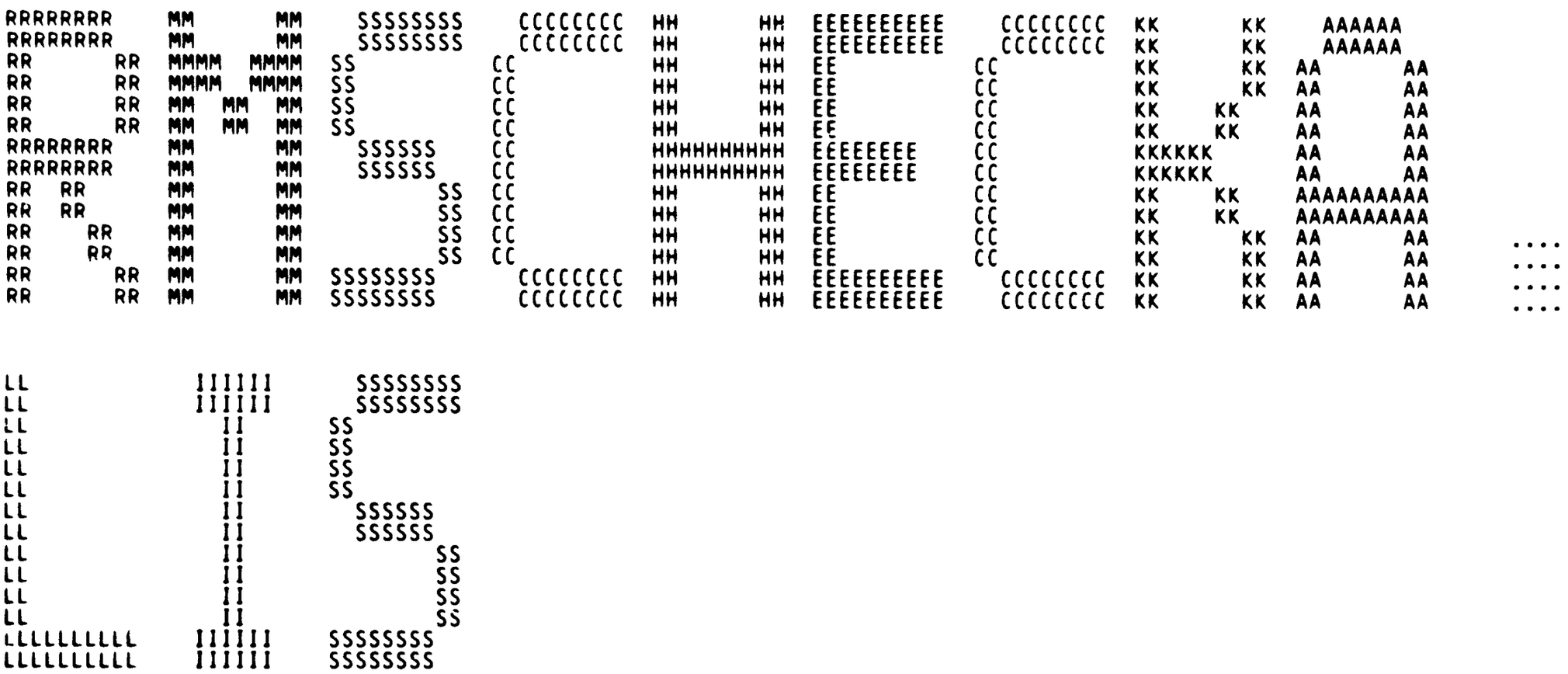
```
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
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
```


RMS-11
LSI

```
;  1     0001  0 %title 'RMSCHECKA - Check a File Structure'
;  2     0002  0         module rmschecka (
;  3     0003  1                           ident='V04-000') = begin
;  4     0004  1
;  5     0005  1 !
;  6     0006  1 !*****************************************************************************
;  7     0007  1 !*                                                                           *
;  8     0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  9     0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
; 10     0010  1 !*  ALL RIGHTS RESERVED.                                                     *
; 11     0011  1 !*                                                                           *
; 12     0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 13     0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 14     0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 15     0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
; 16     0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
; 17     0017  1 !*  TRANSFERRED.                                                             *
; 18     0018  1 !*                                                                           *
; 19     0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 20     0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 21     0021  1 !*  CORPORATION.                                                             *
; 22     0022  1 !*                                                                           *
; 23     0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 24     0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 25     0025  1 !*                                                                           *
; 26     0026  1 !*                                                                           *
; 27     0027  1 !*****************************************************************************
; 28     0028  1 !
; 29     0029  1
; 30     0030  1 !++
; 31     0031  1 ! Facility:     VAX/VMS Analyze Facility, Check a File Structure
; 32     0032  1 !
; 33     0033  1 ! Abstract:     This module is responsible for checking the structure of
; 34     0034  1 !               an RMS file as requested via /CHECK.  It also prepares a
; 35     0035  1 !               report of the results.
; 36     0036  1 !
; 37     0037  1 !
; 38     0038  1 ! Environment:
; 39     0039  1 !
; 40     0040  1 ! Author: Paul C. Anagnostopoulos, Creation Date: 20 February 1981
; 41     0041  1 !
; 42     0042  1 ! Modified By:
; 43     0043  1 !
; 44     0044  1 !       V03-006 DGB0047         Donald G. Blair         08-May-1984
; 45     0045  1 !               Fix condition handling so ANALYZRMS returns the correct
; 46     0046  1 !               error status at image exit.  Change condition handler
; 47     0047  1 !               from ANL$CONDITION_HANDLER to ANL$UNWIND_HANDLER.
; 48     0048  1 !
; 49     0049  1 !       V03-005 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
; 50     0050  1 !               Change the message prefix to ANLRMS$  to ensure that
; 51     0051  1 !               message symbols are unique across all ANALYZEs.  This
; 52     0052  1 !               is necessitated by the new merged message files.
; 53     0053  1 !
; 54     0054  1 !       V03-004 PCA1001         Paul C. Anagnostopoulos  5-Nov-1982
; 55     0055  1 !               Add code to support the new /SUMMARY mode.
; 56     0056  1 !               Trim the command line before including it in the report.
; 57     0057  1 !
```

```
;    58          0058  1 !       V03-003 PCA0061         Paul Anagnostopoulos    29-Mar-1982
;    59          0059  1 !               Fix bug in ANL$PROLOG_CHECKSUMS so it calculates the
;    60          0060  1 !               number of prologue blocks correctly.
;    61          0061  1 !
;    62          0062  1 !       V03-002 PCA0006         Paul Anagnostopoulos    16-Mar-1982
;    63          0063  1 !               Modify table of stream record delimiters to conform
;    64          0064  1 !               to new standard.
;    65          0065  1 !
;    66          0066  1 !       V03-001 PCA0005         Paul Anagnostopoulos    16-Mar-1982
;    67          0067  1 !               Fix relative file checking so that it doesn't choke
;    68          0068  1 !               on a file with no records.
;    69          0069  1 !--
```

```
;    71          0070  1 %sbttl 'Module Declarations'
;    72          0071  1 !
;    73          0072  1 ! Libraries and Requires:
;    74          0073  1 !
;    75          0074  1
;    76          0075  1 library 'lib';
;    77          0076  1 require 'rmsreq';
;    78          0585  1
;    79          0586  1 !
;    80          0587  1 ! Table of Contents:
;    81          0588  1 !
;    82          0589  1
;    83          0590  1 forward routine
;    84          0591  1         anl$check_mode: novalue,
;    85          0592  1         anl$seq_check: novalue,
;    86          0593  1         anl$seq_data_record,
;    87          0594  1         anl$stream_record_length,
;    88          0595  1         anl$rel_check: novalue,
;    89          0596  1         anl$rel_prolog: novalue,
;    90          0597  1         anl$rel_cell,
;    91          0598  1         anl$prolog_checksums: novalue;
;    92          0599  1
;    93          0600  1 !
;    94          0601  1 ! External References:
;    95          0602  1 !
;    96          0603  1
;    97          0604  1 external routine
;    98          0605  1         anl$bucket,
;    99          0606  1         anl$check_flags,
;   100          0607  1         anl$error_count,
;   101          0608  1         anl$format_error,
;   102          0609  1         anl$format_file_attributes,
;   103          0610  1         anl$format_file_header,
;   104          0611  1         anl$format_flags,
;   105          0612  1         anl$format_hex,
;   106          0613  1         anl$format_line,
;   107          0614  1         anl$format_skip,
;   108          0615  1         anl$idx_check,
;   109          0616  1         anl$open_next_rms_file,
;   110          0617  1         anl$prepare_report_file,
;   111          0618  1         anl$unwind_handler,
;   112          0619  1         cli$get_value: addressing_mode(general),
;   113          0620  1         lib$establish: addressing_mode(general),
;   114          0621  1         str$trim: addressing_mode(general);
;   115          0622  1
;   116          0623  1 external
;   117          0624  1         anl$gb_mode: byte,
;   118          0625  1         anl$gl_fat: ref block[,byte];
;   119          0626  1
;   120          0627  1 !
;   121          0628  1 ! Own Variables:
;   122          0629  1 !
```

```
;   124          0630  1 %sbttl 'ANL$CHECK_MODE - Check an RMS File Structure'
;   125          0631  1 !++
;   126          0632  1 ! Functional Description:
;   127          0633  1 !       This is the main routine for checking the structure of an RMS file
;   128          0634  1 !       and producing a report of the results.  It is responsible for
;   129          0635  1 !       cycling through the input files and calling the appropriate routine.
;   130          0636  1 !
;   131          0637  1 !       This routine is also called for /STATISTICS mode.  This mode is
;   132          0638  1 !       a superset of /CHECK mode, doing all the same checking in addition
;   133          0639  1 !       to printing some statistics.
;   134          0640  1 !
;   135          0641  1 ! Formal Parameters:
;   136          0642  1 !       none
;   137          0643  1 !
;   138          0644  1 ! Implicit Inputs:
;   139          0645  1 !       global data
;   140          0646  1 !
;   141          0647  1 ! Implicit Outputs:
;   142          0648  1 !       global data
;   143          0649  1 !
;   144          0650  1 ! Returned Value:
;   145          0651  1 !       none
;   146          0652  1 !
;   147          0653  1 ! Side Effects:
;   148          0654  1 !
;   149          0655  1 !--
;   150          0656  1
;   151          0657  1
;   152          0658  2 global routine anl$check_mode: novalue = begin
;   153          0659  2
;   154          0660  2 local
;   155          0661  2         status: long;
;   156          0662  2
;   157          0663  2
;   158          0664  2 ! We go into a loop, once through for each input file.
;   159          0665  2
;   160          0666  3 loop (
;   161          0667  3         local
;   162          0668  3                 local_described_buffer(resultant_file_spec,nam$c_maxrss);
;   163          0669  3
;   164          0670  3         status = anl$open_next_rms_file(resultant_file_spec);
;   165          0671  3
;   166          0672  3 exitif (not .status);
;   167          0673  3
;   168          0674  3         ! Prepare the file to receive the report.  We use a different
;   169          0675  3         ! page heading for each of the modes.
;   170          0676  3
;   171          0677  4         anl$prepare_report_file((selectoneu .anl$gb_mode of set
;   172          0678  4                                 [anl$k_check]:          anlrms$_checkhdg;
;   173          0679  4                                 [anl$k_statistics]:     anlrms$_stathdg;
;   174          0680  4                                 [anl$k_summary]:        anlrms$_summaryhdg;
;   175          0681  3                                 tes),
;   176          0682  3                                 resultant_file_spec);
;   177          0683  3
;   178          0684  3         ! Print the file header information and RMS attribute information in
;   179          0685  3         ! the report.
;   180          0686  3
```

```
;  181          0687  3          anl$format_file_header();
;  182          0688  3
;  183          0689  3          anl$format_skip(0);
;  184          0690  3          anl$format_skip(0);
;  185          0691  3          anl$format_file_attributes();
;  186          0692  3
;  187          0693  3          ! Now call the appropriate checking routine.
;  188          0694  3
;  189          0695  3          selectoneu .anl$gl_fat[fat$v_fileorg] of set
;  190          0696  3          [fat$c_sequential]:     anl$seq_check();
;  191          0697  3          [fat$c_relative]:       anl$rel_check();
;  192          0698  3          [fat$c_indexed]:        anl$idx_check();
;  193          0699  3          tes;
;  194          0700  3
;  195          0701  3          ! Tell the user how many structure errors were discovered.
;  196          0702  3
;  197          0703  3          anl$format_skip(0);
;  198          0704  3          anl$format_skip(0);
;  199          0705  3          anl$error_count();
;  200          0706  3
;  201          0707  3          ! Finally, include the command line at the end of the report.
;  202          0708  3
;  203          0709  4          begin
;  204          0710  4          local
;  205          0711  4                  local_described_buffer(command_line,80);
;  206          0712  4
;  207          0713  4          cli$get_value(describe('$LINE'),command_line);
;  208          0714  4          str$trim(command_line,command_line,command_line);
;  209          0715  4          anl$format_skip(0);
;  210          0716  4          anl$format_skip(0);
;  211          0717  4          anl$format_line(0,0,anlrms$_anything,command_line);
;  212          0718  3          end;
;  213          0719  3
;  214          0720  2 );
;  215          0721  2
;  216          0722  2 return;
;  217          0723  2
;  218          0724  1 end;

                                                    .TITLE  RMSCHECKA RMSCHECKA - Check a File Structure
                                                    .IDENT  \V04-000\

                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                        '           45  4E  49  4C  24  00000 P.AAB:  .ASCII  \$LINE\                                ;
                        -                           00005           .BLKB   3
                                          00000005  00008 P.AAA:    .LONG   5                                         ;
                                          00000000' 0000C           .ADDRESS P.AAB                                    ;

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

