```
   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  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
RRRRRRRR    MM      MM    SSSSSSSS    IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
RR      RR  MMMM  MMMM  SS                II    NN      NN      TT      EE          RR      RR
RR      RR  MMMM  MMMM  SS                II    NN      NN      TT      EE          RR      RR
RR      RR  MM  MM  MM  SS                II    NNNN    NN      TT      EE          RR      RR
RR      RR  MM  MM  MM  SS                II    NNNN    NN      TT      EE          RR      RR
RRRRRRRR    MM      MM    SSSSSS          II    NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
RRRRRRRR    MM      MM    SSSSSS          II    NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
RR  RR      MM      MM          SS        II    NN    NNNN      TT      EE          RR  RR
RR  RR      MM      MM          SS        II    NN    NNNN      TT      EE          RR  RR
RR    RR    MM      MM          SS        II    NN      NN      TT      EE          RR    RR        ....
RR    RR    MM      MM          SS        II    NN      NN      TT      EE          RR    RR        ....
RR      RR  MM      MM    SSSSSSSS    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR      ....
RR      RR  MM      MM    SSSSSSSS    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR      ....

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 'RMSINTER - Interactive Analysis Mode'
;  2     0002  0         module rmsinter (
;  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  RELI BILITY 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, Interactive Analysis Mode
; 32     0032  1 !
; 33     0033  1 ! Abstract:     This module handles the interactive mode of analysis
; 34     0034  1 !               requested via the /INTERACTIVE qualifier.  This mode
; 35     0035  1 !               allows the user to interactively peruse the structure
; 36     0036  1 !               of any RMS file.
; 37     0037  1 !
; 38     0038  1 !
; 39     0039  1 ! Environment:
; 40     0040  1 !
; 41     0041  1 ! Author: Paul C. Anagnostopoulos, Creation Date:  20 May 1981
; 42     0042  1 !
; 43     0043  1 ! Modified By:
; 44     0044  1 !
; 45     0045  1 !       V03-006 DGB0050         Donald G. Blair         08-May-1984
; 46     0046  1 !               Fix condition handling so ANALYZRMS returns the correct
; 47     0047  1 !               error status at image exit.  Change condition handler
; 48     0048  1 !               from ANL$CONDITION_HANDLER to ANL$UNWIND_HANDLER.
; 49     0049  1 !
; 50     0050  1 !       V03-005 PCA1012         Paul C. Anagnostopoulos  6-Apr-1983
; 51     0051  1 !               Remove redundant cases from ANL$INTERACTIVE_DOWN, so that
; 52     0052  1 !               common algorithms for moving down from a structure are
; 53     0053  1 !               not repeated.
; 54     0054  1 !
; 55     0055  1 !       V03-004 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
; 56     0056  1 !               Change the message prefix to ANLRMS$_ to ensure that
; 57     0057  1 !               message symbols are unique across all ANALYZEs.  This
```

```
;    58          0058  1 !                    is necessitated by the new merged message files.
;    59          0059  1 !
;    60          0060  1 !        V03-003 PCA1007         Paul C. Anagnostopoulos 10 Feb 1983
;    61          0061  1 !                Needed to make a small change to the way deleted primary
;    62          0062  1 !                data records were detected in prologue 3 files.  This
;    63          0063  1 !                change was necessitated by recovery unit enhancements.
;    64          0064  1 !
;    65          0065  1 !        V03-002 PCA1001         Paul C. Anagnostopoulos 12-Oct-1982
;    66          0066  1 !                Add code to support SIDR records for prologue 3 indexed
;    67          0067  1 !                files.
;    68          0068  1 !
;    69          0069  1 !        V03-001 PCA0010         Paul Anagnostopoulos    16-Mar-1982
;    70          0070  1 !                Fix the code that goes down into the buckets of a
;    71          0071  1 !                relative file.  There may not be any.
;    72          0072  1 !--
```

```
;    74          0073  1 %sbttl 'Module Declarations'
;    75          0074  1 !
;    76          0075  1 ! Libraries and Requires:
;    77          0076  1 !
;    78          0077  1
;    79          0078  1 library 'lib';
;    80          0079  1 library 'tpamac';
;    81          0080  1 require 'rmsreq';
;    82          0589  1
;    83          0590  1 !
;    84          0591  1 ! Table of Contents:
;    85          0592  1 !
;    86          0593  1
;    87          0594  1 forward routine
;    88          0595  1         anl$interactive_mode: novalue,
;    89          0596  1         anl$interactive_driver: novalue,
;    90          0597  1         anl$interactive_command: novalue,
;    91          0598  1         anl$interactive_display: novalue,
;    92          0599  1         anl$interactive_down,
;    93          0600  1         anl$interactive_dump: novalue,
;    94          0601  1         anl$interactive_help: novalue;
;    95          0602  1
;    96          0603  1 !
;    97          0604  1 ! External References:
;    98          0605  1 !
;    99          0606  1
;   100          0607  1 external routine
;   101          0608  1         anl$area_descriptor,
;   102          0609  1         anl$bucket,
;   103          0610  1         anl$2bucket_header,
;   104          0611  1         anl$3bucket_header,
;   105          0612  1         anl$3format_data_bytes,
;   106          0613  1         anl$format_file_attributes,
;   107          0614  1         anl$format_file_header,
;   108          0615  1         anl$format_hex,
;   109          0616  1         anl$format_line,
;   110          0617  1         anl$format_skip,
;   111          0618  1         anl$idx_prolog,
;   112          0619  1         anl$unwind_handler,
;   113          0620  1         anl$2index_record,
;   114          0621  1         anl$3index_record,
;   115          0622  1         anl$internalize_number,
;   116          0623  1         anl$key_descriptor,
;   117          0624  1         anl$open_next_rms_file,
;   118          0625  1         anl$prepare_report_file,
;   119          0626  1         anl$2primary_data_record,
;   120          0627  1         anl$3primary_data_record,
;   121          0628  1         anl$3reclaimed_bucket_header,
;   122          0629  1         anl$rel_cell,
;   123          0630  1         anl$rel_prolog,
;   124          0631  1         anl$seq_data_record,
;   125          0632  1         anl$2sidr_pointer,
;   126          0633  1         anl$3sidr_pointer,
;   127          0634  1         anl$2sidr_record,
;   128          0635  1         anl$3sidr_record,
;   129          0636  1         cli$get_value: addressing_mode(general),
;   130          0637  1         lbr$output_help: addressing_mode(general),
```

```
;  131          0638  1          lib$establish: addressing_mode(general),
;  132          0639  1          lib$get_input: addressing_mode(general),
;  133          0640  1          lib$put_output: addressing_mode(general),
;  134          0641  1          lib$tparse: addressing_mode(general),
;  135          0642  1          str$upcase: addressing_mode(general);
;  136          0643  1
;  137          0644  1  external literal
;  138          0645  1          lib$_syntaxerr;
;  139          0646  1
;  140          0647  1  external
;  141          0648  1          anl$gl_fat: ref block[,byte],
;  142          0649  1          anl$gw_prolog: word;
;  143          0650  1
;  144          0651  1  !
;  145          0652  1  ! Macro Definitions:
;  146          0653  1  !
;  147          0654  1  ! The following macro is simply a shorthand:
;  148          0655  1
;  149          0656  1  macro text[] = uplit byte (%ascic %remaining) %;
```

```
;  151     0657  1 !
;  152     0658  1 ! Own Variables:
;  153     0659  1 !
;  154     0660  1 ! The following two tables control the interactive perusal of a file by
;  155     0661  1 ! describing the hierarchical structure of the three RMS file types.
;  156     0662  1 !
;  157     0663  1 ! The first table describes each of the structures in an RMS file.
;  158     0664  1 ! For our purposes, a structure is basically defined as any context in
;  159     0665  1 ! which we are able to discretely display an identifiable piece of a file.
;  160     0666  1 ! Examples are the RMS file attribute area or a indexed file key descriptor.
;  161     0667  1 ! THE INDICES OF ENTRIES IN THIS TABLE ARE USED IN THE BSD AS THE
;  162     0668  1 ! STRUCTURE TYPE INDICATOR.
;  163     0669  1
;  164     0670  1 ! There is a vector of four items for each table entry, as follows:
;  165     0671  1 !       0)      The number of a routine that can effect the display
;  166     0672  1 !               of this structure (routines reside in ANL$INTERACTIVE_DISPLAY).
;  167     0673  1 !       1-3)    A list of 0-3 indices into the PATH_TABLE.  This list
;  168     0674  1 !               defines the ways in which you can go down from this structure.
;  169     0675  1
;  170     0676  1 structure matrix[i,j; n] =
;  171     0677  1                 [n*4]
;  172     0678  1                 (matrix+(i*4+j))<0,8,0>;
;  173     0679  1
;  174     0680  1 own
;  175     0681  1         structure_table: matrix[35] initial(byte (
;  176     0682  1                 0,0,0,0,                                ! 0 - unused
;  177     0683  1                  1,     1,0,0,                          ! 1 - File header
;  178     0684  1                  2,     2,0,0,                          ! 2 - RMS attributes
;  179     0685  1                  3,     0,0,0,                          ! 3 - Seq rec
;  180     0686  1                  4,     3,0,0,                          ! 4 - Rel prolog
;  181     0687  1                  5,     4,0,0,                          ! 5 - Rel bkts
;  182     0688  1                  6,     0,0,0,                          ! 6 - Rel cells
;  183     0689  1                  7,     5,6,0,                          ! 7 - Idx prolog
;  184     0690  1                  8,    23,0,0,                          ! 8 - Idx area descriptor
;  185     0691  1                  9,     7,8,0,                          ! 9 - Idx key descriptor
;  186     0692  1                 10,     9,0,0,                          !10 - 2Idx primary index bkt
;  187     0693  1                 11,     9,0,0,                          !11 - 2Idx secondary index bkt
;  188     0694  1                 12,    11,0,0,                          !12 - 2Idx primary data bkt
;  189     0695  1                 13,    14,0,0,                          !13 - 2Idx SIDR bkt
;  190     0696  1                 14,    10,0,0,                          !14 - 2Idx primary index rec
;  191     0697  1                 15,    10,0,0,                          !15 - 2Idx secondary index rec
;  192     0698  1                 16,    12,13,0,                         !16 - 2Idx primary data rec
;  193     0699  1                 17,     0,0,0,                          !17 - 2Idx actual data bytes
;  194     0700  1                 18,    15,0,0,                          !18 - 2Idx SIDR rec
;  195     0701  1                 19,     0,0,0,                          !19 - 2Idx SIDR pointer
;  196     0702  1                 20,    16,0,0,                          !20 - 3Idx primary index bkt
;  197     0703  1                 21,    16,0,0,                          !21 - 3Idx secondary index bkt
;  198     0704  1                 22,    18,0,0,                          !22 - 3Idx primary data bkt
;  199     0705  1                 23,    21,0,0,                          !23 - 3Idx SIDR bkt
;  200     0706  1                 24,    17,0,0,                          !24 - 3Idx primary index rec
;  201     0707  1                 25,    17,0,0,                          !25 - 3Idx secondary index rec
;  202     0708  1                 26,    19,20,0,                         !26 - 3Idx primary data rec
;  203     0709  1                 27,     0,0,0,                          !27 - 3Idx actual data bytes
;  204     0710  1                 28,    22,0,0,                          !28 - 3Idx SIDR rec
;  205     0711  1                 29,     0,0,0,                          !29 - 3Idx SIDR pointer
;  206     0712  1                 30,     0,0,0                           !30 - Idx reclaimed bkt
;  207     0713  1                 ));
```

```
;  208          0714  1
;  209          0715  1
;  210          0716  1 ! This second table contains an entry for each downward path in the file
;  211          0717  1 ! structure.  A downward path is a method for descending from a given
;  212          0718  1 ! structure in the file down deeper to a new structure in the file.
;  213          0719  1 ! An example is the pointer from an index entry to its associated data
;  214          0720  1 ! bucket.
;  215          0721  1 !
;  216          0722  1 ! Each entry in the table contains four items, as follows:
;  217          0723  1 !       0)      The symbolic name of the path.
;  218          0724  1 !       1)      A short description of the path.
;  219          0725  1 !       2)      The number of the routine that can effect the downward
;  220          0726  1 !               movement (routines are in ANL$INTERACTIVE_DOWN).
;  221          0727  1 !       3)      The number of the entry in the STRUCTURE_TABLE that
;  222          0728  1 !               describes where you end up when you go down.
;  223          0729  1 !               If zero, the value is computed in ANL$INTERACTIVE_DOWN.
;  224          0730  1
;  225          0731  1 field path_fields = set
;  226          0732  1         path_name       = [0,0,32,0],
;  227          0733  1         path_text       = [4,0,32,0],
;  228          0734  1         path_routine    = [8,0, 8,0],
;  229          0735  1         path_result     = [9,0, 8,0]
;  230          0736  1 tes;
;  231          0737  1
;  232          0738  1 own
;  233          0739  1         path_table: blockvector[25,10,byte] field(path_fields) initial(
;  234          0740  1 0, 0,                                                  byte(0,0),      !unused
;  235          0741  1 text('ATTRIBUTES'), text('RMS file attribute area'),   byte(1,2),      !  1
;  236          0742  1 text('BLOCKS'), text('Depends on file organization'),  byte(2,0),      !  2
;  237          0743  1 text('BUCKETS'), text('Data buckets'),                 byte(3,5),      !  3
;  238          0744  1 text('CELLS'), text('Record cells'),                   byte(4,6),      !  4
;  239          0745  1 text('AREAS'), text('Area descriptors'),               byte(5,8),      !  5
;  240          0746  1 text('KEYS'), text('Key descriptors'),                 byte(6,9),      !  6
;  241          0747  1 text('INDEX'), text('Root index bucket'),              byte(7,0),      !  7
;  242          0748  1 text('DATA'), text('Data buckets'),                    byte(8,0),      !  8
;  243          0749  1 text('RECORDS'), text('Index records'),                byte(9,0),      !  9
;  244          0750  1 text('DEEPER'), text('Index or data buckets'),         byte(10,0),     !  10
;  245          0751  1 text('RECORDS'), text('Primary data records'),         byte(11,16),    !  11
;  246          0752  1 text('BYTES'), text('Actual data record bytes'),       byte(12,17),    !  12
;  247          0753  1 text('RRV'), text('RRV data bucket'),                  byte(13,12),    !  13
;  248          0754  1 text('SIDRS'), text('SIDR record'),                    byte(14,18),    !  14
;  249          0755  1 text('POINTER'), text('Record pointer'),               byte(15,19),    !  15
;  250          0756  1 text('RECORDS'), text('Index records'),                byte(16,0),     !  16
;  251          0757  1 text('DEEPER'), text('Index or data buckets'),         byte(17,0),     !  17
;  252          0758  1 text('RECORDS'), text('Primary data records'),         byte(11,26),    !  18
;  253          0759  1 text('BYTES'), text('Actual data record bytes'),       byte(18,27),    !  19
;  254          0760  1 text('RRV'), text('RRV data bucket'),                  byte(19,22),    !  20
;  255          0761  1 text('SIDRS'), text('SIDR record'),                    byte(14,28),    !  21
;  256          0762  1 text('POINTER'), text('Record pointer'),               byte(21,29),    !  22
;  257          0763  1 text('RECLAIMED'), text('Reclaimed buckets'),          byte(22,30)     !  23
;  258          0764  1 );
;  259          0765  1
;  260          0766  1 ! The hierarchical perusal of the file will be controlled by three stacks
;  261          0767  1 ! of BSDs.  FIRST_STACK contains BSDs that describe the first structure
;  262          0768  1 ! that we encountered on a given level when we went down to it.
;  263          0769  1 ! CURRENT_STACK describes the current structure on a given level.
;  264          0770  1 ! NEXT_STACK describes the next structure that we will encounter on a
```

```
;  265          0771  1 ! given level.
;  266          0772  1
;  267          0773  1 literal
;  268          0774  1         stack_size = 32;
;  269          0775  1 own
;  270          0776  1         top: signed long initial(0),
;  271          0777  1         first_stack: blockvector[stack_size,bsd$c_size,byte] field(bsd_fields),
;  272          0778  1         current_stack: blockvector[stack_size,bsd$c_size,byte] field(bsd_fields),
;  273          0779  1         next_stack: blockvector[stack_size,bsd$c_size,byte] field(bsd_fields),
;  274          0780  1         key_level: long;
```

```
;  276          0781  1 %sbttl 'ANL$INTERACTIVE_MODE - Control Interactive Mode Analysis'
;  277          0782  1 !++
;  278          0783  1 ! Functional Description:
;  279          0784  1 !       This routine is the controlling routine for /INTERACTIVE mode
;  280          0785  1 !       analysis.  We allow the user to analyze the file specified
;  281          0786  1 !       on the command line.
;  282          0787  1 !
;  283          0788  1 ! Formal Parameters:
;  284          0789  1 !       none
;  285          0790  1 !
;  286          0791  1 ! Implicit Inputs:
;  287          0792  1 !       global data
;  288          0793  1 !
;  289          0794  1 ! Implicit Outputs:
;  290          0795  1 !       global data
;  291          0796  1 !
;  292          0797  1 ! Returned Value:
;  293          0798  1 !       none
;  294          0799  1 !
;  295          0800  1 ! Side Effects:
;  296          0801  1 !
;  297          0802  1 !--
;  298          0803  1
;  299          0804  1
;  300          0805  2 global routine anl$interactive_mode: novalue = begin
;  301          0806  2
;  302          0807  2 local
;  303          0808  2         status;
;  304          0809  2
;  305          0810  2
;  306          0811  2 ! Begin by opening the file to be analyzed.  If the user blew it, just quit.
;  307          0812  2
;  308          0813  3 begin
;  309          0814  3 local
;  310          0815  3         local_described_buffer(resultant_file_spec,nam$c_maxrss);
;  311          0816  3
;  312          0817  3 if not anl$open_next_rms_file(resultant_file_spec) then
;  313          0818  3         return;
;  314          0819  3
;  315          0820  3 ! Now we can prepare the report file to receive a transcript of the session.
;  316          0821  3
;  317          0822  3 anl$prepare_report_file(anlrms$_interhdg,resultant_file_spec);
;  318          0823  2 end;
;  319          0824  2
;  320          0825  2 ! Interactively analyze the file.
;  321          0826  2
;  322          0827  2 anl$interactive_driver();
;  323          0828  2
;  324          0829  2 return;
;  325          0830  2
;  326          0831  1 end;


                                                .TITLE  RMSINTER RMSINTER - Interactive Analysis Mode
                                                .IDENT  \V04-000\

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

```
                                53 45 54 55 42 49 52 54 54 41 0A 00000 P.AAA:  .ASCII  <10>\ATTRIBUTES\
69 72 74 74 61 20 65 6C 69 66 20 53 4D 52 17 0000B P.AAB:  .ASCII  <23>\RMS file attribute area\
                   61 65 72 61 20 65 74 75 62 0001A
                                53 4B 43 4F 4C 42 06 00023 P.AAC:  .ASCII  <6>\BLOCKS\
6C 69 66 20 6E 6F 20 73 64 6E 65 70 65 44 1C 0002A P.AAD:  .ASCII  <28>\Depends on file organization\
   6E 6F 69 74 61 7A 69 6E 61 67 72 6F 20 65 00039
                                53 54 45 4B 43 55 42 07 00047 P.AAE:  .ASCII  <7>\BUCKETS\
         73 74 65 6B 63 75 62 20 61 74 61 44 0C 0004F P.AAF:  .ASCII  <12>\Data buckets\
                                53 4C 4C 45 43 05 0005C P.AAG:  .ASCII  <5>\CELLS\
         73 6C 6C 65 63 20 64 72 6F 63 65 52 0C 00062 P.AAH:  .ASCII  <12>\Record cells\
                                53 41 45 52 41 05 0006F P.AAI:  .ASCII  <5>\AREAS\
6F 74 70 69 72 63 73 65 64 20 61 65 72 41 10 00075 P.AAJ:  .ASCII  <16>\Area descriptors\
                                                  73 72 00084
                                53 59 45 4B 04 00086 P.AAK:  .ASCII  <4>\KEYS\
72 6F 74 70 69 72 63 73 65 64 20 79 65 4B 0F 0008B P.AAL:  .ASCII  <15>\Key descriptors\
                                                  73 0009A
                                58 45 44 4E 49 05 0009B P.AAM:  .ASCII  <5>\INDEX\
63 75 62 20 78 65 64 6E 69 20 74 6F 6F 52 11 000A1 P.AAN:  .ASCII  <17>\Root index bucket\
                                              74 65 6B 000B0
                                41 54 41 44 04 000B3 P.AAO:  .ASCII  <4>\DATA\
         73 74 65 6B 63 75 62 20 61 74 61 44 0C 000B8 P.AAP:  .ASCII  <12>\Data buckets\
                                53 44 52 4F 43 45 52 07 000C5 P.AAQ:  .ASCII  <7>\RECORDS\
   73 64 72 6F 63 65 72 20 78 65 64 6E 49 0D 000CD P.AAR:  .ASCII  <13>\Index records\
                                52 45 50 45 45 44 06 000DB P.AAS:  .ASCII  <6>\DEEPER\
20 61 74 61 64 20 72 6F 20 78 65 64 6E 49 15 000E2 P.AAT:  .ASCII  <21>\Index or data buckets\
                                73 74 65 6B 63 75 62 000F1
                                53 44 52 4F 43 45 52 07 000F8 P.AAU:  .ASCII  <7>\RECORDS\
72 20 61 74 61 64 20 79 72 61 6D 69 72 50 14 00100 P.AAV:  .ASCII  <20>\Primary data records\
                                73 64 72 6F 63 65 0010F
                                53 45 54 59 42 05 00115 P.AAW:  .ASCII  <5>\BYTES\
65 72 20 61 74 61 64 20 6C 61 75 74 63 41 18 0011B P.AAX:  .ASCII  <24>\Actual data record bytes\
                   73 65 74 79 62 20 64 72 6F 63 0012A
                                56 52 52 03 00134 P.AAY:  .ASCII  <3>\RRV\
65 6B 63 75 62 20 61 74 61 64 20 56 52 52 0F 00138 P.AAZ:  .ASCII  <15>\RRV data bucket\
                                                  74 00147
                                53 52 44 49 53 05 00148 P.ABA:  .ASCII  <5>\SIDRS\
         64 72 6F 63 65 72 20 52 44 49 53 0B 0014E P.ABB:  .ASCII  <11>\SIDR record\
                                52 45 54 4E 49 4F 50 07 0015A P.ABC:  .ASCII  <7>\POINTER\
72 65 74 6E 69 6F 70 20 64 72 6F 63 65 52 0E 00162 P.ABD:  .ASCII  <14>\Record pointer\
                                53 44 52 4F 43 45 52 07 00171 P.ABE:  .ASCII  <7>\RECORDS\
   73 64 72 6F 63 65 72 20 78 65 64 6E 49 0D 00179 P.ABF:  .ASCII  <13>\Index records\
                                52 45 50 45 45 44 06 00187 P.ABG:  .ASCII  <6>\DEEPER\
20 61 74 61 64 20 72 6F 20 78 65 64 6E 49 15 0018E P.ABH:  .ASCII  <21>\Index or data buckets\
                                73 74 65 6B 63 75 62 0019D
                                53 44 52 4F 43 45 52 07 001A4 P.ABI:  .ASCII  <7>\RECORDS\
72 20 61 74 61 64 20 79 72 61 6D 69 72 50 14 001AC P.ABJ:  .ASCII  <20>\Primary data records\
                                73 64 72 6F 63 65 001BB
                                53 45 54 59 42 05 001C1 P.ABK:  .ASCII  <5>\BYTES\
65 72 20 61 74 61 64 20 6C 61 75 74 63 41 18 001C7 P.ABL:  .ASCII  <24>\Actual data record bytes\
                   73 65 74 79 62 20 64 72 6F 63 001D6
                                56 52 52 03 001E0 P.ABM:  .ASCII  <3>\RRV\
65 6B 63 75 62 20 61 74 61 64 20 56 52 52 0F 001E4 P.ABN:  .ASCII  <15>\RRV data bucket\
                                                  74 001F3
                                53 52 44 49 53 05 001F4 P.ABO:  .ASCII  <5>\SIDRS\
         64 72 6F 63 65 72 20 52 44 49 53 0B 001FA P.ABP:  .ASCII  <11>\SIDR record\
                                52 45 54 4E 49 4F 50 07 00206 P.ABQ:  .ASCII  <7>\POINTER\
```

