B3      04    B6      66  28 00166          MOVC3   RESULTANT_SPEC, @RESULTANT_SPEC+4, @4(R3) ; 0806
                          50      01  D0 0016C          MOVL    #1, R0                                   ; 0808
                                      04 0016F          RET                                              ;
                                  50  D4 00170  12$:    CLRL    R0                                       ; 0810
```

```
                                    04 00172        RET                                             ;

; Routine Size:  371 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  308     0811  1 %sbttl 'ANL$PROLOG_INFO - Set Up Global Information from Prolog'
;  309     0812  1 !++
;  310     0813  1 ! Functional Description:
;  311     0814  1 !       This routine is responsible for obtaining information from the
;  312     0815  1 !       prolog of a file and setting it up in global places.  We do this
;  313     0816  1 !       because the information is used a lot all over the place.
;  314     0817  1 !
;  315     0818  1 ! Formal Parameters:
;  316     0819  1 !       none
;  317     0820  1 !
;  318     0821  1 ! Implicit Inputs:
;  319     0822  1 !       global data                                                     7
;  320     0823  1 !
;  321     0824  1 ! Implicit Outputs:
;  322     0825  1 !       global data
;  323     0826  1 !                                                                       6
;  324     0827  1 ! Returned Value:
;  325     0828  1 !       none                                                            7
;  326     0829  1 !
;  327     0830  1 ! Side Effects:
;  328     0831  1 !
;  329     0832  1 !--
;  330     0833  1
;  331     0834  1
;  332     0835  2 global routine anl$prolog_info: novalue = begin
;  333     0836  2
;  334     0837  2 local
;  335     0838  2         p: bsd,
;  336     0839  2         pp: ref block[,byte];
;  337     0840  2
;  338     0841  2
;  339     0842  2 ! We want to set up a global variable with the prolog version.  If it's
;  340     0843  2 ! a sequential file, the version is 1.  Otherwise we have to get the
;  341     0844  2 ! file's first prolog block.
;  342     0845  2
;  343     0846  2 if .anl$gl_fat[fat$v_fileorg] eqlu fat$c_sequential then
;  344     0847  2         anl$gw_prolog = 1
;  345     0848  3 else (
;  346     0849  3
;  347     0850  3         ! Set up a BSD to read in the prolog of the file.  Get it.
;  348     0851  3
;  349     0852  3         init_bsd(p);
;  350     0853  3         p[bsd$w_size] = 1;
;  351     0854  3         p[bsd$l_vbn] = 1;
;  352     0855  3         anl$bucket(p,0);
;  353     0856  3
;  354     0857  3         ! Put the prolog version in our global variable.
;  355     0858  3
;  356     0859  3         pp = .p[bsd$l_bufptr];
;  357     0860  3         anl$gw_prolog = .pp[plg$w_ver_no];
;  358     0861  3
;  359     0862  3         ! If this in a relative file, then the prolog block contains the
;  360     0863  3         ! end-of-file VBN.  It should have been in the damn file header,
;  361     0864  3         ! as it is for sequential files.  Let's put it there.
;  362     0865  3
;  363     0866  3         anl$gl_fat[fat$l_efblk] = .pp[plg$l_eof];
;  364     0867  3
```

```
;  365          0868  3          anl$bucket(p,-1);
;  366          0869  2 );
;  367          0870  2
;  368          0871  2 return;
;  369          0872  2
;  370          0873  1 end;


                                                003C 00000          .ENTRY  ANL$PROLOG_INFO, Save R2,R3,R4,R5      ; 0835
                                    5E          18   C2 00002          SUBL2   #24, SP                                ;
                                F0  8F      0000'   DF   93 00005          BITB    @ANL$GL_FAT, #240              ; 0846
                                                    06   12 00008          BNEQ    1$                             ;
                            0000'   CF              01   B0 0000D          MOVW    #1, ANL$GW_PROLOG              ; 0847
                                                    04   00012          RET                                    ;
            18          00          6E              00   2C 00013 1$:     MOVC5   #0, (SP), #0, #24, P           ; 0852
                                                    6E       00018      ;                                      ;
                                02  AE              01   B0 00019          MOVW    #1, P+2                        ; 0853
                                04  AE              01   D0 0001D          MOVL    #1, P+4                        ; 0854
                                                    7E   D4 00021          CLRL    -(SP)                          ; 0855
                                        04  AE      9F   00023          PUSHAB  P                              ;
                            0000V   CF              02   FB 00026          CALLS   #2, ANL$BUCKET                 ;
                                    51          0C  AE   D0 0002B          MOVL    P+12, PP                       ; 0859
                            0000'   CF          74  A1   B0 0002F          MOVW    116(PP), ANL$GW_PROLOG         ; 0860
                                    50      0000'   CF   D0 00035          MOVL    ANL$GL_FAT, R0                 ; 0866
                                08  A0          70  A1   D0 0003A          MOVL    112(PP), 8(R0)                 ;
                                    7E              01   CE 0003F          MNEGL   #1, -(SP)                      ; 0868
                                        04  AE      9F   00042          PUSHAB  P                              ;
                            0000V   CF              02   FB 00045          CALLS   #2, ANL$BUCKET                 ;
                                                    04   0004A          RET                                    ; 0873

; Routine Size:  75 bytes,    Routine Base:  $CODE$ + 0173
```

```
;  372     0874  1 %sbttl 'ANL$BUCKET - Handle Acquire & Release of Buckets'
;  373     0875  1 !++
;  374     0876  1 ! Functional Description:
;  375     0877  1 !       This routine is called to acquire and/or release buckets from
;  376     0878  1 !       the file being analyzed.  It handles reading and caching of
;  377     0879  1 !       buckets when they are acquired, and freeing of buffers when
;  378     0880  1 !       they are released.
;  379     0881  1 !
;  380     0882  1 !       Caching enhances performance when the same bucket is acquired
;  381     0883  1 !       by multiple routines around the same time (as in /INTERACTIVE mode).
;  382     0884  1 !
;  383     0885  1 !       This routine also handles a special call, indicated by no
;  384     0886  1 !       arguments, that causes us to flush the cache.
;  385     0887  1 !
;  386     0888  1 ! Formal Parameters:
;  387     0889  1 !       the_bsd         Address of a BSD describing the buckets to be
;  388     0890  1 !                       acquired and/or released.
;  389     0891  1 !       operation       positive:       If the BSD points to a bucket buffer,
;  390     0892  1 !                                       it is to be released.  Then the
;  391     0893  1 !                                       bucket described by the BSD size/VBN
;  392     0894  1 !                                       is to be acquired.  The positive value
;  393     0895  1 !                                       is the VBN of the parent of the new
;  394     0896  1 !                                       structure.
;  395     0897  1 !                       zero:           Same as positive, except that there is
;  396     0898  1 !                                       no parent or it is unknown.
;  397     0899  1 !                       -1:             If the BSD points to a bucket buffer,
;  398     0900  1 !                                       it is to be released.  No new bucket
;  399     0901  1 !                                       is acquired.
;  400     0902  1 !
;  401     0903  1 ! Implicit Inputs:
;  402     0904  1 !       global data
;  403     0905  1 !
;  404     0906  1 ! Implicit Outputs:
;  405     0907  1 !       global data
;  406     0908  1 !
;  407     0909  1 ! Returned Value:
;  408     0910  1 !       none
;  409     0911  1 !
;  410     0912  1 ! Side Effects:
;  411     0913  1 !
;  412     0914  1 !--
;  413     0915  1
;  414     0916  1
;  415     0917  2 global routine anl$bucket(the_bsd,operation): novalue = begin
;  416     0918  2
;  417     0919  2 bind
;  418     0920  2         b = .the_bsd: bsd;
;  419     0921  2
;  420     0922  2 literal
;  421     0923  2         slot_count = 32;
;  422     0924  2 own
;  423     0925  2         high_slot:      signed long initial(-1),
;  424     0926  2         cached_size:    vector[slot_count,byte],
;  425     0927  2         cached_vbn:     vector[slot_count,long],
;  426     0928  2         cached_bufptr:  vector[slot_count,long] initial(rep slot_count of long (0)),
;  427     0929  2         cached_refs:    vector[slot_count,byte] initial(rep slot_count of byte (0));
;  428     0930  2
```

```
;  429          0931  2 local
;  430          0932  2         status: long,
;  431          0933  2         i: long,
;  432          0934  2         available: signed long, release: signed long, acquire: signed long;
;  433          0935  2
;  434          0936  2 builtin
;  435          0937  2         nullparameter;
;  436          0938  2
;  437          0939  2
;  438          0940  2 ! If we are called with no parameters, then we are to flush the cache.
;  439          0941  2 ! This is done between files, in case a drastic structure error causes
;  440          0942  2 ! us to quit in the middle of an analysis.
;  441          0943  2
;  442          0944  3 if nullparameter(1) then (
;  443          0945  3         high_slot = -1;
;  444          0946  3         return,
;  445          0947  2 );
```