```
.EXTRN  ANLRMS$_BKTKEY, ANLRMS$_BKTLEVEL
.EXTRN  ANLRMS$_BKTNEXT
.EXTRN  ANLRMS$_BKTPTRSIZE
.EXTRN  ANLRMS$_BKTRECID
.EXTRN  ANLRMS$_BKTRECID3
.EXTRN  ANLRMS$_BKTSAMPLE
.EXTRN  ANLRMS$_BKTVBNFREE
.EXTRN  ANLRMS$_BUCKETSIZE
.EXTRN  ANLRMS$_CELL, ANLRMS$_CELLDATA
.EXTRN  ANLRMS$_CELLFLAGS
.EXTRN  ANLRMS$_CHECKHDG
.EXTRN  ANLRMS$_CONTIG, ANLRMS$_CREATION
.EXTRN  ANLRMS$_CTLSIZE
.EXTRN  ANLRMS$_DATAREC
.EXTRN  ANLRMS$_DATABKTVBN
.EXTRN  ANLRMS$_DUMPHEADING
.EXTRN  ANLRMS$_EOF, ANLRMS$_ERRORCOUNT
.EXTRN  ANLRMS$_ERRORNONE
.EXTRN  ANLRMS$_ERRORS, ANLRMS$_EXPIRATION
.EXTRN  ANLRMS$_FILEATTR
.EXTRN  ANLRMS$_FILEHDR
.EXTRN  ANLRMS$_FILEID, ANLRMS$_FILEORG
.EXTRN  ANLRMS$_FILESPEC
.EXTRN  ANLRMS$_FLAG, ANLRMS$_GLOBALBUFS
.EXTRN  ANLRMS$_HEXDATA
.EXTRN  ANLRMS$_HEXHEADING1
.EXTRN  ANLRMS$_HEXHEADING2
.EXTRN  ANLRMS$_IDXAREA
.EXTRN  ANLRMS$_IDXAREAALLOC
.EXTRN  ANLRMS$_IDXAREABKTSZ
.EXTRN  ANLRMS$_IDXAREANEXT
.EXTRN  ANLRMS$_IDXAREANOALLOC
.EXTRN  ANLRMS$_IDXAREAQTY
.EXTRN  ANLRMS$_IDXARERECL
.EXTRN  ANLRMS$_IDXAREUSED
.EXTRN  ANLRMS$_IDXKEY, ANLRMS$_IDXKEYAREAS
.EXTRN  ANLRMS$_IDXKEYBKTSZ
.EXTRN  ANLRMS$_IDXKEYBYTES
.EXTRN  ANLRMS$_IDXKEYITYPE
.EXTRN  ANLRMS$_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$_FACILITY
                .EXTRN  ANL$BUCKET, ANL$CHECK_FLAGS
                .EXTRN  ANL$ERROR_COUNT
```

```
                                                .EXTRN  ANL$FORMAT_ERROR
                                                .EXTRN  ANL$FORMAT_FILE_ATTRIBUTES
                                                .EXTRN  ANL$FORMAT_FILE_HEADER
                                                .EXTRN  ANL$FORMAT_FLAGS
                                                .EXTRN  ANL$FORMAT_HEX, ANL$FORMAT_LINE
                                                .EXTRN  ANL$FORMAT_SKIP
                                                .EXTRN  ANL$IDX_CHECK, ANL$OPEN_NEXT_RMS_FILE
                                                .EXTRN  ANL$PREPARE_REPORT_FILE
                                                .EXTRN  ANL$UNWIND_HANDLER
                                                .EXTRN  CLI$GET_VALUE, LIB$ESTABLISH
                                                .EXTRN  STR$TRIM, ANL$GB_MODE
                                                .EXTRN  ANL$GL_FAT

                                                .PSECT  $CODE$,NOWRT,2

                                    001C 00000      .ENTRY  ANL$CHECK_MODE, Save R2,R3,R4                   ; 0658
                54      0000G   CF  9E 00002        MOVAB   ANL$FORMAT_SKIP, R4                             ;
                5E      FEA0    CE  9E 00007        MOVAB   -352(SP), SP                                    ;
             58 AE        FF    8F  9A 0000C 1$:    MOVZBL  #255, RESULTANT_FILE_SPEC                       ; 0668
             5C AE        60    AE  9E 00011        MOVAB   RESULTANT_FILE_SPEC+8, -                        ;
                                                            RESULTANT_FILE_SPEC+4                           ;
                          58    AE  9F 00016        PUSHAB  RESULTANT_FILE_SPEC                             ; 0670
          0000G CF                01 FB 00019        CALLS   #1, ANL$OPEN_NEXT_RMS_FILE                      ;
                53                50 D0 0001E        MOVL    R0, STATUS                                      ;
                01                53 E8 00021        BLBS    STATUS, 2$                                      ; 0672
                                  04 00024        RET                                                     ;
                          58    AE  9F 00025 2$:    PUSHAB  RESULTANT_FILE_SPEC                             ; 0677
                50      0000G   CF  9A 00028        MOVZBL  ANL$GB_MODE, R0                                 ;
                01                50 91 0002D        CMPB    R0, #1                                          ; 0678
                                  08 12 00030        BNEQ    3$                                              ;
                       00000000G  8F  DD 00032        PUSHL   #ANLRMS$_CHECKHDG                               ;
                                  1D 11 00038        BRB     6$                                              ;
                04                50 91 0003A 3$:     CMPB    R0, #4                                          ; 0679
                                  08 12 0003D        BNEQ    4$                                              ;
                       00000000G  8F  DD 0003F        PUSHL   #ANLRMS$_STATHDG                                ;
                                  10 11 00045        BRB     6$                                              ;
                05                50 91 00047 4$:     CMPB    R0, #5                                          ; 0680
                                  05 13 0004A        BEQL    5$                                              ;
                7E                01 CE 0004C        MNEGL   #1, -(SP)                                       ;
                                  06 11 0004F        BRB     6$                                              ;
                       00000000G  8F  DD 00051 5$:    PUSHL   #ANLRMS$_SUMMARYHDG                             ;
          0000G CF                02 FB 00057 6$:    CALLS   #2, ANL$PREPARE_REPORT_FILE                     ; 0677
          0000G CF                00 FB 0005C        CALLS   #0, ANL$FORMAT_FILE_HEADER                      ; 0687
                                  7E D4 00061        CLRL    -(SP)                                           ; 0689
                64                01 FB 00063        CALLS   #1, ANL$FORMAT_SKIP                             ;
                                  7E D4 00066        CLRL    -(SP)                                           ; 0690
                64                01 FB 00068        CALLS   #1, ANL$FORMAT_SKIP                             ;
          0000G CF                00 FB 0006B        CALLS   #0, ANL$FORMAT_FILE_ATTRIBUTES                  ; 0691
52        0000G DF          04    04 EF 00070        EXTZV   #4, #4, @ANL$GL_FAT, R2                         ; 0695
                                  07 12 00077        BNEQ    7$                                              ; 0696
          0000V CF                00 FB 00079        CALLS   #0, ANL$SEQ_CHECK                               ;
                                  16 11 0007E        BRB     9$                                              ;
                01                52 D1 00080 7$:    CMPL    R2, #1                                          ; 0697
                                  07 12 00083        BNEQ    8$                                              ;
          0000V CF                00 FB 00085        CALLS   #0, ANL$REL_CHECK                               ;
                                  0A 11 0008A        BRB     9$                                              ;
                02                52 D1 0008  8$:    CMPL    R2, #2                                          ; 0698
```

```
                                 05  12 0008F          BNEQ    9$
                        0000G    CF     00  FB 00091          CALLS   #0, ANL$IDX_CHECK
                                        7E  D4 00096 9$:      CLRL    -(SP)                                                   ; 0703
                                 64     01  FB 00098          CALLS   #1, ANL$FORMAT_SKIP
                                        7E  D4 0009B          CLRL    -(SP)                                                   ; 0704
                                 64     01  FB 0009D          CALLS   #1, ANL$FORMAT_SKIP
                        0000G    CF     00  FB 000A0          CALLS   #0, ANL$ERROR_COUNT                                     ; 0705
                                 6E  50 8F  9A 000A5          MOVZBL  #80, COMMAND_LINE                                       ; 0711
                          04     AE  08 AE  9E 000A9          MOVAB   COMMAND_LINE+8, COMMAND_LINE+4
                                        5E  DD 000AE          PUSHL   SP                                                      ; 0713
                                 0000'  CF  9F 000B0          PUSHAB  P.AAA
                00000000G        00     02  FB 000B4          CALLS   #2, CLI$GET_VALUE
                                        5E  DD 000BB          PUSHL   SP                                                      ; 0714
                                   04   AE  9F 000BD          PUSHAB  COMMAND_LINE
                                   08   AE  9F 000C0          PUSHAB  COMMAND_LINE
                00000000G        00     03  FB 000C3          CALLS   #3, STR$TRIM
                                        7E  D4 000CA          CLRL    -(SP)                                                   ; 0715
                                 64     01  FB 000CC          CALLS   #1, ANL$FORMAT_SKIP
                                        7E  D4 000CF          CLRL    -(SP)                                                   ; 0716
                                 64     01  FB 000D1          CALLS   #1, ANL$FORMAT_SKIP
                                        5E  DD 000D4          PUSHL   SP                                                      ; 0717
                              00000000G 8F  DD 000D6          PUSHL   #ANLRMS$_ANYTHING
                                        7E  7C 000DC          CLRQ    -(SP)
                        0000G    CF     04  FB 000DE          CALLS   #4, ANL$FORMAT_LINE
                                     FF26   31 000E3          BRW     1$                                                      ; 0661
                                            04 000E6          RET                                                             ; 0724