```
72 65 74 6E 69 6F 70 20 64 72 6F 63 65 52 0E  0020E P.ABR:  .ASCII  <14>\Record pointer\                    ;
                  44 45 4D 49 41 4C 43 45 52 09  0021D P.ABS:  .ASCII  <9>\RECLAIMED\                      ;
6B 63 75 62 20 64 65 6D 69 61 6C 63 65 52 11  00227 P.ABT:  .ASCII  <17>\Reclaimed buckets\                ;
                                    73 74 65  00236                                                      ;

                                                          .PSECT  $OWN$,NOEXE,2

00 00 03 00 00 02 02 00 00 01 01 00 00 00 00  00000 STRUCTURE_TABLE:
                                                          .BYTE   0, 0, 0, 0, 1, 1, 0, 0, 2, 2, 0, 0, 3, 0, - ;
05 07 00 00 00 06 00 00 04 05 00 00 03 04 00  0000F                0, 0, 4, 3, 0, 0, 5, 4, 0, 0, 6, 0, 0, 0, - ;
0B 00 00 09 0A 00 08 07 09 00 00 17 08 00 06  0001E                7, 5, 6, 0, 8, 23, 0, 0, 9, 7, 8, 0, 10, -  ;
00 00 0A 0E 00 00 0E 0D 00 00 0B 0C 00 00 09  0002D                9, 0, 0, 11, 9, 0, 0, 12, 11, 0, 0, 13, -   ;
00 0F 12 00 00 00 11 00 0D 0C 10 00 00 0A 0F  0003C                14, 0, 0, 14, 10, 0, 0, 15, 10, 0, 0, 16, - ;
12 16 00 00 10 15 00 00 10 14 00 00 00 13 00  0004B                12, 13, 0, 17, 0, 0, 0, 18, 15, 0, 0, 19, - ;
1A 00 00 11 19 00 00 11 18 00 00 15 17 00 00  0005A                0, 0, 0, 20, 16, 0, 0, 21, 16, 0, 0, 22, -  ;
00 00 00 1D 00 00 16 1C 00 00 00 1B 00 14 13  00069                18, 0, 0, 23, 21, 0, 0, 24, 17, 0, 0, 25, - ;
                                    00 00 00 1E  00078                17, 0, 0, 26, 19, 20, 0, 27, 0, 0, 0, 28, - ;
                                                                      22, 0, 0, 29, 0, 0, 0, 30, 0, 0, 0          ;
                                                0007C     .BLKB   16
                                00000000  00000000  0008C PATH_TABLE:
                                                          .LONG   0, 0                                       ;
                                    00 00  00094          .BYTE   0, 0
                                00000000' 00000000' 00096 .ADDRESS P.AAA, P.AAB
                                    02 01  0009E          .BYTE   1, 2
                                00000000' 00000000' 000A0 .ADDRESS P.AAC, P.AAD
                                    00 02  000A8          .BYTE   2, 0
                                00000000' 00000000' 000AA .ADDRESS P.AAE, P.AAF
                                    05 03  000B2          .BYTE   3, 5
                                00000000' 00000000' 000B4 .ADDRESS P.AAG, P.AAH
                                    06 04  000BC          .BYTE   4, 6
                                00000000' 00000000' 000BE .ADDRESS P.AAI, P.AAJ
                                    08 05  000C6          .BYTE   5, 8
                                00000000' 00000000' 000C8 .ADDRESS P.AAK, P.AAL
                                    09 06  000D0          .BYTE   6, 9
                                00000000' 00000000' 000D2 .ADDRESS P.AAM, P.AAN
                                    00 07  000DA          .BYTE   7, 0
                                00000000' 00000000' 000DC .ADDRESS P.AAO, P.AAP
                                    00 08  000E4          .BYTE   8, 0
                                00000000' 00000000' 000E6 .ADDRESS P.AAQ, P.AAR
                                    00 09  000EE          .BYTE   9, 0
                                00000000' 00000000' 000F0 .ADDRESS P.AAS, P.AAT
                                    00 0A  000F8          .BYTE   10, 0
                                00000000' 00000000' 000FA .ADDRESS P.AAU, P.AAV
                                    10 0B  00102          .BYTE   11, 16
                                00000000' 00000000' 00104 .ADDRESS P.AAW, P.AAX
                                    11 0C  0010C          .BYTE   12, 17
                                00000000' 00000000' 0010E .ADDRESS P.AAY, P.AAZ
                                    0C 0D  00116          .BYTE   13, 12
                                00000000' 00000000' 00118 .ADDRESS P.ABA, P.ABB
                                    12 0E  00120          .BYTE   14, 18
                                00000000' 00000000' 00122 .ADDRESS P.ABC, P.ABD
                                    13 0F  0012A          .BYTE   15, 19
                                00000000' 00000000' 0012C .ADDRESS P.ABE, P.ABF
                                    00 10  00134          .BYTE   16, 0
                                00000000' 00000000' 00136 .ADDRESS P.ABG, P.ABH
                                    00 11  0013E          .BYTE   17, 0
                                00000000' 00000000' 00140 .ADDRESS P.ABI, P.ABJ
```

```
                      1A  0B  00148             .BYTE    11, 26                        ;
        00000000' 00000000'   0014A             .ADDRESS P.ABK, P.ABL                  ;
                      1B  12  00152             .BYTE    18, 27                        ;
        00000000' 00000000'   00154             .ADDRESS P.ABM, P.ABN                  ;
                      16  13  0015C             .BYTE    19, 22                        ;
        00000000' 00000000'   0015E             .ADDRESS P.ABO, P.ABP                  ;
                      1C  0E  00166             .BYTE    14, 28                        ;
        00000000' 00000000'   00168             .ADDRESS P.ABQ, P.ABR                  ;
                      1D  15  00170             .BYTE    21, 29                        ;
        00000000' 00000000'   00172             .ADDRESS P.ABS, P.ABT                  ;
                      1E  16  0017A             .BYTE    22, 30                        ;
                              0017C             .BLKB    10
                              00186             .BLKB    2
                   00000000   00188 TOP:        .LONG    0                             ;
                              0018C FIRST_STACK:
                                                .BLKB    768
                              0048C CURRENT_STACK:
                                                .BLKB    768
                              0078C NEXT_STACK:
                                                .BLKB    768
                              00A8C KEY_LEVEL:
                                                .BLKB    4

                                                .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$_IDXAREARECL
.EXTRN  ANLRMS$_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$AREA_DESCRIPTOR
.EXTRN  ANL$BUCKET, ANL$2BUCKET_HEADER
.EXTRN  ANL$3BUCKET_HEADER
.EXTRN  ANL$3FORMAT_DATA_BYTES
.EXTRN  ANL$FORMAT_FILE_ATTRIBUTES
.EXTRN  ANL$FORMAT_FILE_HEADER
.EXTRN  ANL$FORMAT_HEX, ANL$FORMAT_LINE
.EXTRN  ANL$FORMAT_SKIP
.EXTRN  ANL$IDX_PROLOG, ANL$UNWIND_HANDLER
.EXTRN  ANL$2INDEX_RECORD
.EXTRN  ANL$3INDEX_RECORD
.EXTRN  ANL$INTERNALIZE_NUMBER
.EXTRN  ANL$KEY_DESCRIPTOR
.EXTRN  ANL$OPEN_NEXT_RMS_FILE
.EXTRN  ANL$PREPARE_REPORT_FILE
.EXTRN  ANL$2PRIMARY_DATA_RECORD
.EXTRN  ANL$3PRIMARY_DATA_RECORD
.EXTRN  ANL$3RECLAIMED_BUCKET_HEADER
.EXTRN  ANL$REL_CELL, ANL$REL_PROLOG
.EXTRN  ANL$SEQ_DATA_RECORD
.EXTRN  ANL$2SIDR_POINTER
.EXTRN  ANL$3SIDR_POINTER
.EXTRN  ANL$2SIDR_RECORD
.EXTRN  ANL$3SIDR_RECORD
.EXTRN  CLI$GET_VALUE, LBR$OUTPUT_HELP
.EXTRN  LIB$ESTABLISH, LIB$GET_INPUT
.EXTRN  LIB$PUT_OUTPUT, LIB$TPARSE
.EXTRN  STR$UPCASE, LIB$_SYNTAXERR
.EXTRN  ANL$GL_FAT, ANL$GW_PROLOG
```

```
                                                                        .PSECT  $CODE$,NOWRT,2

                                                0000 00000              .ENTRY  ANL$INTERACTIVE_MODE, Save nothing                ; 0805
                                    5E      FEFC   CE   9E 00002        MOVAB   -260(SP), SP                                      ;
                                    7E        FF   8F   9A 00007        MOVZBL  #255, RESULTANT_FILE_SPEC                         ; 0815
                                04  AE        08   AE   9E 0000B        MOVAB   RESULTANT_FILE_SPEC+8, -                          ;
                                                                                RESULTANT_FILE_SPEC+4                             ;
                                                   5E   DD 00010        PUSHL   SP                                                ; 0817
                              0000G CF                 01   FB 00012     CALLS   #1, ANL$OPEN_NEXT_RMS_FILE                        ;
                                    12                 50   E9 00017     BLBC    R0, 1$                                            ;
                                                       5E   DD 0001A     PUSHL   SP                                                ; 0822
                                        00000000G      8F   DD 0001C     PUSHL   #ANLRMS$_INTERHDG                                 ;
                              0000G CF                 02   FB 00022     CALLS   #2, ANL$PREPARE_REPORT_FILE                       ;
                              0000V CF                 00   FB 00027     CALLS   #0, ANL$INTERACTIVE_DRIVER                        ; 0827
                                                            04 0002C 1$:  RET                                                       ; 0831

; Routine Size:  45 bytes,    Routine Base:  $CODE$ + 0000
```

```
;   328          0832  1 %sbttl 'ANL$INTERACTIVE_DRIVER - Drive Interactive Analysis of a File'
;   329          0833  1 !++
;   330          0834  1 ! Functional Description:
;   331          0835  1 !       This routine drives the iteractive analysis of a single RMS file.
;   332          0836  1 !       It accepts commands from the user and displays file information
;   333          0837  1 !       accordingly.
;   334          0838  1 !
;   335          0839  1 ! Formal Parameters:
;   336          0840  1 !       none
;   337          0841  1 !
;   338          0842  1 ! Implicit Inputs:
;   339          0843  1 !       global data
;   340          0844  1 !
;   341          0845  1 ! Implicit Outputs:
;   342          0846  1 !       global data
;   343          0847  1 !
;   344          0848  1 ! Returned Value:
;   345          0849  1 !       none
;   346          0850  1 !
;   347          0851  1 ! Side Effects:
;   348          0852  1 !
;   349          0853  1 !--
;   350          0854  1
;   351          0855  1
;   352          0856  2 global routine anl$interactive_driver: novalue = begin
;   353          0857  2
;   354          0858  2
;   355          0859  2 local
;   356          0860  2         status: long,
;   357          0861  2         command_number: long,
;   358          0862  2         display: byte;
;   359          0863  2
;   360          0864  2
;   361          0865  2 ! Initialization is not very difficult.  We have to set up the zeroth
;   362          0866  2 ! entry on the stack as if we just went "down" into the file header of
;   363          0867  2 ! the file.  This means we need a BSD describing the file header, and
;   364          0868  2 ! it must be present on the FIRST and CURRENT stacks.
;   365          0869  2
;   366          0870  2 init_bsd(first_stack[.top,0,0,0,0]);
;   367          0871  2 first_stack[.top,bsd$w_type] = 1;
;   368          0872  2 init_bsd(current_stack[.top,0,0,0,0]);
;   369          0873  2 current_stack[.top,bsd$w_type] = 1;
;   370          0874  2 init_bsd(next_stack[.top,0,0,0,0]);
```

```
;  372     0875  2 ! OK, now we can actually begin the analysis.  The main loop is traversed
;  373     0876  2 ! once for each user command.  We quit when we get an EXIT command or
;  374     0877  2 ! CTRL/Z.
;  375     0878  2
;  376     0879  2 display = true;
;  377     0880  3 loop (
;  378     0881  3         local
;  379     0882  3                 local_described_buffer(command_arguments,80);
;  380     0883  3
;  381     0884  3
;  382     0885  3         ! Usually we have to display the current structure.  The display
;  383     0886  3         ! routine will format the contents of the structure, and then
;  384     0887  3         ! update the BSD to describe the next structure on the current
;  385     0888  3         ! level.  This is why we pass it the BSD on the NEXT stack.
;  386     0889  3         ! The display routine also needs the BSD for the parent of the
;  387     0890  3         ! current structure.
;  388     0891  3
;  389     0892  4         if .display then (
;  390     0893  4                 anl$format_skip(0);
;  391     0894  4                 copy_bucket(current_stack[.top,0,0,0,0],next_stack[.top,0,0,0,0]);
;  392     0895  4                 anl$interactive_display(next_stack[.top,0,0,0,0],current_stack[.top-1,0,0,0,0]);
;  393     0896  4                 anl$format_skip(0);
;  394     0897  3         ) else
;  395     0898  3                 display = true;
;  396     0899  3
;  397     0900  3         ! Now we can actually get a command from the user.  The command
;  398     0901  3         ! routine returns the command number and a descriptor of any
;  399     0902  3         ! command arguments.
;  400     0903  3
;  401     0904  3         anl$interactive_command(command_number,command_arguments);
;  402     0905  3
;  403     0906  3         ! Now we can case on the command.
;  404     0907  3
;  405     0908  3         case .command_number from 1 to 11 of set
;  406     0909  3
;  407     0910  3         [1]:    ! The AGAIN command is trivial.  In fact, we don't have to
;  408     0911  3                 ! do a thing.
;  409     0912  3
;  410     0913  3                 ;
;  411     0914  3
;  412     0915  3
;  413     0916  3         [2]:    ! This command number is reserved for the BUCKET command.
;  414     0917  3
;  415     0918  3                 ;
;  416     0919  3
;  417     0920  3
;  418     0921  3         [3]:    ! The DOWN command is very complicated.
;  419     0922  3
;  420     0923  5                 (if .top eqlu stack_size then (
;  421     0924  5
;  422     0925  5                         ! No more room on the stack.  Sorry user.
;  423     0926  5
;  424     0927  5                         signal (anlrms$_stackfull);
;  425     0928  5                         display = false;
;  426     0929  5                 ) else (
;  427     0930  5
;  428     0931  5                         ! The following routine will build a new BSD on the
```

```
;  429          0932  5                          ! FIRST stack describing the lower structure.
;  430          0933  5
;  431          0934  5                          status = anl$interactive_down(command_arguments,
;  432          0935  5                                                        current_stack[.top,0,0,0,0],first_stack[.top+1,0,0,0,0],.top+1);
;  433          0936  6                          if .status then (
;  434          0937  6
;  435          0938  6                                  ! We could go down.  Initialize the CURRENT
;  436          0939  6                                  ! and NEXT stacks, and set the CURRENT stack
;  437          0940  6                                  ! to the first structure on the new level.
;  438          0941  6
;  439          0942  6                                  increment (top);
;  440          0943  6                                  init_bsd(current_stack[.top,0,0,0,0]);
;  441          0944  6                                  copy_bucket(first_stack[.top,0,0,0,0],current_stack[.top,0,0,0,0]);
;  442          0945  6                                  init_bsd(next_stack[.top,0,0,0,0]);
;  443          0946  5                          ) else
;  444          0947  5
;  445          0948  5                                  ! Something prevented us from going down.
;  446          0949  5
;  447          0950  5                                  display = false;
;  448          0951  3                  ););
;  449          0952  3
;  450          0953  3
;  451          0954  3          [4]:    ! The DUMP command is easy here, because we just call
;  452          0955  3                  ! a routine to do it, passing the user's argument.
;  453          0956  3
;  454          0957  4                  (anl$interactive_dump(command_arguments);
;  455          0958  3                  display = false;);
;  456          0959  3
;  457          0960  3
;  458          0961  3          [5]:    ! The EXIT command is real easy.  Just return.
;  459          0962  3
;  460          0963  3                  return;
;  461          0964  3
;  462          0965  3
;  463          0966  3          [6]:    ! The FIRST command is easy.  Just copy the FIRST stack
;  464          0967  3                  ! into the CURRENT stack.
;  465          0968  3
;  466          0969  3                  copy_bucket(first_stack[.top,0,0,0,0],current_stack[.top,0,0,0,0]);
;  467          0970  3
;  468          0971  3
;  469          0972  3          [7]:    ! The HELP command is easy here, because we just call a
;  470          0973  3                  ! routine to do it, passing the user's arguments.
;  471          0974  3
;  472          0975  4                  (anl$interactive_help(command_arguments);
;  473          0976  3                  display = false;);
;  474          0977  3
;  475          0978  3
;  476          0979  3          [8]:    ! The NEXT command is easy.  If there is no next structure,
;  477          0980  3                  ! tell the user.  If there is, simply copy the NEXT stack
;  478          0981  3                  ! into the CURRENT stack.
;  479          0982  3
;  480          0983  3                  if .next_stack[.top,bsd$l_vbn] eqlu .current_stack[.top,bsd$l_vbn] and
;  481          0984  4                     .next_stack[.top,bsd$l_offset] eqlu .current_stack[.top,bsd$l_offset] then (
;  482          0985  4                          signal (anlrms$_nonext);
;  483          0986  4                          display = false;
;  484          0987  3                  ) else
;  485          0988  3                          copy_bucket(next_stack[.top,0,0,0,0],current_stack[.top,0,0,0,0]);
```

```
;  486          0989  3
;  487          0990  3
;  488          0991  3          [9]:    ! The REST command is a little harder.  We sit in a loop,
;  489          0992  3                  ! displaying structures and moving on to the next one,
;  490          0993  3                  ! until there is no next one.
;  491          0994  3
;  492          0995  4                  (until .next_stack[.top,bsd$l_vbn] eglu .current_stack[.top,bsd$l_vbn] and
;  493          0996  5                         .next_stack[.top,bsd$l_offset] eglu .current_stack[.top,bsd$l_offset] do (
;  494          0997  5                          copy_bucket(next_stack[.top,0,0,0,0],current_stack[.top,0,0,0,0]);
;  495          0998  5                          anl$format_skip(0);
;  496          0999  5                          anl$interactive_display(next_stack[.top,0,0,0,0],current_stack[.top-1,0,0,0,0]);
;  497          1000  4                  );
;  498          1001  4                  signal (anlrms$_restdone);
;  499          1002  3                  display = false;);
;  500          1003  3
;  501          1004  3
;  502          1005  3          [10]:   ! The TOP command requires a loop to pop each stack entry
;  503          1006  3                  ! down to the original one.
;  504          1007  3
;  505          1008  4                  while .top gtru 0 do (
;  506          1009  4                          anl$bucket(first_stack[.top,0,0,0,0],-1);
;  507          1010  4                          anl$bucket(current_stack[.top,0,0,0,0],-1);
;  508          1011  4                          anl$bucket(next_stack[.top,0,0,0,0],-1);
;  509          1012  4                          decrement (top);
;  510          1013  3                  );
;  511          1014  3
;  512          1015  3
;  513          1016  3          [11]:   ! The UP command is easy.  Just pop the stacks, unless we
;  514          1017  3                  ! already are at the top.
;  515          1018  3
;  516          1019  4                  if .top eglu 0 then (
;  517          1020  4                          signal (anlrms$_attop);
;  518          1021  4                          display = false;
;  519          1022  4                  ) else (
;  520          1023  4                          anl$bucket(first_stack[.top,0,0,0,0],-1);
;  521          1024  4                          anl$bucket(current_stack[.top,0,0,0,0],-1);
;  522          1025  4                          anl$bucket(next_stack[.top,0,0,0,0],-1);
;  523          1026  4                          decrement (top);
;  524          1027  3                  );
;  525          1028  3
;  526          1029  3              tes;
;  527          1030  3
;  528          1031  2 );
;  529          1032  2
;  530          1033  2 return;
;  531          1034  2
;  532          1035  1 end;


                                                0FFC 00000          .ENTRY  ANL$INTERACTIVE_DRIVER, Save R2,R3,R4,R5,-      ; 0856
                                                                            R6,R7,R8,R9,R10,R11                                 ;
                                5B      0000G   CF   9E 00002       MOVAB   ANL$BUCKET, R11                                     ;
                                5A      0000'   CF   9E 00007       MOVAB   TOP, R10                                            ;
                                5E        A4    AE   9E 0000C       MOVAB   -92(SP), SP                                         ;
```