```
;  447     0948  2 ! To begin with, we have to locate three slots in the valid cached vectors:
;  448     0949  2 !       available:      The index of the first available slot.
;  449     0950  2 !       release:        The index of the slot describing the buffer to be
;  450     0951  2 !                       released (if the BSD references a buffer).
;  451     0952  2 !       acquire:        The index of the slot describing the buffer to be
;  452     0953  2 !                       acquired (if th' new bucket is indeed cached).
;  453     0954  2
;  454     0955  2 available = release = acquire = -1;
;  455     0956  3 incr i from 0 to .high_slot do (
;  456     0957  3
;  457     0958  3         if .cached_refs[.i] eglu 0 then
;  458     0959  3                 available = minu(.available,.i)
;  459     0960  3         else
;  460     0961  3                 if .b[bsd$l_bufptr] eqla .cached_bufptr[.i] then
;  461     0962  3                         release = .i;
;  462     0963  3
;  463     0964  3         if .b[bsd$w_size] eglu .cached_size[.i] and .b[bsd$l_vbn] eglu .cached_vbn[.i] then
;  464     0965  3                 acquire = .i;
;  465     0966  2 );
;  466     0967  2
;  467     0968  2 ! Alright, now we may need to release a buffer.
;  468     0969  2
;  469     0970  3 if .release neq -1 then (
;  470     0971  3
;  471     0972  3         ! We have a buffer to release.  Decrement its reference count.
;  472     0973  3
;  473     0974  3         decrement (cached_refs[.release]);
;  474     0975  3
;  475     0976  3         ! Clear the buffer pointer in the BSD as a positive statement
;  476     0977  3         ! that it no longer references a buffer.
;  477     0978  3
;  478     0979  3         b[bsd$l_bufptr] = 0;
;  479     0980  3
;  480     0981  2 ) else
;  481     0982  2         if .b[bsd$l_bufptr] neqa 0 then
;  482     0983  2
;  483     0984  2                 ! Oops.  The BSD references a buffer, but we didn't find it
;  484     0985  2                 ! in the cache.  This is a logic error.
;  485     0986  2
;  486     0987  2                 signal(anlrms$_cacherelfail);
```

```
;  488          0988  2 ! Now the caller may want us to acquire a new bucket.  We allow buckets
;  489          0989  2 ! to be of size zero, which is useful when the BSD describes something
;  490          0990  2 ! that isn't actually within the virtual blocks of the file (e.g., the
;  491          0991  2 ! file header).
;  492          0992  2
;  493          0993  3 if .operation neq -1 and .b[bsd$w_size] nequ 0 then (
;  494          0994  3
;  495          0995  3         ! If the desired bucket is already cached, then we can just return
;  496          0996  3         ! a reference to the extant buffer.  We must increment the reference
;  497          0997  3         ! count to record this new reference.
;  498          0998  3
;  499          0999  4         if .acquire neq -1 then (
;  500          1000  4                 b[bsd$l_bufptr] = .cached_bufptr[.acquire];
;  501          1001  4                 increment (cached_refs[.acquire]);
;  502          1002  4
;  503          1003  4         ) else (
;  504          1004  4
;  505          1005  4                 ! We know the desired bucket must be read.  Use the first
;  506          1006  4                 ! unused slot if any; otherwise use the available slot found
;  507          1007  4                 ! above.  If we have run out of slots, that's trouble.
;  508          1008  4
;  509          1009  4                 if .high_slot lss slot_count-1 then
;  510          1010  5                         available = increment (high_slot)
;  511          1011  4                 else
;  512          1012  4                         if .available eql -1 then
;  513          1013  4                                 signal (anlrms$_cachefull);
;  514          1014  4
;  515          1015  4                 ! If the slot we are going to use for this new bucket still
;  516          1016  4                 ! describes an old buffer, free up the buffer.
;  517          1017  4
;  518          1018  5                 if .cached_bufptr[.available] nega 0 then (
;  519          1019  5                         status = lib$free_vm(%ref(.cached_size[.available]*512),
;  520          1020  5                                              cached_bufptr[.available]);
;  521          1021  5                         check (.status, .status);
;  522          1022  4                 );
;  523          1023  4
;  524          1024  4                 ! Allocate a buffer for the bucket and read it in.
;  525          1025  4                 ! We force reads of VBN 0 to fail rather than read the next
;  526          1026  4                 ! stupid block.
;  527          1027  4
;  528          1028  4                 status = lib$get_vm(%ref(.b[bsd$w_size]*512),rms_rab[rab$l_ubf]);
;  529          1029  4                 check (.status, .status);
;  530          1030  4                 rms_rab[rab$w_usz] = .b[bsd$w_size]*512;
;  531          1031  4                 rms_rab[rab$l_bkt] = .b[bsd$l_vbn];
;  532          1032  5                 status =        (if .b[bsd$l_vbn] eqlu 0 then
;  533          1033  5                                         false
;  534          1034  5                                 else
;  535          1035  4                                         $read(rab=rms_rab));
;  536          1036  4
;  537          1037  4                 ! If the read failed, we have to generate and error message
;  538          1038  4                 ! and treat it as a drastic structure error.  A read fails
;  539          1039  4                 ! if the status is bad, or if it didn't read in the number
;  540          1040  4                 ! of blocks we asked for.  If we know the parent VBN, it is
;  541          1041  4                 ! included in the message.
;  542          1042  4
;  543          1043  5                 if not .status or (.rms_rab[rab$w_rsz]+511)/512 nequ .b[bsd$w_size] then (
;  544          1044  5                         lib$free_vm(%ref(.b[bsd$w_size]*512),rms_rab[rab$l_ubf]);
```

```
;  545          1045  5                        if .operation neq 0 then
;  546          1046  5                                anl$format_error(anlrms$_badreadpar,.operation,.b[bsd$l_vbn])
;  547          1047  5                        else
;  548          1048  5                                anl$format_error(anlrms$_badreadnopar,.b[bsd$l_vbn]);
;  549          1049  5                        signal (anlrms$_unwind);
;  550          1050  4                );
;  551          1051  4
;  552          1052  4                ! Now we can fill in the cached vectors with a description of
;  553          1053  4                ! the bucket we just read.
;  554          1054  4
;  555          1055  4                cached_size[.available] =       .b[bsd$w_size];
;  556          1056  4                cached_vbn[.available] =        .b[bsd$l_vbn];
;  557          1057  4                cached_bufptr[.available] =     .rms_rab[rab$l_ubf];
;  558          1058  4                cached_refs[.available] =       1;
;  559          1059  4
;  560          1060  4                ! Finally, reference the buffer in the BSD.
;  561          1061  4
;  562          1062  4                b[bsd$l_bufptr] = .rms_rab[rab$l_ubf];
;  563          1063  3          );
;  564          1064  3
;  565          1065  3          ! As a friendly gesture, set the end pointer in the BSD to point
;  566          1066  3          ! at the byte following the buffer.
;  567          1067  3
;  568          1068  3          b[bsd$l_erdptr] = .b[bsd$l_bufptr] + .b[bsd$w_size]*512;
;  569          1069  2 );
;  570          1070  2
;  571          1071  2 return;
;  572          1072  2
;  573          1073  1 end;


                                        .PSECT  $OWN$,NOEXE,2

                FFFFFFFF  00528 HIGH_SLOT:
                                        .LONG   -1                                              ;
                          0052C CACHED_SIZE:
                                        .BLKB   32
                          0054C CACHED_VBN:
                                        .BLKB   128
                00000000# 005CC CACHED_BUFPTR:
                                        .LONG   0[32]                                           ;
                      00# 0064C CACHED_REFS:
                                        .BYTE   0[32]                                           ;

                                        .EXTRN  SYS$READ

                                        .PSECT  $CODE$,NOWRT,2

                          0FFC 00000    .ENTRY  ANL$BUCKET, Save R2,R3,R4,R5,R6,R7,R8,R9,-       ; 0917
                                                R10,R11
5B 00000000G   00   9E 00002            MOVAB   LIB$FREE_VM, R11                                ;
5A 00000000G   00   9E 00009            MOVAB   LIB$SIGNAL, R10                                 ;
59       0000'  CF   9E 00010            MOVAB   HIGH_SLOT, R9                                   ;
5E              04   C2 00015            SUBL2   #4, SP                                          ;
54        04    AC   D0 00018            MOVL    THE_BSD, R4                                     ; 0920
                6C   95 0001C            TSTB    (AP)                                            ; 0944
```