; Routine Size:  231 bytes,    Routine Base:  $CODE$ + 0000
```

1

```
;  220          0725  1 %sbttl 'ANL$SEQ_CHECK - Sequential File Check'
;  221          0726  1 !++
;  222          0727  1 ! Functional Description:
;  223          0728  1 !       This routine is responsible for performing the structure check
;  224          0729  1 !       on an RMS sequential file.
;  225          0730  1 !
;  226          0731  1 ! Formal Parameters:
;  227          0732  1 !       none
;  228          0733  1 !
;  229          0734  1 ! Implicit Inputs:
;  230          0735  1 !       global data
;  231          0736  1 !
;  232          0737  1 ! Implicit Outputs:
;  233          0738  1 !       global data
;  234          0739  1 !
;  235          0740  1 ! Returned Value:
;  236          0741  1 !       none
;  237          0742  1 !
;  238          0743  1 ! Side Effects:
;  239          0744  1 !
;  240          0745  1 !--
;  241          0746  1
;  242          0747  1
;  243          0748  2 global routine anl$seq_check: novalue = begin
;  244          0749  2
;  245          0750  2 local
;  246          0751  2         b: bsd;
;  247          0752  2
;  248          0753  2
;  249          0754  2 ! Establish the condition handler for drastic structure errors.
;  250          0755  2
;  251          0756  2 lib$establish(anl$unwind_handler);
;  252          0757  2
;  253          0758  2 ! If we are running in /SUMMARY mode, then the user is not interested in
;  254          0759  2 ! spending time to read through the entire file.
;  255          0760  2
;  256          0761  2 if .anl$gb_mode eqlu anl$k_summary then
;  257          0762  2         return;
;  258          0763  2
;  259          0764  2 ! If the file is null, just forget it.
;  260          0765  2
;  261          0766  2 if .anl$gl_fat[fat$l_efblk] eqlu 1 and .anl$gl_fat[fat$w_ffbyte] eqlu 0 then
;  262          0767  2         return;
;  263          0768  2
;  264          0769  2 ! We are going to scan the blocks of the file, treating the file as if
;  265          0770  2 ! it contains single block buckets.  Get the first block.
;  266          0771  2
;  267          0772  2 init_bsd(b);
;  268          0773  2 b[bsd$w_size] = 1;
;  269          0774  2 b[bsd$l_vbn] = 1;
;  270          0775  2 anl$bucket(b,0);
;  271          0776  2
;  272          0777  2 ! Now we sit in a loop, checking each record.
;  273          0778  2
;  274          0779  2 while anl$seq_data_record(b,false) do;
;  275          0780  2
;  276          0781  2 anl$bucket(b,-1);
```

```
;  277          0782  2
;  278          0783  2 return;
;  279          0784  2
;  280          0785  1 end;


                                                003C 00000          .ENTRY  ANL$SEQ_CHECK, Save R2,R3,R4,R5     ; 0748
                                    5E                  18  C2 00002  SUBL2   #24, SP
                                              0000G     CF  9F 00005  PUSHAB  ANL$UNWIND_HANDLER                  ; 0756
                                00000000G     00        01  FB 00009  CALLS   #1, LIB$ESTABLISH
                                    05        0000G     CF  91 00010  CMPB    ANL$GB_MODE, #5                   ; 0761
                                                        40  13 00015  BEQL    3$
                                    50        0000G     CF  D0 00017  MOVL    ANL$GL_FAT, R0                    ; 0766
                                    01          08      A0  D1 0001C  CMPL    8(R0), #1
                                                        05  12 00020  BNEQ    1$
                                                  0C    A0  B5 00022  TSTW    12(R0)
                                                        30  13 00025  BEQL    3$
                18                  00                  6E  00  2C 00027 1$:  MOVC5   #0, (SP), #0, #24, B      ; 0772
                                                        6E      0002C
                                          02        AE      01  B0 0002D  MOVW    #1, B+2                       ; 0773
                                          04        AE      01  D0 00031  MOVL    #1, B+4                       ; 0774
                                                        7E  D4 00035  CLRL    -(SP)                             ; 0775
                                                  04    AE  9F 00037  PUSHAB  B
                                        0000G     CF          02  FB 0003A  CALLS   #2, ANL$BUCKET
                                                        7E  D4 0003F 2$:  CLRL    -(SP)                         ; 0779
                                                  04    AE  9F 00041  PUSHAB  B
                                        0000V     CF          02  FB 00044  CALLS   #2, ANL$SEQ_DATA_RECORD
                                                  F3        50  E8 00049  BLBS    R0, 2$
                                                  7E        01  CE 0004C  MNEGL   #1, -(SP)                      ; 0781
                                                  04    AE  9F 0004F  PUSHAB  B
                                        0000G     CF          02  FB 00052  CALLS   #2, ANL$BUCKET
                                                              04  00057 3$:  RET                               ; 0785

; Routine Size:  88 bytes,    Routine Base:  $CODE$ + 00E7
```

```
;  282          0786  1 %sbttl 'ANL$SEQ_DATA_RECORD - Display & Check a Sequential Data Record'
;  283          0787  1 !++
;  284          0788  1 ! Functional Description:
;  285          0789  1 !       This routine is responsible for analyzing a data record from a
;  286          0790  1 !       sequential file.  It can display the record and also check its
;  287          0791  1 !       validity.
;  288          0792  1 !
;  289          0793  1 ! Formal Parameters:
;  290          0794  1 !       rec_bsd         A BSD describing the record.  We update it to
;  291          0795  1 !                       describe the next record.
;  292          0796  1 !       report          A boolean, true if we are to format the record.
;  293          0797  1 !       indent_level    The level of indentation of the report.
;  294          0798  1 !
;  295          0799  1 ! Implicit Inputs:
;  296          0800  1 !       global data
;  297          0801  1 !
;  298          0802  1 ! Implicit Outputs:
;  299          0803  1 !       global data
;  300          0804  1 !
;  301          0805  1 ! Returned Value:
;  302          0806  1 !       True if there is a record following this one; false otherwise.
;  303          0807  1 !
;  304          0808  1 ! Side Effects:
;  305          0809  1 !
;  306          0810  1 !--
;  307          0811  1
;  308          0812  1
;  309          0813  2 global routine anl$seq_data_record(rec_bsd,report,indent_level) = begin
;  310          0814  2
;  311          0815  2 bind
;  312          0816  2         b = .rec_bsd: bsd;
;  313          0817  2
;  314          0818  2 local
;  315          0819  2         sp: ref block[,byte],
;  316          0820  2         filler: byte,
;  317          0821  2         data_length: long, length: long,
;  318          0822  2         d: bsd,
;  319          0823  2         next_vbn: long, next_offset: long;
;  320          0824  2
;  321          0825  2
;  322          0826  2 ! Set up a pointer to the record.
;  323          0827  2
;  324          0828  2 sp = .b[bsd$l_bufptr] + .b[bsd$l_offset];
;  325          0829  2
;  326          0830  2 ! Now we will set up three items.  FILLER will tell us whether this is a
;  327          0831  2 ! block filler for a no-span file.  DATA_LENGTH will tell us the length
;  328          0832  2 ! of the actual data bytes.  LENGTH will give us the overall length of the
;  329          0833  2 ! record.
;  330          0834  2
;  331          0835  2 selectoneu .anl$gl_fat[fat$v_rtype] of set
;  332          0836  3 [fat$c_undefined]:       (filler = false;
;  333          0837  2                         length = data_length = 512;);
;  334          0838  2
;  335          0839  3 [fat$c_fixed]:           (filler = .anl$gl_fat[fat$v_nospan] and
;  336          0840  3                                 .anl$gl_fat[fat$w_maxrec] gtru 512-.b[bsd$l_offset];
;  337          0841  3                         data_length = .anl$gl_fat[fat$w_maxrec];
;  338          0842  2                         length = .data_length + .data_length mod 2;);
```

```
;  339          0843  2
;  340          0844  2 [fat$c_variable,
;  341          0845  3  fat$c_vfc]:            (filler = .anl$gl_fat[fat$v_nospan] and
;  342          0846  3                                .sp[0,0,16,0] eglu %x'ffff';
;  343          0847  3                         data_length = .sp[0,0,16,0];
;  344          0848  2                         length = 2 + .data_length + .data_length mod 2;);
;  345          0849  2 [fat$c_stream,
;  346          0850  2  fat$c_streamlf,
;  347          0851  3  fat$c_streamcr]:       (filler = false;
;  348          0852  2                         length = data_length = anl$stream_record_length(b););
;  349          0853  2 tes;
;  350          0854  2
;  351          0855  2 ! Now we must check to see if a record in a no-span file spills across
;  352          0856  2 ! a block.  If so, we have a drastic structure error.
;  353          0857  2
;  354          0858  3 if .anl$gl_fat[fat$v_nospan] and not .filler and .b[bsd$l_offset]+.length gtru 512 then (
;  355          0859  3         anl$format_error(anlrms$_spanerror,.b[bsd$l_vbn]);
;  356          0860  3         signal (anlrms$_unwind);
;  357          0861  2 );
```

```
;  359          0862  2 ! Now we can format the record if requested.
;  360          0863  2
;  361          0864  2 if .report then
;  362          0865  2
;  363          0866  2         if .filler then
;  364          0867  2
;  365          0868  2                 ! If it's a nospan filler, just format a heading.
;  366          0869  2
;  367          0870  2                 anl$format_line(0,.indent_level,anlrms$_nospanfiller,
;  368          0871  2                                 .b[bsd$l_vbn],.b[bsd$l_offset])
;  369          0872  2
;  370          0873  3         else (
;  371          0874  3
;  372          0875  3                 ! If it's a real record, we need to bring the entire thing
;  373          0876  3                 ! into memory.  Build a BSD describing the record and get it.
;  374          0877  3
;  375          0878  3                 init_bsd(d);
;  376          0879  3                 d[bsd$w_size] = (.b[bsd$l_offset] + .length + 511) / 512;
;  377          0880  3                 d[bsd$l_vbn] = .b[bsd$l_vbn];
;  378          0881  3                 d[bsd$l_offset] = .b[bsd$l_offset];
;  379          0882  3                 anl$bucket(d,0);
;  380          0883  3
;  381          0884  3                 ! Now we can format a heading and the data in hex.
;  382          0885  3
;  383          0886  4                 begin
;  384          0887  4                 local
;  385          0888  4                         data_dsc: descriptor;
;  386          0889  4
;  387          0890  4                 anl$format_line(4,.indent_level,anlrms$_datarec,
;  388          0891  4                                 .b[bsd$l_vbn],.b[bsd$l_offset]);
;  389          0892  4                 anl$format_skip(0);
;  390          0893  4                 build_descriptor(data_dsc, .length,.d[bsd$l_bufptr]+.d[bsd$l_offset]);
;  391          0894  4                 anl$format_hex(.indent_level+2,data_dsc);
;  392          0895  3                 end;
;  393          0896  3
;  394          0897  3                 anl$bucket(d,-1);
;  395          0898  2         );
```

```
;  397          0899  2 ! Now check the length of a VFC record to make sure the header can fit.
;  398          0900  2
;  399          0901  2 if .anl$gl_fat[fat$v_rtype] eqlu fat$c_vfc and .data_length lssu .anl$gl_fat[fat$b_vfcsize] then
;  400          0902  2         anl$format_error(anlrms$_vfctooshort,.b[bsd$l_vbn]);
;  401          0903  2
;  402          0904  2 ! Now we need to calculate the VBN and offset of the potential next record
;  403          0905  2 ! in the file.
;  404          0906  2
;  405          0907  3 if .filler then (
;  406          0908  3         next_vbn = .b[bsd$l_vbn] + 1;
;  407          0909  3         next_offset = 0;
;  408          0910  3 ) else (
;  409          0911  3         next_vbn = .b[bsd$l_vbn] + (.b[bsd$l_offset]+.length)/512;
;  410          0912  3         next_offset = (.b[bsd$l_offset]+.length) mod 512;
;  411          0913  2 );
;  412          0914  2
;  413          0915  2 ! At this point, we would like to ensure that the record didn't spill past
;  414          0916  2 ! the end of file.  However, some VMS files violate the end of file
;  415          0917  2 ! convention as defined by ODS II (e.g., images linked with debug).
;  416          0918  2 ! Therefore we can't make the check, or we'll for sure generate a ton
;  417          0919  2 ! of SPRs.  Instead, we will simply assume that there isn't another record
;  418          0920  2 ! if we are at or past the end of file.
;  419          0921  2
;  420          0922  2 if (.next_vbn eqlu .anl$gl_fat[fat$l_efblk] and .next_offset gequ .anl$gl_fat[fat$w_ffbyte]) or
;  421          0923  2    (.next_vbn gtru .anl$gl_fat[fat$l_efblk])    then
;  422          0924  2         return false;
;  423          0925  2
;  424          0926  2 ! Well, it appears we do indeed have another record.  Update the caller's BSD
;  425          0927  2 ! to describe it.
;  426          0928  2
;  427          0929  2 b[bsd$l_vbn] = .next_vbn;
;  428          0930  2 b[bsd$l_offset] = .next_offset;
;  429          0931  2 anl$bucket(b,0);
;  430          0932  2
;  431          0933  2 return true;
;  432          0934  2
;  433          0935  1 end;