```
                56              6A              18  C5 00010          MULL3   #24, TOP, R6                                    ; 0870
        18      00              6E              00  2C 00014          MOVC5   #0, (SP), #0, #24, FIRST_STACK[R6]
                                                04 AA46   00019
                                                04 AA46  9F 0001C     PUSHAB  FIRST_STACK[R6]                                 ; 0871
                                9E                  01  B0 00020     MOVW    #1, @(SP)+
        18      00              6E                  00  2C 00023     MOVC5   #0, (SP), #0, #24, CURRENT_STACK[R6]             ; 0872
                                              0304 CA46   00028
                                              0304 CA46  9F 0002C     PUSHAB  CURRENT_STACK[R6]                               ; 0873
                                9E                  01  B0 00031     MOVW    #1, @(SP)+
        18      00              6E                  00  2C 00034     MOVC5   #0, (SP), #0, #24, NEXT_STACK[R6]               ; 0874
                                              0604 CA46   00039
                                58                  01  90 0003D     MOVB    #1, DISPLAY                                     ; 0879
                        04      AE          50      8F  9A 00040 1$:  MOVZBL  #80, COMMAND_ARGUMENTS                          ; 0882
                        08      AE          0C      AE  9E 00045     MOVAB   COMMAND_ARGUMENTS+8, COMMAND_ARGUMENTS+4
                                4B                  58  E9 0004A     BLBC    DISPLAY, 2$                                     ; 0892
                                                    7E  D4 0004D     CLRL    -(SP)                                           ; 0893
                    0000G       CF                  01  FB 0004F     CALLS   #1, ANL$FORMAT_SKIP
                50              6A                  18  C5 00054     MULL3   #24, TOP, R0                                    ; 0894
                                51              0304 CA40  9E 00058     MOVAB   CURRENT_STACK[R0], R1
                                50              0604 CA40  9E 0005E     MOVAB   NEXT_STACK[R0], R0
                                60                  61  7D 00064     MOVQ    (R1), (R0)
                        08      A0          08      A1  D0 00067     MOVL    8(R1), 8(R0)
                        14      A0          14      A1  D0 0006C     MOVL    20(R1), 20(R0)
                                                    7E  D4 00071     CLRL    -(SP)
                                                    50  DD 00073     PUSHL   R0
                                6B                  02  FB 00075     CALLS   #2, ANL$BUCKET
                50              6A                  18  C5 00078     MULL3   #24, TOP, R0                                    ; 0895
                                              02EC CA40  9F 0007C     PUSHAB  CURRENT_STACK-24[R0]
                50              6A                  18  C5 00081     MULL3   #24, TOP, R0
                                              0604 CA40  9F 00085     PUSHAB  NEXT_STACK[R0]
                    0000V       CF                  02  FB 0008A     CALLS   #2, ANL$INTERACTIVE_DISPLAY
                                                    7E  D4 0008F     CLRL    -(SP)                                           ; 0896
                    0000G       CF                  01  FB 00091     CALLS   #1, ANL$FORMAT_SKIP
                                                    03  11 00096     BRB     3$                                              ; 0892
                                58                  01  90 00098 2$:  MOVB    #1, DISPLAY                                     ; 0898
                        04      AE          9F      0009B 3$:         PUSHAB  COMMAND_ARGUMENTS                               ; 0904
                        04      AE          9F      0009E             PUSHAB  COMMAND_NUMBER
                    0000V       CF                  02  FB 000A1     CALLS   #2, ANL$INTERACTIVE_COMMAND
                0A              01                  CF  6E 000A6     CASEL   COMMAND_NUMBER, #1, #10                         ; 0908
0088        0018            FF96            FF96        000AA 4$:     .WORD   1$-4$,-
00AE        00A3            0092            01E1        000B2         1$-4$,-
            019A            0166            00FC        000BA         5$-4$,-
                                                                      8$-4$,-
                                                                      25$-4$,-
                                                                      9$-4$,-
                                                                      10$-4$,-
                                                                      12$-4$,-
                                                                      16$-4$,-
                                                                      19$-4$,-
                                                                      20$-4$
                                                    6E  11 000C0     BRB     7$
                                52                  6A  D0 000C2 5$:  MOVL    TOP, R2                                       ; 0923
                                20                  52  D1 000C5     CMPL    R2, #32
                                                    09  12 000C8     BNEQ    6$
                        00000000G                   8F  DD 000CA     PUSHL   #ANLRMS$_STACKFULL                            ; 0927
                                              017C      31 000D0     BRW     21$
                                        01          A2  9F 000D3 6$:  PUSHAB  1(R2)                                         ; 0935
```

```
                                50          01    A2  9E 000D6          MOVAB   1(R2), R0                                    ;
                                50              18  C4 000DA          MULL2   #24, R0                                     ;
                                            04  AA40 9F 000DD          PUSHAB  FIRST_STACK[R0]                             ;
                          50                52      18 C5 000E1       MULL3   #24, R2, R0                                 ;
                                          0304  CA40 9F 000E5          PUSHAB  CURRENT_STACK[R0]                           ;
                                            10    AE 9F 000EA          PUSHAB  COMMAND_ARGUMENTS                           ; 0934
                                0000V       CF      04 FB 000ED       CALLS   #4, ANL$INTERACTIVE_DOWN                    ; 0935
                                59              50  D0 000F2          MOVL    R0, STATUS                                  ;
                                5D              59  E9 000F5          BLBC    STATUS, 11$                                 ; 0936
                                6A              D6  000F8            INCL    TOP                                         ; 0942
                          56                6A      18 C5 000FA       MULL3   #24, TOP, R6                                ; 0943
                                57          0304  CA46 9E 000FE       MOVAB   CURRENT_STACK[R6], R7                       ;
                18        00                6E      00 2C 00104       MOVC5   #0, (SP), #0, #24, (R7)                     ;
                                                  67    00109                                                             ;
                                50          04  AA46 9E 0010A          MOVAB   FIRST_STACK[R6], R0                         ; 0944
                                67              60  7D 0010F          MOVQ    (R0), (R7)                                  ;
                                      08    A7    08  A0 D0 00112      MOVL    8(R0), 8(R7)                                ;
                                      14    A7    14  A0 D0 00117      MOVL    20(R0), 20(R7)                              ;
                                              7E  D4 0011C            CLRL    -(SP)                                       ;
                                              57  DD 0011E            PUSHL   R7                                          ;
                                6B              02  FB 00120         CALLS   #2, ANL$BUCKET                              ;
                          50                6A      18 C5 00123       MULL3   #24, TOP, R0                                ; 0945
                18        00                6E      00 2C 00127       MOVC5   #0, (SP), #0, #24, NEXT_STACK[R0]           ;
                                          0604  CA40      0012C                                                           ;
                                              71  11 00130 7$:         BRB     15$                                         ; 0936
                                            04    AE 9F 00132 8$:       PUSHAB  COMMAND_ARGUMENTS                           ; 0957
                                0000V       CF      01 FB 00135       CALLS   #1, ANL$INTERACTIVE_DUMP                    ;
                                              19  11 0013A            BRB     11$                                         ; 0958
                          50                6A      18 C5 0013C 9$:   MULL3   #24, TOP, R0                                ; 0969
                                51          04  AA40 9E 00140          MOVAB   FIRST_STACK[R0], R1                         ;
                                50        0304  CA40 9E 00145          MOVAB   CURRENT_STACK[R0], R0                       ;
                                              42  11 0014B            BRB     14$                                         ;
                                            04    AE 9F 0014D 10$:      PUSHAB  COMMAND_ARGUMENTS                           ; 0975
                              0000V         CF      01 FB 00150       CALLS   #1, ANL$INTERACTIVE_HELP                    ;
                                            00FE    31 00155 11$:     BRW     22$                                         ; 0976
                52                6A              18  C5 00158 12$:   MULL3   #24, TOP, R2                                ; 0983
                                          0308  CA42 9F 0015C          PUSHAB  CURRENT_STACK+4[R2]                         ;
                                          0608  CA42 9F 00161          PUSHAB  NEXT_STACK+4[R2]                            ;
                                9E              9E  D1 00166          CMPL    @(SP)+, @(SP)+                              ;
                                              18  12 00169            BNEQ    13$                                         ;
                                          030C  CA42 9F 0016B          PUSHAB  CURRENT_STACK+8[R2]                         ; 0984
                                          060C  CA42 9F 00170          PUSHAB  NEXT_STACK+8[R2]                            ;
                                9E              9E  D1 00175          CMPL    @(SP)+, @(SP)+                              ;
                                              09  12 00178            BNEQ    13$                                         ;
                                    00000000G   8F  DD 0017A          PUSHL   #ANLRMS$_NONEXT                             ; 0985
                                            00CC    31 00180          BRW     21$                                         ;
                                51        0604  CA42 9E 00183 13$:     MOVAB   NEXT_STACK[R2], R1                          ; 0988
                                50        0304  CA42 9E 00189          MOVAB   CURRENT_STACK[R2], R0                       ;
                                60              61  7D 0018F 14$:     MOVQ    (R1), (R0)                                  ;
                                      08    A0    08  A1 D0 00192      MOVL    8(R1), 8(R0)                                ;
                                      14    A0    14  A1 D0 00197      MOVL    20(R1), 20(R0)                              ;
                                              7E  D4 0019C            CLRL    -(SP)                                       ;
                                              50  DD 0019E            PUSHL   R0                                          ;
                                6B              02  FB 001A0         CALLS   #2, ANL$BUCKET                              ;
                                            FE9A    31 001A3 15$:     BRW     1$                                          ; 0983
                50                6A              18  C5 001A6 16$:   MULL3   #24, TOP, R0                                ; 0995
                                          0308  CA40 9F 001AA          PUSHAB  CURRENT_STACK+4[R0]                         ;
```

```
                                0608 CA40  9F 001AF              PUSHAB  NEXT_STACK+4[R0]                        ;
                          9E          9E  D1 001B4              CMPL    @(SP)+, @(SP)+                          ;
                                      0F  12 001B7              BNEQ    17$                                     ;
                                030C CA40  9F 001B9              PUSHAB  CURRENT_STACK+8[R0]                     ; 0996
                                060C CA40  9F 001BE              PUSHAB  NEXT_STACK+8[R0]                        ;
                          9E          9E  D1 001C3              CMPL    @(SP)+, @(SP)+                          ;
                                      40  13 001C6              BEQL    18$                                     ;
                          51    0604 CA40  9E 001C8 17$:          MOVAB   NEXT_STACK[R0], R1                      ; 0997
                          50    0304 CA40  9E 001CE               MOVAB   CURRENT_STACK[R0], R0                   ;
                          60          61  7D 001D4               MOVQ    (R1), (R0)                              ;
                    08    A0    08    A1  D0 001D7               MOVL    8(R1), 8(R0)                            ;
                    14    A0    14    A1  D0 001DC               MOVL    20(R1), 20(R0)                          ;
                                      7E  D4 001E1               CLRL    -(SP)                                   ;
                                      50  DD 001E3               PUSHL   R0                                      ;
                          6B          02  FB 001E5               CALLS   #2, ANL$BUCKET                          ;
                                      7E  D4 001E8               CLRL    -(SP)                                   ; 0998
                    0000G CF          01  FB 001EA               CALLS   #1, ANL$FORMAT_SKIP                     ;
               50         6A          18  C5 001EF               MULL3   #24, TOP, R0                            ; 0999
                                02EC CA40  9F 001F3               PUSHAB  CURRENT_STACK-24[R0]                    ;
               50         6A          18  C5 001F8               MULL3   #24, TOP, R0                            ;
                                0604 CA40  9F 001FC               PUSHAB  NEXT_STACK[R0]                          ;
                    0000V CF          02  FB 00201               CALLS   #2, ANL$INTERACTIVE_DISPLAY             ;
                                      9E  11 00206               BRB     16$                                     ; 0995
                         00000000G    8F  DD 00208 18$:          PUSHL   #ANLRMS$_RESTDONE                       ; 1001
                                      3F  11 0020E               BRB     21$                                     ;
                          50          6A  D0 00210 19$:          MOVL    TOP, R0                                 ; 1008
                                      8E  13 00213               BEQL    15$                                     ;
                          7E          01  CE 00215               MNEGL   #1, -(SP)                               ; 1009
                          50          18  C4 00218               MULL2   #24, R0                                 ;
                                04 AA40  9F 0021B               PUSHAB  FIRST_STACK[R0]                         ;
                          6B          02  FB 0021F               CALLS   #2, ANL$BUCKET                          ;
                          7E          01  CE 00222               MNEGL   #1, -(SP)                               ; 1010
               50         6A          18  C5 00225               MULL3   #24, TOP, R0                            ;
                                0304 CA40  9F 00229               PUSHAB  CURRENT_STACK[R0]                       ;
                          6B          02  FB 0022E               CALLS   #2, ANL$BUCKET                          ;
                          7E          01  CE 00231               MNEGL   #1, -(SP)                               ; 1011
               50         6A          18  C5 00234               MULL3   #24, TOP, R0                            ;
                                0604 CA40  9F 00238               PUSHAB  NEXT_STACK[R0]                          ;
                          6B          02  FB 0023D               CALLS   #2, ANL$BUCKET                          ;
                                      6A  D7 00240               DECL    TOP                                     ; 1012
                                      CC  11 00242               BRB     19$                                     ; 1008
                          52          6A  D0 00244 20$:          MOVL    TOP, R2                                 ; 1019
                                      11  12 00247               BNEQ    23$                                     ;
                         00000000G    8F  DD 00249               PUSHL   #ANLRMS$_ATTOP                          ; 1020
                    00000000G   00    01  FB 0024F 21$:          CALLS   #1, LIB$SIGNAL                          ;
                                      58  94 00256 22$:          CLRB    DISPLAY                                 ; 1021
                                      2E  11 00258               BRB     24$                                     ; 1019
                          7E          01  CE 0025A 23$:          MNEGL   #1, -(SP)                               ; 1023
               50         52          18  C5 0025D               MULL3   #24, R2, R0                             ;
                                04 AA40  9F 00261               PUSHAB  FIRST_STACK[R0]                         ;
                          6B          02  FB 00265               CALLS   #2, ANL$BUCKET                          ;
                          7E          01  CE 00268               MNEGL   #1, -(SP)                               ; 1024
               50         6A          18  C5 0026B               MULL3   #24, TOP, R0                            ;
                                0304 CA40  9F 0026F               PUSHAB  CURRENT_STACK[R0]                       ;
                          6B          02  FB 00274               CALLS   #2, ANL$BUCKET                          ;
                          7E          01  CE 00277               MNEGL   #1, -(SP)                               ; 1025
               50         6A          18  C5 0027A               MULL3   #24, TOP, R0                            ;
```

```
                          0604 CA40  9F 0027E          PUSHAB  NEXT_STACK[R0]                  ;
                     6B         02  FB 00283          CALLS   #2, ANL$BUCKET                  ;
                                6A  D7 00286          DECL    TOP                             ; 1026
                              FDB5  31 00288 24$:     BRW     1$                              ; 1019
                                    04 0028B 25$:     RET                                     ; 1035

; Routine Size:  652 bytes,    Routine Base:  $CODE$ + 002D
```

```
;   534         1036  1 %sbttl 'ANL$INTERACTIVE_COMMAND - Get a Command From the User'
;   535         1037  1 !++
;   536         1038  1 ! Functional Description:
;   537         1039  1 !       This routine is responsible for obtaining a command from the user,
;   538         1040  1 !       parsing it, checking it, and returning information about it.
;   539         1041  1 !
;   540         1042  1 ! Formal Parameters:
;   541         1043  1 !       number          Address of a longword in which to return the command
;   542         1044  1 !                       identification number.
;   543         1045  1 !       arguments       Address of a descriptor of a buffer in which to
;   544         1046  1 !                       return any command arguments.
;   545         1047  1 !
;   546         1048  1 ! Implicit Inputs:
;   547         1049  1 !       global data
;   548         1050  1 !
;   549         1051  1 ! Implicit Outputs:
;   550         1052  1 !       global data
;   551         1053  1 !
;   552         1054  1 ! Returned Value:
;   553         1055  1 !       none
;   554         1056  1 !
;   555         1057  1 ! Side Effects:
;   556         1058  1 !
;   557         1059  1 !--
;   558         1060  1
;   559         1061  1
;   560         1062  2 global routine anl$interactive_command(number,arguments): novalue = begin
;   561         1063  2
;   562         1064  2 bind
;   563         1065  2         arguments_dsc = .arguments: descriptor;
;   564         1066  2
;   565         1067  2 own
;   566         1068  2         tparse_block: block[tpa$k_length0,byte] initial(
;   567         1069  2                                 tpa$k_count0,
;   568         1070  2                                 tpa$m_blanks + tpa$m_abbrev),
;   569         1071  2         command_number: long;
;   570         1072  2
;   571         1073  2 local
;   572         1074  2         status: long;
```

```
;  574          1075  2 ! The following data structure is the parsing table used to analyze a
;  575          1076  2 ! command from the user.  The command numbers cannot be changed.
;  576          1077  2
;  577          1078  2 $init_state(command_state,command_key);
;  578          1079  2
;  579        P 1080  2 $state  (,
;  580        P 1081  2                  (tpa$_blank),
;  581        P 1082  2                  (tpa$_lambda)
;  582          1083  2          );
;  583          1084  2
;  584        P 1085  2 $state  (,
;  585        P 1086  2                  (tpa$_eos,      noargs,,        8,command_number),
;  586        P 1087  2                  ('AGAIN',       noargs,,        1,command_number),
;  587        P 1088  2 ! Command number 2 is reserved for BUCKET.
;  588        P 1089  2                  ('DOWN',          args,,        3,command_number),
;  589        P 1090  2                  ('DUMP',          args,,        4,command_number),
;  590        P 1091  2                  ('EXIT',        noargs,,        5,command_number),
;  591        P 1092  2                  ('FIRST',       noargs,,        6,command_number),
;  592        P 1093  2                  ('HELP',          args,,        7,command_number),
;  593        P 1094  2                  ('NEXT',        noargs,,        8,command_number),
;  594        P 1095  2                  ('REST',        noargs,,        9,command_number),
;  595        P 1096  2                  ('TOP',         noargs,,       10,command_number),
;  596        P 1097  2                  ('UP',          noargs,,       11,command_number)
;  597          1098  2          );
;  598          1099  2
;  599        P 1100  2 $state  (noargs,
;  600        P 1101  2                  (tpa$_blank),
;  601        P 1102  2                  (tpa$_lambda)
;  602          1103  2          );
;  603        P 1104  2 $state  (,
;  604        P 1105  2                  (tpa$_eos,tpa$_exit)
;  605          1106  2          );
;  606          1107  2
;  607        P 1108  2 $state  (args,
;  608        P 1109  2                  (tpa$_blank,tpa$_exit),
;  609        P 1110  2                  (tpa$_lambda,tpa$_exit)
;  610          1111  2          );
```

```
;  612          1112  2 ! Sit in a loop until we get a valid command.
;  613          1113  2
;  614          1114  3 begin
;  615          1115  3 local
;  616          1116  3         local_described_buffer(command_buffer,80);
;  617          1117  3
;  618          1118  4 loop (
;  619          1119  4
;  620          1120  4         ! Get the command string.
;  621          1121  4
;  622          1122  4         command_buffer[len] = 80;
;  623          1123  4         status = lib$get_input(command_buffer,describe('ANALYZE> '),command_buffer);
;  624          1124  4
;  625          1125  4         ! If we got an end-of-file, then just tell the caller we got an EXIT
;  626          1126  4         ! command.
;  627          1127  4
;  628          1128  5         if .status eqlu rms$_eof then (
;  629          1129  5                 .number = 5;
;  630          1130  5                 return;
;  631          1131  4         );
;  632          1132  4         check (.status, .status);
;  633          1133  4
;  634          1134  4         ! Set up for parsing the command.  Don't forget to uppercase it.
;  635          1135  4
;  636          1136  4         tparse_block[tpa$l_stringcnt] = .command_buffer[len];
;  637          1137  4         tparse_block[tpa$l_stringptr] = .command_buffer[ptr];
;  638          1138  4         str$upcase(tparse_block[tpa$l_stringcnt],tparse_block[tpa$l_stringcnt]);
;  639          1139  4         command_number = 0;
;  640          1140  4         status = lib$tparse(tparse_block,command_state,command_key);
;  641          1141  4
;  642          1142  4         ! If we didn't get a syntax error, then we're all set.
;  643          1143  4         ! Otherwise try again.
;  644          1144  4
;  645          1145  4 exitif (.status eqlu ss$_normal);
;  646          1146  4         signal (anlrms$_badcmd);
;  647          1147  3 );
;  648          1148  3
;  649          1149  3 ! We have a command, so let's echo it into the transcript file, if present.
;  650          1150  3 ! The -1 widow control prevents the line from appearing on screen.
;  651          1151  3
;  652          1152  3 anl$format_line(-1,0,anlrms$_intercommand,command_buffer);
;  653          1153  2 end;
;  654          1154  2
;  655          1155  2 ! OK, return the command number.  Also place any command arguments into
;  656          1156  2 ! the caller's buffer.
;  657          1157  2
;  658          1158  2 .number = .command_number;
;  659          1159  2 arguments_dsc[len] = .tparse_block[tpa$l_stringcnt];
;  660          1160  2 ch$move(.tparse_block[tpa$l_stringcnt],.tparse_block[tpa$l_stringptr], .arguments_dsc[ptr]);
;  661          1161  2
;  662          1162  2 return;
;  663          1163  2
;  664          1164  1 end;