```
                                 05  13 0001E          BEQL    1$                                                      ;
                              04 AC  D5 00020          TSTL    4(AP)                                                   ;
                                 04  12 00023          BNEQ    2$                                                      ;
                           69        01  CE 00025 1$:  MNEGL   #1, HIGH_SLOT                                          ; 0945
                                     04  00028          RET                                                             ; 0944
                           53        01  CE 00029 2$:  MNEGL   #1, ACQUIRE                                            ; 0955
                           51        01  CE 0002C       MNEGL   #1, RELEASE                                            ;
                           52        01  CE 0002F       MNEGL   #1, AVAILABLE                                          ;
                           50        01  CE 00032       MNEGL   #1, I                                                  ; 0956
                                     39  11 00035       BRB     7$                                                     ;
                        0124 C940    95  00037 3$:      TSTB    CACHED_REFS[I]                                         ; 0958
                                     10  12 0003C       BNEQ    5$                                                     ;
                           55        52  D0 0003E       MOVL    AVAILABLE, R5                                          ; 0959
                           50        55  D1 00041       CMPL    R5, I                                                  ;
                                     03  1B 00044       BLEQU   4$                                                     ;
                           55        50  D0 00046       MOVL    I, R5                                                  ;
                           52        55  D0 00049 4$:   MOVL    R5, AVAILABLE                                          ;
                                     0C  11 0004C       BRB     6$                                                     ;
                     00A4 C940    0C   A4  D1 0004E 5$:  CMPL    12(R4), CACHED_BUFPTR[I]                               ; 0961
                                     03  12 00055       BNEQ    6$                                                     ;
                           51        50  D0 00057       MOVL    I, RELEASE                                             ; 0962
                           55     04 A940  9A 0005A 6$:  MOVZBL  CACHED_SIZE[I], R5                                    ; 0964
                        02 A4           55  B1 0005F       CMPW    R5, 2(R4)                                              ;
                                     0B  12 00063       BNEQ    7$                                                     ;
                     24 A940    04   A4  D1 00065       CMPL    4(R4), CACHED_VBN[I]                                   ;
                                     03  12 0006B       BNEQ    7$                                                     ;
                           53        50  D0 0006D       MOVL    I, ACQUIRE                                             ; 0965
                     C3    50        69  F3 00070 7$:   AOBLEQ  HIGH_SLOT, I, 3$                                      ; 0956
                        FFFFFFFF 8F     51  D1 00074       CMPL    RELEASE, #-1                                           ; 0970
                                     0D  13 0007B       BEQL    8$                                                     ;
                        0124 C941    97  0007D       DECB    CACHED_REFS[RELEASE]                                   ; 0974
                           57     0C   A4  9E 00082       MOVAB   12(R4), R7                                             ; 0979
                                     67  D4 00086       CLRL    (R7)                                                   ;
                                     11  11 00088       BRB     9$                                                     ; 0970
                           57        0C   A4  9E 0008A 8$:   MOVAB   12(R4), R7                                        ; 0982
                                     67  D5 0008E       TSTL    (R7)                                                   ;
                                     09  13 00090       BEQL    9$                                                     ;
                              00000000G  8F  DD 00092       PUSHL   #ANLRMS$_CACHERELFAIL                                  ; 0987
                           6A                01  FB 00098       CALLS   #1, LIB$SIGNAL                                 ;
                        FFFFFFFF 8F     08   AC  D1 0009B 9$:   CMPL    OPERATION, #-1                                 ; 0993
                                     03  13 000A3       BEQL    10$                                                    ;
                                 02   A4  B5 000A5       TSTW    2(R4)                                                  ;
                                     01  12 000A8 10$:   BNEQ    11$                                                    ;
                                     04  000AA          RET                                                            ;
                        FFFFFFFF 8F         53  D1 000AB 11$:   CMPL    ACQUIRE, #-1                                   ; 0999
                                     0E  13 000B2       BEQL    12$                                                    ;
                           67     00A4 C943    D0 000B4       MOVL    CACHED_BUFPTR[ACQUIRE], (R7)                   ; 1000
                                 0124 C943    96 000BA       INCB    CACHED_REFS[ACQUIRE]                           ; 1001
                                   0103  31 000BF       BRW     23$                                                    ; 0999
                           1F            69  D1 000C2 12$:   CMPL    HIGH_SLOT, #31                                 ; 1009
                                     0C  18 000C5       BGEQ    13$                                                    ;
                     50    69        01  C1 000C7       ADDL3   #1, HIGH_SLOT, R0                                      ; 1010
                           69        50  D0 000CB       MOVL    R0, HIGH_SLOT                                          ;
                           52        50  D0 000CE       MOVL    R0, AVAILABLE                                          ;
                                     12  11 000D1       BRB     14$                                                    ;
                        FFFFFFFF 8F         52  D1 000D3 13$:   CMPL    AVAILABLE, #-1                                 ; 1012
                                     09  12 000DA       BNEQ    14$                                                    ;
```

```
                                00000000G  8F   DD 000DC          PUSHL   #ANLRMS$_CACHEFULL                       ; 1013
                                      6A   01   FB 000E2          CALLS   #1, LIB$SIGNAL                          ;
                                      58 00A4 C942   DE 000E5 14$:  MOVAL   CACHED_BUFPTR[AVAILABLE], R8            ; 1018
                                                 68   D5 000EB          TSTL    (R8)                               ;
                                                 1D   13 000ED          BEQL    15$                                ;
                                                 58   DD 000EF          PUSHL   R8                                 ; 1020
                                      50   04 A942   9A 000F1          MOVZBL  CACHED_SIZE[AVAILABLE], R0          ; 1019
                              04   AE      50   09   78 000F6          ASHL    #9, R0, 4(SP)                      ;
                                        04   AE   9F 000FB          PUSHAB  4(SP)                              ;
                                      6B        02   FB 000FE          CALLS   #2, LIB$FREE_VM                    ; 1020
                                      55        50   D0 00101          MOVL    R0, STATUS                         ;
                                      05        55   E8 00104          BLBS    STATUS, 15$                        ; 1021
                                                55   DD 00107          PUSHL   STATUS                             ;
                                      6A        01   FB 00109          CALLS   #1, LIB$SIGNAL                     ;
                                           FED0   C9   9F 0010C 15$:    PUSHAB  RMS_RAB+36                         ; 1028
                                      56   02   A4   3C 00110          MOVZWL  2(R4), R6                          ;
                              56      56        09   78 00114          ASHL    #9, R6, R6                         ;
                                   04   AE      56   D0 00118          MOVL    R6, 4(SP)                          ;
                                        04   AE   9F 0011C          PUSHAB  4(SP)                              ;
                                00000000G  00   02   FB 0011F          CALLS   #2, LIB$GET_VM                     ;
                                      55        50   D0 00126          MOVL    R0, STATUS                         ;
                                      05        55   E8 00129          BLBS    STATUS, 16$                        ; 1029
                                                55   DD 0012C          PUSHL   STATUS                             ;
                                      6A        01   FB 0012E          CALLS   #1, LIB$SIGNAL                     ;
                                   FECC   C9      56   B0 00131 16$:    MOVW    R6, RMS_RAB+32                     ; 1030
                                      53   04   A4   D0 00136          MOVL    4(R4), R3                          ; 1031
                                   FEE4   C9      53   D0 0013A          MOVL    R3, RMS_RAB+56                     ;
                                                04   12 0013F          BNEQ    17$                                ; 1032
                                                55   D4 00141          CLRL    STATUS                             ;
                                                0E   11 00143          BRB     18$                                ;
                                           FEAC   C9   9F 00145 17$:    PUSHAB  RMS_RAB                            ; 1035
                                00000000G  00        01   FB 00149          CALLS   #1, SYS$READ                       ;
                                      55        50   D0 00150          MOVL    R0, STATUS                         ;
                                      19        55   E9 00153 18$:    BLBC    STATUS, 19$                        ; 1043
                                      50   FECE   C9   3C 00156          MOVZWL  RMS_RAB+34, R0                     ;
                                      50   01FF   C0   9E 0015B          MOVAB   511(R0), R0                        ;
                                      50 00000200 8F   C6 00160          DIVL2   #512, R0                           ;
50         02   A4                   10           00   ED 00167          CMPZV   #0, #16, 2(R4), R0                 ;
                                                3B   13 0016D          BEQL    22$                                ;
                                           FED0   C9   9F 0016F 19$:    PUSHAB  RMS_RAB+36                         ; 1044
                                   04   AE           56   D0 00173          MOVL    R6, 4(SP)                          ;
                                             04   AE   9F 00177          PUSHAB  4(SP)                              ;
                                      6B        02   FB 0017A          CALLS   #2, LIB$FREE_VM                    ;
                                             08   AC   D5 0017D          TSTL    OPERATION                          ; 1045
                                                12   13 00180          BEQL    20$                                ;
                                                53   DD 00182          PUSHL   R3                                 ; 1046
                                             08   AC   DD 00184          PUSHL   OPERATION                          ;
                                00000000G       8F   DD 00187          PUSHL   #ANLRMS$_BADREADPAR                ;
                                   0000G   CF           03   FB 0018D          CALLS   #3, ANL$FORMAT_ERROR               ;
                                                0D   11 00192          BRB     21$                                ;
                                                53   DD 00194 20$:    PUSHL   R3                                 ; 1048
                                00000000G       8F   DD 00196          PUSHL   #ANLRMS$_BADREADNOPAR              ;
                                   0000G   CF           02   FB 0019C          CALLS   #2, ANL$FORMAT_ERROR               ;
                                00000000G       8F   DD 001A1 21$:    PUSHL   #ANLRMS$_UNWIND                    ; 1049
                                      6A        01   FB 001A7          CALLS   #1, LIB$SIGNAL                     ;
                                04 A942      02   A4   90 001AA 22$:    MOVB    2(R4), CACHED_SIZE[AVAILABLE]      ; 1055
                                24 A942           53   D0 001B0          MOVL    R3, CACHED_VBN[AVAILABLE]          ; 1056
```

```
                                 68     FED0    C9  D0 001B5          MOVL    RMS_RAB+36, (R8)                        ; 1057
                          0124 C942             01  90 001BA          MOVB    #1, CACHED_REFS[AVAILABLE]              ; 1058
                                 67     FED0    C9  D0 001C0          MOVL    RMS_RAB+36, (R7)                        ; 1062
                                 50       02    A4  3C 001C5 23$:      MOVZWL  2(R4), R0                       ; 1068
                    50           50             09  78 001C9          ASHL    #9, R0, R0                              ;
                10  A4           50             67  C1 001CD          ADDL3   (R7), R0, 16(R4)                        ;
                                                    04 001D2          RET                                     ; 1073