                                                  0FFC 00000          .ENTRY  ANL$SEQ_DATA_RECORD, Save R2,R3,R4,R5,R6,-      ; 0813
                                                                              R7,R8,R9,R10,R11                                    ;
                                  5B      0000G   CF   9E 00002       MOVAB   ANL$GL_FAT, R11                                     ;
                                  5E              20   C2 00007       SUBL2   #32, SP                                             ;
                                  57        04    AC   D0 0000A       MOVL    REC_BSD, R7                                         ; 0816
                                  5A        08    A7   9E 0000E       MOVAB   8(R7), R10                                          ; 0828
                        53        0C    A7       6A   C1 00012       ADDL3   (R10), 12(R7), SP                                   ;
                                        50       6B   D0 00017       MOVL    ANL$GL_FAT, R0                                      ; 0835
                                        0F       60   93 0001A       BITB    (R0), #15                                           ; 0836
                                                 0E   12 0001D       BNEQ    1$                                                  ;
                                                 59   94 0001F       CLRB    FILLER                                              ;
                                  56      0200   8F   3C 00021       MOVZWL  #512, DATA_LENGTH                                   ; 0837
                                  58      0200   8F   3C 00026       MOVZWL  #512, LENGTH                                        ;
                                                 77   11 0002B       BRB     5$                                                  ; 0835
                01                60            04   00   ED 0002D 1$:    CMPZV   #0, #4, (R0), #1                                    ; 0839
```

```
                                38 12 00032        BNEQ    3$
         51          6A 00000200 8F C3 00034       SUBL3   #512, (R10), R1                ; 0840
                     51          51 CE 0003C       MNEGL   R1, R1                         ;
                                 52 D4 0003F       CLRL    R2                             ;
51     10    A0      10          00 ED 00041       CMPZV   #0, #16, 16(R0), R1            ;
                                 02 1B 00047       BLEQU   2$                             ;
                                 52 D6 00049       INCL    R2                             ;
54     01    A0      01          03 EF 0004B 2$:   EXTZV   #3, #1, 1(R0), R4              ;
                     54          54 D2 00051       MCOML   R4, R4                         ;
         59          52          54 8B 00054       BICB3   R4, R2, FILLER                 ;
                     56       10 A0 3C 00058       MOVZWL  16(R0), DATA_LENGTH            ; 0841
7E       00          56          01 7A 0005C       EMUL    #1, DATA_LENGTH, #0, -(SP)     ; 0842
50       50          8E          02 7B 00061       EDIV    #2, (SP)+, R0, R0              ;
         58          50          56 C1 00066       ADDL3   DATA_LENGTH, R0, LENGTH        ;
                                 57 11 0006A       BRB     7$                             ; 0835
02       60          04          00 ED 0006C 3$:   CMPZV   #0, #4, (R0), #2               ; 0844
                                 33 1F 00071       BLSSU   6$                             ;
03       60          04          00 ED 00073       CMPZV   #0, #4, (R0), #3               ;
                                 2C 1A 00078       BGTRU   6$                             ;
                                 51 D4 0007A       CLRL    R1                             ; 0846
              FFFF   8F          63 B1 0007C       CMPW    (SP), #65535                   ;
                                 02 12 00081       BNEQ    4$                             ;
                                 51 D6 00083       INCL    R1                             ;
52     01    A0      01          03 EF 00085 4$:   EXTZV   #3, #1, 1(R0), R2              ;
                     52          52 D2 0008B       MCOML   R2, R2                         ;
         59          51          52 8B 0008E       BICB3   R2, R1, FILLER                 ;
                     56          63 3C 00092       MOVZWL  (SP), DATA_LENGTH              ; 0847
7E       00          56          01 7A 00095       EMUL    #1, DATA_LENGTH, #0, -(SP)     ; 0848
50       50          8E          02 7B 0009A       EDIV    #2, (SP)+, R0, R0              ;
                     58     02 A046 9E 0009F       MOVAB   2(R0)[DATA_LENGTH], LENGTH     ;
                                 1D 11 000A4 5$:   BRB     7$                             ; 0835
04       60          04          00 ED 000A6 6$:   CMPZV   #0, #4, (R0), #4               ; 0849
                                 16 1F 000AB       BLSSU   7$                             ;
06       60          04          00 ED 000AD       CMPZV   #0, #4, (R0), #6               ;
                                 0F 1A 000B2       BGTRU   7$                             ;
                                 59 94 000B4       CLRB    FILLER                         ; 0851
                                 57 DD 000B6       PUSHL   R7                             ; 0852
              0000V  CF          01 FB 000B8       CALLS   #1, ANL$STREAM_RECORD_LENGTH   ;
                     56          50 D0 000BD       MOVL    R0, DATA_LENGTH                ;
                     58          56 D0 000C0       MOVL    DATA_LENGTH, LENGTH            ;
                     50          6B D0 000C3 7$:   MOVL    ANL$GL_FAT, R0                 ; 0858
         2B     01   A0          03 E1 000C6       BBC     #3, 1(R0), 8$                  ;
                     28          59 E8 000CB       BLBS    FILLER, 8$                     ;
         51          6A          58 C1 000CE       ADDL3   LENGTH, (R10), R1              ;
           00000200  8F          51 D1 000D2       CMPL    R1, #512                       ;
                                 1B 1B 000D9       BLEQU   8$                             ;
                              04 A7 DD 000DB       PUSHL   4(R7)                          ; 0859
                      00000000G 8F DD 000DE        PUSHL   #ANLRMS$_SPANERROR             ;
              0000G  CF          02 FB 000E4       CALLS   #2, ANL$FORMAT_ERROR           ;
                      00000000G 8F DD 000E9        PUSHL   #ANLRMS$_UNWIND                ; 0860
          00000000G  00          01 FB 000EF       CALLS   #1, LIB$SIGNAL                 ;
                     18       08 AC E9 000F6 8$:   BLBC    REPORT, 9$                     ; 0864
                     17          59 E9 000FA       BLBC    FILLER, 10$                    ; 0866
                                 6A DD 000FD       PUSHL   (R10)                          ; 0871
                              04 A7 DD 000FF       PUSHL   4(R7)                          ;
                      00000000G 8F DD 00102        PUSHL   #ANLRMS$_NOSPANFILLER          ; 0870
                              0C AC DD 00108       PUSHL   INDENT_LEVEL                   ;
```

```
                                    7E  D4 0010B        CLRL    -(SP)                                   ;
                     0000G  CF          05  FB 0010D        CALLS   #5, ANL$FORMAT_LINE                     ;
                                    6C  11 00112 9$:    BRB     11$                                     ;
 18        00        6E             00  2C 00114 10$:   MOVC5   #0, (SP), #0, #24, D                    ; 0878
                               08   AE     00119                                                        ;
           51        6A             58  C1 0011B        ADDL3   LENGTH, (R10), R1                       ; 0879
                     51       01FF  C1  9E 0011F        MOVAB   511(R1), R1                             ;
           50        51   00000200  8F  C7 00124        DIVL3   #512, R1, R0                            ;
                0A   AE             50  B0 0012C        MOVW    R0, D+2                                 ;
                0C   AE         04  A7  D0 00130        MOVL    4(R7), D+4                              ; 0880
                10   AE             6A  D0 00135        MOVL    (R10), D+8                              ; 0881
                                    7E  D4 00139        CLRL    -(SP)                                   ; 0882
                                0C  AE  9F 0013B        PUSHAB  D                                       ;
                     0000G  CF          02  FB 0013E        CALLS   #2, ANL$BUCKET                          ;
                                    6A  DD 00143        PUSHL   (R10)                                   ; 0891
                                04  A7  DD 00145        PUSHL   4(R7)                                   ;
                         00000000G  8F  DD 00148        PUSHL   #ANLRMS$_DATAREC                        ; 0890
                                0C  AC  DD 0014E        PUSHL   INDENT_LEVEL                            ;
                                    04  DD 00151        PUSHL   #4                                      ;
                     0000G  CF          05  FB 00153        CALLS   #5, ANL$FORMAT_LINE                     ;
                                    7E  D4 00158        CLRL    -(SP)                                   ; 0892
                     0000G  CF          01  FB 0015A        CALLS   #1, ANL$FORMAT_SKIP                     ;
                            6E          58  D0 0015F        MOVL    LENGTH, DATA_DSC                        ; 0893
       04  AE        14     AE      10  AE  C1 00162        ADDL3   D+8, D+12, DATA_DSC+4                   ;
                                    5E  DD 00169        PUSHL   SP                                      ; 0894
           7E        0C     AC          02  C1 0016B        ADDL3   #2, INDENT_LEVEL, -(SP)                 ;
                     0000G  CF          02  FB 00170        CALLS   #2, ANL$FORMAT_HEX                      ;
                            7E          01  CE 00175        MNEGL   #1, -(SP)                               ; 0897
                                0C  AE  9F 00178        PUSHAB  D                                       ;
                     0000G  CF          02  FB 0017B        CALLS   #2, ANL$BUCKET                          ;
                            50          6B  D0 00180 11$:   MOVL    ANL$GL_FAT, R0                          ; 0901
 03        60               04          00  ED 00183        CMPZV   #0, #4, (R0), #3                        ;
                                    16  12 00188        BNEQ    12$                                     ;
 56   0F   A0               08          00  ED 0018A        CMPZV   #0, #8, 15(R0), DATA_LENGTH             ;
                                    0E  1B 00190        BLEQU   12$                                     ;
                                04  A7  DD 00192        PUSHL   4(R7)                                   ; 0902
                         00000000G  8F  DD 00195        PUSHL   #ANLRMS$_VFCTOOSHORT                    ;
                     0000G  CF          02  FB 0019B        CALLS   #2, ANL$FORMAT_ERROR                    ;
                            0C          59  E9 001A0 12$:   BLBC    FILLER, 13$                             ; 0907
                            52      04  A7  9E 001A3        MOVAB   4(R7), R2                               ; 0908
           54               62          01  C1 001A7        ADDL3   #1, (R2), NEXT_VBN                      ;
                                    53  D4 001AB        CLRL    NEXT_OFFSET                             ; 0909
                                    22  11 001AD        BRB     14$                                     ; 0907
                            52      04  A7  9E 001AF 13$:   MOVAB   4(R7), R2                               ; 0911
           50               6A          58  C1 001B3        ADDL3   LENGTH, (R10), R0                       ;
           51               50  00000200  8F  C7 001B7        DIVL3   #512, R0, R1                            ;
           54               51          62  C1 001BF        ADDL3   (R2), R1, NEXT_VBN                      ;
 7E        00               50          01  7A 001C3        EMUL    #1, R0, #0, -(SP)                       ; 0912
 53        53               8E  00000200  8F  7B 001C8        EDIV    #512, (SP)+, NEXT_OFFSET, NEXT_OFFSET   ;
                            50          6B  D0 001D1 14$:   MOVL    ANL$GL_FAT, R0                          ; 0922
                08   A0                 54  D1 001D4        CMPL    NEXT_VBN, 8(R0)                         ;
                                    08  12 001D8        BNEQ    15$                                     ;
 53   0C   A0               10          00  ED 001DA        CMPZV   #0, #16, 12(R0), NEXT_OFFSET            ;
                                    19  1B 001E0        BLEQU   16$                                     ;
                08   A0                 54  D1 001E2 15$:   CMPL    NEXT_VBN, 8(R0)                         ; 0923
                                    13  1A 001E6        BGTRU   16$                                     ;
                            62          54  D0 001E8        MOVL    NEXT_VBN, (R2)                          ; 0929
```

```
                                 6A            53  D0 001EB          MOVL    NEXT_OFFSET, (R10)                      ; 0930
                                               7E  D4 001EE          CLRL    -(SP)                                   ; 0931
                                               57  DD 001F0          PUSHL   R7                                      ;
                              0000G  CF        02  FB 001F2          CALLS   #2, ANL$BUCKET                          ;
                                     50        01  D0 001F7          MOVL    #1, R0                                  ; 0933
                                               04  001FA          RET                                             ;
                                               50  D4 001FB 16$:    CLRL    R0                                      ; 0935
                                               04  001FD          RET                                             ;