                                                .PSECT  _LIB$KEY1$,NOWRT,  SHR,  PIC,1
```

```
                                00000 ;TPA$KEYSTO
                                      U.9:      .BLKB   0
                     4E 49 41 47 41 00000 ;TPA$KEYST
                                      U.11:     .ASCII  \AGAIN\                 ;
                              FF      00005             .BYTE   -1              ;
                                      00006 ;TPA$KEYSTO
                                      U.16:     .BLKB   0
                        4E 57 4F 44   00006 ;TPA$KEYST
                                      U.18:     .ASCII  \DOWN\                  ;
                              FF      0000A             .BYTE   -1              ;
                                      0000B ;TPA$KEYSTO
                                      U.24:     .BLKB   0
                        50 4D 55 44   0000B ;TPA$KEYST
                                      U.26:     .ASCII  \DUMP\                  ;
                              FF      0000F             .BYTE   -1              ;
                                      00010 ;TPA$KEYSTO
                                      U.31:     .BLKB   0
                        54 49 58 45   00010 ;TPA$KEYST
                                      U.33:     .ASCII  \EXIT\                  ;
                              FF      00014             .BYTE   -1              ;
                                      00015 ;TPA$KEYSTO
                                      U.38:     .BLKB   0
                     54 53 52 49 46   00015 ;TPA$KEYST
                                      U.40:     .ASCII  \FIRST\                 ;
                              FF      0001A             .BYTE   -1              ;
                                      0001B ;TPA$KEYSTO
                                      U.45:     .BLKB   0
                     50 4C 45 48      0001B ;TPA$KEYST
                                      U.47:     .ASCII  \HELP\                  ;
                              FF      0001F             .BYTE   -1              ;
                                      00020 ;TPA$KEYSTO
                                      U.52:     .BLKB   0
                     54 58 45 4E      00020 ;TPA$KEYST
                                      U.54:     .ASCII  \NEXT\                  ;
                              FF      00024             .BYTE   -1              ;
                                      00025 ;TPA$KEYSTO
                                      U.59:     .BLKB   0
                     54 53 45 52      00025 ;TPA$KEYST
                                      U.61:     .ASCII  \REST\                  ;
                              FF      00029             .BYTE   -1              ;
                                      0002A ;TPA$KEYSTO
                                      U.66:     .BLKB   0
                        50 4F 54      0002A ;TPA$KEYST
                                      U.68:     .ASCII  \TOP\                   ;
                              FF      0002D             .BYTE   -1              ;
                                      0002E ;TPA$KEYSTO
                                      U.73:     .BLKB   0
                           50 55      0002E ;TPA$KEYST
                                      U.75:     .ASCII  \UP\                    ;
                              FF      00030             .BYTE   -1              ;
                              FF      00031 ;TPA$KEYFILL
                                      U.80:     .BYTE   -1                      ;

                                                .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                00000 COMMAND_STATE::
```

```
                                                        .BLKB   0
                     01F2  00000 ;TPA$TYPE
                                 U.2:       .WORD   498                                 ;
                     05F6  00002 ;TPA$TYPE
                                 U.3:       .WORD   1526                                ;
                     71F7  00004 ;TPA$TYPE
                                 U.4:       .WORD   29175                               ;
                00000000* 00006 ;TPA$ADDR
                                 U.5:       .LONG   <<COMMAND_NUMBER-U.5>-4>            ;
                00000008  0000A ;TPA$MASK
                                 U.6:       .LONG   8                                   ;
                    0000* 0000E ;TPA$TARGET
                                 U.8:       .WORD   <<U.7-U.8>-2>                       ;
                     7100  00010 ;TPA$TYPE
                                 U.12:      .WORD   28928                               ;
                00000000* 00012 ;TPA$ADDR
                                 U.13:      .LONG   <<COMMAND_NUMBER-U.13>-4>           ;
                00000001  00016 ;TPA$MASK
                                 U.14:      .LONG   1                                   ;
                    0000* 0001A ;TPA$TARGET
                                 U.15:      .WORD   <<U.7-U.15>-2>                      ;
                     7101  0001C ;TPA$TYPE
                                 U.19:      .WORD   28929                               ;
                00000000* 0001E ;TPA$ADDR
                                 U.20:      .LONG   <<COMMAND_NUMBER-U.20>-4>           ;
                00000003  00022 ;TPA$MASK
                                 U.21:      .LONG   3                                   ;
                    0000* 00026 ;TPA$TARGET
                                 U.23:      .WORD   <<U.22-U.23>-2>                     ;
                     7102  00028 ;TPA$TYPE
                                 U.27:      .WORD   28930                               ;
                00000000* 0002A ;TPA$ADDR
                                 U.28:      .LONG   <<COMMAND_NUMBER-U.28>-4>           ;
                00000004  0002E ;TPA$MASK
                                 U.29:      .LONG   4                                   ;
                    0000* 00032 ;TPA$TARGET
                                 U.30:      .WORD   <<U.22-U.30>-2>                     ;
                     7103  00034 ;TPA$TYPE
                                 U.34:      .WORD   28931                               ;
                00000000* 00036 ;TPA$ADDR
                                 U.35:      .LONG   <<COMMAND_NUMBER-U.35>-4>           ;
                00000005  0003A ;TPA$MASK
                                 U.36:      .LONG   5                                   ;
                    0000* 0003E ;TPA$TARGET
                                 U.37:      .WORD   <<U.7-U.37>-2>                      ;
                     7104  00040 ;TPA$TYPE
                                 U.41:      .WORD   28932                               ;
                00000000* 00042 ;TPA$ADDR
                                 U.42:      .LONG   <<COMMAND_NUMBER-U.42>-4>           ;
                00000006  00046 ;TPA$MASK
                                 U.43:      .LONG   6                                   ;
                    0000* 0004A ;TPA$TARGET
                                 U.44:      .WORD   <<U.7-U.44>-2>                      ;
                     7105  0004C ;TPA$TYPE
                                 U.48:      .WORD   28933                               ;
                00000000* 0004E ;TPA$ADDR
                                 U.49:      .LONG   <<COMMAND_NUMBER-U.49>-4>           ;
```

```
                                00000007  00052 ;TPA$MASK
                                          U.50:    .LONG   7                                       ;
                                    0000* 00056 ;TPA$TARGET
                                          U.51:    .WORD   <<U.22-U.51>-2>                         ;
                                    7106  00058 ;TPA$TYPE
                                          U.55:    .WORD   28934                                   ;
                                00000000* 0005A ;TPA$ADDR
                                          U.56:    .LONG   <<COMMAND_NUMBER-U.56>-4>               ;
                                00000008  0005E ;TPA$MASK
                                          U.57:    .LONG   8                                       ;
                                    0000* 00062 ;TPA$TARGET
                                          U.58:    .WORD   <<U.7-U.58>-2>                          ;
                                    7107  00064 ;TPA$TYPE
                                          U.62:    .WORD   28935                                   ;
                                00000000* 00066 ;TPA$ADDR
                                          U.63:    .LONG   <<COMMAND_NUMBER-U.63>-4>               ;
                                00000009  0006A ;TPA$MASK
                                          U.64:    .LONG   9                                       ;
                                    0000* 0006E ;TPA$TARGET
                                          U.65:    .WORD   <<U.7-U.65>-2>                          ;
                                    7108  00070 ;TPA$TYPE
                                          U.69:    .WORD   28936                                   ;
                                00000000* 00072 ;TPA$ADDR
                                          U.70:    .LONG   <<COMMAND_NUMBER-U.70>-4>               ;
                                0000000A  00076 ;TPA$MASK
                                          U.71:    .LONG   10                                      ;
                                    0000* 0007A ;TPA$TARGET
                                          U.72:    .WORD   <<U.7-U.72>-2>                          ;
                                    7509  0007C ;TPA$TYPE
                                          U.76:    .WORD   29961                                   ;
                                00000000* 0007E ;TPA$ADDR
                                          U.77:    .LONG   <<COMMAND_NUMBER-U.77>-4>               ;
                                0000000B  00082 ;TPA$MASK
                                          U.78:    .LONG   11                                      ;
                                    0000* 00086 ;TPA$TARGET
                                          U.79:    .WORD   <<U.7-U.79>-2>                          ;
                                          00088 ;NOARGS
                                          U.7:     .BLKB   0
                                    01F2  00088 ;TPA$TYPE
                                          U.81:    .WORD   498                                     ;
                                    05F6  0008A ;TPA$TYPE
                                          U.82:    .WORD   1526                                    ;
                                    15F7  0008C ;TPA$TYPE
                                          U.83:    .WORD   5623                                    ;
                                    FFFF  0008E ;TPA$TARGET
                                          U.84:    .WORD   -1                                      ;
                                          00090 ;ARGS
                                          U.22:    .BLKB   0
                                    11F2  00090 ;TPA$TYPE
                                          U.85:    .WORD   4594                                    ;
                                    FFFF  00092 ;TPA$TARGET
                                          U.86:    .WORD   -1                                      ;
                                    15F6  00094 ;TPA$TYPE
                                          U.87:    .WORD   5622                                    ;
                                    FFFF  00096 ;TPA$TARGET
                                          U.88:    .WORD   -1                                      ;
```

```
                                                        .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                                                00000 COMMAND_KEY::
                                                        .BLKB   0
                                                00000 ;TPA$KEY0
                                                U.1:    .BLKB   0
                                          0000* 00000 ;TPA$KEY
                                                U.10:   .WORD   <U.9-U.1>                               ;
                                          0000* 00002 ;TPA$KEY
                                                U.17:   .WORD   <U.16-U.1>                              ;
                                          0000* 00004 ;TPA$KEY
                                                U.25:   .WORD   <U.24-U.1>                              ;
                                          0000* 00006 ;TPA$KEY
                                                U.32:   .WORD   <U.31-U.1>                              ;
                                          0000* 00008 ;TPA$KEY
                                                U.39:   .WORD   <U.38-U.1>                              ;
                                          0000* 0000A ;TPA$KEY
                                                U.46:   .WORD   <U.45-U.1>                              ;
                                          0000* 0000C ;TPA$KEY
                                                U.53:   .WORD   <U.52-U.1>                              ;
                                          0000* 0000E ;TPA$KEY
                                                U.60:   .WORD   <U.59-U.1>                              ;
                                          0000* 00010 ;TPA$KEY
                                                U.67:   .WORD   <U.66-U.1>                              ;
                                          0000* 00012 ;TPA$KEY
                                                U.74:   .WORD   <U.73-U.1>                              ;

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                20  3E  45  5A  59  4C  41  4E  41  00239 P.ABV:  .ASCII  \ANALYZE> \                     ;
                                                00242           .BLKB   2
                                     00000009   00244 P.ABU:    .LONG   9                               ;
                                     00000000'  00248           .ADDRESS P.ABV                          ;

                                                        .PSECT  $OWN$,NOEXE,2

                          00000003  00000008  00A90 TPARSE_BLOCK:
                                                        .LONG   8, 3                                    ;
                                                00A98           .BLKB   28
                                                00AB4 COMMAND_NUMBER:
                                                        .BLKB   4


                                                        .PSECT  $CODE$,NOWRT,2

                                                00FC 00000      .ENTRY  ANL$INTERACTIVE_COMMAND, Save R2,R3,R4,R5,- ; 1062
                                                                R6,R7                                   ;
                          57 00000000G   00  9E 00002           MOVAB   LIB$SIGNAL, R7                  ;
                          56      0000'  CF  9E 00009           MOVAB   TPARSE_BLOCK+8, R6              ;
                          5E        AC   AE  9E 0000E           MOVAB   -84(SP), SP                     ;
                          52        08   AC  D0 00012           MOVL    ARGUMENTS, R2                   ; 1065
                          7E        50   8F  9A 00016           MOVZBL  #80, COMMAND_BUFFER             ; 1116
                    04    AE        08   AE  9E 0001A           MOVAB   COMMAND_BUFFER+8, COMMAND_BUFFER+4 ;
                          6E        50   8F  9B 0001F 1$:       MOVZBW  #80, COMMAND_BUFFER             ; 1122
                                         5E  DD 00023           PUSHL   SP                              ; 1123
                                  0000'  CF  9F 00025           PUSHAB  P.ABU                           ;
```

```
                                    08    AE  9F 00029         PUSHAB  COMMAND_BUFFER                                          ;
                          00000000G 00          03  FB 0002C         CALLS   #3, LIB$GET_INPUT                               ;
                                    53          50  D0 00033         MOVL    R0, STATUS                                      ;
                          0001827A  8F          53  D1 00036         CMPL    STATUS, #98938                                  ; 1128
                                                05  12 0003D         BNEQ    2$                                              ;
                                 04 BC          05  D0 0003F         MOVL    #5, @NUMBER                                     ; 1129
                                                04  00043         RET                                                     ; 1128
                                    05          53  E8 00044 2$:     BLBS    STATUS, 3$                                      ; 1132
                                    53          53  DD 00047         PUSHL   STATUS                                          ;
                                    67          01  FB 00049         CALLS   #1, LIB$SIGNAL                                  ;
                                    66          6E  3C 0004C 3$:     MOVZWL  COMMAND_BUFFER, TPARSE_BLOCK+8                  ; 1136
                                 04 A6       04 AE  D0 0004F         MOVL    COMMAND_BUFFER+4, TPARSE_BLOCK+12               ; 1137
                                    56          56  DD 00054         PUSHL   R6                                              ; 1138
                                    56          56  DD 00056         PUSHL   R6                                              ;
                          00000000G 00          02  FB 00058         CALLS   #2, STR$UPCASE                                  ;
                                             1C A6  D4 0005F         CLRL    COMMAND_NUMBER                                  ; 1139
                                       0000' CF  9F 00062         PUSHAB  COMMAND_KEY                                     ; 1140
                                       0000' CF  9F 00066         PUSHAB  COMMAND_STATE                                   ;
                                          F8 A6  9F 0006A         PUSHAB  TPARSE_BLOCK                                    ;
                          00000000G 00          03  FB 0006D         CALLS   #3, LIB$TPARSE                                  ;
                                    53          50  D0 00074         MOVL    R0, STATUS                                      ;
                                    01          53  D1 00077         CMPL    STATUS, #1                                      ; 1145
                                                0B  13 0007A         BEQL    4$                                              ;
                                00000000G       8F  DD 0007C         PUSHL   #ANLRMS$_BADCMD                                 ; 1146
                                    67          01  FB 00082         CALLS   #1, LIB$SIGNAL                                  ;
                                                98  11 00085         BRB     1$                                              ; 1116
                                                5E  DD 00087 4$:     PUSHL   SP                                              ; 1152
                                00000000G       8F  DD 00089         PUSHL   #ANLRMS$_INTERCOMMAND                           ;
                                                7E  D4 0008F         CLRL    -(SP)                                           ;
                                    7E          01  CE 00091         MNEGL   #1, -(SP)                                       ;
                                0000G CF        04  FB 00094         CALLS   #4, ANL$FORMAT_LINE                             ;
                                   04 BC     1C A6  D0 00099         MOVL    COMMAND_NUMBER, @NUMBER                         ; 1158
                                    62          66  B0 0009E         MOVW    TPARSE_BLOCK+8, (R2)                            ; 1159
                             04 B2     04 B6       66  28 000A1         MOVC3   TPARSE_BLOCK+8, @TPARSE_BLOCK+12, @4(R2)        ; 1160
                                                04  000A7         RET                                                     ; 1164

; Routine Size:  168 bytes,    Routine Base:  $CODE$ + 02B9
```

```
;  666          1165  1 %sbttl 'ANL$INTERACTIVE_DISPLAY - Display a File Structure'
;  667          1166  1 !++
;  668          1167  1 ! Functional Description:
;  669          1168  1 !       This routine is responsible for displaying the various structures
;  670          1169  1 !       that exist in an RMS file.  It is also responsible for determining
;  671          1170  1 !       the location of the structure following the one it displays.
;  672          1171  1 !
;  673          1172  1 ! Formal Parameters:
;  674          1173  1 !       structure_bsd   Address of BSD describing the structure to display.
;  675          1174  1 !                       It is updated to describe the following structure.
;  676          1175  1 !       parent_bsd      Address of BSD describing the parent of the structure.
;  677          1176  1 !
;  678          1177  1 ! Implicit Inputs:
;  679          1178  1 !       global data
;  680          1179  1 !
;  681          1180  1 ! Implicit Outputs:
;  682          1181  1 !       global data
;  683          1182  1 !
;  684          1183  1 ! Returned Value:
;  685          1184  1 !       none
;  686          1185  1 !
;  687          1186  1 ! Side Effects:
;  688          1187  1 !
;  689          1188  1 !--
;  690          1189  1
;  691          1190  1
;  692          1191  2 global routine anl$interactive_display(structure_bsd,parent_bsd): novalue = begin
;  693          1192  2
;  694          1193  2 bind
;  695          1194  2         s = .structure_bsd: bsd,
;  696          1195  2         p = .parent_bsd: bsd;
;  697          1196  2
;  698          1197  2 local
;  699          1198  2         sp: ref block[,byte],
;  700          1199  2         i: long;
;  701          1200  2
;  702          1201  2
;  703          1202  2 ! Set up the condition handler for drastic structure errors.
;  704          1203  2
;  705          1204  2 lib$establish(anl$unwind_handler);
;  706          1205  2
;  707          1206  2 ! Set up a pointer to the structure to be displayed.
;  708          1207  2
;  709          1208  2 sp = .s[bsd$l_bufptr] + .s[bsd$l_offset];
;  710          1209  2
;  711          1210  2 ! Because it requires a different routine to display each of the structures,
;  712          1211  2 ! this process is table-driven.  The structure type code in the BSD is
;  713          1212  2 ! an index into the STRUCTURE_TABLE, which contains a routine number for
;  714          1213  2 ! displaying the structure.  We simply case on that number.
;  715          1214  2
;  716          1215  2 case .structure_table[.s[bsd$w_type],0] from 1 to 30 of set
;  717          1216  2
;  718          1217  2 [1]:     ! Routine number 1 is for displaying the file header.  No updating
;  719          1218  2         ! of the BSD is necessary, since there is no "next" structure.
;  720          1219  2
;  721          1220  2         anl$format_file_header();
;  722          1221  2
```

```
;  723          1222  2
;  724          1223  2 [2]:     ! Routine number 2 is for displaying the RMS file attributes.
;  725          1224  2          ! No updating of the BSD is necessary.
;  726          1225  2
;  727          1226  2          anl$format_file_attributes();
;  728          1227  2
;  729          1228  2
;  730          1229  2 [3]:     ! Routine number 3 is for displaying a record from a sequential
;  731          1230  2          ! file.  The following routine will do so and update the BSD.
;  732          1231  2
;  733          1232  2          anl$seq_data_record(s,true,1);
;  734          1233  2
;  735          1234  2
;  736          1235  2 [4]:     ! Routine number 4 is for displaying the prolog of a relative file.
;  737          1236  2          ! The following routine will do it.
;  738          1237  2
;  739          1238  2          anl$rel_prolog(s,true,0);
;  740          1239  2
;  741          1240  2
;  742          1241  2 [5]:     ! Routine number 5 is for displaying the buckets of a relative file.
;  743          1242  2          ! This consists of nothing more than a heading.
;  744          1243  2
;  745          1244  3          (local
;  746          1245  3                  pp: ref block[,byte];
;  747          1246  3
;  748          1247  3          anl$format_line(3,0,anlrms$_relbucket,.s[bsd$l_vbn]);
;  749          1248  3
;  750          1249  3          ! Now we move on to the next bucket if there is one.  We can tell
;  751          1250  3          ! by looking at the end-of-file VBN in the prolog.
;  752          1251  3
;  753          1252  3          pp = .p[bsd$l_bufptr] + .p[bsd$l_offset];
;  754          1253  4          if .s[bsd$l_vbn]+2*.s[bsd$w_size] lequ .pp[plg$l_eof] then (
;  755          1254  4                  s[bsd$l_vbn] = .s[bsd$l_vbn] + .s[bsd$w_size];
;  756          1255  4                  s[bsd$l_offset] = 0;
;  757          1256  4                  anl$bucket(s,0);
;  758          1257  2          ););
;  759          1258  2
;  760          1259  2
;  761          1260  2 [6]:     ! Routine number 6 is for displaying the cells of a relative file.
;  762          1261  2          ! The following routine will do the work and update the BSD.
;  763          1262  2
;  764          1263  2          anl$rel_cell(s,true,1);
;  765          1264  2
;  766          1265  2
;  767          1266  2 [7]:     ! Routine number 7 is for displaying the prolog of an indexed file.
;  768          1267  2          ! The following routine will do it.
;  769          1268  2
;  770          1269  2          anl$idx_prolog(s,true,0);
;  771          1270  2
;  772          1271  2
;  773          1272  2 [8]:     ! Routine number 8 is for displaying an area descriptor in an indexed
;  774          1273  2          ! file.  The following routine will do it and update the BSD.
;  775          1274  2
;  776          1275  2          anl$area_descriptor(s,.sp[area$b_areaid],true,0);
;  777          1276  2
;  778          1277  2
;  779          1278  2 [9]:     ! Routine number 9 is for displaying a key descriptor in an indexed
```

```
;   780          1279  2          ! file.  The following routine will do it and update the BSD.
;   781          1280  2
;   782          1281  2          anl$key_descriptor(s,.sp[key$b_keyref],0,true,0);
;   783          1282  2
;   784          1283  2
;   785          1284  2 [10,
;   786          1285  2  11,
;   787          1286  2  12,
;   788          1287  2  13]:    ! Routine numbers 10 thru 13 are for displaying the bucket
;   789          1288  2          ! headers for primary index, secondary index, primary data, and
;   790          1289  2          ! secondary data buckets, respectively.  The following routine
;   791          1290  2          ! will do it and update the BSD.  This is for prolog 2.
;   792          1291  2
;   793          1292  2          anl$2bucket_header(s,.sp[bkt$b_areano],.sp[bkt$b_level],true,0);
;   794          1293  2
;   795          1294  2
;   796          1295  2 [14,
;   797          1296  2  15]:    ! Routine numbers 14 and 15 are for displaying the index records in
;   798          1297  2          ! primary and secondary indexes, respectively.  The following
;   799          1298  2          ! routine will do it and update the BSD.  The routine needs the key
;   800          1299  2          ! descriptor.  This is for prolog 2.
;   801          1300  2
;   802          1301  2          anl$2index_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   803          1302  2
;   804          1303  2
;   805          1304  2 [16]:    ! Routine number 16 is for displaying the primary data records in a
;   806          1305  2          ! primary data bucket.  The following routine will do it and update
;   807          1306  2          ! the BSD.  This is for prolog 2.
;   808          1307  2
;   809          1308  2          anl$2primary_data_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   810          1309  2
;   811          1310  2
;   812          1311  2 [17]:    ! Routine number 17 is for displaying the actual data record bytes
;   813          1312  2          ! in a primary data record.  The BSD points at the data record,
;   814          1313  2          ! which we will format in hex.  This is for prolog 2.
;   815          1314  2
;   816          1315  3          (local
;   817          1316  3                  rec_dsc: descriptor;
;   818          1317  3
;   819          1318  3          selectoneu .anl$gl_fat[fat$v_rtype] of set
;   820          1319  3          [fat$c_fixed]:          build_descriptor(rec_dsc,.anl$gl_fat[fat$w_maxrec],.sp);
;   821          1320  3
;   822          1321  3          [fat$c_variable,
;   823          1322  3           fat$c_vfc]:            build_descriptor(rec_dsc,2+.sp[0,0,16,0],.sp);
;   824          1323  3          tes;
;   825          1324  2          anl$format_hex(1,rec_dsc););
;   826          1325  2
;   827          1326  2
;   828          1327  2 [18]:    ! Routine number 18 is for displaying a SIDR record fixed portion.
;   829          1328  2          ! The following routine will do it, and update the BSD.
;   830          1329  2          ! It needs the key descriptor for this index.  This is for prolog 2.
;   831          1330  2
;   832          1331  2          anl$2sidr_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   833          1332  2
;   834          1333  2
;   835          1334  2 [19]:    ! Routine number 19 is for displaying a SIDR pointer.  The following
;   836          1335  2          ! routine will do it and update the BSD.  This is for prolog 2.
```