; Routine Size:  467 bytes,    Routine Base:  $CODE$ + 01BE
```

```
;  575          1074  1 %sbttl 'ANL$FORMAT_FILE_HEADER - Print Nice File Header'
;  576          1075  1 !++
;  577          1076  1 ! Functional Description:
;  578          1077  1 !       This routine is called to print a nicely formatted file header.
;  579          1078  1 !       This does NOT include the information in the user file attribute
;  580          1079  1 !       area.  We assume that the file header is valid, since it's
;  581          1080  1 !       ANALYZE/DISK_STRUCTURE's business to verify it.
;  582          1081  1 !
;  583          1082  1 ! Formal Parameters:
;  584          1083  1 !       none
;  585          1084  1 !
;  586          1085  1 ! Implicit Inputs:
;  587          1086  1 !       global data
;  588          1087  1 !
;  589          1088  1 ! Implicit Outputs:
;  590          1089  1 !       global data
;  591          1090  1 !
;  592          1091  1 ! Returned Value:
;  593          1092  1 !       none
;  594          1093  1 !
;  595          1094  1 ! Side Effects:
;  596          1095  1 !
;  597          1096  1 !--
;  598          1097  1
;  599          1098  1
;  600          1099  2 global routine anl$format_file_header: novalue = begin
;  601          1100  2
;  602          1101  2 bind
;  603          1102  2         null= uplit byte(%ascic ''),
;  604          1103  2         none= uplit byte(%ascic ' none');
;  605          1104  2
;  606          1105  2 local
;  607          1106  2         ai,bi,at,ru,only_ru,never_ru: word,
;  608          1107  2         bi_comma,at_comma,ru_comma,only_ru_comma,never_ru_comma:word,
;  609          1108  2         flag:byte;
;  610          1109  2
;  611          1110  2 ! Start by putting out a little heading line.
;  612          1111  2
;  613          1112  2 anl$format_line(3,0,anlrms$_filehdr);
;  614          1113  2 anl$format_skip(0);
;  615          1114  2
;  616          1115  2 ! We will begin with the complete file specification.
;  617          1116  2
;  618          1117  2 anl$format_line(0,1,anlrms$_filespec,resultant_spec);
;  619          1118  2
;  620          1119  2 ! Now we include the file ID.
;  621          1120  2
;  622          1121  2 anl$format_line(0,1,anlrms$_fileid,.rms_nam[nam$w_fid_num],.rms_nam[nam$w_fid_seq],.rms_nam[nam$w_fid_rvn]);
;  623          1122  2
;  624          1123  2 ! Now the owner UIC.
;  625          1124  2
;  626          1125  2 anl$format_line(0,1,anlrms$_owneruic,.rms_xabpro[xab$w_grp],.rms_xabpro[xab$w_mbm]);
;  627          1126  2
;  628          1127  2 ! Now the file protection mask.
;  629          1128  2
;  630          1129  2 anl$format_protection_mask(1,anlrms$_prot,.rms_xabpro[xab$w_pro]);
;  631          1130  2
```

```
;  632          1131  2 ! Now the creation, revision, expiration, and backup dates.  We also include
;  633          1132  2 ! the revision number.  Some dates might not be present.
;  634          1133  2
;  635          1134  2 anl$format_line(0,1,anlrms$_creation,rms_xabdat[xab$q_cdt]);
;  636          1135  2 anl$format_line(0,1 anlrms$_revision,rms_xabdat[xab$q_rdt],,rms_xabdat[xab$w_rvn]);
;  637          1136  2 if .rms_xabdat[xab$l_edt0] eqlu 0 and .rms_xabdat[xab$l_edt4] eqlu 0 then
;  638          1137  2         anl$format_line(0,1,anlrms$_noexpiration)
;  639          1138  2 else
;  640          1139  2         anl$format_line(0,1,anlrms$_expiration,rms_xabdat[xab$q_edt]);
;  641          1140  2 if .rms_xabdat[xab$l_bdt0] eqlu 0 and .rms_xabdat[xab$l_bdt4] eqlu 0 then
;  642          1141  2         anl$format_line(0,1,anlrms$_nobackup)
;  643          1142  2 else
;  644          1143  2         anl$format_line(0,1,anlrms$_backup,rms_xabdat[xab$q_bdt]);
;  645          1144  2
;  646          1145  2 ! Now the contiguity options, performance options, and reliability options.
;  647          1146  2
;  648          1147  2 selectoneu true of set
;  649          1148  2 [.rms_fab[fab$v_cbt]]:  anl$format_line(0,1,anlrms$_contig,uplit byte(%ascic 'contiguous-best-try'));
;  650          1149  2
;  651          1150  2 [.rms_fab[fab$v_ctg]]:  anl$format_line(0,1,anlrms$_contig,uplit byte(%ascic 'contiguous'));
;  652          1151  2
;  653          1152  2 [otherwise]:           anl$format_line(0,1,anlrms$_contig,uplit byte(%ascic 'none'));
;  654          1153  2 tes;
;  655          1154  2
;  656          1155  2 if .rms_fab[fab$v_dfw] then
;  657          1156  2         anl$format_line(0,1,anlrms$_perform,uplit byte(%ascic 'deferred-write'))
;  658          1157  2 else
;  659          1158  2         anl$format_line(0,1,anlrms$_perform,uplit byte(%ascic 'none'));
;  660          1159  2
;  661          1160  2 if .rms_fab[fab$v_rck] then
;  662          1161  2         if .rms_fab[fab$v_wck] then
;  663          1162  2                 anl$format_line(0,1,anlrms$_reliab,uplit byte(%ascic 'read-check, write-check'))
;  664          1163  2         else
;  665          1164  2                 anl$format_line(0,1,anlrms$_reliab,uplit byte(%ascic 'read-check'))
;  666          1165  2 else
;  667          1166  2         if .rms_fab[fab$v_wck] then
;  668          1167  2                 anl$format_line(0,1,anlrms$_reliab,uplit byte(%ascic 'write-check'))
;  669          1168  2         else
;  670          1169  2                 anl$format_line(0,1,anlrms$_reliab,uplit byte(%ascic 'none'));
;  671          1170  2
;  672          1171  2 ! Now display the journaling option bits and journaling file names
;  673          1172  2
;  674          1173  2
;  675          1174  2 if .rms_xabjnl[xab$w_jop] eqlu 0 then
;  676          1175  2         anl$format_line(0,1,anlrms$_jnl,none,null,null,null,null,null,
;  677          1176  2                         null,null,null,null,null)
;  678          1177  3 else begin
;  679          1178  3         ai=bi=at=ru=only_ru=never_ru= uplit byte(%ascic '');
;  680          1179  3         bi_comma=at_comma=ru_comma=only_ru_comma=never_ru_comma= uplit byte(%ascic '');
;  681          1180  3
;  682          1181  4         if .rms_xabjnl[xab$v_ai] then begin
;  683          1182  4             ai= uplit byte(%ascic ' AI');
;  684          1183  4             flag=1;
;  685          1184  3             end;
;  686          1185  3
;  687          1186  4         if .rms_xabjnl[xab$v_bi] then begin
;  688          1187  4             bi= uplit byte(%ascic ' BI');
```

```
;  689          1188  5              bi_comma= (if .flag then
;  690          1189  5                         uplit byte(%ascic ',')
;  691          1190  5                     else
;  692          1191  4                         uplit byte(%ascic ''));
;  693          1192  4              flag=1;
;  694          1193  3              end;
;  695          1194  3
;  696          1195  4          if .rms_xabjnl[xab$v_at] then begin
;  697          1196  4              at= uplit byte(%ascic ' AT');
;  698          1197  5              at_comma= (if .flag then
;  699          1198  5                         uplit byte(%ascic ',')
;  700          1199  5                     else
;  701          1200  4                         uplit byte(%ascic ''));
;  702          1201  4              flag=1;
;  703          1202  3              end;
;  704          1203  3
;  705          1204  4          if .rms_xabjnl[xab$v_ru] then begin
;  706          1205  4              ru= uplit byte(%ascic ' RU');
;  707          1206  5              ru_comma= (if .flag then
;  708          1207  5                         uplit byte(%ascic ',')
;  709          1208  5                     else
;  710          1209  4                         uplit byte(%ascic ''));
;  711          1210  3              end;
;  712          1211  3
;  713          1212  4          if .rms_xabjnl[xab$v_only_ru] then begin
;  714          1213  4              only_ru= uplit byte(%ascic ' ONLY_RU');
;  715          1214  5              only_ru_comma= (if .flag then
;  716          1215  5                         uplit byte(%ascic ',')
;  717          1216  5                     else
;  718          1217  4                         uplit byte(%ascic ''));
;  719          1218  3              end;
;  720          1219  3
;  721          1220  4          if .rms_xabjnl[xab$v_never_ru] then begin
;  722          1221  4              never_ru= uplit byte(%ascic ' NEVER_RU');
;  723          1222  5              never_ru_comma= (if .flag then
;  724          1223  5                         uplit byte(%ascic ',')
;  725          1224  5                     else
;  726          1225  4                         uplit byte(%ascic ''));
;  727          1226  3              end;
;  728          1227  3
;  729          1228  3          anl$format_line(0,1,anlrms$_jnl,.ai,.bi_comma,.bi,.at_comma,.at,
;  730          1229  3                              .ru_comma,.ru,.never_ru_comma,.never_ru,
;  731          1230  3                              .only_ru_comma,.only_ru);
;  732          1231  3
;  733          1232  2      end;
;  734          1233  2
;  735          1234  2 if .rms_xabjnl[xab$b_ail] neq 0 then
;  736          1235  2          anl$format_line(0,1,anlrms$_aijnl,.rms_xabjnl[xab$B_aiS],
;  737          1236  2          .rms_xabjnl[xab$l_aia]);
;  738          1237  2
;  739          1238  2 if .rms_xabjnl[xab$b_bil] neq 0 then
;  740          1239  2          anl$format_line(0,1,anlrms$_bijnl,.rms_xabjnl[xab$B_biS],
;  741          1240  2          .rms_xabjnl[xab$l_bia]);
;  742          1241  2
;  743          1242  2 if .rms_xabjnl[xab$b_atl] neq 0 then
;  744          1243  2          anl$format_line(0,1,anlrms$_atjnl,.rms_xabjnl[xab$B_atS],
;  745          1244  2          .rms_xabjnl[xab$l_ata]);
```