; Routine Size:  510 bytes,    Routine Base:  $CODE$ + 013F
```

```
;  435          0936  1 %sbttl 'ANL$STREAM_RECORD_LENGTH - Determine Length of Stream Record'
;  436          0937  1 !++
;  437          0938  1 ! Functional Description:
;  438          0939  1 !       This routine is responsible for determining the length of a
;  439          0940  1 !       record in a sequential file with stream record format.  Records
;  440          0941  1 !       end with one of three sets of delimiters.
;  441          0942  1 !
;  442          0943  1 ! Formal Parameters:
;  443          0944  1 !       rec_bsd         Address of BSD describing beginning of record.
;  444          0945  1 !
;  445          0946  1 ! Implicit Inputs:
;  446          0947  1 !       global data
;  447          0948  1 !
;  448          0949  1 ! Implicit Outputs:
;  449          0950  1 !       global data
;  450          0951  1 !
;  451          0952  1 ! Returned Value:
;  452          0953  1 !       length of the record.
;  453          0954  1 !
;  454          0955  1 ! Side Effects:
;  455          0956  1 !
;  456          0957  1 !--
;  457          0958  1
;  458          0959  1
;  459          0960  2 global routine anl$stream_record_length(rec_bsd) = begin
;  460          0961  2
;  461          0962  2 bind
;  462          0963  2         b = .rec_bsd: bsd;
;  463          0964  2
;  464          0965  2 bind
;  465          0966  2         delimiter_table = ch$transtable(
;  466          0967  2                                 rep 10 of (false),
;  467          0968  2                                 true,                  ! line feed
;  468          0969  2                                 true,                  ! vertical tab
;  469          0970  2                                 true,                  ! form feed
;  470          0971  2                                 true,                  ! carriage return
;  471          0972  2                                 rep 242 of (false)
;  472          0973  2                                 );
;  473          0974  2 local
;  474          0975  2         r: bsd,
;  475          0976  2         rp: ref block[,byte],
;  476          0977  2         found: byte;
;  477          0978  2
;  478          0979  2 builtin
;  479          0980  2         scanc;
;  480          0981  2
;  481          0982  2
;  482          0983  2 ! We have to scan the bytes in this block and succeeding ores until we
;  483          0984  2 ! find a delimiter.  Copy the caller's BSD because we don't want to
;  484          0985  2 ! modify it.
;  485          0986  2
;  486          0987  2 init_bsd(r);
;  487          0988  2 copy_bucket(b,r);
;  488          0989  2
;  489          0990  2 ! We go through our scanning loop once for each potential delimiter we
;  490          0991  2 ! encounter.
;  491          0992  2
```