```
;   837     1336  2
;   838     1337  2          anl$2sidr_pointer(s,true,2);
;   839     1338  2
;   840     1339  2
;   841     1340  2 [20,
;   842     1341  2  21,
;   843     1342  2  22,
;   844     1343  2  23]:    ! Routines number 20 through 23 are for displaying primary and
;   845     1344  2          ! secondary index buckets, and primary and secondary data buckets.
;   846     1345  2          ! The following routine will do it and update the BSD.  This is
;   847     1346  2          ! for prolog 3.
;   848     1347  2
;   849     1348  3          (bind
;   850     1349  3                  k = current_stack[.key_level,0,0,0,0]: bsd,
;   851     1350  3                  kp = .k[bsd$l_bufptr] + .k[bsd$l_offset]: block[,byte];
;   852     1351  3
;   853     1352  2          anl$3bucket_header(s,.sp[bkt$b_indexno],.kp[key$v_dupkeys],.sp[bkt$b_level],true,0););
;   854     1353  2
;   855     1354  2
;   856     1355  2 [24,
;   857     1356  2  25]:    ! Routines number 24 and 25 are for displaying the index records
;   858     1357  2          ! in primary and secondary indexes, respectively.  The following
;   859     1358  2          ! routine will do it and update the BSD.  It needs the key
;   860     1359  2          ! descriptor.  This is for prolog 3.
;   861     1360  2
;   862     1361  2          anl$3index_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   863     1362  2
;   864     1363  2
;   865     1364  2 [26]:    ! Routine number 26 is for displaying the primary data records in a
;   866     1365  2          ! primary data bucket.  The following routine will do it and update
;   867     1366  2          ! the BSD.  It needs the key descriptor.  This is for prolog 3.
;   868     1367  2
;   869     1368  2          anl$3primary_data_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   870     1369  2
;   871     1370  2
;   872     1371  2 [27]:    ! Routine number 27 is for displaying the actual data record bytes
;   873     1372  2          ! in a primary data record.  We call a routine to do it.  This is
;   874     1373  2          ! for prolog 3.
;   875     1374  2
;   876     1375  2          anl$3format_data_bytes(1,s,current_stack[.key_level,0,0,0,0]);
;   877     1376  2
;   878     1377  2
;   879     1378  2 [28]:    ! Routine number 28 is for displaying a SIDR record fixed portion
;   880     1379  2          ! for prolog 3.  The following routine will do it, and update the BSD.
;   881     1380  2          ! It needs the key descriptor for this index.
;   882     1381  2
;   883     1382  2          anl$3sidr_record(s,current_stack[.key_level,0,0,0,0],true,1);
;   884     1383  2
;   885     1384  2
;   886     1385  2 [29]:    ! Routine number 29 is for displaying a SIDR pointer for prologue 3.
;   887     1386  2          ! The following routine will do it and update the BSD.
;   888     1387  2
;   889     1388  2          anl$3sidr_pointer(s,true,2);
;   890     1389  2
;   891     1390  2
;   892     1391  2 [30]:    ! Routine number 30 is for displaying the header of a reclaimed
;   893     1392  2          ! bucket on the available chain off an area descriptor.  This
```

```
;  894          1393  2          ! routine works for all prologs.
;  895          1394  2
;  896          1395  2          anl$3reclaimed_bucket_header(s,true,0);
;  897          1396  2 tes;
;  898          1397  2
;  899          1398  2 return;
;  900          1399  2
;  901          1400  1 end;


                                                003C 00000          .ENTRY  ANL$INTERACTIVE_DISPLAY, Save R2,R3,R4,R5    ; 1191
                                    55  0000'    CF   9E 00002       MOVAB   KEY_LEVEL, R5                                ;
                                    5E          08   C2 00007       SUBL2   #8, SP                                       ;
                                    52    04    AC   D0 0000A       MOVL    STRUCTURE_BSD, R2                            ; 1194
                                    54    08    AC   D0 0000E       MOVL    PARENT_BSD, R4                               ; 1195
                                          0000G CF   9F 00012       PUSHAB  ANL$UNWIND_HANDLER                           ; 1204
                            00000000G   00          01   FB 00016   CALLS   #1, LIB$ESTABLISH                          ;
                                53    0C    A2    08   A2 C1 0001D   ADDL3   8(R2), 12(R2), SP                          ; 1208
                                          50          62   3C 00023   MOVZWL  (R2), R0                                  ; 1215
                                          F574 C540   DF   00026     PUSHAL  STRUCTURE_TABLE[R0]                          ;
                                    1D              9E   8F 0002B     CASEB   @(SP)+, #1, #29                              ;
                          0054          0048          003C    0002F 1$: .WORD   2$-1$,-                                  ;
                          00AE          00A3          005F    00037           3$-1$,-                                    ;
                          00CE          00CE          00BD    0003F           4$-1$,-                                    ;
                          00F6          00E1          00CE    00047           5$-1$,-                                    ;
                          015F          0153          010B    0004F           6$-1$,-                                    ;
                          0188          015F          015F    00057           8$-1$,-                                    ;
                          01C5          01B2          0188    0005F           9$-1$,-                                    ;
                                        01E6          01DA    00067           10$-1$,-                                   ;
                                                              11$-1$,-                                                   ;
                                                              12$-1$,-                                                   ;
                                                              12$-1$,-                                                   ;
                                                              12$-1$,-                                                   ;
                                                              12$-1$,-                                                   ;
                                                              13$-1$,-                                                   ;
                                                              13$-1$,-                                                   ;
                                                              14$-1$,-                                                   ;
                                                              15$-1$,-                                                   ;
                                                              19$-1$,-                                                   ;
                                                              20$-1$,-                                                   ;
                                                              21$-1$,-                                                   ;
                                                              21$-1$,-                                                   ;
                                                              21$-1$,-                                                   ;
                                                              21$-1$,-                                                   ;
                                                              21$-1$,-                                                   ;
                                                              22$-1$,-                                                   ;
                                                              22$-1$,-                                                   ;
                                                              23$-1$,-                                                   ;
                                                              24$-1$,-                                                   ;
                                                              25$-1$,-                                                   ;
                                                              26$-1$,-                                                   ;
                                                              27$-1$                                                     ;
                          0000G CF          00  FB 0006B 2$:  CALLS   #0, ANL$FORMAT_FILE_HEADER                       ; 1220
                                              04 00070        RET                                                      ;
                          0000G CF          00  FB 00071 3$:  CALLS   #0, ANL$FORMAT_FILE_ATTRIBUTES                   ; 1226
```

```
                                                 04 00076                RET
                                              01 DD 00077 4$:            PUSHL   #1                                          ; 1232
                                              01 DD 00079                PUSHL   #1                                          ;
                                              52 DD 0007B                PUSHL   R2                                          ;
                                     0000G    CF    03 FB 0007D          CALLS   #3, ANL$SEQ_DATA_RECORD                     ;
                                                 04 00082                RET                                                 ;
                                           7E    01 7D 00083 5$:        MOVQ    #1, -(SP)                                   ; 1238
                                              52 DD 00086                PUSHL   R2                                          ;
                                     0000G    CF    03 FB 00088          CALLS   #3, ANL$REL_PROLOG                          ;
                                                 04 0008D                RET                                                 ;
                                        04    A2 DD 0008E 6$:            PUSHL   4(R2)                                       ; 1247
                                 00000000G    8F DD 00091                PUSHL   #ANLRMS$_RELBUCKET                          ;
                                           7E    03 7D 00097            MOVQ    #3, -(SP)                                   ;
                                     0000G    CF    04 FB 0009A          CALLS   #4, ANL$FORMAT_LINE                         ;
                                  51    0C    A4 08 A4 C1 0009F          ADDL3   8(R4), 12(R4), PP                           ; 1252
                                        50    02 A2 3C 000A5             MOVZWL  2(R2), R0                                   ; 1253
                                        54    04 B240 3E 000A9           MOVAW   @4(R2)[R0], R4                              ;
                                        70    A1 54 D1 000AE             CMPL    R4, 112(PP)                                 ;
                                              01 1B 000B2                BLEQU   7$                                          ;
                                                 04 000B4                RET                                                 ;
                                        04    A2 50 C0 000B5 7$:         ADDL2   R0, 4(R2)                                   ; 1254
                                           08 A2 D4 000B9                CLRL    8(R2)                                       ; 1255
                                           7E D4 000BC                   CLRL    -(SP)                                       ; 1256
                                              52 DD 000BE                PUSHL   R2                                          ;
                                     0000G    CF    02 FB 000C0          CALLS   #2, ANL$BUCKET                              ;
                                                 04 000C5                RET                                                 ; 1215
                                              01 DD 000C6 8$:            PUSHL   #1                                          ; 1263
                                              01 DD 000C8                PUSHL   #1                                          ;
                                              52 DD 000CA                PUSHL   R2                                          ;
                                     0000G    CF    03 FB 000CC          CALLS   #3, ANL$REL_CELL                            ;
                                                 04 000D1                RET                                                 ;
                                           7E    01 7D 000D2 9$:        MOVQ    #1, -(SP)                                   ; 1269
                                              52 DD 000D5                PUSHL   R2                                          ;
                                     0000G    CF    03 FB 000D7          CALLS   #3, ANL$IDX_PROLOG                          ;
                                                 04 000DC                RET                                                 ;
                                           7E    01 7D 000DD 10$:       MOVQ    #1, -(SP)                                   ; 1275
                                           7E 02 A3 9A 000E0            MOVZBL  2(SP), -(SP)                                ;
                                              52 DD 000E4                PUSHL   R2                                          ;
                                     0000G    CF    04 FB 000E6          CALLS   #4, ANL$AREA_DESCRIPTOR                     ;
                                                 04 000EB                RET                                                 ;
                                           7E    01 7D 000EC 11$:       MOVQ    #1, -(SP)                                   ; 1281
                                           7E    7E D4 000EF            CLRL    -(SP)                                       ;
                                           7E 15 A3 9A 000F1            MOVZBL  21(SP), -(SP)                               ;
                                              52 DD 000F5                PUSHL   R2                                          ;
                                     0000G    CF    05 FB 000F7          CALLS   #5, ANL$KEY_DESCRIPTOR                      ;
                                                 04 000FC                RET                                                 ;
                                           7E    01 7D 000FD 12$:       MOVQ    #1, -(SP)                                   ; 1292
                                           7E 0C A3 9A 00100            MOVZBL  12(SP), -(SP)                               ;
                                           7E 01 A3 9A 00104            MOVZBL  1(SP), -(SP)                                ;
                                              52 DD 00108                PUSHL   R2                                          ;
                                     0000G    CF    05 FB 0010A          CALLS   #5, ANL$2BUCKET_HEADER                      ;
                                                 04 0010F                RET                                                 ;
                                              01 DD 00110 13$:           PUSHL   #1                                          ; 1301
                                              01 DD 00112                PUSHL   #1                                          ;
                                  50          65    18 C5 00114          MULL3   #24, KEY_LEVEL, R0                          ;
                                           FA00 C540 9F 00118            PUSHAB  CURRENT_STACK[R0]                           ;
                                              52 DD 0011D                PUSHL   R2                                          ;
```

```
                    0000G  CF              04  FB 0011F              CALLS   #4, ANL$2INDEX_RECORD                                ;
                                            04 00124                RET                                                         ;
                                        01  DD 00125 14$:           PUSHL   #1                                                  ; 1308
                                        01  DD 00127                PUSHL   #1                                                  ;
                50                      65  18  C5 00129            MULL3   #24, KEY_LEVEL, R0                                  ;
                                    FA00 C540  9F 0012D             PUSHAB  CURRENT_STACK[R0]                                   ;
                                        52  DD 00132                PUSHL   R2                                                  ;
                    0000G  CF              04  FB 00134             CALLS   #4, ANL$2PRIMARY_DATA_RECORD                        ;
                                            04 00139                RET                                                         ;
                        50      0000G  CF  D0 0013A 15$:            MOVL    ANL$GL_FAT, R0                                      ; 1318
          51            60              04  00 0013F               EXTZV   #0, #4, (R0), R1                                    ;
                                        01  51 D1 00144             CMPL    R1, #1                                              ; 1319
                                            06 12 00147             BNEQ    16$                                                 ;
                                        6E   10 A0 3C 00149         MOVZWL  16(R0), REC_DSC                                     ;
                                            10 11 0014D             BRB     17$                                                 ;
                                        02  51 D1 0014F 16$:        CMPL    R1, #2                                              ; 1321
                                            0F 1F 00152             BLSSU   18$                                                 ;
                                        03  51 D1 00154             CMPL    R1, #3                                              ;
                                            0A 1A 00157             BGTRU   18$                                                 ;
                                        6E  63 3C 00159             MOVZWL  (SP), REC_DSC                                       ; 1322
                                        6E  02 C0 0015C             ADDL2   #2, REC_DSC                                         ;
                                  04    AE  53 D0 0015F 17$:        MOVL    SP, REC_DSC+4                                       ;
                                            5E DD 00163 18$:        PUSHL   SP                                                  ; 1324
                                        01  DD 00165                PUSHL   #1                                                  ;
                    0000G  CF              02  FB 00167             CALLS   #2, ANL$FORMAT_HEX                                  ;
                                            04 0016C                RET                                                         ; 1215
                                        01  DD 0016D 19$:           PUSHL   #1                                                  ; 1331
                                        01  DD 0016F                PUSHL   #1                                                  ;
                50                      65  18  C5 00171            MULL3   #24, KEY_LEVEL, R0                                  ;
                                    FA00 C540  9F 00175             PUSHAB  CURRENT_STACK[R0]                                   ;
                                        52  DD 0017A                PUSHL   R2                                                  ;
                    0000G  CF              04  FB 0017C             CALLS   #4, ANL$2SIDR_RECORD                                ;
                                            04 00181                RET                                                         ;
                                        02  DD 00182 20$:           PUSHL   #2                                                  ; 1337
                                        01  DD 00184                PUSHL   #1                                                  ;
                                        52  DD 00186                PUSHL   R2                                                  ;
                    0000G  CF              03  FB 00188             CALLS   #3, ANL$2SIDR_POINTER                               ;
                                            04 0018D                RET                                                         ;
                50                      65  18  C5 0018E 21$:      MULL3   #24, KEY_LEVEL, R0                                  ; 1349
                                        50  FA00 C540  9E 00192      MOVAB   CURRENT_STACK[R0], R0                               ;
                50          0C          A0   08  A0  C1 00198      ADDL3   8(R0), T2(R0), R0                                   ; 1350
                                        7E  01  7D 0019E            MOVQ    #1, -(SP)                                           ; 1352
                                        7E   0C  A3 9A 001A1        MOVZBL  12(SP), -(SP)                                       ;
          7E        10  A0              01  00  EF 001A5            EXTZV   #0, #1, 16(R0), -(SP)                               ;
                                        7E   01  A3 9A 001AB        MOVZBL  1(SP), -(SP)                                        ;
                                        52  DD 001AF                PUSHL   R2                                                  ;
                    0000G  CF              06  FB 001B1             CALLS   #6, ANL$3BUCKET_HEADER                              ;
                                            04 001B6                RET                                                         ; 1215
                                        01  DD 001B7 22$:           PUSHL   #1                                                  ; 1361
                                        01  DD 001B9                PUSHL   #1                                                  ;
                50                      65  18  C5 001BB            MULL3   #24, KEY_LEVEL, R0                                  ;
                                    FA00 C540  9F 001BF             PUSHAB  CURRENT_STACK[R0]                                   ;
                                        52  DD 001C4                PUSHL   R2                                                  ;
                    0000G  CF              04  FB 001C6             CALLS   #4, ANL$3INDEX_RECORD                               ;
                                            04 001CB                RET                                                         ;
                                        01  DD 001CC 23$:           PUSHL   #1                                                  ; 1368
                                        01  DD 001CE                PUSHL   #1                                                  ;
```

```
                50              65              18  C5 001D0           MULL3   #24, KEY_LEVEL, R0                      ;
                                    FA00 C540   9F 001D4           PUSHAB  CURRENT_STACK[R0]                       ;
                                            52  DD 001D9           PUSHL   R2                                      ;
                        0000G   CF              04  FB 001DB           CALLS   #4, ANL$3PRIMARY_DATA_RECORD            ;
                                                04 001E0           RET                                             ;
                50              65              18  C5 001E1 24$:      MULL3   #24, KEY_LEVEL, R0                      ; 1375
                                    FA00 C540   9F 001E5           PUSHAB  CURRENT_STACK[R0]                       ;
                                            52  DD 001EA           PUSHL   R2                                      ;
                                            01  DD 001EC           PUSHL   #1                                      ;
                        0000G   CF              03  FB 001EE           CALLS   #3, ANL$3FORMAT_DATA_BYTES              ;
                                                04 001F3           RET                                             ;
                                            01  DD 001F4 25$:      PUSHL   #1                                      ; 1382
                                            01  DD 001F6           PUSHL   #1                                      ;
                50              65              18  C5 001F8           MULL3   #24, KEY_LEVEL, R0                      ;
                                    FA00 C540   9F 001FC           PUSHAB  CURRENT_STACK[R0]                       ;
                                            52  DD 00201           PUSHL   R2                                      ;
                        0000G   CF              04  FB 00203           CALLS   #4, ANL$3SIDR_RECORD                    ;
                                                04 00208           RET                                             ;
                                            02  DD 00209 26$:      PUSHL   #2                                      ; 1388
                                            01  DD 0020B           PUSHL   #1                                      ;
                                            52  DD 0020D           PUSHL   R2                                      ;
                        0000G   CF              03  FB 0020F           CALLS   #3, ANL$3SIDR_POINTER                   ;
                                                04 00214           RET                                             ;
                                7E              01  7D 00215 27$:      MOVQ    #1, -(SP)                           ; 1395
                                            52  DD 00218           PUSHL   R2                                      ;
                        0000G   CF              03  FB 0021A           CALLS   #3, ANL$3RECLAIMED_BUCKET_HEADER        ;
                                                04 0021F           RET                                             ; 1400

; Routine Size:  544 bytes,    Routine Base:  $CODE$ + 0361
```

```
;  903          1401  1 %sbttl 'ANL$INTERACTIVE_DOWN - Handle DOWN Command'
;  904          1402  1 !++
;  905          1403  1 ! Functional Description:
;  906          1404  1 !       This routine handles the interactive DOWN command.  It is responsible
;  907          1405  1 !       for determining the path that the user wants to take, and constructing
;  908          1406  1 !       a BSD that describes the resulting structure.
;  909          1407  1 !
;  910          1408  1 ! Formal Parameters:
;  911          1409  1 !       path            Address of descriptor of desired path name.
;  912          1410  1 !       current_bsd     Address of BSD describing current structure.
;  913          1411  1 !       down_bsd        Address of BSD to fill in with the down structure.
;  914          1412  1 !       new_level       The stack level of the BSD to fill.
;  915          1413  1 !
;  916          1414  1 ! Implicit Inputs:
;  917          1415  1 !       global data
;  918          1416  1 !
;  919          1417  1 ! Implicit Outputs:
;  920          1418  1 !       global data
;  921          1419  1 !
;  922          1420  1 ! Returned Value:
;  923          1421  1 !       True if there is a down structure, false if not.
;  924          1422  1 !
;  925          1423  1 ! Side Effects:
;  926          1424  1 !
;  927          1425  1 !--
;  928          1426  1
;  929          1427  1
;  930          1428  2 global routine anl$interactive_down(path,current_bsd,down_bsd,new_level) = begin
;  931          1429  2
;  932          1430  2 bind
;  933          1431  2         path_dsc = .path: descriptor,
;  934          1432  2         c = .current_bsd: bsd,
;  935          1433  2         d = .down_bsd: bsd;
;  936          1434  2
;  937          1435  2 local
;  938          1436  2         i: long, j: long,
;  939          1437  2         path_index: long,
;  940          1438  2         cp: ref block[,byte],
;  941          1439  2         hp: ref block[,byte],
;  942          1440  2         sp: ref block[,byte];
;  943          1441  2
;  944          1442  2
;  945          1443  2 ! Establish the condition handler for drastic structure errors.
;  946          1444  2
;  947          1445  2 lib$establish(anl$unwind_handler);
;  948          1446  2
;  949          1447  2 ! The first thing we need to check is whether there are any possible
;  950          1448  2 ! paths down from the current structure.  If not, that's an error.
;  951          1449  2
;  952          1450  3 if .structure_table[.c[bsd$w_type],1] eqlu 0 then (
;  953          1451  3         signal (anlrms$_nodown);
;  954          1452  3         return false;
;  955          1453  2 );
;  956          1454  2
;  957          1455  2 ! Now, if the user has entered the command DOWN ?, or has not entered
;  958          1456  2 ! any path name at all and there is more than one way down, we need to
;  959          1457  2 ! display a list of possible paths.
```