```
;  746          1245  2
;  747          1246  2 return;
;  748          1247  2
;  749          1248  1 end;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                        00  00020 P.AAE:  .ASCII  <0>
                                                    65  6E  6F  6E  20  05  00021 P.AAF:  .ASCII  <5>\ none\
73  65  62  2D  73  75  6F  75  67  69  74  6E  6F  63  13  00027 P.AAG:  .ASCII  <19>\contiguous-best-try\
                                        79  72  74  2D  74  00036
                    73  75  6F  75  67  69  74  6E  6F  63  0A  0003B P.AAH:  .ASCII  <10>\contiguous\
                                        65  6E  6F  6E  04  00046 P.AAI:  .ASCII  <4>\none\
65  74  69  72  77  2D  64  65  72  72  65  66  65  64  0E  0004B P.AAJ:  .ASCII  <14>\deferred-write\
                                        65  6E  6F  6E  04  0005A P.AAK:  .ASCII  <4>\none\
72  77  20  2C  6B  63  65  68  63  2D  64  61  65  72  17  0005F P.AAL:  .ASCII  <23>\read-check, write-check\
                            6B  63  65  68  63  2D  65  74  69  0006E
                    6B  63  65  68  63  2D  64  61  65  72  0A  00077 P.AAM:  .ASCII  <10>\read-check\
                6B  63  65  68  63  2D  65  74  69  72  77  0B  00082 P.AAN:  .ASCII  <11>\write-check\
                                        65  6E  6F  6E  04  0008E P.AAO:  .ASCII  <4>\none\
                                                    00  00093 P.AAP:  .ASCII  <0>
                                                    00  00094 P.AAQ:  .ASCII  <0>
                                            49  41  20  03  00095 P.AAR:  .ASCII  <3>\ AI\
                                            49  42  20  03  00099 P.AAS:  .ASCII  <3>\ BI\
                                                    2C  01  0009D P.AAT:  .ASCII  <1>\,\
                                                    00  0009F P.AAU:  .ASCII  <0>
                                            54  41  20  03  000A0 P.AAV:  .ASCII  <3>\ AT\
                                                    2C  01  000A4 P.AAW:  .ASCII  <1>\,\
                                                    00  000A6 P.AAX:  .ASCII  <0>
                                            55  52  20  03  000A7 P.AAY:  .ASCII  <3>\ RU\
                                                    2C  01  000AB P.AAZ:  .ASCII  <1>\,\
                                                    00  000AD P.ABA:  .ASCII  <0>
                55  52  5F  59  4C  4E  4F  20  08  000AE P.ABB:  .ASCII  <8>\ ONLY_RU\
                                                    2C  01  000B7 P.ABC:  .ASCII  <1>\,\
                                                    00  000B9 P.ABD:  .ASCII  <0>
            55  52  5F  52  45  56  45  4E  20  09  000BA P.ABE:  .ASCII  <9>\ NEVER_RU\
                                                    2C  01  000C4 P.ABF:  .ASCII  <1>\,\
                                                    00  000C6 P.ABG:  .ASCII  <0>

                                                NULL=           P.AAE
                                                NONE=           P.AAF


                                                                .PSECT  $CODE$,NOWRT,2

                                                    0FFC 00000          .ENTRY  ANL$FORMAT_FILE_HEADER, Save R2,R3,R4,R5,-  ; 1099
                                                                                R6,R7,R8,R9,R10,R11
                                            5E                  04  C2  00002        SUBL2   #4, SP
                                                00000000G       8F  DD  00005        PUSHL   #ANLRMS$_FILEHDR                ; 1112
                                            7E                  03  7D  0000B        MOVQ    #3, -(SP)
                                    0000G   CF                  03  FB  0000E        CALLS   #3, ANL$FORMAT_LINE
                                                                7E  D4  00013        CLRL    -(SP)                           ; 1113
                                    0000G   CF                  01  FB  00015        CALLS   #1, ANL$FORMAT_SKIP
                                                    0000'       CF  9F  0001A        PUSHAB  RESULTANT_SPEC                  ; 1117
                                                00000000G       8F  DD  0001E        PUSHL   #ANLRMS$_FILESPEC
```

```
                                 01  DD 00024          PUSHL   #1
                                 7E  D4 00026          CLRL    -(SP)
                        0000G    CF     04 FB 00028    CALLS   #4, ANL$FORMAT_LINE
                                 7E  0000'  CF 3C 0002D MOVZWL  RMS_NAM+40, -(SP)                    1121
                                 7E  0000'  CF 3C 00032 MOVZWL  RMS_NAM+38, -(SP)
                                 7E  0000'  CF 3C 00037 MOVZWL  RMS_NAM+36, -(SP)
                                    00000000G 8F DD 0003C PUSHL   #AN[RMS$_FILEID
                                 01  DD 00042          PUSHL   #1
                                 7E  D4 00044          CLRL    -(SP)
                        0000G    CF     06 FB 00046    CALLS   #6, ANL$FORMAT_LINE
                                 7E  0000'  CF 3C 0004B MOVZWL  RMS_XABPRO+12, -(SP)                 1125
                                 7E  0000'  CF 3C 00050 MOVZWL  RMS_XABPRO+14, -(SP)
                                    00000000G 8F DD 00055 PUSHL   #AN[RMS$_OWNERUIC
                                 01  DD 0005B          PUSHL   #1
                                 7E  D4 0005D          CLRL    -(SP)
                        0000G    CF     05 FB 0005F    CALLS   #5, ANL$FORMAT_LINE
                                 7E  0000'  CF 3C 00064 MOVZWL  RMS_XABPRO+8, -(SP)                  1129
                                    00000000G 8F DD 00069 PUSHL   #AN[RMS$_PROT
                                 01  DD 0006F          PUSHL   #1
                        0000G    CF     03 FB 00071    CALLS   #3, ANL$FORMAT_PROTECTION_MASK
                                    0000'  CF 9F 00076 PUSHAB  RMS_XABDAT+20                        1134
                                    00000000G 8F DD 0007A PUSHL   #AN[RMS$_CREATION
                                 01  DD 00080          PUSHL   #1
                                 7E  D4 00082          CLRL    -(SP)
                        0000G    CF     04 FB 00084    CALLS   #4, ANL$FORMAT_LINE
                                 7E  0000'  CF 3C 00089 MOVZWL  RMS_XABDAT+8, -(SP)                  1135
                                    0000'  CF 9F 0008E PUSHAB  RMS_XABDAT+12
                                    00000000G 8F DD 00092 PUSHL   #AN[RMS$_REVISION
                                 01  DD 00098          PUSHL   #1
                                 7E  D4 0009A          CLRL    -(SP)
                        0000G    CF     05 FB 0009C    CALLS   #5, ANL$FORMAT_LINE
                                    0000'  CF D5 000A1 TSTL    RMS_XABDAT+28                        1136
                                 17  12 000A5          BNEQ    1$
                                    0000'  CF D5 000A7 TSTL    RMS_XABDAT+32
                                 11  12 000AB          BNEQ    1$
                                    00000000G 8F DD 000AD PUSHL   #ANLRMS$_NOEXPIRATION              1137
                                 01  DD 000B3          PUSHL   #1
                                 7E  D4 000B5          CLRL    -(SP)
                        0000G    CF     03 FB 000B7    CALLS   #3, ANL$FORMAT_LINE
                                 13  11 000BC          BRB     2$
                                    0000'  CF 9F 000BE 1$:  PUSHAB  RMS_XABDAT+28                   1139
                                    00000000G 8F DD 000C2 PUSHL   #AN[RMS$_EXPIRATION
                                 01  DD 000C8          PUSHL   #1
                                 7E  D4 000CA          CLRL    -(SP)
                        0000G    CF     04 FB 000CC    CALLS   #4, ANL$FORMAT_LINE
                                    0000'  CF D5 000D1 2$:  TSTL    RMS_XABDAT+36                   1140
                                 17  12 000D5          BNEQ    3$
                                    0000'  CF D5 000D7 TSTL    RMS_XABDAT+40
                                 11  12 000DB          BNEQ    3$
                                    00000000G 8F DD 000DD PUSHL   #ANLRMS$_NOBACKUP                  1141
                                 01  DD 000E3          PUSHL   #1
                                 7E  D4 000E5          CLRL    -(SP)
                        0000G    CF     03 FB 000E7    CALLS   #3, ANL$FORMAT_LINE
                                 13  11 000EC          BRB     4$
                                    0000'  CF 9F 000EE 3$:  PUSHAB  RMS_XABDAT+36                   1143
                                    00000000G 8F DD 000F2 PUSHL   #AN[RMS$_BACKUP
                                 01  DD 000F8          PUSHL   #1
```