```
;  492     0993  3 do (
;  493     0994  3
;  494     0995  3         ! Search the bytes in this block for any character which could
;  495     0996  3         ! possibly be a delimiter.
;  496     0997  3
;  497     0998  3         rp = scanc(%ref(512-.r[bsd$l_offset]),.r[bsd$l_bufptr]+.r[bsd$l_offset],
;  498     0999  3                 delimiter_table,uplit byte (%x'ff'));
;  499     1000  3
;  500     1001  4         if .rp eqla 0 then (
;  501     1002  4
;  502     1003  4                 ! We didn't find any delimiter in this block.  If this block
;  503     1004  4                 ! contains the end-of-file, then we have a drastic structure
;  504     1005  4                 ! error.
;  505     1006  4
;  506     1007  5                 if .r[bsd$l_vbn] eqlu .anl$gl_fat[fat$l_efblk] then (
;  507     1008  5                         anl$format_error(anl$rms$_badstreameof,.b[bsd$l_vbn]);
;  508     1009  5                         signal (anl$rms$_unwind);
;  509     1010  4                 );
;  510     1011  4
;  511     1012  4                 ! We need to search the next block.  Update our BSD and
;  512     1013  4                 ! read it in.
;  513     1014  4
;  514     1015  4                 increment (r[bsd$l_vbn]);
;  515     1016  4                 r[bsd$l_offset] = 0;
;  516     1017  4                 anl$bucket(r,0);
;  517     1018  4
;  518     1019  4         ) else (
;  519     1020  4
;  520     1021  4                 ! We found a potential delimiter.  To determine if it really
;  521     1022  4                 ! is one, we have to split up on record format.  Set flag
;  522     1023  4                 ! FOUND if it really is.
;  523     1024  4
;  524     1025  5                 found = (selectoneu .anl$gl_fat[fat$v_rtype] of set
;  525     1026  5                         [fat$c_stream]:         ch$rchar(.rp) nequ creturn;
;  526     1027  5                         [fat$c_streamlf]:       ch$rchar(.rp) eqlu linefeed;
;  527     1028  5                         [fat$c_streamcr]:       ch$rchar(.rp) eqlu creturn;
;  528     1029  4                         tes);
;  529     1030  4
;  530     1031  4                 ! Update the offset in the BSD to point at the character
;  531     1032  4                 ! following the one we found.  Then continue searching if
;  532     1033  4                 ! it wasn't a real delimiter.
;  533     1034  4
;  534     1035  4                 r[bsd$l_offset] = .rp - .r[bsd$l_bufptr] + 1;
;  535     1036  3         );
;  536     1037  3
;  537     1038  2 ) until .found;
;  538     1039  2
;  539     1040  2 anl$bucket(r,-1);
;  540     1041  2
;  541     1042  2 ! We found a real delimiter.  BSD B tells us where we started searching,
;  542     1043  2 ! and R tells us where we finished.  Calculate the length of the record.
;  543     1044  2
;  544     1045  2 return (.r[bsd$l_vbn]*512+.r[bsd$l_offset]) - (.b[bsd$l_vbn]*512+.b[bsd$l_offset]);
;  545     1046  2
;  546     1047  1 end;
```

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

                                                        00# 00010 P.AAC:  .BYTE   0[10]                                         ;
                                                  01  01  01  01  0001A          .BYTE   1, 1, 1, 1                              ;
                                                        00# 0001E          .BYTE   0[242]                                       ;
                                                        FF  00110 P.AAD:  .BYTE   -1                                           ;

                                                                DELIMITER_TABLE=    P.AAC


                                                                .PSECT  $CODE$,NOWRT,2

                                                            00FC 00000          .ENTRY  ANL$STREAM_RECORD_LENGTH, Save R2,R3,R4,R5,-; 0960
                                                                                        R6,R7
                                                57      0000G   CF   9E 00002          MOVAB   ANL$BUCKET, R7                  ;
                                                5E              18   C2 00007          SUBL2   #24, SP                         ;
                                                56        04    AC   D0 0000A          MOVL    REC_BSD, R6                     ; 0963
      18                  00                    6E              00   2C 0000E          MOVC5   #0, (SP), #0, #24, R            ; 0987
                                                                6E       00013
                                                6E              66   7D 00014          MOVQ    (R6), T                         ; 0988
                                          08    AE        08    A6   D0 00017          MOVL    8(R6), T+8                      ;
                                          14    AE        14    A6   D0 0001C          MOVL    20(R6), T+20                    ;
                                                          7E    D4   00021            CLRL    -(SP)                           ;
                                                          04    AE   9F 00023          PUSHAB  T                               ;
                                                67              02   FB 00026          CALLS   #2, ANL$BUCKET                  ;
                                                50        08    AE   D0 00029 1$:      MOVL    R+8, R0                         ; 0998
                                                50      FE00    C0   9E 0002D          MOVAB   -512(R0), R0                    ;
                                                50              50   CE 00032          MNEGL   R0, R0                          ;
                                          52    0C    AE        08   AE   C1 00035    ADDL3   R+8, R+12, R2                   ;
    0000'   CF      0000'   CF                62              50   2A 0003B          SCANC   R0, (R2), DELIMITER_TABLE, P.AAD ; 0999
                                                                02   12 00044          BNEQ    2$                              ;
                                                                51   D4 00046          CLRL    R1                              ;
                                                54              51   D0 00048 2$:      MOVL    R1, RP                          ; 0998
                                                                37   12 0004B          BNEQ    4$                              ; 1001
                                                50      0000G   CF   D0 0004D          MOVL    ANL$GL_FAT, R0                  ; 1007
                                          08    A0        04    AE   D1 00052          CMPL    R+4, 8(R0)                      ;
                                                                1B   12 00057          BNEQ    3$                              ;
                                                          04    A6   DD 00059          PUSHL   4(R6)                           ; 1008
                                                00000000G       8F   DD 0005C          PUSHL   #ANLRMS$_BADSTREAMEOF           ;
                                            0000G   CF          02   FB 00062          CALLS   #2, ANL$FORMAT_ERROR            ;
                                                00000000G       8F   DD 00067          PUSHL   #ANLRMS$_UNWIND                 ; 1009
                                      00000000G     00          01   FB 0006D          CALLS   #1, LIB$SIGNAL                  ;
                                                          04    AE   D6 00074 3$:      INCL    R+4                             ; 1015
                                                          08    AE   D4 00077          CLRL    R+8                             ; 1016
                                                          7E    D4   0007A            CLRL    -(SP)                           ; 1017
                                                          04    AE   9F 0007C          PUSHAB  R                               ;
                                                67              02   FB 0007F          CALLS   #2, ANL$BUCKET                  ;
                                                                41   11 00082          BRB     11$                             ; 1001
50      0000G   DF                              04              00   EF 00084 4$:      EXTZV   #0, #4, @ANL$GL_FAT, R0         ; 1025
                                                04              50   D1 00088          CMPL    R0, #4                          ; 1026
                                                                09   12 0008E          BNEQ    5$                              ;
                                                                50   D4 00090          CLRL    R0                              ;
                                                0D              64   91 00092          CMPB    (RP), #13                       ;
                                                                21   13 00095          BEQL    10$                             ;
                                                                1D   11 00097          BRB     9$                              ;
                                                05              50   D1 00099 5$:      CMPL    R0, #5                          ; 1027
```

```
                                 07  12 0009C          BNEQ    6$                                      ;
                                 50  D4 0009E          CLRL    R0                                      ;
                           0A    64  91 000A0          CMPB    (RP), #10                               ;
                                 0F  11 000A3          BRB     8$                                      ;
                           06    50  D1 000A5 6$:      CMPL    R0, #6                                  ; 1028
                                 05  13 000A8          BEQL    7$                                      ;
                           50    01  CE 000AA          MNEGL   #1, R0                                  ;
                                 09  11 000AD          BRB     10$                                     ;
                                 50  D4 000AF 7$:      CLRL    R0                                      ;
                           0D    64  91 000B1          CMPB    (RP), #13                               ;
                                 02  12 000B4 8$:      BNEQ    10$                                     ;
                                 50  D6 000B6 9$:      INCL    R0                                      ;
                           55    50  90 000B8 10$:     MOVB    R0, FOUND                               ; 1025
                     50    54    0C  AE  C3 000BB      SUBL3   R+12, RP, R0                            ; 1035
                     08    AE    01  A0  9E 000C0      MOVAB   1(R0), R+8                              ;
                           03    55  E8 000C5 11$:     BLBS    FOUND, 12$                              ; 1038
                                 FF5E  31 000C8        BRW     1$                                      ;
                           7E    01  CE 000CB 12$:     MNEGL   #1, -(SP)                               ; 1040
                           04    AE  9F 000CE          PUSHAB  R                                       ;
                           67    02  FB 000D1          CALLS   #2, ANL$BUCKET                          ;
                     51    04    AE  09  78 000D4      ASHL    #9, R+4, R1                             ; 1045
                           51    08  AE  C0 000D9      ADDL2   R+8, R1                                 ;
                     50    04    A6  09  78 000DD      ASHL    #9, 4(R6), R0                           ;
                           50    08  A6  C0 000E2      ADDL2   8(R6), R0                               ;
                           51        50  C2 000E6      SUBL2   R0, R1                                  ;
                           50        51  D0 000E9      MOVL    R1, R0                                  ;
                                     04 000EC          RET                                             ; 1047

; Routine Size:  237 bytes,    Routine Base:  $CODE$ + 033D
```

```
;  548          1048  1 %sbttl 'ANL$REL_CHECK - Relative File Check'
;  549          1049  1 !++
;  550          1050  1 ! Functional Description:
;  551          1051  1 !       This routine is responsible for performing a check of the structure
;  552          1052  1 !       of a relative file.
;  553          1053  1 !
;  554          1054  1 ! Formal Parameters:
;  555          1055  1 !       none
;  556          1056  1 !
;  557          1057  1 ! Implicit Inputs:
;  558          1058  1 !       global data
;  559          1059  1 !
;  560          1060  1 ! Implicit Outputs:
;  561          1061  1 !       global data
;  562          1062  1 !
;  563          1063  1 ! Returned Value:
;  564          1064  1 !
;  565          1065  1 !
;  566          1066  1 ! Side Effects:
;  567          1067  1 !
;  568          1068  1 !--
;  569          1069  1
;  570          1070  1
;  571          1071  2 global routine anl$rel_check: novalue = begin
;  572          1072  2
;  573          1073  2 local
;  574          1074  2         status: long,
;  575          1075  2         b: bsd,
;  576          1076  2         pp: ref block[,byte];
;  577          1077  2
;  578          1078  2
;  579          1079  2 ! Establish the condition handler for drastic structure errors.
;  580          1080  2
;  581          1081  2 lib$establish(anl$unwind_handler);
;  582          1082  2
;  583          1083  2 ! First call a routine to check the checksum in the prolog block.
;  584          1084  2
;  585          1085  2 anl$prolog_checksums();
;  586          1086  2
;  587          1087  2 ! Read in the prolog block and check the fields relevent to relative files.
;  588          1088  2
;  589          1089  2 init_bsd(b);
;  590          1090  2 b[bsd$w_size] = 1;
;  591          1091  2 b[bsd$l_vbn] = 1;
;  592          1092  2 anl$bucket(b,0);
;  593          1093  2 pp = .b[bsd$l_bufptr];
;  594          1094  2
;  595          1095  2 anl$format_skip(0);
;  596          1096  2 anl$format_skip(0);
;  597          1097  2 anl$rel_prolog(b,true,0);
;  598          1098  2
;  599          1099  2 ! If we are running in /SUMMARY mode, then the user is not interested in
;  600          1100  2 ! spending the time to read through the entire file.
;  601          1101  2
;  602          1102  2 if .anl$gb_mode eqlu anl$k_summary then
;  603          1103  2         return;
;  604          1104  2
```

```
;  605          1105  2 ! Now let's read in the first data bucket and check all cells.
;  606          1106  2 ! There are no cells if the file isn't big enough for even one bucket.
;  607          1107  2
;  608          1108  3 if .anl$gl_fat[fat$l_hiblk]-1 gequ .anl$gl_fat[fat$b_bktsize] then (
;  609          1109  3         b[bsd$w_size] = .anl$gl_fat[fat$b_bktsize];
;  610          1110  3         b[bsd$l_vbn] = .pp[plg$w_dvbn];
;  611          1111  3         anl$bucket(b,0);
;  612          1112  3
;  613          1113  3         while anl$rel_cell(b,false) do;
;  614          1114  2 );
;  615          1115  2
;  616          1116  2 anl$bucket(b,-1);
;  617          1117  2
;  618          1118  2 return;
;  619          1119  2
;  620          1120  1 end;
```

```
                                                  007C 00000          .ENTRY  ANL$REL_CHECK, Save R2,R3,R4,R5,R6      ; 1071
                                  56      0000G   CF   9E 00002       MOVAB   ANL$BUCKET, R6                          ;
                                  5E              18   C2 00007       SUBL2   #24, SP                                 ;
                                          0000G   CF   9F 0000A       PUSHAB  ANL$UNWIND_HANDLER                      ; 1081
                              00000000G   00          01   FB 0000E   CALLS   #1, LIB$ESTABLISH                      ;
                                  0000V   CF          00   FB 00015   CALLS   #0, ANL$PROLOG_CHECKSUMS                ; 1085
                18                00      6E          00   2C 0001A   MOVC5   #0, (SP), #0, #24, B                    ; 1089
                                                      6E       0001F                                                  ;
                                      02  AE          01   B0 00020   MOVW    #1, B+2                                 ; 1090
                                      04  AE          01   D0 00024   MOVL    #1, B+4                                 ; 1091
                                                      7E   D4 00028   CLRL    -(SP)                                   ; 1092
                                              04      AE   9F 0002A   PUSHAB  B                                       ;
                                          66          02   FB 0002D   CALLS   #2, ANL$BUCKET                         ;
                                          52      0C  AE   D0 00030   MOVL    B+12, PP                                ; 1093
                                                      7E   D4 00034   CLRL    -(SP)                                   ; 1095
                                  0000G   CF          01   FB 00036   CALLS   #1, ANL$FORMAT_SKIP                     ;
                                                      7E   D4 0003B   CLRL    -(SP)                                   ; 1096
                                  0000G   CF          01   FB 0003D   CALLS   #1, ANL$FORMAT_SKIP                     ;
                                          7E          01   7D 00042   MOVQ    #1, -(SP)                               ; 1097
                                              08      AE   9F 00045   PUSHAB  B                                       ;
                                  0000V   CF          03   FB 00048   CALLS   #3, ANL$RE_PROLOG                       ;
                                          05      0000G   CF   91 0004D   CMPB    ANL$GB_MODE, #5                     ; 1102
                                                          3A   13 00052   BEQL    3$                                  ;
                                          50      0000G   CF   D0 00054   MOVL    ANL$GL_FAT, R0                      ; 1108
                                      51      04  A0       01   C3 00059   SUBL3   #1, 4(R0), R1                      ;