```
;  960          1458  2
;  961          1459  2 if (.path_dsc[len] gequ 1 and ch$rchar(.path_dsc[ptr]) eqlu '?') or
;  962          1460  3    (.path_dsc[len] eqlu 0 and .structure_table[.c[bsd$w_type],2] nequ 0) then (
;  963          1461  3         signal (anlrms$_downhelp);
;  964          1462  3         incru i from 1 to 3 do
;  965          1463  3                 if (j = .structure_table[.c[bsd$w_type],.i]) nequ 0 then
;  966          1464  3                         signal (anlrms$_downpath,2,.path_table[.j,path_name],.path_table[.j,path_text]);
;  967          1465  3         return false;
;  968          1466  2 );
;  969          1467  2
;  970          1468  2 ! Now, if the user has entered a path name, we need to figure which path
;  971          1469  2 ! they have specified.  If they didn't enter one, we know at this point
;  972          1470  2 ! that there is only one way down.
;  973          1471  2
;  974          1472  3 if .path_dsc[len] gtru 0 then (
;  975          1473  3         local
;  976          1474  3                 length: long;
;  977          1475  3
;  978          1476  3         ! Now loop through the down paths specified by this structure entry.
;  979          1477  3         ! We are looking for a path name that matches what the user entered.
;  980          1478  3
;  981          1479  3         path_index = 0;
;  982          1480  3         incru i from 1 to 3 do
;  983          1481  4                 if (j = .structure_table[.c[bsd$w_type],.i]) nequ 0 then (
;  984          1482  4                         bind
;  985          1483  4                                 a_path_name = .path_table[.j,path_name];
;  986          1484  4                         length = minu(ch$rchar(a_path_name),.path_dsc[len]);
;  987          1485  5                         if ch$eql(.length,.path_dsc[ptr], .length,a_path_name+1,' ') then (
;  988          1486  5                                 path_index = .j;
;  989          1487  5         exitloop;
;  990          1488  4                         );
;  991          1489  3                 );
;  992          1490  3
;  993          1491  2 ) else
;  994          1492  2         path_index = .structure_table[.c[bsd$w_type],1];
```

```
;  996          1493  2 ! Let's set up a pointer to the current structure.  Also we sometimes need
;  997          1494  2 ! one to the bucket header.
;  998          1495  2
;  999          1496  2 cp = .c[bsd$l_bufptr] + .c[bsd$l_offset];
; 1000          1497  2 hp = .c[bsd$l_bufptr];
; 1001          1498  2
; 1002          1499  2 ! OK, now we can case on the path routine number and actually effect
; 1003          1500  2 ! the downward movement.  We are to fill in the down_bsd with a description
; 1004          1501  2 ! of the resulting structure.  The BSD type is specified in the path table.
; 1005          1502  2
; 1006          1503  2 init_bsd(d);
; 1007          1504  2 d[bsd$w_type] = .path_table[.path_index,path_result];
; 1008          1505  2
; 1009          1506  2 case .path_table[.path_index,path_routine] from 0 to 22 of set
; 1010          1507  2
; 1011          1508  2 [0]:     ! If the path_index wasn't set to a valid path number, then the
; 1012          1509  2          ! user must have entered a bad path name.
; 1013          1510  2
; 1014          1511  3          (signal (anlrms$_badpath);
; 1015          1512  2          return false;);
; 1016          1513  2
; 1017          1514  2
; 1018          1515  2 [1]:     ! Downward path 1 is from the file header to the RMS attribute
; 1019          1516  2          ! area.  All we need to fill in is the type, which was done above.
; 1020          1517  2
; 1021          1518  2          ;
; 1022          1519  2
; 1023          1520  2
; 1024          1521  2 [2]:     ! Downward path 2 is from the RMS attribute area to the actual
; 1025          1522  2          ! blocks of the file.  The structure type depends on file organization.
; 1026          1523  2          ! If it's a sequential file, we have to check that there are
; 1027          1524  2          ! any records at all.
; 1028          1525  2
; 1029          1526  3          (d[bsd$w_type] =
; 1030          1527  4                  (selectoneu .anl$gl_fat[fat$v_fileorg] of set
; 1031          1528  4
; 1032          1529  5                  [fat$c_sequential]:    (if .anl$gl_fat[fat$l_efblk] eqlu 1 and
; 1033          1530  6                                             .anl$gl_fat[fat$w_ffbyte] eqlu 0 then (
; 1034          1531  6                                                 signal (anlrms$_norecs);
; 1035          1532  6                                                 return false;
; 1036          1533  5                                             );
; 1037          1534  4                                             3);
; 1038          1535  4
; 1039          1536  4                  [fat$c_relative]:      4;
; 1040          1537  4
; 1041          1538  4                  [fat$c_indexed]:       7;
; 1042          1539  3                  tes);
; 1043          1540  3          d[bsd$w_size] = 1;
; 1044          1541  2          d[bsd$l_vbn] = 1;);
; 1045          1542  2
; 1046          1543  2 [3]:     ! Downward path 3 is from a relative file prolog to its first
; 1047          1544  2          ! data bucket.  There may not be any.
; 1048          1545  2
; 1049          1546  3          if .anl$gl_fat[fat$l_hiblk]-1 lssu .anl$gl_fat[fat$b_bktsize] then (
; 1050          1547  3                  signal (anlrms$_norecs);
; 1051          1548  3                  return false;
; 1052          1549  3          ) else (
```

```
;  1053          1550  3                   d[bsd$w_size] = .anl$gl_fat[fat$b_bktsize];
;  1054          1551  3                   d[bsd$l_vbn] = .cp[plg$w_dvbn];
;  1055          1552  2           );
;  1056          1553  2
;  1057          1554  2
;  1058          1555  2 [4]:     ! Downward path 4 is from a relative file bucket to the first
;  1059          1556  2           ! first cell in the bucket.
;  1060          1557  2
;  1061          1558  3           (d[bsd$w_size] = .c[bsd$w_size];
;  1062          1559  2           d[bsd$l_vbn] = .c[bsd$l_vbn];);
;  1063          1560  2
;  1064          1561  2
;  1065          1562  2 [5]:     ! Downward path 5 is from an indexed file prolog to the first
;  1066          1563  2           ! area descriptor.
;  1067          1564  2
;  1068          1565  3           (d[bsd$w_size] = 1;
;  1069          1566  2           d[bsd$l_vbn] = .cp[plg$b_avbn];);
;  1070          1567  2
;  1071          1568  2
;  1072          1569  2 [6]:     ! Downward path 6 is from an indexed file prolog to the first
;  1073          1570  2           ! key descriptor.  We need to remember the stack level of the
;  1074          1571  2           ! BSD we are creating, because lots of other folks need to get
;  1075          1572  2           ! at the key descriptor.
;  1076          1573  2
;  1077          1574  3           (d[bsd$w_size] = 1;
;  1078          1575  3           d[bsd$l_vbn] = 1;
;  1079          1576  2           key_level = .new_level);
;  1080          1577  2
;  1081          1578  2
;  1082          1579  2 [7]:     ! Downward path 7 is from an indexed file key descriptor to either
;  1083          1580  2           ! the primary or secondary index buckets.  We must distinguish
;  1084          1581  2           ! between prolog 2 and 3 files and worry about uninitialized indexes.
;  1085          1582  2
;  1086          1583  3           if .cp[key$v_initidx] then (
;  1087          1584  3                   signal (anlrms$_uninitindex);
;  1088          1585  3                   return false;
;  1089          1586  3           ) else (
;  1090          1587  4                   d[bsd$w_type] = (if .anl$gw_prolog eqlu plg$c_ver_3 then
;  1091          1588  4                                       if .cp[key$b_keyref] eqlu 0 then 20 else 21
;  1092          1589  4                                   else
;  1093          1590  3                                       if .cp[key$b_keyref] eqlu 0 then 10 else 11);
;  1094          1591  3                   d[bsd$w_size] = .cp[key$b_idxbktsz];
;  1095          1592  3                   d[bsd$l_vbn] = .cp[key$l_rootvbn];
;  1096          1593  2           );
;  1097          1594  2
;  1098          1595  2
;  1099          1596  2 [8]:     ! Downward path 8 is from an indexed file key descriptor to either
;  1100          1597  2           ! the primary or secondary data buckets.  We must distinguish
;  1101          1598  2           ! between prolog 2 and 3 files and worry about uninitialized indexes.
;  1102          1599  2
;  1103          1600  3           if .cp[key$v_initidx] then (
;  1104          1601  3                   signal (anlrms$_uninitindex);
;  1105          1602  3                   return false;
;  1106          1603  3           ) else (
;  1107          1604  4                   d[bsd$w_type] = (if .anl$gw_prolog eqlu plg$c_ver_3 then
;  1108          1605  4                                       if .cp[key$b_keyref] eqlu 0 then 22 else 23
;  1109          1606  4                                   else
```

```
;  1110          1607  3                                                  if .cp[key$b_keyref] eqlu 0 then 12 else 13);
;  1111          1608  3                      d[bsd$w_size] = .cp[key$b_datbktsz];
;  1112          1609  3                      d[bsd$l_vbn] = .cp[key$l_[dvbn];
;  1113          1610  2              );
;  1114          1611  2
;  1115          1612  2
;  1116          1613  2 [9]:      ! Downward path 9 is from an index file index bucket to the first
;  1117          1614  2           ! index entry in the bucket.  This is for prolog 2.
;  1118          1615  2
;  1119          1616  3           (d[bsd$w_type] = (if .c[bsd$w_type] eqlu 10 then 14 else 15);
;  1120          1617  3           d[bsd$w_size] = .c[bsd$w_size];
;  1121          1618  3           d[bsd$l_vbn] = .c[bsd$l_vbn];
;  1122          1619  2           d[bsd$l_offset] = bkt$c_overhdsz;);
;  1123          1620  2
;  1124          1621  2
;  1125          1622  2 [10]:     ! Downward path 10 is from a primary or secondary index record to
;  1126          1623  2           ! the index or data bucket pointed to by it.  This is for prolog 2.
;  1127          1624  2
;  1128          1625  4           (if .hp[bkt$b_level] gequ 2 then (
;  1129          1626  4
;  1130          1627  4                   ! The next lower level is another index bucket.  Set the
;  1131          1628  4                   ! type according to whether it's primary or secondary.
;  1132          1629  4                   ! Set the size the same as the current index bucket.
;  1133          1630  4
;  1134          1631  4                   d[bsd$w_type] = (if .c[bsd$w_type] eqlu 14 then 10 else 11);
;  1135          1632  4                   d[bsd$w_size] = .c[bsd$w_size];
;  1136          1633  4           ) else (
;  1137          1634  4
;  1138          1635  4                   ! The next lower level is the data buckets.  Set the type
;  1139          1636  4                   ! according to whether it's a primary or secondary bucket.
;  1140          1637  4                   ! The size has to be found from the key descriptor.
;  1141          1638  4
;  1142          1639  4                   d[bsd$w_type] = (if .c[bsd$w_type] eqlu 14 then 12 else 13);
;  1143          1640  5                   begin
;  1144          1641  5                   bind
;  1145          1642  5                           k = current_stack[.key_level,0,0,0,0]: bsd,
;  1146          1643  5                           kp = .k[bsd$l_bufptr] + .k[bsd$l_offset]: block[,byte];
;  1147          1644  5
;  1148          1645  5                   d[bsd$w_size] = .kp[key$b_datbktsz];
;  1149          1646  4                   end;
;  1150          1647  3           );
;  1151          1648  3
;  1152          1649  3           ! Now we set up the VBN of the downward structure by looking in the
;  1153          1650  3           ! index record.
;  1154          1651  3
;  1155          1652  4           d[bsd$l_vbn] =  (case .cp[irc$v_ptrsz] from 0 to 2 of set
;  1156          1653  4                           [0]:    .cp[1,0,16,0];
;  1157          1654  4                           [1]:    .cp[1,0,24,0];
;  1158          1655  4                           [2]:    .cp[1,0,32,0];
;  1159          1656  3                           tes);
;  1160          1657  2           d[bsd$l_offset] = 0;);
;  1161          1658  2
;  1162          1659  2
;  1163          1660  2 [11]:     ! Downward path 11 is from a primary data bucket to the first record
;  1164          1661  2           ! in the bucket.  There might not be any.
;  1165          1662  2
;  1166          1663  3           if .hp[bkt$w_freespace] eqlu bkt$c_overhdsz then (
```

```
; 1167          1664  3                   signal (anlrms$_emptybkt);
; 1168          1665  3                   return false;
; 1169          1666  3           ) else (
; 1170          1667  3                   d[bsd$w_size] = .c[bsd$w_size];
; 1171          1668  3                   d[bsd$l_vbn] = .c[bsd$l_vbn];
; 1172          1669  3                   d[bsd$l_offset] = bkt$c_overhdsz;
; 1173          1670  2           );
; 1174          1671  2
; 1175          1672  2
; 1176          1673  2 [12]:     ! Downward path 12 is from a primary data record to the actual
; 1177          1674  2           ! record bytes.  They may not exist.  This is for prolog 2.
; 1178          1675  2
; 1179          1676  3           if .cp[irc$v_deleted] or .cp[irc$v_rrv] then (
; 1180          1677  3                   signal (anlrms$_nodata);
; 1181          1678  3                   return false;
; 1182          1679  3           ) else (
; 1183          1680  3                   d[bsd$w_size] = .c[bsd$w_size];
; 1184          1681  3                   d[bsd$l_vbn] = .c[bsd$l_vbn];
; 1185          1682  3                   d[bsd$l_offset] = .c[bsd$l_offset] +
; 1186          1683  3                                     1 +
; 1187          1684  3                                     1 +
; 1188          1685  3                                     (if .cp[irc$v_noptrsz] then 0 else .cp[irc$v_ptrsz]+3);
; 1189          1686  2           );
; 1190          1687  2
; 1191          1688  2
; 1192          1689  2 [13]:     ! Downward path 13 is from a primary data record to the data bucket
; 1193          1690  2           ! pointed at by the RRV.  The pointer may not exist.  This is for
; 1194          1691  2           ! prolog 2.
; 1195          1692  2
; 1196          1693  3           if .cp[irc$v_noptrsz] then (
; 1197          1694  3                   signal (anlrms$_norrv);
; 1198          1695  3                   return false;
; 1199          1696  3           ) else (
; 1200          1697  3                   d[bsd$w_size] = .c[bsd$w_size];
; 1201          1698  4                   d[bsd$l_vbn] =  (case .cp[irc$v_ptrsz] from 0 to 2 of set
; 1202          1699  4                                   [0]:    .cp[3,0,16,0];
; 1203          1700  4                                   [1]:    .cp[3,0,24,0];
; 1204          1701  4                                   [2]:    .cp[3,0,32,0];
; 1205          1702  3                                   tes);
; 1206          1703  2           );
; 1207          1704  2
; 1208          1705  2
; 1209          1706  2 [14]:     ! Downward path 14 is from a secondary data bucket to the first record
; 1210          1707  2           ! in the bucket.  The data bucket can be empty.
; 1211          1708  2
; 1212          1709  3           if .hp[bkt$w_freespace] eqlu bkt$c_overhdsz then (
; 1213          1710  3                   signal (anlrms$_emptybkt);
; 1214          1711  3                   return false;
; 1215          1712  3           ) else (
; 1216          1713  3                   d[bsd$w_size] = .c[bsd$w_size];
; 1217          1714  3                   d[bsd$l_vbn] = .c[bsd$l_vbn];
; 1218          1715  3                   d[bsd$l_offset] = bkt$c_overhdsz;
; 1219          1716  2           );
; 1220          1717  2
; 1221          1718  2
; 1222          1719  2 [15]:     ! Downward path 15 is from a SIDR record to the first pointer in the
; 1223          1720  2           ! pointer array.  We have to get the key length to figure out where
```

```
;  1224          1721  2          ! the first pointer is.  The work longword in the BSD must be
;  1225          1722  2          ! initialized to the number of pointer bytes so people can tell
;  1226          1723  2          ! where they end.  This is for prolog 2.
;  1227          1724  2
;  1228          1725  3          (d[bsd$w_size] = .c[bsd$w_size];
;  1229          1726  3          d[bsd$l_vbn] = .c[bsd$l_vbn];
;  1230          1727  3
;  1231          1728  4          begin
;  1232          1729  4          bind
;  1233          1730  4                  k = current_stack[.key_level,0,0,0,0]: bsd,
;  1234          1731  4                  kp = .k[bsd$l_bufptr] + .k[bsd$l_offset]: block[,byte];
;  1235          1732  4
;  1236          1733  4          d[bsd$l_offset] =       .c[bsd$l_offset] +
;  1237          1734  4                                  1 +
;  1238          1735  4                                  1 +
;  1239          1736  4                                  (if .cp[irc$v_noptrsz] then 0 else 4) +
;  1240          1737  4                                  2 +
;  1241          1738  4                                  .kp[key$b_keysz];
;  1242          1739  4          d[bsd$l_work] = (if .cp[irc$v_noptrsz] then .cp[2,0,16,0] else .cp[6,0,16,0]) -
;  1243          1740  4                          .kp[key$b_keysz];
;  1244          1741  2          end;);
;  1245          1742  2
;  1246          1743  2
;  1247          1744  2 [16]:    ! Downward path 16 is from an index bucket to the first index
;  1248          1745  2          ! entry in the bucket.  We must set the work longword to zero to
;  1249          1746  2          ! indicate we are on the zeroth record.  This is for prolog 3.
;  1250          1747  2
;  1251          1748  3          (d[bsd$w_type] = (if .c[bsd$w_type] eqlu 20 then 24 else 25);
;  1252          1749  3          d[bsd$w_size] = .c[bsd$w_size];
;  1253          1750  3          d[bsd$l_vbn] = .c[bsd$l_vbn];
;  1254          1751  3          d[bsd$l_offset] = bkt$c_overhdsz;
;  1255          1752  2          d[bsd$l_work] = 0;);
;  1256          1753  2
;  1257          1754  2
;  1258          1755  2 [17]:    ! Downward path 17 is from a primary or secondary index record to
;  1259          1756  2          ! the index or data bucket pointed to by it.  This is for prolog 3.
;  1260          1757  2
;  1261          1758  4          (if .hp[bkt$b_level] gequ 2 then (
;  1262          1759  4
;  1263          1760  4                  ! The next lower level is another index bucket.  Set the
;  1264          1761  4                  ! type according to whether it's primary or secondary.
;  1265          1762  4                  ! Set the size the same as the current index bucket.
;  1266          1763  4
;  1267          1764  4                  d[bsd$w_type] = (if .c[bsd$w_type] eqlu 24 then 20 else 21);
;  1268          1765  4                  d[bsd$w_size] = .c[bsd$w_size];
;  1269          1766  4          ) else (
;  1270          1767  4
;  1271          1768  4                  ! The next lower level is the data buckets.  Set the type
;  1272          1769  4                  ! according to whether it's a primary or secondary bucket.
;  1273          1770  4                  ! The size has to be found from the key descriptor.
;  1274          1771  4
;  1275          1772  4                  d[bsd$w_type] = (if .c[bsd$w_type] eqlu 24 then 22 else 23);
;  1276          1773  4
;  1277          1774  5                  begin
;  1278          1775  5                  bind
;  1279          1776  5                          k = current_stack[.key_level,0,0,0,0]: bsd,
;  1280          1777  5                          kp = .k[bsd$l_bufptr] + .k[bsd$l_offset]: block[,byte];
```

```
;  1281          1778  5
;  1282          1779  5                   d[bsd$w_size] = .kp[key$b_datbktsz];
;  1283          1780  4                   end;
;  1284          1781  3           );
;  1285          1782  3
;  1286          1783  3           ! Now we set up the VBN of the downward structure by looking in the
;  1287          1784  3           ! VBN list and extracting the appropriate VBN.  The work longword
;  1288          1785  3           ! in the BSD tells us which key we are on.
;  1289          1786  3
;  1290          1787  3           sp = (.c[bsd$l_endptr]-4) - (.c[bsd$l_work]+1) * (.hp[bkt$v_ptr_sz]+2);
;  1291          1788  4           d[bsd$l_vbn] =  (case .hp[bkt$v_ptr_sz] from 0 to 2 of set
;  1292          1789  4                           [0]:    .sp[0,0,16,0];
;  1293          1790  4                           [1]:    .sp[0,0,24,0];
;  1294          1791  4                           [2]:    .sp[0,0,32,0];
;  1295          1792  3                           tes);
;  1296          1793  2           d[bsd$l_offset] = 0;);
;  1297          1794  2
;  1298          1795  2
;  1299          1796  2 [18]:     ! Downward path 18 is from a primary data record to the actual
;  1300          1797  2           ! data bytes.  They may not exist.  This is for prolog 3.
;  1301          1798  2
;  1302          1799  3           if .cp[irc$v_deleted] or .cp[irc$v_ru_delete] or .cp[irc$v_rrv] then (
;  1303          1800  3                   signal (anlrms$_nodata);
;  1304          1801  3                   return false;
;  1305          1802  3           ) else (
;  1306          1803  3
;  1307          1804  3                   ! The BSD for the data bytes is identical to that for the
;  1308          1805  3                   ! complete record, because we need all the record information
;  1309          1806  3                   ! to display the bytes.
;  1310          1807  3
;  1311          1808  3                   d[bsd$w_size] = .c[bsd$w_size];
;  1312          1809  3                   d[bsd$l_vbn] = .c[bsd$l_vbn];
;  1313          1810  3                   d[bsd$l_offset] = .c[bsd$l_offset];
;  1314          1811  2           );
;  1315          1812  2
;  1316          1813  2
;  1317          1814  2 [19]:     ! Downward path 19 is from a primary data record to the data bucket
;  1318          1815  2           ! pointed at by the RRV.  The pointer may not exist.  This is for
;  1319          1816  2           ! prolog 3.
;  1320          1817  2
;  1321          1818  3           if .cp[irc$v_noptrsz] then (
;  1322          1819  3                   signal (anlrms$_norrv);
;  1323          1820  3                   return false;
;  1324          1821  3           ) else (
;  1325          1822  3                   d[bsd$w_size] = .c[bsd$w_size];
;  1326          1823  4                   d[bsd$l_vbn] =  (case .cp[irc$v_ptrsz] from 0 to 2 of set
;  1327          1824  4                                   [0]:    .cp[5,0,16,0];
;  1328          1825  4                                   [1]:    .cp[5,0,24,0];
;  1329          1826  4                                   [2]:    .cp[5,0,32,0];
;  1330          1827  3                                   tes);
;  1331          1828  2           );
;  1332          1829  2
;  1333          1830  2
;  1334          1831  2 [20]:     ! AVAILABLE FOR FUTURE USE.
;  1335          1832  2
;  1336          1833  2           ;
;  1337          1834  2
```