```
                                7E  D4 000FA          CLRL    -(SP)                          ;
            0000G  CF           04  FB 000FC          CALLS   #4, ANL$FORMAT_LINE            ;
      06    0000'  CF           05  E1 00101 4$:      BBC     #5, RMS_FAB+6, 5$              ; 1148
                        0000'   CF  9F 00107          PUSHAB  P.AAG                          ;
                                10  11 0010B          BRB     7$                             ;
      06    0000'  CF           04  E1 0010D 5$:      BBC     #4, RMS_FAB+6, 6$              ; 1150
                        0000'   CF  9F 00113          PUSHAB  P.AAH                          ;
                                04  11 00117          BRB     7$                             ;
                        0000'   CF  9F 00119 6$:      PUSHAB  P.AAI                          ; 1152
                    00000000G   8F  DD 0011D 7$:      PUSHL   #ANLRMS$_CONTIG                ;
                                01  DD 00123          PUSHL   #1                             ;
                                7E  D4 00125          CLRL    -(SP)                          ;
            0000G  CF           04  FB 00127          CALLS   #4, ANL$FORMAT_LINE            ;
      06    0000'  CF           05  E1 0012C          BBC     #5, RMS_FAB+4, 8$              ; 1155
                        0000'   CF  9F 00132          PUSHAB  P.AAJ                          ; 1156
                                04  11 00136          BRB     9$                             ;
                        0000'   CF  9F 00138 8$:      PUSHAB  P.AAK                          ; 1158
                    00000000G   8F  DD 0013C 9$:      PUSHL   #ANLRMS$_PERFORM               ;
                                01  DD 00142          PUSHL   #1                             ;
                                7E  D4 00144          CLRL    -(SP)                          ;
            0000G  CF           04  FB 00146          CALLS   #4, ANL$FORMAT_LINE            ;
                        0000'   CF  95 0014B          TSTB    RMS_FAB+6                      ; 1160
                                12  18 0014F          BGEQ    11$                            ;
      06    0000'  CF           01  E1 00151          BBC     #1, RMS_FAB+5, 10$             ; 1161
                        0000'   CF  9F 00157          PUSHAB  P.AAL                          ; 1162
                                16  11 0015B          BRB     13$                            ;
                        0000'   CF  9F 0015D 10$:     PUSHAB  P.AAM                          ; 1164
                                10  11 00161          BRB     13$                            ;
      06    0000'  CF           01  E1 00163 11$:     BBC     #1, RMS_FAB+5, 12$             ; 1166
                        0000'   CF  9F 00169          PUSHAB  P.AAN                          ; 1167
                                04  11 0016D          BRB     13$                            ;
                        0000'   CF  9F 0016F 12$:     PUSHAB  P.AAO                          ; 1169
                    00000000G   8F  DD 00173 13$:     PUSHL   #ANLRMS$_RELIAB                ;
                                01  DD 00179          PUSHL   #1                             ;
                                7E  D4 0017B          CLRL    -(SP)                          ;
            0000G  CF           04  FB 0017D          CALLS   #4, ANL$FORMAT_LINE            ;
                        0000'   CF  B5 00182          TSTW    RMS_XABJNL+8                   ; 1174
                                2F  12 00186          BNEQ    14$                            ;
                        0000'   CF  9F 00188          PUSHAB  NULL                           ; 1175
                        0000'   CF  9F 0018C          PUSHAB  NULL                           ;
                        0000'   CF  9F 00190          PUSHAB  NULL                           ;
                        0000'   CF  9F 00194          PUSHAB  NULL                           ;
                        0000'   CF  9F 00198          PUSHAB  NULL                           ;
                        0000'   CF  9F 0019C          PUSHAB  NULL                           ;
                        0000'   CF  9F 001A0          PUSHAB  NULL                           ;
                        0000'   CF  9F 001A4          PUSHAB  NULL                           ;
                        0000'   CF  9F 001A8          PUSHAB  NULL                           ;
                        0000'   CF  9F 001AC          PUSHAB  NULL                           ;
                        0000'   CF  9F 001B0          PUSHAB  NONE                           ;
                          00DE  31 001B4              BRW     29$                            ;
             50         0000'   CF  9E 001B7 14$:     MOVAB   P.AAP, R0                      ; 1178
             56                 50  B0 001BC          MOVW    R0, NEVER_RU                   ;
             57                 50  D0 001BF          MOVL    R0, ONLY_RU                    ;
             58                 50  D0 001C2          MOVL    R0, RU                         ;
             59                 50  D0 001C5          MOVL    R0, AT                         ;
             5A                 50  D0 001C8          MOVL    R0, BI                         ;
             5B                 50  D0 001CB          MOVL    R0, AI                         ;
```

```
                     50      0000'  CF  9E 001CE          MOVAB   P.AAQ, R0                       ; 1179
                     55             50  B0 001D3          MOVW    R0, NEVER_RU_COMMA              ;
                     51             50  D0 001D6          MOVL    R0, ONLY_RU_COMMA               ;
                     52             50  D0 001D9          MOVL    R0, RU_COMMA                    ;
                     53             50  D0 001DC          MOVL    R0, AT_COMMA                    ;
                     54             50  D0 001DF          MOVL    R0, BI_COMMA                    ;
                 68      0000'  CF      03  E1 001E2      BBC     #3, RMS_XABJNL+8, 15$           ; 1181
                     5B      0000'  CF  9E 001E8          MOVAB   P.AAR, AI                       ; 1182
                     50             01  90 001ED          MOVB    #1, FLAG                        ; 1183
                 17      0000'  CF      02  E1 001F0  15$:  BBC     #2, RMS_XABJNL+8, 18$           ; 1186
                     5A      0000'  CF  9E 001F6          MOVAB   P.AAS, BI                       ; 1187
                     07             50  E9 001FB          BLBC    FLAG, 16$                       ; 1188
                     54      0000'  CF  9E 001FE          MOVAB   P.AAT, BI_COMMA                 ; 1189
                                    05  11 00203          BRB     17$                             ;
                     54      0000'  CF  9E 00205  16$:    MOVAB   P.AAU, BI_COMMA                 ; 1191
                     50             01  90 0020A  17$:    MOVB    #1, FLAG                        ; 1192
                 17      0000'  CF      04  E1 0020D  18$:  BBC     #4, RMS_XABJNL+8, 21$           ; 1195
                     59      0000'  CF  9E 00213          MOVAB   P.AAV, AT                       ; 1196
                     07             50  E9 00218          BLBC    FLAG, 19$                       ; 1197
                     53      0000'  CF  9E 0021B          MOVAB   P.AAW, AT_COMMA                 ; 1198
                                    05  11 00220          BRB     20$                             ;
                     53      0000'  CF  9E 00222  19$:    MOVAB   P.AAX, AT_COMMA                 ; 1200
                     50             01  90 00227  20$:    MOVB    #1, FLAG                        ; 1201
                 14      0000'  CF      01  E1 0022A  21$:  BBC     #1, RMS_XABJNL+8, 23$           ; 1204
                     58      0000'  CF  9E 00230          MOVAB   P.AAY, RU                       ; 1205
                     07             50  E9 00235          BLBC    FLAG, 22$                       ; 1206
                     52      0000'  CF  9E 00238          MOVAB   P.AAZ, RU_COMMA                 ; 1207
                                    05  11 0023D          BRB     23$                             ;
                     52      0000'  CF  9E 0023F  22$:    MOVAB   P.ABA, RU_COMMA                 ; 1209
                     14      0000'  CF  E9 00244  23$:    BLBC    RMS_XABJNL+8, 25$               ; 1212
                     57      0000'  CF  9E 00249          MOVAB   P.ABB, ONLY_RU                  ; 1213
                     07             50  E9 0024E          BLBC    FLAG, 24$                       ; 1214
                     51      0000'  CF  9E 00251          MOVAB   P.ABC, ONLY_RU_COMMA            ; 1215
                                    05  11 00256          BRB     25$                             ;
                     51      0000'  CF  9E 00258  24$:    MOVAB   P.ABD, ONLY_RU_COMMA            ; 1217
             1A      0000'  CF      05  E1 0025D  25$:    BBC     #5, RMS_XABJNL+8, 28$           ; 1220
                     6E      0000'  CF  9E 00263          MOVAB   P.ABE, (SP)                     ; 1221
                     56             6E  B0 00268          MOVW    (SP), NEVER_RU                  ;
                     07             50  E9 0026B          BLBC    FLAG, 26$                       ; 1222
                     50      0000'  CF  9E 0026E          MOVAB   P.ABF, R0                       ; 1223
                                    05  11 00273          BRB     27$                             ;
                     50      0000'  CF  9E 00275  26$:    MOVAB   P.ABG, R0                       ; 1225
                     55             50  B0 0027A  27$:    MOVW    R0, NEVER_RU_COMMA              ; 1222
                             0082   8F  BB 0027D  28$:    PUSHR   #^M<R1,R7>                      ; 1230
                     7E             56  3C 00281          MOVZWL  NEVER_RU, -(SP)                 ; 1229
                     7E             55  3C 00284          MOVZWL  NEVER_RU_COMMA, -(SP)           ;
                             0104   8F  BB 00287          PUSHR   #^M<R2,R8>                      ;
                             0208   8F  BB 0028B          PUSHR   #^M<R3,R9>                      ; 1228
                             0410   8F  BB 0028F          PUSHR   #^M<R4,R10>                     ;
                                    5B  DD 00293          PUSHL   AI                              ;
                         00000000G  8F  DD 00295  29$:    PUSHL   #ANLRMS$_JNL                    ;
                                    01  DD 0029B          PUSHL   #1                              ;
                                    7E  D4 0029D          CLRL    -(SP)                           ;
                 0000G   CF             0E  FB 0029F      CALLS   #14, ANL$FORMAT_LINE            ;
                             0000'  CF  95 002A4          TSTB    RMS_XABJNL+21                   ; 1234
                                    18  13 002A8          BEQL    30$                             ;
                             0000'  CF  DD 002AA          PUSHL   RMS_XABJNL+24                   ; 1236
```

```
                                7E      0000'   CF  9A 002AE          MOVZBL  RMS_XABJNL+20, -(SP)                ; 1235
                           00000000G    8F  DD 002B3                  PUSHL   #AN[RMS$_AIJNL                      ;
                                        01  DD 002B9                  PUSHL   #1                                  ;
                                        7E  D4 002BB                  CLRL    -(SP)                               ;
                                0000G   CF      05  FB 002BD          CALLS   #5, ANL$FORMAT_LINE                 ;
                                        0000'   CF  95 002C2 30$:     TSTB    RMS_XABJNL+13                       ; 1238
                                        18  13 002C6                  BEQL    31$                                 ;
                                        0000'   CF  DD 002C8          PUSHL   RMS_XABJNL+16                       ; 1240
                                7E      0000'   CF  9A 002CC          MOVZBL  RMS_XABJNL+12, -(SP)                ; 1239
                           00000000G    8F  DD 002D1                  PUSHL   #AN[RMS$_BIJNL                      ;
                                        01  DD 002D7                  PUSHL   #1                                  ;
                                        7E  D4 002D9                  CLRL    -(SP)                               ;
                                0000G   CF      05  FB 002DB          CALLS   #5, ANL$FORMAT_LINE                 ;
                                        0000'   CF  95 002E0 31$:     TSTB    RMS_XABJNL+29                       ; 1242
                                        18  13 002E4                  BEQL    32$                                 ;
                                        0000'   CF  DD 002E6          PUSHL   RMS_XABJNL+32                       ; 1244
                                7E      0000'   CF  9A 002EA          MOVZBL  RMS_XABJNL+28, -(SP)                ; 1243
                           00000000G    8F  DD 002EF                  PUSHL   #AN[RMS$_ATJNL                      ;
                                        01  DD 002F5                  PUSHL   #1                                  ;
                                        7E  D4 002F7                  CLRL    -(SP)                               ;
                                0000G   CF      05  FB 002F9          CALLS   #5, ANL$FORMAT_LINE                 ;
                                                04 002FE 32$:         RET                                         ; 1248