                51            0E    A0          08       00   ED 0005E   CMPZV   #0, #8, 14(R0), R1                  ;
                                                          1F   1A 00064   BGTRU   2$                                  ;
                                      02  AE      0E  A0   9B   00066   MOVZBW  14(R0), B+2                          ; 1109
                                      04  AE      68  A2   3C   00068   MOVZWL  104(PP), B+4                         ; 1110
                                                          7E   D4 00070   CLRL    -(SP)                               ; 1111
                                              04      AE   9F 00072   PUSHAB  B                                       ;
                                          66          02   FB 00075   CALLS   #2, ANL$BUCKET                         ;
                                                          7E   D4 00078 1$: CLRL    -(SP)                           ; 1113
                                              04      AE   9F 0007A   PUSHAB  B                                       ;
                                  0000V   CF          02   FB 0007D   CALLS   #2, ANL$REL_CELL                        ;
                                          F3              50   E8 00082   BLBS    R0, 1$                              ;
```

```
                 7E              01  CE 00085 2$:    MNEGL   #1, -(SP)                          ; 1116
                         04      AE  9F 00088        PUSHAB  B                                  ;
                 66              02  FB 0008B        CALLS   #2, ANL$BUCKE                      ;
                                     04 0008E 3$:    RET                                        ; 1120

; Routine Size:  143 bytes,    Routine Base:  $CODE$ + 042A
```

```
;  622          1121  1 %sbttl 'ANL$REL_PROLOG - Format and Check a Relative File Prolog'
;  623          1122  1 !++
;  624          1123  1 ! Functional Description:
;  625          1124  1 !       This routine is responsible for formatting and checkirg the pro.og
;  626          1125  1 !       of a relative file.  This is fairly simple.
;  627          1126  1 !
;  628          1127  1 ! Formal Parameters:
;  629          1128  1 !       prolog_bsd      A BSD describing the prolog.
;  630          1129  1 !       report          A boolean, true if we are to format a report.
;  631          1130  1 !       indent_level    The indentation level of the report.
;  632          1131  1 !
;  633          1132  1 ! Implicit Inputs:
;  634          1133  1 !       global data
;  635          1134  1 !
;  636          1135  1 ! Implicit Outputs:
;  637          1136  1 !       global data
;  638          1137  1 !
;  639          1138  1 ! Returned Value:
;  640          1139  1 !       none
;  641          1140  1 !
;  642          1141  1 ! Side Effects:
;  643          1142  1 !
;  644          1143  1 !--
;  645          1144  1
;  646          1145  1
;  647          1146  2 global routine anl$rel_prolog(prolog_bsd,report,indent_level): novalue = begin
;  648          1147  2
;  649          1148  2 bind
;  650          1149  2         p = .prolog_bsd: bsd;
;  651          1150  2
;  652          1151  2 own
;  653          1152  2         flags_def: vector[2,long] initial(
;  654          1153  2                                 0,
;  655          1154  2                                 uplit byte (%ascic 'PLG$V_NOEXTEND')
;  656          1155  2                                 );
;  657          1156  2
;  658          1157  2 local
;  659          1158  2         sp: ref block[,byte];
;  660          1159  2
;  661          1160  2
;  662          1161  2 ! We can start right off and format the prolog if requested.  Begin with
;  663          1162  2 ! a nice heading.
;  664          1163  2
;  665          1164  2 sp = .p[bsd$l_bufptr] + .p[bsd$l_offset];
;  666          1165  3 if .report then (
;  667          1166  3         anl$format_line(3,.indent_level,anlrms$_relprolog);
;  668          1167  3         anl$format_skip(0);
;  669          1168  3
;  670          1169  3         ! Format the flags.
;  671          1170  3
;  672          1171  3         anl$format_flags(.indent_level+1,anlrms$_prologflags,.sp[plg$b_flags],flags_def);
;  673          1172  3
;  674          1173  3         ! Format the first data bucket VBN.
;  675          1174  3
;  676          1175  3         anl$format_line(0,.indent_level+1,anlrms$_databktvbn,.sp[plg$w_dvbn]);
;  677          1176  3
;  678          1177  3         ! Format the maximum record number.
```

```
;  679          1178  3
;  680          1179  3          anl$format_line(0,.indent_level+1,anlrms$_relmaxrec,.sp[plg$l_mrn]);
;  681          1180  3
;  682          1181  3          ! Format the end-of-file VBN.
;  683          1182  3
;  684          1183  3          anl$format_line(0,.indent_level+1,anlrms$_releofvbn,.sp[plg$l_eof]);
;  685          1184  3
;  686          1185  3          ! Format the prolog version number.
;  687          1186  3
;  688          1187  3          anl$format_line(0,.indent_level+1,anlrms$_prologver,.sp[plg$w_ver_no]);
;  689          1188  2 );
```

```
;  691          1189  2 ! Now we can check the prolog.
;  692          1190  2 ! Make sure the flag bits are valid.
;  693          1191  2
;  694          1192  2 anl$check_flags(.p[bsd$l_vbn],.sp[plg$b_flags],flags_def);
;  695          1193  2
;  696          1194  2 ! If an extension error occurred, tell the user.
;  697          1195  2
;  698          1196  2 if .sp[plg$v_noextend] then
;  699          1197  2         anl$format_error(anlrms$_extenderr);
;  700          1198  2
;  701          1199  2 return;
;  702          1200  2
;  703          1201  1 end;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2
44 4E 45 54 58 45 4F 4E 5F 56 24 47 4C 50 0E  00111 P.AAE:    .ASCII  <14>\PLG$V_NOEXTEND\                    ;
                                                                .PSECT  $OWN$,NOEXE,2
                                              00000000  00000 FLAGS_DEF:
                                                                .LONG   0                                       ;
                                              00000000' 00004           .ADDRESS P.AAE                        ;


                                                                .PSECT  $CODE$,NOWRT,2
                                                  003C 00000            .ENTRY  ANL$REL_PROLOG, Save R2,R3,R4,R5        ; 1146
                                           55       0000G  CF   9E 00002 MOVAB   ANL$FORMAT_LINE, R5
                                           53         04   AC   D0 00007 MOVL    PROLOG_BSD, R3                  ; 1149
                                52         0C   A3         08   A3   C1 0000B ADDL3   8(R3), 12(R3), SP        ; 1164
                                           71         08   AC   E9 00011 BLBC    REPORT, 1$                      ; 1165
                                              00000000G  8F   DD 00015 PUSHL   #ANLRMS$_RELPROLOG              ; 1166
                                                   0C    AC   DD 0001B PUSHL   INDENT_LEVEL
                                                         03   DD 0001E PUSHL   #3
                                           65            03   FB 00020 CALLS   #3, ANL$FORMAT_LINE
                                                         7E   D4 00023 CLRL    -(SP)                           ; 1167
                                        00UUG  CF        01   FB 00025 CALLS   #1, ANL$FORMAT_SKIP
                                                   0000'  CF   9F 0002A PUSHAB  FLAGS_DEF                       ; 1171
                                           7E        10   A2   9A 0002E MOVZBL  16(SP), -(SP)
                                              00000000G  8F   DD 00032 PUSHL   #ANLRMS$_PROLOGFLAGS
                                54         0C   AC         01   C1   00038 ADDL3   #1, INDENT_LEVEL, R4
                                                         54   DD 0003D PUSHL   R4
                                        0000G  CF        04   FB 0003F CALLS   #4, ANL$FORMAT_FLAGS
                                           7E        68   A2   3C 00044 MOVZWL  104(SP), -(SP)                  ; 1175
                                              00000000G  8F   DD 00048 PUSHL   #ANLRMS$_DATABKTVBN
                                                         54   DD 0004E PUSHL   R4
                                                         7E   D4 00050 CLRL    -(SP)
                                           65            04   FB 00052 CALLS   #4, ANL$FORMAT_LINE
                                                   6C    A2   DD 00055 PUSHL   108(SP)                         ; 1179
                                              00000000G  8F   DD 00058 PUSHL   #ANLRMS$_RELMAXREC
                                                         54   DD 0005E PUSHL   R4
                                                         7E   D4 00060 CLRL    -(SP)
                                           65            04   FB 00062 CALLS   #4, ANL$FORMAT_LINE
```

```
                                    70   A2  DD 00065          PUSHL   112(SP)                                ; 1183
                           00000000G     8F  DD 00068          PUSHL   #ANLRMS$_RELEOFVBN                    ;
                                         54  DD 0006E          PUSHL   R4                                    ;
                                         7E  D4 00070          CLRL    -(SP)                                 ;
                                65       04  FB 00072          CALLS   #4, ANL$FORMAT_LINE                   ;
                                7E       74  A2  3C 00075          MOVZWL  116(SP), -(SP)                    ; 1187
                           00000000G     8F  DD 00079          PUSHL   #ANLRMS$_PROLOGVER                    ;
                                         54  DD 0007F          PUSHL   R4                                    ;
                                         7E  D4 00081          CLRL    -(SP)                                 ;
                                65       04  FB 00083          CALLS   #4, ANL$FORMAT_LINE                   ;
                                    0000'    CF  9F 00086 1$:       PUSHAB  FLAGS_DEF                         ; 1192
                                7E       10  A2  9A 0008A          MOVZBL  16(SP), -(SP)                     ;
                                         04  A3  DD 0008E          PUSHL   4(R3)                             ;
                           0000G CF              03  FB 00091          CALLS   #3, ANL$CHECK_FLAGS               ;
                                0B       10  A2  E9 00096          BLBC    16(SP), 2$                        ; 1196
                           00000000G     8F  DD 0009A          PUSHL   #ANLRMS$_EXTENDERR                    ; 1197
                           0000G CF              01  FB 000A0          CALLS   #1, ANL$FORMAT_ERROR              ;
                                                 04  000A5 2$:        RET                                       ; 1201

; Routine Size:  166 bytes,    Routine Base:  $CODE$ + 04B9
```

```
;  705          1202  1 %sbttl 'ANL$REL_CELL - Format and Check a Relative File Cell'
;  706          1203  1 !++
;  707          1204  1 ! Functional Description:
;  708          1205  1 !       This routine is responsible for producing a report and checking the
;  709          1206  1 !       contents of a record cell in a relative file.
;  710          1207  1 !
;  711          1208  1 ! Formal Parameters:
;  712          1209  1 !       cell_bsd        Address of a BSD describing the cell.
;  713          1210  1 !                       We update it to describe the following cell.
;  714          1211  1 !       report          A boolean, true if we are to produce a report.
;  715          1212  1 !       indent_level    The indentation level of the report.
;  716          1213  1 !
;  717          1214  1 ! Implicit Inputs:
;  718          1215  1 !       global data
;  719          1216  1 !
;  720          1217  1 ! Implicit Outputs:
;  721          1218  1 !       global data
;  722          1219  1 !
;  723          1220  1 ! Returned Value:
;  724          1221  1 !       True if there is another cell, false if not.
;  725          1222  1 !
;  726          1223  1 ! Side Effects:
;  727          1224  1 !
;  728          1225  1 !--
;  729          1226  1
;  730          1227  1
;  731          1228  2 global routine anl$rel_cell(cell_bsd,report,indent_level) = begin
;  732          1229  2
;  733          1230  2 bind
;  734          1231  2         c = .cell_bsd: bsd;
;  735          1232  2
;  736          1233  2 own
;  737          1234  2         cell_flags_def: vector[5,long] initial(
;  738          1235  2                                                3,
;  739          1236  2                                                0,
;  740          1237  2                                                0,
;  741          1238  2                                                uplit byte (%ascic 'DLC$V_DELETED'),
;  742          1239  2                                                uplit byte (%ascic 'DLC$V_REC')
;  743          1240  2                                                );
;  744          1241  2
;  745          1242  2 local
;  746          1243  2         sp: ref block[,byte],
;  747          1244  2         data_length: long, overhead: long, length: long;
;  748          1245  2
;  749          1246  2
;  750          1247  2 ! We begin by calculating two lengths.  The first is the length of only the
;  751          1248  2 ! actual data.  The second is the overall length of the cell.
;  752          1249  2
;  753          1250  2 sp = .c[bsd$l_bufptr] + .c[bsd$l_offset];
;  754          1251  3 data_length = (selectoneu .anl$gl_fat[fat$v_rtype] of set
;  755          1252  3                 [fat$c_fixed]:   .anl$gl_fat[fat$w_maxrec];
;  756          1253  3
;  757          1254  3                 [fat$c_variable,
;  758          1255  3                  fat$c_vfc]:     .sp[1,0,16,0];
;  759          1256  2                 tes);
;  760          1257  2 overhead = 1 +
;  761          1258  2             (if .anl$gl_fat[fat$v_rtype] eqlu fat$c_fixed then 0 else 2);
```

```
;  762          1259  2 length = .overhead +
;  763          1260  2           (if .anl$gl_fat[fat$v_rtype] eqlu fat$c_vfc then .anl$gl_fat[fat$b_vfcsize] else 0) +
;  764          1261  2           .anl$gl_fat[fat$w_maxrec];
;  765          1262  2
;  766          1263  2 ! We will assume that the cell fits in the bucket.  Otherwise we would have
;  767          1264  2 ! gone on to the next bucket last time.
```

```
;  769          1265  2 ! Now we can format the cell if requested.  This will begin with a heading
;  770          1266  2 ! and the flag byte.
;  771          1267  2
;  772          1268  3 if .report then (
;  773          1269  3         anl$format_line(4,.indent_level,anlrms$_cell,.c[bsd$l_vbn],.c[bsd$l_offset]);
;  774          1270  3         anl$format_skip(0);
;  775          1271  3         anl$format_flags(.indent_level+1,anlrms$_cellflags,.sp[0,0,8,0],cell_flags_def);
;  776          1272  3
;  777          1273  3         ! Now we can dump the record with a nice heading.  Only can do if the
;  778          1274  3         ! record is present and not deleted.
;  779          1275  3
;  780          1276  4         if .sp[dlc$v_rec] and not .sp[dlc$v_deleted] then (
;  781          1277  4                 local
;  782          1278  4                         data_dsc: descriptor;
;  783          1279  4
;  784          1280  4                 anl$format_line(0,.indent_level+1,anlrms$_celldata);
;  785        P 1281  4                 build_descriptor(data_dsc,
;  786        P 1282  4                                  (if .anl$gl_fat[fat$v_rtype] eqlu fat$c_fixed then 0 else 2) +
;  787        P 1283  4                                  .data_length,
;  788          1284  4                                  .sp + 1);
;  789          1285  4                 anl$format_hex(.indent_level+2,data_dsc);
;  790          1286  3         );
;  791          1287  2 );
```

```
;  793          1288  2 ! OK, now we have a few checks to make.  Start with the flags.
;  794          1289  2
;  795          1290  2 anl$check_flags(.c[bsd$l_vbn],.sp[0,0,8,0],cell_flags_def);
;  796          1291  2
;  797          1292  2 ! Make sure that the data fits within the fixed cell size.
;  798          1293  2
;  799          1294  2 if .data_length gtru .length-.overhead then
;  800          1295  2         anl$format_error(anlrms$_badcellfit,.c[bsd$l_vbn]);
;  801          1296  2
;  802          1297  2 ! Make sure that a VFC record is large enough to contain the header.
;  803          1298  2
;  804          1299  2 if .c[dlc$v_rec] and not .c[dlc$v_deleted] then
;  805          1300  2         if .anl$gl_fat[fat$v_rtype] eqlu fat$c_vfc and
;  806          1301  2            .data_length lssu .anl$gl_fat[fat$b_vfcsize] then
;  807          1302  2                 anl$format_error(anlrms$_vfctooshort,.c[bsd$l_vbn]);
;  808          1303  2
;  809          1304  2 ! If another cell can fit in this bucket, then update the BSD to describe it.
;  810          1305  2
;  811          1306  3 if .sp + 2*.length lequa .c[bsd$l_endptr] then (
;  812          1307  3         c[bsd$l_offset] = .c[bsd$l_offset] + .length;
;  813          1308  3         return true;
;  814          1309  2 );
;  815          1310  2
;  816          1311  2 ! If there are more buckets, advance to the first cell in the next one.
;  817          1312  2 ! Otherwise we're done.  When we opened the file, we placed the
;  818          1313  2 ! end-of-file VBN in the FAT block, like for a sequential file.
;  819          1314  2
;  820          1315  3 if .c[bsd$l_vbn] + 2*.c[bsd$w_size] lequ .anl$gl_fat[fat$l_efblk] then (
;  821          1316  3         c[bsd$l_vbn] = .c[bsd$l_vbn] + .c[bsd$w_size];
;  822          1317  3         c[bsd$l_offset] = 0;
;  823          1318  3         anl$bucket(c,0);
;  824          1319  3         return true;
;  825          1320  2 ) else
;  826          1321  2         return false;
;  827          1322  2
;  828          1323  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

    44  45  54  45  4C  45  44  5F  56  24  43  4C  44  0D  00120 P.AAF:  .ASCII  <13>\DLC$V_DELETED\                          ;
                    43  45  52  5F  56  24  43  4C  44  09  0012E P.AAG:  .ASCII  <9>\DLC$V_REC\                              ;