```
;  1338          1835  2
;  1339          1836  2 [21]:   ! Downward path 21 is from a prolog 3 SIDR record to the first
;  1340          1837  2         ! pointer in the pointer array.  We have to determine the key
;  1341          1838  2         ! length in order to figure out where the first pointer starts.
;  1342          1839  2         ! The work longword in the BSD must be initialized to the
;  1343          1840  2         ! number of pointer bytes so the end of the SIDR record can be
;  1344          1841  2         ! found.
;  1345          1842  2
;  1346          1843  3         (d[bsd$w_size] = .c[bsd$w_size];
;  1347          1844  3         d[bsd$l_vbn] = .c[bsd$l_vbn];
;  1348          1845  3
;  1349          1846  4         begin
;  1350          1847  4         bind
;  1351          1848  4                 k = current_stack[.key_level,0,0,0,0]: bsd,
;  1352          1849  4                 kp = .k[bsd$l_bufptr] + .k[bsd$l_offset]: block[,byte];
;  1353          1850  4
;  1354          1851  4         local
;  1355          1852  4                 key_length: long;
;  1356          1853  4
;  1357          1854  5         key_length =    (if .kp[key$v_key_compr] then
;  1358          1855  5                             .cp[2,0,8,0] + irc$c_keycmpovh
;  1359          1856  5                         else
;  1360          1857  4                             .kp[key$b_keysz]);
;  1361          1858  4         d[bsd$l_offset] =       .c[bsd$l_offset] +
;  1362          1859  4                                 2 +
;  1363          1860  4                                 .key_length;
;  1364          1861  4         d[bsd$l_work] =         .cp[0,0,16,0] -
;  1365          1862  4                                 .key_length;
;  1366          1863  2         end;);
;  1367          1864  2
;  1368          1865  2
;  1369          1866  2 [22]:   ! Downward path 22 is from an area descriptor to the first reclaimed
;  1370          1867  2         ! bucket on the available list (if any).  This works for both prologs.
;  1371          1868  2
;  1372          1869  3         if .cp[area$l_avail] eglu 0 then (
;  1373          1870  3                 signal (anlrms$_noreclaimed);
;  1374          1871  3                 return false;
;  1375          1872  3         ) else (
;  1376          1873  3                 d[bsd$w_size] = .cp[area$b_arbktsz];
;  1377          1874  3                 d[bsd$l_vbn] = .cp[area$l_avail];
;  1378          1875  2         );
;  1379          1876  2 tes;
;  1380          1877  2
;  1381          1878  2 ! Now we can read in the bucket which was set up.
;  1382          1879  2
;  1383          1880  2 anl$bucket(d,.c[bsd$l_vbn]);
;  1384          1881  2
;  1385          1882  2 return true;
;  1386          1883  2
;  1387          1884  1 end;


                                  0FFC 00000        .ENTRY  ANL$INTERACTIVE_DOWN, Save R2,R3,R4,R5,R6,- ; 1428
                                                            R7,R8,R9,R10,R11                            ;
```

```
                      5E             04  C2 00002          SUBL2   #4, SP                          ;
                      55        04   AC  D0 00005          MOVL    PATH, R5                        ; 1431
                      59        08   AC  D0 00009          MOVL    CURRENT_BSD, R9                 ; 1432
                      58        0C   AC  D0 0000D          MOVL    DOWN_BSD, R8                    ; 1433
                             0000G   CF  9F 00011          PUSHAB  ANL$UNWIND_HANDLER              ; 1445
            00000000G 00             01  FB 00015          CALLS   #1, LIB$ESTABLISH               ;
                      5A             69  3C 0001C          MOVZWL  (R9), R10                       ; 1450
                           0000'CF4A     DF 0001F          PUSHAL  STRUCTURE_TABLE+1[R10]          ;
                                     9E  95 00024          TSTB    @(SP)+                          ;
                                     09  12 00026          BNEQ    1$                              ;
                      00000000G      8F  DD 00028          PUSHL   #ANLRMS$_NODOWN                 ; 1451
                                  04A8   31 0002E          BRW     112$                            ;
                                     65  B5 00031 1$:      TSTW    (R5)                            ; 1459
                                     06  13 00033          BEQL    2$                              ;
                      3F        04   B5  91 00035          CMPB    @4(R5), #63                     ;
                                     0D  13 00039          BEQL    3$                              ;
                                     65  B5 0003B 2$:      TSTW    (R5)                            ; 1460
                                     50  12 0003D          BNEQ    6$                              ;
                           0000'CF4A     DF 0003F          PUSHAL  STRUCTURE_TABLE+2[R10]          ;
                                     9E  95 00044          TSTB    @(SP)+                          ;
                                     47  13 00046          BEQL    6$                              ;
                      00000000G      8F  DD 00048 3$:      PUSHL   #ANLRMS$_DOWNHELP               ; 1461
            00000000G 00             01  FB 0004E          CALLS   #1, LIB$SIGNAL                  ;
                      52             01  D0 00055          MOVL    #1, I                           ; 1462
                      50          624A   DE 00058 4$:      MOVAL   (I)[R10], R0                    ; 1463
                      54   0000'CF40     9A 0005C          MOVZBL  STRUCTURE_TABLE[R0], J          ;
                                     21  13 00062          BEQL    5$                              ;
             50       54             0A  C5 00064          MULL3   #10, J, R0                      ; 1464
                           0000'CF40     9F 00068          PUSHAB  PATH_TABLE+4[R0]                ;
                                     9E  DD 0006D          PUSHL   @(SP)+                          ;
                           0000'CF40     9F 0006F          PUSHAB  PATH_TABLE[R0]                  ;
                                     9E  DD 00074          PUSHL   @(SP)+                          ;
                                     02  DD 00076          PUSHL   #2                              ;
                      00000000G      8F  DD 00078          PUSHL   #ANLRMS$_DOWNPATH               ;
            00000000G 00             04  FB 0007E          CALLS   #4, LIB$SIGNAL                  ;
                                     52  D6 00085 5$:      INCL    I                               ; 1463
                      03             52  D1 00087          CMPL    I, #3                           ;
                                     CC  1B 0008A          BLEQU   4$                              ;
                                  046B   31 0008C          BRW     115$                            ; 1465
                                     65  B5 0008F 6$:      TSTW    (R5)                            ; 1472
                                     3F  13 00091          BEQL    10$                             ;
                      56             01  7D 00093          MOVQ    #1, I                           ; 1480
                      50          664A   DE 00096 7$:      MOVAL   (I)[R10], R0                    ; 1481
                      54   0000'CF40     9A 0009A          MOVZBL  STRUCTURE_TABLE[R0], J          ;
                                     27  13 000A0          BEQL    9$                              ;
             50       54             0A  C5 000A2          MULL3   #10, J, R0                      ; 1483
                           0000'CF40     9F 000A6          PUSHAB  PATH_TABLE[R0]                  ;
                      51             9E  D0 000AB          MOVL    @(SP)+, R1                      ;
                      50             61  9A 000AE          MOVZBL  (R1), R0                        ; 1484
                      50             65  B1 000B1          CMPW    (R5), R0                        ;
                                     03  1E 000B4          BGEQU   8$                              ;
                      50             65  3C 000B6          MOVZWL  (R5), R0                        ;
                      5B             50  D0 000B9 8$:      MOVL    R0, LENGTH                      ;
       01    A1        04            B5  5B  29 000BC      CMPC3   LENGTH, @4(R5), 1(R1)           ; 1485
                                     05  12 000C2          BNEQ    9$                              ;
                      57             54  D0 000C4          MOVL    J, PATH_INDEX                   ; 1486
                                     11  11 000C7          BRB     11$                             ; 1485
```

```
                              56  D6 000C9 9$:    INCL    I                                  ; 1481
            03                56  D1 000CB        CMPL    I, #3                              ;
                              C6  1B 000CE        BLEQU   7$                                 ;
                              08  11 000D0        BRB     11$                                ; 1472
                       0000'CF4A  DF 000D2 10$:   PUSHAL  STRUCTURE_TABLE+1[R10]             ; 1492
            57                9E  9A 000D7        MOVZBL  @(SP)+, PATH_INDEX                 ;
            5B        08      A9  D0 000DA 11$:   MOVL    8(R9), R11                         ; 1496
      56    5B        0C      A9  C1 000DE        ADDL3   12(R9), R11, CP                    ;
            6E        0C      A9  D0 000E3        MOVL    12(R9), HP                         ; 1497
18    00    6E                00  2C 000E7        MOVC5   #0, (SP), #0, #24, (R8)            ; 1503
                              68     000EC                                                   ;
            57                0A  C4 000ED        MULL2   #10, R7                            ; 1504
            68         0000'CF47  9B 000F0        MOVZBW  PATH_TABLE+9[R7], (R8)             ;
      16    00         0000'CF47  8F 000F6        CASEB   PATH_TABLE+8[R7], #0, #22          ; 1506
0079  0036  03EF            002E     000FD 12$:   .WORD   13$-12$,-                          ;
00C6  00B6  00AB            009F     00105                114$-12$,-                         ;
01BC  0151  013E            00FD     0010D                14$-12$,-                          ;
0245  01BC  01FF            01CD     00115                20$-12$,-                          ;
0360  0341  02B3            028F     0011D                24$-12$,-                          ;
      03D1  0393            03EF     00125                25$-12$,-                          ;
                                                          26$-12$,-                          ;
                                                          28$-12$,-                          ;
                                                          34$-12$,-                          ;
                                                          43$-12$,-                          ;
                                                          46$-12$,-                          ;
                                                          58$-12$,-                          ;
                                                          59$-12$,-                          ;
                                                          64$-12$,-                          ;
                                                          58$-12$,-                          ;
                                                          75$-12$,-                          ;
                                                          80$-12$,-                          ;
                                                          84$-12$,-                          ;
                                                          96$-12$,-                          ;
                                                          99$-12$,-                          ;
                                                          114$-12$,-                         ;
                                                          107$-12$,-                         ;
                                                          111$-12$                           ;
                 00000000G  8F  DD 0012B 13$:   PUSHL   #ANLRMS$_BADPATH                   ; 1511
                              5B  11 00131        BRB     22$                                ;
            50         0000G  CF  D0 00133 14$:   MOVL    ANL$GL_FAT, R0                     ; 1527
       F0   8F                60  93 00138        BITB    (R0), #240                         ; 1529
                              10  12 0013C        BNEQ    16$                                ;
            01        08      A0  D1 0013E        CMPL    8(R0), #1                          ;
                              05  12 00142        BNEQ    15$                                ;
                      0C      A0  B5 00144        TSTW    12(R0)                             ; 1530
                              3F  13 00147        BEQL    21$                                ;
            50                03  D0 00149 15$:   MOVL    #3, R0                             ; 1529
                              1B  11 0014C        BRB     19$                                ;
01    60    04                04  ED 0014E 16$:   CMPZV   #4, #4, (R0), #1                   ; 1536
                              05  12 00153        BNEQ    17$                                ;
            50                04  D0 00155        MOVL    #4, R0                             ;
                              0F  11 00158        BRB     19$                                ;
02    60    04                04  ED 0015A 17$:   CMPZV   #4, #4, (R0), #2                   ; 1538
                              05  13 0015F        BEQL    18$                                ;
            50                01  CE 00161        MNEGL   #1, R0                             ;
                              03  11 00164        BRB     19$                                ;
            50                07  D0 00166 18$:   MOVL    #7, R0                             ;
```

```
                                    68              50  B0 00169 19$:    MOVW    R0, (R8)                                ; 1527
                                02  A8              01  B0 0016C         MOVW    #1, 2(R8)                               ; 1540
                                04  A8              01  D0 00170         MOVL    #1, 4(R8)                               ; 1541
                                                    4B  11 00174         BRB     27$                                     ; 1506
                                    52      0000G   CF  D0 00176 20$:    MOVL    ANL$GL_FAT, R2                          ; 1546
                            50      04      A2      01  C3 0017B         SUBL3   #1, 4(R2), R0                           ;
                    50      0E      A2              08  00  ED 00180     CMPZV   #0, #8, 14(R2), R0                      ;
                                                        08  1B 00186     BLEQU   23$                                     ;
                                        00000000G   8F  DD 00188 21$:    PUSHL   #ANLRMS$_NORECS                         ; 1547
                                                    75  11 0018E 22$:    BRB     36$                                     ;
                                02  A8          0E  A2  9B 00190 23$:    MOVZBW  14(R2), 2(R8)                           ; 1550
                                04  A8          68  A6  3C 00195         MOVZWL  104(CP), 4(R8)                          ; 1551
                                                    5C  11 0019A         BRB     33$                                     ; 1546
                                02  A8          02  A9  B0 0019C 24$:    MOVW    2(R9), 2(R8)                            ; 1558
                                04  A8          04  A9  D0 001A1         MOVL    4(R9), 4(R8)                            ; 1559
                                                    50  11 001A6         BRB     33$                                     ; 1506
                                02  A8              01  B0 001A8 25$:    MOVW    #1, 2(R8)                               ; 1565
                                04  A8          66  A6  9A 001AC         MOVZBL  102(CP), 4(R8)                          ; 1566
                                                    45  11 001B1         BRB     33$                                     ; 1506
                                02  A8              01  B0 001B3 26$:    MOVW    #1, 2(R8)                               ; 1574
                                04  A8              01  D0 001B7         MOVL    #1, 4(R8)                               ; 1575
                            0000'   CF          10  AC  D0 001BB         MOVL    NEW_LEVEL, KEY_LEVEL                    ; 1576
                                                    75  11 001C1 27$:    BRB     42$                                     ;
                    37      10      A6              04  E0 001C3 28$:    BBS     #4, 16(CP), 35$                         ; 1583
                                    03      0000G   CF  B1 001C8         CMPW    ANL$GW_PROLOG, #3                       ; 1587
                                                    0F  12 001CD         BNEQ    30$                                     ;
                                            15      A6  95 001CF         TSTB    21(CP)                                  ; 1588
                                                    05  12 001D2         BNEQ    29$                                     ;
                                    50              14  D0 001D4         MOVL    #20, R0                                 ;
                                                    12  11 001D7         BRB     32$                                     ;
                                    50              15  D0 001D9 29$:    MOVL    #21, R0                                 ;
                                                    0D  11 001DC         BRB     32$                                     ;
                                            15      A6  95 001DE 30$:    TSTB    21(CP)                                  ; 1590
                                                    05  12 001E1         BNEQ    31$                                     ;
                                    50              0A  D0 001E3         MOVL    #10, R0                                 ;
                                                    03  11 001E6         BRB     32$                                     ;
                                    50              0B  D0 001E8 31$:    MOVL    #11, R0                                 ;
                                    68              50  B0 001EB 32$:    MOVW    R0, (R8)                                ; 1587
                                02  A8          0A  A6  9B 001EE         MOVZBW  10(CP), 2(R8)                           ; 1591
                                04  A8          0C  A6  D0 001F3         MOVL    12(CP), 4(R8)                           ; 1592
                                                    3E  11 001F8 33$:    BRB     42$                                     ; 1583
                09      10      A6              04  E1 001FA 34$:    BBC     #4, 16(CP), 37$                         ; 1600
                                        00000000G   8F  DD 001FF 35$:    PUSHL   #ANLRMS$_UNINITINDEX                    ; 1601
                                                02D1    31  00205 36$:    BRW     112$                                    ;
                                    03      0000G   CF  B1 00208 37$:    CMPW    ANL$GW_PROLOG, #3                       ; 1604
                                                    0F  12 0020D         BNEQ    39$                                     ;
                                            15      A6  95 0020F         TSTB    21(CP)                                  ; 1605
                                                    05  12 00212         BNEQ    38$                                     ;
                                    50              16  D0 00214         MOVL    #22, R0                                 ;
                                                    12  11 00217         BRB     41$                                     ;
                                    50              17  D0 00219 38$:    MOVL    #23, R0                                 ;
                                                    0D  11 0021C         BRB     41$                                     ;
                                            15      A6  95 0021E 39$:    TSTB    21(CP)                                  ; 1607
                                                    05  12 00221         BNEQ    40$                                     ;
                                    50              0C  D0 00223         MOVL    #12, R0                                 ;
                                                    03  11 00226         BRB     41$                                     ;
                                    50              0D  D0 00228 40$:    MOVL    #13, R0                                 ;
```

```
                                    68        50  B0 0022B 41$:    MOVW    R0, (R8)                        ; 1604
                                02  A8     0B  A6  9B 0022E         MOVZBW  11(CP), 2(R8)                   ; 1608
                                04  A8     54  A6  D0 00233         MOVL    84(CP), 4(R8)                   ; 1609
                                          02B1  31 00238 42$:       BRW     114$                            ; 1600
                                    0A        5A  B1 0023B 43$:     CMPW    R10, #10                        ; 1616
                                              05  12 0023E         BNEQ    44$                             ;
                                    50        0E  D0 00240         MOVL    #14, R0                         ;
                                              03  11 00243         BRB     45$                             ;
                                    50        0F  D0 00245 44$:     MOVL    #15, R0                         ;
                                    68        50  B0 00248 45$:     MOVW    R0, (R8)                        ;
                                          00E4  31 0024B         BRW     73$                             ; 1617
                              50    6E        0C  C1 0024E 46$:     ADDL3   #12, HP, R0                     ; 1625
                                    02        60  91 00252         CMPB    (R0), #2                        ;
                                              17  1F 00255         BLSSU   49$                             ;
                                    0E        5A  B1 00257         CMPW    R10, #14                        ; 1631
                                              05  12 0025A         BNEQ    47$                             ;
                                    50        0A  D0 0025C         MOVL    #10, R0                         ;
                                              03  11 0025F         BRB     48$                             ;
                                    50        0B  D0 00261 47$:     MOVL    #11, R0                         ;
                                    68        50  B0 00264 48$:     MOVW    R0, (R8)                        ;
                                02  A8     02  A9  B0 00267         MOVW    2(R9), 2(R8)                    ; 1632
                                              27  11 0026C         BRB     52$                             ; 1625
                                    0E        5A  B1 0026E 49$:     CMPW    R10, #14                        ; 1639
                                              05  12 00271         BNEQ    50$                             ;
                                    50        0C  D0 00273         MOVL    #12, R0                         ;
                                              03  11 00276         BRB     51$                             ;
                                    50        0D  D0 00278 50$:     MOVL    #13, R0                         ;
                                    68        50  B0 0027B 51$:     MOVW    R0, (R8)                        ;
                          50    0000'   CF        18  C5 0027E         MULL3   #24, KEY_LEVEL, R0              ; 1642
                                    50  0000'CF40  9E 00284         MOVAB   CURRENT_STACK[R0], R0           ;
                          50    0C  A0    08  A0  C1 0028A         ADDL3   8(R0), T2(R0), R0               ; 1643
                                02  A8    0B  A0  9B 00290         MOVZBW  11(R0), 2(R8)                   ; 1645
51                        66      02        00  EF 00295 52$:     EXTZV   #0, #2, (CP), R1                ; 1652
                          02        00        51  CF 0029A         CASEL   R1, #0, #2                      ;
                      0014      000C      0006      0029E 53$:     .WORD   54$-53$,-                       ;
                                                                          55$-53$,-                       ;
                                                                          56$-53$                         ;
                                    50        01  A6  3C 002A4 54$:     MOVZWL  1(CP), R0                       ; 1653
                                              0C  11 002A8         BRB     57$                             ;
50                  01  A6        18        00  EF 002AA 55$:     EXTZV   #0, #24, 1(CP), R0              ; 1654
                                              04  11 002B0         BRB     57$                             ;
                                    50        01  A6  D0 002B2 56$:     MOVL    1(CP), R0                       ; 1655
                                          017C  31 002B6 57$:     BRW     95$                             ; 1652
                                    50        6E  04  C1 002B9 58$:     ADDL3   #4, HP, R0                      ; 1663
                                    0E        60  B1 002BD         CMPW    (R0), #14                       ;
                                              70  12 002C0         BNEQ    73$                             ;
                                  00000000G   8F  DD 002C2         PUSHL   #ANLRMS$_EMPTYBKT               ; 1664
                                              3C  11 002C8         BRB     66$                             ;
                                    04        66  02  E0 002CA 59$:     BBS     #2, (CP), 60$                   ; 1676
                                    08        66  03  E1 002CE         BBC     #3, (CP), 61$                   ;
                                  00000000G   8F  DD 002D2 60$:     PUSHL   #ANLRMS$_NODATA                 ; 1677
                                              2C  11 002D8         BRB     66$                             ;
                                02  A8        02  A9  B0 002DA 61$:     MOVW    2(R9), 2(R8)                    ; 1680
                                04  A8        04  A9  D0 002DF         MOVL    4(R9), 4(R8)                    ; 1681
                                    04        66  04  E1 002E4         BBC     #4, (CP), 62$                   ; 1685
                                              50  D4 002E8         CLRL    R0                              ;
                                              08  11 002EA         BRB     63$                             ;
```