; Routine Size:  767 bytes,    Routine Base:  $CODE$ + 0391
```

```
;  751          1249  1 %sbttl 'ANL$FDL_FILE - Generate FILE Primary for FDL'
;  752          1250  1 !++
;  753          1251  1 ! Functional Description:
;  754          1252  1 !       This routine is responsible for generating the file primary of an
;  755          1253  1 !       FDL spec.  The file primary describes the attributes of the file.
;  756          1254  1 !       The rest of the FDL routines are in module RMSFDL.
;  757          1255  1 !
;  758          1256  1 ! Formal Parameters:
;  759          1257  1 !       none
;  760          1258  1 !
;  761          1259  1 ! Implicit Inputs:
;  762          1260  1 !       global data
;  763          1261  1 !
;  764          1262  1 ! Implicit Outputs:
;  765          1263  1 !       global data
;  766          1264  1 !
;  767          1265  1 ! Returned Value:
;  768          1266  1 !       none
;  769          1267  1 !
;  770          1268  1 ! Side Effects:
;  771          1269  1 !
;  772          1270  1 !--
;  773          1271  1
;  774          1272  1
;  775          1273  2 global routine anl$fdl_file: novalue = begin
;  776          1274  2
;  777          1275  2 own
;  778          1276  2         yes_no: vector[2,long] initial(
;  779          1277  2                         uplit byte (%ascic 'no'),
;  780          1278  2                         uplit byte (%ascic 'yes')
;  781          1279  2                         ),
;  782          1280  2         cluster_size: long;
;  783          1281  2
;  784          1282  2 local
;  785          1283  2         p: bsd,
;  786          1284  2         pp: ref block[,byte];
;  787          1285  2
;  788          1286  2 bind
;  789          1287  2         get_cluster_size = uplit(word(4),word(dvi$_cluster),
;  790          1288  2                                  long(cluster_size),
;  791          1289  2                                  long(0),
;  792          1290  2                                  long(0));
;  793          1291  2
;  794          1292  2
;  795          1293  2 ! All we have to do is put out a line for each item in the file primary.
;  796          1294  2 ! We cannot, of course, put out lines for items which are not stored in
;  797          1295  2 ! the file header or RMS attribute area.
;  798          1296  2
;  799          1297  2 anl$format_line(0,0,anlrms$_fdlfile);
;  800          1298  2 anl$format_line(0,1,anlrms$_fdlalloc,.anl$gl_fat[fat$l_hiblk]);
;  801          1299  2 anl$format_line(0,1,anlrms$_fdlbesttry,.yes_no[.rms_fab[fab$v_cbt] and 1]);
;  802          1300  2 if .anl$gl_fat[fat$v_fileorg] eqlu fat$c_relative or
;  803          1301  2    .anl$gl_fat[fat$v_fileorg] eqlu fat$c_indexed        then
;  804          1302  2         anl$format_line(0,1,anlrms$_fdlbucketsize,.anl$gl_fat[fat$b_bktsize]);
;  805          1303  2
;  806          1304  2 ! To get the cluster size for the devie on which the file resides, we use
;  807          1305  2 ! the $GETDVI system service.  The device name is stored in the NAM block.
```