```
                50              66              02              00  EF 002EC 62$:    EXTZV   #0, #2, (CP), R0                        ;
                                                50              03  C0 002F1         ADDL2   #3, R0                                  ;
                                        08      A8      02 A04B  9E 002F4 63$:    MOVAB   2(R0)[R11], 8(R8)                       ; 1684
                                                                44  11 002FA         BRB     74$                                     ; 1676
                                09              66              04  E1 002FC 64$:    BBC     #4, (CP), 67$                           ; 1693
                                                        00000000G   8F  DD 00300 65$:    PUSHL   #ANLRMS$_NORRV                          ; 1694
                                                                01D0  31 00306 66$:    BRW     112$                                    ;
                                        02      A8      02      A9  B0 00309 67$:    MOVW    2(R9), 2(R8)                            ; 1697
                51              66              02              00  EF 0030E         EXTZV   #0, #2, (CP), R1                        ; 1698
                                02              00              51  CF 00313         CASEL   R1, #0, #2                              ;
                            0014            000C            0006      00317 68$:    .WORD   69$-68$,-                               ;
                                                                                              70$-68$,-                               ;
                                                                                              71$-68$                                 ;
                                                50      03      A6  3C 0031D 69$:    MOVZWL  3(CP), R0                               ; 1699
                                                                0C  11 00321         BRB     72$                                     ;
                50      03      A6              18              00  EF 00323 70$:    EXTZV   #0, #24, 3(CP), R0                      ; 1700
                                                                04  11 00329         BRB     72$                                     ;
                                                50      03      A6  D0 0032B 71$:    MOVL    3(CP), R0                               ; 1701
                                                        0158    31  0032F 72$:    BRW     105$                                    ; 1698
                                        02      A8      02      A9  B0 00332 73$:    MOVW    2(R9), 2(R8)                            ; 1713
                                        04      A8      04      A9  D0 00337         MOVL    4(R9), 4(R8)                            ; 1714
                                        08      A8              0E  D0 0033C         MOVL    #14, 8(R8)                              ; 1715
                                                                6B  11 00340 74$:    BRB     83$                                     ; 1709
                                        02      A8      02      A9  B0 00342 75$:    MOVW    2(R9), 2(R8)                            ; 1725
                                        04      A8      04      A9  D0 00347         MOVL    4(R9), 4(R8)                            ; 1726
                        50      0000'   CF              18      C5  0034C         MULL3   #24, KEY_LEVEL, R0                      ; 1730
                                                50  0000'CF40   9E  00352         MOVAB   CURRENT_STACK[R0], R0                   ;
                        50      0C      A0      08      A0      C1  00358         ADDL3   8(R0), T2(R0), R0                       ; 1731
                        04              66              04      E1  0035E         BBC     #4, (CP), 76$                           ; 1736
                                                                51  D4  00362         CLRL    R1                                      ;
                                                                03  11  00364         BRB     77$                                     ;
                                                51              04  D0  00366 76$:    MOVL    #4, R1                                  ;
                        52              5B              51      C1  00369 77$:    ADDL3   R1, R11, R2                             ; 1735
                                                51      14      A0  9A  0036D         MOVZBL  20(R0), R1                              ; 1738
                                        08      A8      04 A142  9E  00371         MOVAB   4(R1)[R2], 8(R8)                        ; 1737
                        06              66              04      E1  00377         BBC     #4, (CP), 78$                           ; 1739
                                                50      02      A6  3C  0037B         MOVZWL  2(CP), R0                               ;
                                                                04  11  0037F         BRB     79$                                     ;
                                                50      06      A6  3C  00381 78$:    MOVZWL  6(CP), R0                               ;
                    14  A8                      50              51  C3  00385 79$:    SUBL3   R1, R0, 20(R8)                          ; 1740
                                                                21  11  0038A         BRB     83$                                     ; 1506
                                                14              5A  B1  0038C 80$:    CMPW    R10, #20                                ; 1748
                                                                05  12  0038F         BNEQ    81$                                     ;
                                                50              18  D0  00391         MOVL    #24, R0                                 ;
                                                                03  11  00394         BRB     82$                                     ;
                                                50              19  D0  00396 81$:    MOVL    #25, R0                                 ;
                                                68              50  B0  00399 82$:    MOVW    R0, (R8)                                ;
                                        02      A8      02      A9  B0  0039C         MOVW    2(R9), 2(R8)                            ; 1749
                                        04      A8      04      A9  D0  003A1         MOVL    4(R9), 4(R8)                            ; 1750
                                        08      A8              0E  D0  003A6         MOVL    #14, 8(R8)                              ; 1751
                                                        14      A8  D4  003AA         CLRL    20(R8)                                  ; 1752
                                                            013C    31  003AD 83$:    BRW     114$                                    ; 1506
                        50              6E              0C      C1  003B0 84$:    ADDL3   #12, HP, R0                             ; 1758
                                                02              60  91  003B4         CMPB    (R0), #2                                ;
                                                                17  1F  003B7         BLSSU   87$                                     ;
                                                18              5A  B1  003B9         CMPW    R10, #24                                ; 1764
                                                                05  12  003BC         BNEQ    85$                                     ;
```

```
                        50            14  D0 003BE         MOVL    #20, R0                         ;
                                      03  11 003C1         BRB     86$                             ;
                        50            15  D0 003C3 85$:    MOVL    #21, R0                         ;
                        68            50  B0 003C6 86$:    MOVW    R0, (R8)                        ;
                 02     A8      02    A9  B0 003C9         MOVW    2(R9), 2(R8)                    ; 1765
                                      27  11 003CE         BRB     90$                             ; 1758
                        18            5A  B1 003D0 87$:    CMPW    R10, #24                        ; 1772
                                      05  12 003D3         BNEQ    88$                             ;
                        50            16  D0 003D5         MOVL    #22, R0                         ;
                                      03  11 003D8         BRB     89$                             ;
                        50            17  D0 003DA 88$:    MOVL    #23, R0                         ;
                        68            50  B0 003DD 89$:    MOVW    R0, (R8)                        ;
           50    0000'  CF            18  C5 003E0         MULL3   #24, KEY_LEVEL, R0              ; 1776
                        50     0000'CF40  9E 003E6         MOVAB   CURRENT_STACK[R0], R0           ;
           50    0C     A0      08    A0  C1 003EC         ADDL3   8(R0), 12(R0), R0               ; 1777
           02           A8      0B    A0  9B 003F2         MOVZBW  11(R0), 2(R8)                   ; 1779
           51    14     A9            01  C1 003F7 90$:    ADDL3   #1, 20(R9), R1                  ; 1787
           52           6E            0D  C1 003FC         ADDL3   #13, HP, R2                     ;
     50    62           02            03  EF 00400         EXTZV   #3, #2, (R2), R0                ;
                        50            02  C0 00405         ADDL2   #2, R0                          ;
                        50            51  C4 00408         MULL2   R1, R0                          ;
           50    10     A9            50  C3 0040B         SUBL3   R0, 16(R9), R0                  ;
                        50            04  C2 00410         SUBL2   #4, SP                          ;
           52           6E            0D  C1 00413         ADDL3   #13, HP, R2                     ; 1788
     51    62           02            03  EF 00417         EXTZV   #3, #2, (R2), R1                ;
           02           00            51  CF 0041C         CASEL   R1, #0, #2                      ;
         0012         000B          0006     00420 91$:    .WORD   92$-91$,-                       ;
                                                                   93$-91$,-                       ;
                                                                   94$-91$                         ;
                        50            60  3C 00426 92$:    MOVZWL  (SP), R0                        ; 1789
                                      0A  11 00429         BRB     95$                             ;
     50    60           18            00  EF 0042B 93$:    EXTZV   #0, #24, (SP), R0               ; 1790
                                      03  11 00430         BRB     95$                             ;
                        50            60  D0 00432 94$:    MOVL    (SP), R0                        ; 1791
                 04     A8            50  D0 00435 95$:    MOVL    R0, 4(R8)                       ; 1788
                                08    A8  D4 00439         CLRL    8(R8)                           ; 1793
                                      50  11 0043C         BRB     106$                            ; 1506
           03           66            02  E1 0043E 96$:    BBC     #2, (CP), 98$                   ; 1799
                                    FE8D  31 00442 97$:    BRW     60$                             ;
           F9           66            05  E0 00445 98$:    BBS     #5, (CP), 97$                   ;
           F5           66            03  E0 00449         BBS     #3, (CP), 97$                   ;
                 02     A8      02    A9  B0 0044D         MOVW    2(R9), 2(R8)                    ; 1808
                 04     A8      04    A9  D0 00452         MOVL    4(R9), 4(R8)                    ; 1809
                 08     A8            5B  D0 00457         MOVL    R11, 8(R8)                      ; 1810
                                      6F  11 0045B         BRB     110$                            ; 1799
           03           66            04  E1 0045D 99$:    BBC     #4, (CP), 100$                  ; 1818
                                    FE9C  31 00461         BRW     65$                             ;
                 02     A8      02    A9  B0 00464 100$:   MOVW    2(R9), 2(R8)                    ; 1822
     51    66           02            00  EF 00469         EXTZV   #0, #2, (CP), R1                ; 1823
           02           00            51  CF 0046E         CASEL   R1, #0, #2                      ;
         0014         000C          0006     00472 101$:   .WORD   102$-101$,-                     ;
                                                                   103$-101$,-                     ;
                                                                   104$-101$                       ;
                        50      05    A6  3C 00478 102$:   MOVZWL  5(CP), R0                       ; 1824
                                      0C  11 0047C         BRB     105$                            ;
     50     05   A6     18            00  EF 0047E 103$:   EXTZV   #0, #24, 5(CP), R0              ; 1825
                                      04  11 00484         BRB     105$                            ;
```

```
                                50        05    A6  D0 00486 104$:   MOVL    5(CP), R0                          ; 1826
                            04  A8              50  D0 0048A 105$:   MOVL    R0, 4(R8)                          ; 1823
                                              5C  11 0048E 106$:   BRB     114$                               ; 1818
                            02  A8        02    A9  B0 00490 107$:   MOVW    2(R9), 2(R8)                       ; 1843
                            04  A8        04    A9  D0 00495         MOVL    4(R9), 4(R8)                       ; 1844
                    50      0000'  CF           18  C5 0049A         MULL3   #24, KEY_LEVEL, R0                 ; 1848
                                   50   0000'CF40  9E 004A0         MOVAB   CURRENT_STACK[R0], R0              ;
                    50      0C     A0    08    A0  C1 004A6         ADDL3   8(R0), T2(R0), R0                  ; 1849
                    09      10     A0          06  E1 004AC         BBC     #6, 16(R0), 108$                   ; 1854
                                   50    02    A6  9A 004B1         MOVZBL  2(CP), KEY_LENGTH                  ; 1855
                                   50          02  C0 004B5         ADDL2   #2, KEY_LENGTH                     ;
                                               04  11 004B8         BRB     109$                               ;
                                   50    14    A0  9A 004BA 108$:   MOVZBL  20(R0), KEY_LENGTH                 ; 1857
                            08     A8    02 A04B  9E 004BE 109$:   MOVAB   2(KEY_LENGTH)[R11], 8(R8)          ; 1859
                                   51          66  3C 004C4         MOVZWL  (CP), R1                           ; 1862
                14  A8             51          50  C3 004C7         SUBL3   KEY_LENGTH, R1, 20(R8)             ;
                                               1E  11 004CC 110$:   BRB     114$                               ; 1506
                                         08    A6  D5 004CE 111$:   TSTL    8(CP)                              ; 1869
                                               0F  12 004D1         BNEQ    113$                               ;
                                00000000G      8F  DD 004D3         PUSHL   #ANLRMS$_NORECLAIMED               ; 1870
                        00000000G .    00          01  FB 004D9 112$:   CALLS   #1, LIB$SIGNAL                ;
                                               18  11 004E0         BRB     115$                               ; 1871
                            02  A8        03    A6  9B 004E2 113$:   MOVZBW  3(CP), 2(R8)                       ; 1873
                            04  A8        08    A6  D0 004E7         MOVL    8(CP), 4(R8)                       ; 1874
                                         04    A9  DD 004EC 114$:   PUSHL   4(R9)                              ; 1880
                                               58  DD 004EF         PUSHL   R8                                 ;
                        0000G  CF                  02  FB 004F1         CALLS   #2, ANL$BUCKET                 ;
                                50                 01  D0 004F6         MOVL    #1, R0                         ; 1882
                                                   04  004F9         RET                                        ;
                                                   50  D4 004FA 115$:   CLRL    R0                             ; 1884
                                                   04  004FC         RET                                        ;
                                                                                                              ;

; Routine Size:  1277 bytes,    Routine Base:  $CODE$ + 0581
```

```
;  1389     1885  1 %sbttl 'ANL$INTERACTIVE_DUMP - Dump a Block in Hex'
;  1390     1886  1 !++
;  1391     1887  1 ! Functional Description:
;  1392     1888  1 !       This routine handles the interactive DUMP command, which allows the
;  1393     1889  1 !       user to dump a single virtual block in hex.
;  1394     1890  1 !
;  1395     1891  1 ! Formal Parameters:
;  1396     1892  1 !       argument        A descriptor of the argument supplied by the user.
;  1397     1893  1 !                       It should be the VBN of the block to be dumped.
;  1398     1894  1 !
;  1399     1895  1 ! Implicit Inputs:
;  1400     1896  1 !       global data
;  1401     1897  1 !
;  1402     1898  1 ! Implicit Outputs:
;  1403     1899  1 !       global data
;  1404     1900  1 !
;  1405     1901  1 ! Returned Value:
;  1406     1902  1 !       none
;  1407     1903  1 !
;  1408     1904  1 ! Side Effects:
;  1409     1905  1 !
;  1410     1906  1 !--
;  1411     1907  1
;  1412     1908  1
;  1413     1909  2 global routine anl$interactive_dump(argument): novalue = begin
;  1414     1910  2
;  1415     1911  2 bind
;  1416     1912  2         argument_dsc = .argument: descriptor;
;  1417     1913  2
;  1418     1914  2 local
;  1419     1915  2         status: long,
;  1420     1916  2         vbn: long,
;  1421     1917  2         b: bsd;
;  1422     1918  2
;  1423     1919  2
;  1424     1920  2 ! Begin by converting the user's argument to a longword.  If it won't convert,
;  1425     1921  2 ! tell the user and quit.
;  1426     1922  2
;  1427     1923  2 status = anl$internalize_number(argument_dsc,vbn);
;  1428     1924  3 if not .status then (
;  1429     1925  3         signal (anlrms$_badvbn);
;  1430     1926  3         return;
;  1431     1927  2 );
;  1432     1928  2
;  1433     1929  2 ! Now let's constrain the VBN to within the limits of the file.  Because of
;  1434     1930  2 ! a stupidity in RMS block I/O, we have to contrain sequential files to
;  1435     1931  2 ! the end-of-file block, while the others only to the end of the allocation.
;  1436     1932  2
;  1437     1933  2 vbn = minu( maxu(1,.vbn),
;  1438     1934  3                 (if .anl$gl_fat[fat$v_fileorg] eqlu fat$c_sequential then
;  1439     1935  3                         .anl$gl_fat[fat$l_efblk]
;  1440     1936  3                 else
;  1441     1937  2                         .anl$gl_fat[fat$l_hiblk]));
;  1442     1938  2
;  1443     1939  2 ! Build a BSD describing the desired block and read it in.
;  1444     1940  2
;  1445     1941  2 init_bsd(b);
```

```
;  1446          1942  2 b[bsd$w_size] = 1;
;  1447          1943  2 b[bsd$l_vbn] = .vbn;
;  1448          1944  2 anl$bucket(b,0);
;  1449          1945  2
;  1450          1946  2 ! We can format the block in hex, and then free it up.  We'll include a nice
;  1451          1947  2 ! heading also.
;  1452          1948  2
;  1453          1949  2 anl$format_line(3,0,anlrms$_dumpheading,.vbn);
;  1454          1950  2
;  1455          1951  3 begin
;  1456          1952  3 local
;  1457          1953  3         block_dsc: descriptor;
;  1458          1954  3
;  1459          1955  3 build_descriptor(block_dsc,512,.b[bsd$l_bufptr]);
;  1460          1956  3 anl$format_hex(1,block_dsc);
;  1461          1957  2 end;
;  1462          1958  2
;  1463          1959  2 anl$bucket(b,-1);
;  1464          1960  2
;  1465          1961  2 return;
;  1466          1962  2
;  1467          1963  1 end;


                                003C 00000          .ENTRY  ANL$INTERACTIVE_DUMP, Save R2,R3,R4,R5      ; 1909
                             5E 24   C2 00002          SUBL2   #36, SP                                     ;
                             5E   DD 00005          PUSHL   SP                                          ; 1923
                          04 AC   DD 00007          PUSHL   ARGUMENT                                    ;
                   0000G  CF      02 FB 0000A          CALLS   #2, ANL$INTERNALIZE_NUMBER                  ;
                             0E   50 E8 0000F          BLBS    STATUS, 1$                                  ; 1924
                00000000G 8F      DD 00012          PUSHL   #ANLRMS$_BADVBN                             ; 1925
             00000000G    00      01 FB 00018          CALLS   #1, LIB$SIGNAL                              ;
                                04 0001F          RET                                                 ; 1924
                             51   6E D0 00020 1$:       MOVL    VBN, R1                                     ; 1933
                             03   12 00023          BNEQ    2$                                          ;
                             51   01 D0 00025          MOVL    #1, R1                                      ;
                             50 0000G  CF D0 00028 2$:       MOVL    ANL$GL_FAT, R0                      ; 1934
                          F0 8F      60 93 0002D          BITB    (R0), #240                          ;
                             06   12 00031          BNEQ    3$                                          ;
                             50 08 A0 D0 00033          MOVL    8(R0), R0                               ; 1935
                             04   11 00037          BRB     4$                                          ;
                             50 04 A0 D0 00039 3$:       MOVL    4(R0), R0                               ; 1937
                             50   51 D1 0003D 4$:       CMPL    R1, R0                                  ; 1934
                             03   1B 00040          BLEQU   5$                                          ;
                             51   50 D0 00042          MOVL    R0, R1                                      ;
                             6E   51 D0 00045 5$:       MOVL    R1, VBN                                 ; 1933
          18       00          6E   00 2C 00048          MOVC5   #0, (SP), #0, #24, B                    ; 1941
                          0C AE      0004D          ;                                                   ;
                       0E AE   01 B0 0004F          MOVW    #1, B+2                                     ; 1942
                       10 AE   6E D0 00053          MOVL    VBN, B+4                                    ; 1943
                               7E D4 00057          CLRL    -(SP)                                       ; 1944
                          10 AE   9F 00059          PUSHAB  B                                           ;
                   0000G  CF      02 FB 0005C          CALLS   #2, ANL$BUCKET                              ;
                               6E   DD 00061          PUSHL   VBN                                         ; 1949
```

```
                    00000000G  8F  DD 00063      PUSHL   #ANLRMS$_DUMPHEADING          ;
             7E                03  7D 00069      MOVQ    #3, -(SP)                     ;
       0000G CF                04  FB 0006C      CALLS   #4, ANL$FORMAT_LINE           ;
          04 AE        0200    8F  3C 00071      MOVZWL  #512, BLOCK_DSC               ; 1955
          08 AE          18    AE  D0 00077      MOVL    B+12, BLOCK_DSC+4             ;
                         04    AE  9F 0007C      PUSHAB  BLOCK_DSC                     ; 1956
                               01  DD 0007F      PUSHL   #1                            ;
       0000G CF                02  FB 00081      CALLS   #2, ANL$FORMAT_HEX            ;
             7E                01  CE 00086      MNEGL   #1, -(SP)                     ; 1959
                         10    AE  9F 00089      PUSHAB  B                             ;
       0000G CF                02  FB 0008C      CALLS   #2, ANL$BUCKET                ;
                                   04 00091      RET                                   ; 1963

; Routine Size:  146 bytes,    Routine Base:  $CODE$ + 0A7E
```

```
;  1469          1964  1 %sbttl 'ANL$INTERACTIVE_HELP - Handle the HELP Command'
;  1470          1965  1 !++
;  1471          1966  1 ! Functional Description:
;  1472          1967  1 !       This routine is responsible for handling the interactive HELP command.
;  1473          1968  1 !       All the work is done by LBR$OUTPUT_HELP.
;  1474          1969  1 !
;  1475          1970  1 ! Formal Parameters:
;  1476          1971  1 !       arguments       A descriptor of the help keywords as entered by user.
;  1477          1972  1 !
;  1478          1973  1 ! Implicit Inputs:
;  1479          1974  1 !       global data
;  1480          1975  1 !
;  1481          1976  1 ! Implicit Outputs:
;  1482          1977  1 !       global data
;  1483          1978  1 !
;  1484          1979  1 ! Returned Value:
;  1485          1980  1 !       none
;  1486          1981  1 !
;  1487          1982  1 ! Side Effects:
;  1488          1983  1 !
;  1489          1984  1 !--
;  1490          1985  1
;  1491          1986  1
;  1492          1987  2 global routine anl$interactive_help(arguments): novalue = begin
;  1493          1988  2
;  1494          1989  2 bind
;  1495          1990  2         arguments_dsc = .arguments: descriptor;
;  1496          1991  2
;  1497          1992  2 local
;  1498          1993  2         status: long;
;  1499          1994  2
;  1500          1995  2
;  1501          1996  2 ! Simply call the wonderful librarian to do the work.
;  1502          1997  2
;  1503          1998  2 status = lbr$output_help(lib$put_output,0,arguments_dsc,describe('ANLRMSHLP'),
;  1504          1999  2                         0,lib$get_input);
;  1505          2000  2 check (.status, .status);
;  1506          2001  2
;  1507          2002  2 return;
;  1508          2003  2
;  1509          2004  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                50  4C  48  53  4D  52  4C  4E  41  0024C P.ABX:  .ASCII  \ANLRMSHLP\                                    ;
                                                    00255       .BLKB   3
                                            00000009    00258 P.ABW:  .LONG   9                                         ;
                                            00000000'   0025C         .ADDRESS P.ABX                                    ;


                                                        .PSECT  $CODE$,NOWRT,2

                                                0000 00000      .ENTRY  ANL$INTERACTIVE_HELP, Save nothing          ; 1987
                                    00000000G   00   9F 00002   PUSHAB  LIB$GET_INPUT                               ; 1998
```

```
                                    7E  D4 00008          CLRL    -(SP)                                                  ;
                                0000' CF  9F 0000A        PUSHAB  P.ABW                                                  ;
                                   04 AC  DD 0000E        PUSHL   ARGUMENTS                                              ;
                                    7E  D4 00011          CLRL    -(SP)                                                  ;
                            00000000G 00  9F 00013        PUSHAB  LIB$PUT_OUTPUT                                         ;
                00000000G   00        06  FB 00019        CALLS   #6, LBR$OUTPUT_HELP                                    ;
                            09        50  E8 00020        BLBS    STATUS, 1$                                             ; 2000
                                      50  DD 00023        PUSHL   STATUS                                                 ;
                00000000G   00        01  FB 00025        CALLS   #1, LIB$SIGNAL                                         ;
                                      04  0002C 1$:       RET                                                            ; 2004

; Routine Size:  45 bytes,    Routine Base:  $CODE$ + 0B10

; 1510          2005  1
; 1511          2006  0 end eludom



                                                          .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $OWN$                         2744  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         608  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        2877  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  _LIB$KEY0$                      20  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$STATE$                    152  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$KEY1$                      50  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)


;                            Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        60         0        1000        00:01.8
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1          42        25        59          14        00:00.1



;                                  COMMAND QUALIFIERS

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

; 1512          2007  0
; Size:         2877 code + 3574 data bytes
; Run Time:        01:08.1
```

```
; Elapsed Time:    04:00.3
; Lines/CPU Min:    1768
; Lexemes/CPU-Min: 32462
; Memory Used:  500 pages
; Compilation Complete
```

RMSINTER
LIS

RMSCHECKA
LIS

RMSFDL
LIS

RMSCHECKB
LIS

RMSINPUT
LIS

RMSMSG
LIS