```
;  808          1306  2
;  809          1307  3 begin
;  810          1308  3 local
;  811          1309  3         device_dsc: descriptor;
;  812          1310  3
;  813          1311  3 cluster_size = 0;
;  814          1312  3 build_descriptor(device_dsc,.rms_nam[nam$b_dev],.rms_nam[nam$l_dev]);
;  815        P 1313  3 $getdvi(efn=1,
;  816        P 1314  3         devnam=device_dsc,
;  817          1315  3         itmlst=get_cluster_size);
;  818          1316  3 $waitfr(efn=1);
;  819          1317  3 anl$format_line(0,1,anlrms$_fdlclustersize,.cluster_size);
;  820          1318  2 end;
;  821          1319  2
;  822          1320  2 anl$format_line(0,1,anlrms$_fdlcontig,
;  823          1321  2                 .yes_no[not .rms_fab[fab$v_cbt] and .rms_fab[fab$v_ctg] and 1]);
;  824          1322  2 anl$format_line(0,1,anlrms$_fdlextension,.anl$gl_fat[fat$w_defext]);
;  825          1323  2 anl$format_line(0,1,anlrms$_fdlglobalbufs,.rms_fab[fab$w_gbc]);
;  826          1324  2
;  827          1325  2 ! To put out the maximum record number for relative files, we have to get
;  828          1326  2 ! the prolog block.  Set up a BSD, read it, and put out the line.
;  829          1327  2
;  830          1328  3 if .anl$gl_fat[fat$v_fileorg] eqlu fat$c_relative then (
;  831          1329  3         init_bsd(p);
;  832          1330  3         p[bsd$w_size] = 1;
;  833          1331  3         p[bsd$l_vbn] = 1;
;  834          1332  3         anl$bucket(p,0);
;  835          1333  3         pp = .p[bsd$l_bufptr];
;  836          1334  3         anl$format_line(0,1,anlrms$_fdlmaxrecord,.pp[plg$l_mrn]);
;  837          1335  3         anl$bucket(p,-1);
;  838          1336  2 );
;  839          1337  2
;  840          1338  2 ! For the file name, we have to produce a quoted string containing
;  841          1339  2 ! the name.  This goes in the output line along with the NAME keyword.
;  842          1340  2
;  843          1341  3 begin
;  844          1342  3 local
;  845          1343  3         local_described_buffer(string_buf,nam$c_maxrss+4);
;  846          1344  3
;  847          1345  3 anl$prepare_quoted_string(resultant_spec,string_buf);
;  848          1346  3 anl$format_line(0,1,anlrms$_fdlfilename,string_buf);
;  849          1347  2 end;
;  850          1348  2
;  851          1349  2 anl$format_line(0,1,anlrms$_fdlorg,
;  852          1350  3                 (selectoneu .anl$gl_fat[fat$v_fileorg] of set
;  853          1351  3                 [fat$c_sequential]:     uplit byte (%ascic 'sequential');
;  854          1352  3                 [fat$c_relative]:       uplit byte (%ascic 'relative');
;  855          1353  3                 [fat$c_indexed]:        uplit byte (%ascic 'indexed');
;  856          1354  2                 tes));
;  857          1355  2 anl$format_line(0,1,anlrms$_fdlowner,.rms_xabpro[xab$w_grp],.rms_xabpro[xab$w_mbm]);
;  858          1356  2 anl$format_protection_mask(1,anlrms$_fdlprotection,.rms_xabpro[xab$w_pro]);
;  859          1357  2
;  860          1358  2 return;
;  861          1359  2
;  862          1360  1 end;
```

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

                         6F 6E 02  000C7 P.ABH:   .ASCII  <2>\no\                          ;
                      73 65 79 03  000CA P.ABI:   .ASCII  <3>\yes\                         ;
                                   000CE          .BLKB   2
                             0004  000D0 P.ABJ:   .WORD   4                                ;
                             003A  000D2          .WORD   58                               ;
                        00000000'  000D4          .ADDRESS CLUSTER_SIZE                    ;
                         00000000  000D8          .LONG   0                                ;
                         00000000  000DC          .LONG   0                                ;
 6C 61 69 74 6E 65 75 71 65 73 0A  000E0 P.ABK:   .ASCII  <10>\sequential\                 ;
          65 76 69 74 61 6C 65 72 08  000EB P.ABL:   .ASCII  <8>\relative\                 ;
             64 65 78 65 64 6E 69 07  000F4 P.ABM:   .ASCII  <7>\indexed\                  ;

                                                          .PSECT  $OWN$,NOEXE,2

              00000000' 00000000'  0066C YES_NO:  .ADDRESS P.ABH, P.ABI                    ;
                                   00674 CLUSTER_SIZE:
                                                  .BLKB   4

                                         GET_CLUSTER_SIZE=   P.ABJ
                                                  .EXTRN  SYS$GETDVI, SYS$WAITFR

                                                          .PSECT  $CODE$,NOWRT,2

                          01FC  00000             .ENTRY  ANL$FDL_FILE, Save R2,R3,R4,R5,R6,R7,R8    ; 1273
              58      0000'  CF  9E  00002        MOVAB   ANL$GL_FAT, R8                     ;
              57      0000G  CF  9E  00007        MOVAB   ANL$FORMAT_LINE, R7                ;
              56      0000'  CF  9E  0000C        MOVAB   RMS_FAB+4, R6                      ;
              5E      FEDC   CE  9E  00011        MOVAB   -292(SP), SP                       ;
                  00000000G  8F  DD  00016        PUSHL   #ANLRMS$_FDLFILE                   ; 1297
                             7E  7C  0001C        CLRQ    -(SP)                              ;
              67             03  FB  0001E        CALLS   #3, ANL$FORMAT_LINE                ;
              50             68  D0  00021        MOVL    ANL$GL_FAT, R0                     ; 1298
                        04   A0  DD  00024        PUSHL   4(R0)                              ;
                  00000000G  8F  DD  00027        PUSHL   #ANLRMS$_FDLALLOC                  ;
                             01  DD  0002D        PUSHL   #1                                 ;
                             7E  D4  0002F        CLRL    -(SP)                              ;
              67             04  FB  00031        CALLS   #4, ANL$FORMAT_LINE                ;
 50      02  A6   01             05  EF  00034        EXTZV   #5, #1, RMS_FAB+6, R0              ; 1299
                     02E4 C640  DD  0003A        PUSHL   YES_NO[R0]                         ;
                  00000000G  8F  DD  0003F        PUSHL   #ANLRMS$_FDLBESTTRY                ;
                             01  DD  00045        PUSHL   #1                                 ;
                             7E  D4  00047        CLRL    -(SP)                              ;
              67             04  FB  00049        CALLS   #4, ANL$FORMAT_LINE                ;
              50             68  D0  0004C        MOVL    ANL$GL_FAT, R0                     ; 1300
 01           60  04         04  ED  0004F        CMPZV   #4, #4, (R0), #1                   ;
                             07  13  00054        BEQL    1$                                 ;
 02           60  04         04  ED  00056        CMPZV   #4, #4, (R0), #2                   ; 1301
                             11  12  0005B        BNEQ    2$                                 ;
              7E      0E     A0  9A  0005D 1$:    MOVZBL  14(R0), -(SP)                      ; 1302
                  00000000G  8F  DD  00061        PUSHL   #ANLRMS$_FDLBUCKETSIZE             ;
                             01  DD  00067        PUSHL   #1                                 ;
                             7E  D4  00069        CLRL    -(SP)                              ;
              67             04  FB  0006B        CALLS   #4, ANL$FORMAT_LINE                ;
                      02EC   C6  D4  0006E 2$:    CLRL    CLUSTER_SIZE                       ; 1311
```

```
                                 E0  AD       D5  A6  9A 00072          MOVZBL  RMS_NAM+57, DEVICE_DSC                  ; 1312
                                 E4  AD       E0  A6  D0 00077          MOVL    RMS_NAM+68, DEVICE_DSC+4
                                                  7E  7C 0007C          CLRQ    -(SP)                                   ; 1315
                                                  7E  7C 0007E          CLRQ    -(SP)
                                           0000'  CF  9F 00080          PUSHAB  GET_CLUSTER_SIZE
                                             E0   AD  9F 00084          PUSHAB  DEVICE_DSC
                                     7E              01  7D 00087          MOVQ    #1, -(SP)
                          00000000G  00              08  FB 0008A          CALLS   #8, SYS$GETDVI                          ; 0
                                                     01  DD 00091          PUSHL   #1                                      ; 1316
                          00000000G  00              01  FB 00093          CALLS   #1, SYS$WAITFR                          ; 0
                                        02EC         C6  DD 0009A          PUSHL   CLUSTER_SIZE                            ; 1317
                                   00000000G         8F  DD 0009E          PUSHL   #ANLRMS$_FDLCLUSTERSIZE                 ; 0
                                                     01  DD 000A4          PUSHL   #1                                      ; 0
                                                     7E  D4 000A6          CLRL    -(SP)                                   ; 0
                                     67              04  FB 000A8          CALLS   #4, ANL$FORMAT_LINE                     ; 0
     50      02   A6                 01              04  EF 000AB          EXTZV   #4, #1, RMS_FAB+6, R0                   ; 1321
     51      02   A6                 01              05  EF 000B1          EXTZV   #5, #1, RMS_FAB+6, R1
                                     50              51  CA 000B7          BICL2   R1, R0
     50           50                 01              00  EF 000BA          EXTZV   #0, #1, R0, R0
                                           02E4 C640  DD 000BF          PUSHL   YES_NO[R0]
                                   00000000G         8F  DD 000C4          PUSHL   #AN[RMS$_FDLCONTIG                      ; 1320
                                                     01  DD 000CA          PUSHL   #1                                      ; 0
                                                     7E  D4 000CC          CLRL    -(SP)                                   ; 0
                                     67              04  FB 000CE          CALLS   #4, ANL$FORMAT_LINE                     ; 0
                                     50              68  D0 000D1          MOVL    ANL$GL_FAT, R0                          ; 1322
                                     7E        12    A0  3C 000D4          MOVZWL  18(R0), -(SP)                           ; 0
                                   00000000G         8F  DD 000D8          PUSHL   #ANLRMS$_FDLEXTENSION                   ; 0
                                                     01  DD 000DE          PUSHL   #1                                      ; 0
                                                     7E  D4 000E0          CLRL    -(SP)                                   ; 0
                                     67              04  FB 000E2          CALLS   #4, ANL$FORMAT_LINE                     ; 0
                                     7E        44    A6  3C 000E5          MOVZWL  RMS_FAB+72, -(SP)                       ; 1323
                                   00000000G         8F  DD 000E9          PUSHL   #AN[RMS$_FDLGLOBALBUFS                  ; 0
                                                     01  DD 000EF          PUSHL   #1                                      ; 0
                                                     7E  D4 000F1          CLRL    -(SP)                                   ; 0
                                     67              04  FB 000F3          CALLS   #4, ANL$FORMAT_LINE                     ; 0
     01      00   B8                 04              04  ED 000F6          CMPZV   #4, #4, @ANL$G[_FAT, #1                 ; 1328
                                                     38  12 000FC          BNEQ    3$
     18           00                 6E              00  2C 000FE          MOVC5   #0, (SP), #0, #24, P                    ; 1329
                                           E8    AD      00103
                                 EA  AD                  01  B0 00105          MOVW    #1, P+2                                 ; 1330
                                 EC  AD                  01  D0 00109          MOVL    #1, P+4                                 ; 1331
                                                         7E  D4 0010D          CLRL    -(SP)                                   ; 1332
                                           E8    AD      9F  0010F          PUSHAB  P                                       ; 0
                               FA17  CF                  02  FB 00112          CALLS   #2, ANL$BUCKET                          ; 0
                                     50        F4    AD  D0 00117          MOVL    P+12, PP                                ; 1333
                                               6C    A0  DD 0011B          PUSHL   108(PP)                                 ; 1334
                                   00000000G         8F  DD 0011E          PUSHL   #ANLRMS$_FDLMAXRECORD                   ; 0
                                                     01  DD 00124          PUSHL   #1                                      ; 0
                                                     7E  D4 00126          CLRL    -(SP)                                   ; 0
                                     67              04  FB 00128          CALLS   #4, ANL$FORMAT_LINE                     ; 0
                                     7E              01  CE 0012B          MNEGL   #1, -(SP)                               ; 1335
                                           E8    AD  9F  0012E          PUSHAB  P                                       ; 0
                               F9F8  CF                  02  FB 00131          CALLS   #2, ANL$BUCKET                          ; 0
                                     6E        0103  8F  3C 00136  3$:    MOVZWL  #259, STRING_BUF                        ; 1343
                                   04  AE        08  AE  9E 0013B          MOVAB   STRING_BUF+8, STRING_BUF+4
                                                     5E  DD 00140          PUSHL   SP                                      ; 1345
                                           FC78  C6  9F  00142          PUSHAB  RESULTANT_SPEC                          ; 0
```

```
                    0000G  CF              02  FB 00146          CALLS   #2, ANL$PREPARE_QUOTED_STRING
                                            5E  DD 0014B          PUSHL   SP                                          1346
                        00000000G           8F  DD 0014D          PUSHL   #ANLRMS$_FDLFILENAME
                                            01  DD 00153          PUSHL   #1
                                            7E  D4 00155          CLRL    -(SP)
                    67                      04  FB 00157          CALLS   #4, ANL$FORMAT_LINE
          51    00  B8                      04  EF 0015A          EXTZV   #4, #4, @ANL$GL_FAT, R1                     1350
                                            07  12 00160          BNEQ    4$                                          1351
                    50      0000'           CF  9E 00162          MOVAB   P.ABK, R0
                                            1B  11 00167          BRB     7$
                    01                      51  D1 00169 4$:      CMPL    R1, #1                                      1352
                                            07  12 0016C          BNEQ    5$
                    50      0000'           CF  9E 0016E          MOVAB   P.ABL, R0
                                            0F  11 00173          BRB     7$
                    02                      51  D1 00175 5$:      CMPL    R1, #2                                      1353
                                            05  13 00178          BEQL    6$
                    7E                      01  CE 0017A          MNEGL   #1, -(SP)
                                            07  11 0017D          BRB     8$
                    50      0000'           CF  9E 0017F 6$:      MOVAB   P.ABM, R0
                                            50  DD 00184 7$:      PUSHL   R0
                        00000000G           8F  DD 00186 8$:      PUSHL   #ANLRMS$_FDLORG                             1349
                                            01  DD 0018C          PUSHL   #1
                                            7E  D4 0018E          CLRL    -(SP)
                    67                      04  FB 00190          CALLS   #4, ANL$FORMAT_LINE
                    7E      FE24            C6  3C 00193          MOVZWL  RMS_XABPRO+12, -(SP)                        1355
                    7E      FE26            C6  3C 00198          MOVZWL  RMS_XABPRO+14, -(SP)
                        00000000G           8F  DD 0019D          PUSHL   #ANLRMS$_FDLOWNER
                                            01  DD 001A3          PUSHL   #1
                                            7E  D4 001A5          CLRL    -(SP)
                    67                      05  FB 001A7          CALLS   #5, ANL$FORMAT_LINE
                    7E      FE20            C6  3C 001AA          MOVZWL  RMS_XABPRO+8, -(SP)                         1356
                        00000000G           8F  DD 001AF          PUSHL   #ANLRMS$_FDLPROTECTION
                                            01  DD 001B5          PUSHL   #1
                0000G  CF                   03  FB 001B7          CALLS   #3, ANL$FORMAT_PROTECTION_MASK
                                                04 001BC          RET                                                 1360

; Routine Size:  445 bytes,    Routine Base:  $CODE$ + 0690

;  863          1361  1
;  864          1362  0 end eludom


                                                                   .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  $GLOBAL$                         6  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                         1656  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         252  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        2125  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       133         0      1000          00:01.8
```

```
;                                      COMMAND QUALIFIERS

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

; Size:         2125 code + 1914 data bytes
; Run Time:       00:39.0
; Elapsed Time:   02:07.9
; Lines/CPU Min:   2096
; Lexemes/CPU-Min: 21728
; Memory Used:  355 pages
; Compilation Complete
```

RMSINTER
LIS

RMSCHECKA
LIS

RMSFDL
LIS

RMSCHECKB
LIS

RMSINPUT
LIS

RMSMSG
LIS