                                                        .PSECT  $OWN$,NOEXE,2

                            00000000  00000000  00000003  00008 CELL_FLAGS_DEF:
                                                        .LONG   3, 0, 0
                                      00000000' 00000000' 00014         .ADDRESS P.AAF, P.AAG                               ;


                                                        .PSECT  $CODE$,NOWRT,2

                                                  01FC 00000         .ENTRY  ANL$REL_CELL, Save R2,R3,R4,R5,R6,R7,R8      ; 1228
                                      58      0000G CF  9E 00002         MOVAB   ANL$GL_FAT, R8                              ;
                                      5E            08  C2 00007         SUBL2   #8, SP                                      ;
```

```
                              56          04  AC  D0 0000A          MOVL    CELL_BSD, R6                        ; 1231
              52        0C    A6          08  A6  C1 0000E          ADDL3   8(R6), 12(R6), SP                   ; 1250
                              50              68  D0 00014          MOVL    ANL$GL_FAT, R0                      ; 1251
     01       60              04              00  ED 00017          CMPZV   #0, #4, (R0), #1                    ; 1252
                                              06  12 0001C          BNEQ    1$                                  ;
                              55          10  A0  3C 0001E          MOVZWL  16(R0), DATA_LENGTH                 ;
                                              17  11 00022          BRB     4$                                  ;
     02       60              04              00  ED 00024 1$:      CMPZV   #0, #4, (R0), #2                    ; 1254
                                              07  1F 00029          BLSSU   2$                                  ;
     03       60              04              00  ED 0002B          CMPZV   #0, #4, (R0), #3                    ;
                                              05  1B 00030          BLEQU   3$                                  ;
                              55              01  CE 00032 2$:      MNEGL   #1, DATA_LENGTH                     ;
                                              04  11 00035          BRB     4$                                  ;
                              55          01  A2  3C 00037 3$:      MOVZWL  1(SP), DATA_LENGTH                  ; 1255
     01       60              04              00  ED 0003B 4$:      CMPZV   #0, #4, (R0), #1                    ; 1258
                                              04  12 00040          BNEQ    5$                                  ;
                                              54  D4 00042          CLRL    R4                                  ;
                                              03  11 00044          BRB     6$                                  ;
                              54              02  D0 00046 5$:      MOVL    #2, R4                              ;
                                              54  D6 00049 6$:      INCL    OVERHEAD                            ; 1257
     03       60              04              00  ED 0004B          CMPZV   #0, #4, (R0), #3                    ; 1260
                                              06  12 00050          BNEQ    7$                                  ;
                              51          0F  A0  9A 00052          MOVZBL  15(R0), R1                          ;
                                              02  11 00056          BRB     8$                                  ;
                                              51  D4 00058 7$:      CLRL    R1                                  ;
                              51              54  C0 0005A 8$:      ADDL2   OVERHEAD, R1                        ; 1259
                              53          10  A0  3C 0005D          MOVZWL  16(R0), LENGTH                      ; 1261
                              53              51  C0 00061          ADDL2   R1, LENGTH                          ;
                              6F          08  AC  E9 00064          BLBC    REPORT, 11$                         ; 1268
                              7E          04  A6  7D 00068          MOVQ    4(R6), -(SP)                        ; 1269
                                   00000000G  8F  DD 0006C          PUSHL   #ANLRMS$_CELL                       ;
                                          0C  AC  DD 00072          PUSHL   INDENT_LEVEL                        ;
                                              04  DD 00075          PUSHL   #4                                  ;
                        0000G  CF             05  FB 00077          CALLS   #5, ANL$FORMAT_LINE                 ;
                                              7E  D4 0007C          CLRL    -(SP)                               ; 1270
                        0000G  CF             01  FB 0007E          CALLS   #1, ANL$FORMAT_SKIP                 ;
                                       0000'  CF  9F 00083          PUSHAB  CELL_FLAGS_DEF                      ; 1271
                              7E              62  9A 00087          MOVZBL  (SP), -(SP)                         ;
                                   00000000G  8F  DD 0008A          PUSHL   #ANLRMS$_CELLFLAGS                  ;
              57        0C    AC              01  C1 00090          ADDL3   #1, INDENT_LEVEL, R7                ;
                                              57  DD 00095          PUSHL   R7                                  ;
                        0000G  CF             04  FB 00097          CALLS   #4, ANL$FORMAT_FLAGS                ;
              37              62              03  E1 0009C          BBC     #3, (SP), 11$                       ; 1276
              33              62              02  E0 000A0          BBS     #2, (SP), 11$                       ;
                                   00000000G  8F  DD 000A4          PUSHL   #ANLRMS$_CELLDATA                   ; 1280
                                              57  DD 000AA          PUSHL   R7                                  ;
                                              7E  D4 000AC          CLRL    -(SP)                               ;
                        0000G  CF             03  FB 000AE          CALLS   #3, ANL$FORMAT_LINE                 ;
     01        00   B8        04              00  ED 000B3          CMPZV   #0, #4, @ANL$GL_FAT, #1             ; 1284
                                              04  12 000B9          BNEQ    9$                                  ;
                                              50  D4 000BB          CLRL    R0                                  ;
                                              03  11 000BD          BRB     10$                                 ;
                              50              02  D0 000BF 9$:      MOVL    #2, R0                              ;
              6E              50              55  C1 000C2 10$:     ADDL3   DATA_LENGTH, R0, DATA_DSC           ;
                        04    AE          01  A2  9E 000C6          MOVAB   1(R2), DATA_DSC+4                   ;
                                              5E  DD 000CB          PUSHL   SP                                  ; 1285
              7E        0C    AC              02  C1 000CD          ADDL3   #2, INDENT_LEVEL, -(SP)             ;
```

```
                                0000G  CF              02  FB 000D2              CALLS   #2, ANL$FORMAT_HEX
                                     0000'  CF          9F  000D7 11$:           PUSHAB  CELL_FLAGS_DEF                  ;1290
                                7E                      62  9A 000DB            MOVZBL  (SP), -(SP)
                                   04                   A6  DD 000DE            PUSHL   4(R6)
                                0000G  CF               03  FB 000E1            CALLS   #3, ANL$CHECK_FLAGS
                          50                            53  C3 000E6            SUBL3   OVERHEAD, LENGTH, R0            ;1294
                                                        50  D1 000EA            CMPL    DATA_LENGTH, R0
                                                        0E  1B 000ED            BLEQU   12$
                                   04                   A6  DD 000EF            PUSHL   4(R6)                           ;1295
                                000000000G              8F  DD 000F2            PUSHL   #ANLRMS$_BADCELLFIT
                                0000G  CF               02  FB 000F8            CALLS   #2, ANL$FORMAT_ERROR
                          24                            66  03  E1 000FD 12$:   BBC     #3, (R6), 13$                   ;1299
                          20                            66  02  E0 00101        BBS     #2, (R6), 13$
                                                        50  68  D0 00105        MOVL    ANL$GL_FAT, R0                  ;1300
                    03                  60              04  00  ED 00108        CMPZV   #0, #4, (R0), #3
                                                            16  12 0010D        BNEQ    13$
                    55          0F      A0              08  00  ED 0010F        CMPZV   #0, #8, 15(R0), DATA_LENGTH     ;1301
                                                            0E  1B 00115        BLEQU   13$
                                   04                   A6  DD 00117            PUSHL   4(R6)                           ;1302
                                000C0000G               8F  DD 0011A            PUSHL   #ANLRMS$_VFCTOOSHORT
                                0000G  CF               02  FB 00120            CALLS   #2, ANL$FORMAT_ERROR
                                                        50  3E 00125 13$:       MOVAW   (SP)[LENGTH], R0                ;1306
                                   10                   A6  D1 00129            CMPL    R0, 16(R6)
                                                            06  1A 0012D        BGTRU   14$
                                   08                   A6  C0 0012F            ADDL2   LENGTH, 8(R6)                   ;1307
                                                            22  11 00133        BRB     15$                             ;1308
                                                        50  3C 00135 14$:       MOVZWL  2(R6), R0                       ;1315
                                                        52  3E 00139            MOVAW   @4(R6)[R0], R2
                                                        51  68  D0 0013E        MOVL    ANL$GL_FAT, R1
                                   08                   A1  52  D1 00141        CMPL    R2, 8(R1)
                                                            14  1A 00145        BGTRU   16$
                                   04                   A6  50  C0 00147        ADDL2   R0, 4(R6)                       ;1316
                                                        08  A6  D4 0014B        CLRL    8(R6)                           ;1317
                                                            7E  D4 0014E        CLRL    -(SP)                           ;1318
                                                            56  DD 00150        PUSHL   R6
                                0000G  CF                   02  FB 00152        CALLS   #2, ANL$BUCKET
                                                        50  01  D0 00157 15$:   MOVL    #1, R0                          ;1321
                                                                04 0015A        RET
                                                            50  D4 0015B 16$:   CLRL    R0
                                                                04 0015D        RET                                     ;1323

; Routine Size:  350 bytes,    Routine Base:  $CODE$ + 055F
```

```
;  830          1324  1 %sbttl 'ANL$PROLOG_CHECKSUMS - Check Prolog Block Checksums'
;  831          1325  1 !++
;  832          1326  1 ! Functional Description:
;  833          1327  1 !       This routine is responsible for checking the checksums that are
;  834          1328  1 !       present at the end of prolog blocks.  This pertains to both relative
;  835          1329  1 !       and indexed files.
;  836          1330  1 !
;  837          1331  1 ! Formal Parameters:
;  838          1332  1 !       none
;  839          1333  1 !
;  840          1334  1 ! Implicit Inputs:
;  841          1335  1 !       global data
;  842          1336  1 !
;  843          1337  1 ! Implicit Outputs:
;  844          1338  1 !       global data
;  845          1339  1 !
;  846          1340  1 ! Returned Value:
;  847          1341  1 !       none
;  848          1342  1 !
;  849          1343  1 ! Side Effects:
;  850          1344  1 !
;  851          1345  1 !--
;  852          1346  1
;  853          1347  1
;  854          1348  2 global routine anl$prolog_checksums: novalue = begin
;  855          1349  2
;  856          1350  2 local
;  857          1351  2         blocks: long,
;  858          1352  2         p :bsd,
;  859          1353  2         pp: ref block[,byte],
;  860          1354  2         vbn: long,
;  861          1355  2         checksum: word, i: long;
;  862          1356  2
;  863          1357  2
;  864          1358  2 ! We will be reading in the prolog blocks as if they were single block buckets.
;  865          1359  2
;  866          1360  2 init_bsd(p);
;  867          1361  2 p[bsd$w_size] = 1;
;  868          1362  2
;  869          1363  2 ! We begin by calculating the number of prolog blocks.
;  870          1364  2
;  871          1365  2 if .anl$gl_fat[fat$v_fileorg] eqlu fat$c_relative then
;  872          1366  2
;  873          1367  2         ! Relative files always have one prolog block.
;  874          1368  2
;  875          1369  2         blocks = 1
;  876          1370  2
;  877          1371  3 else (
;  878          1372  3
;  879          1373  3         ! The number of prolog blocks in an indexed file depends on the
;  880          1374  3         ! number of keys and areas.  We have to read in the first prolog
;  881          1375  3         ! block to determine this.
;  882          1376  3
;  883          1377  3         p[bsd$l_vbn] = 1;
;  884          1378  3         anl$bucket(p,0);
;  885          1379  3         pp = .p[bsd$l_bufptr];
;  886          1380  3
```

```
;  887          1381  3          ! The key descriptors are first, followed by the area descriptors.
;  888          1382  3          ! Thus we can determine the total prolog blocks from the first
;  889          1383  3          ! area descriptor VBN and the number of areas.
;  890          1384  3
;  891          1385  3          blocks = .pp[plg$b_avbn] +
;  892          1386  3                   (.pp[plg$b_amax]-1) / (512/area$c_bln);
;  893          1387  2 );
;  894          1388  2
;  895          1389  2 ! Now we can loop through the prolog blocks and read them in one at a time.
;  896          1390  2
;  897          1391  3 incru vbn from 1 to .blocks do (
;  898          1392  3
;  899          1393  3          p[bsd$l_vbn] = .vbn;
;  900          1394  3          anl$bucket(p,0);
;  901          1395  3
;  902          1396  3          ! Sum up the first 255 words of the block, ignoring overflow.  This is
;  903          1397  3          ! the correct checksum, but if it's wrong, tell the user.
;  904          1398  3
;  905          1399  4          begin
;  906          1400  4          bind
;  907          1401  4                  word_vector = .p[bsd$l_bufptr]: vector[256,word];
;  908          1402  4
;  909          1403  4          checksum = 0;
;  910          1404  4          incru i from 0 to 254 do
;  911          1405  4                  checksum = .checksum + .word_vector[.i];
;  912          1406  4
;  913          1407  4          if .checksum nequ .word_vector[255] then
;  914          1408  4                  anl$format_error(anlrms$_badchecksum,.vbn);
;  915          1409  3          end;
;  916          1410  2 );
;  917          1411  2
;  918          1412  2 anl$bucket(p,-1);
;  919          1413  2 return;
;  920          1414  2
;  921          1415  1 end;
```

```
                                                 007C 00000          .ENTRY  ANL$PROLOG_CHECKSUMS, Save R2,R3,R4,R5,R6    ; 1348
                                          56     0000G CF   9E 00002  MOVAB   ANL$BUCKET, R6                               ;
                                          5E           18   C2 00007  SUBL2   #24, SP                                     ;
                18                00      6E           00   2C 0000A  MOVC5   #0, (SP), #0, #24, P                       ; 1360
                                                       6E        0000F                                                     ;
                                    02    AE           01   B0 00010  MOVW    #1, P+2                                     ; 1361
                01        0000G   DF      04           04   ED 00014  CMPZV   #4, #4, @ANL$GL_FAT, #1                    ; 1365
                                                       05   12 0001B  BNEQ    1$                                          ;
                                          54           01   D0 0001D  MOVL    #1, BLOCKS                                  ; 1369
                                                       20   11 00020  BRB     2$                                          ;
                                    04    AE           01   D0 00022 1$:    MOVL    #1, P+4                             ; 1377
                                                       7E   D4 00026        CLRL    -(SP)                               ; 1378
                                                04     AE   9F 00028        PUSHAB  P                                   ;
                                          66           02   FB 0002B        CALLS   #2, ANL$BUCKET                      ;
                                          50     0C    AE   DC 0002E        MOVL    P+12, PP                            ; 1379
                                          51     67    A0   9A 00032        MOVZBL  103(PP), R1                         ; 1386
                                                       51   D7 00036        DECL    R1                                  ;
```

```
                     51          08  C6 00038          DIVL2   #8, R1
                     54      66  A0  9A 0003B          MOVZBL  102(PP), BLOCKS
                     54          51  C0 0003F          ADDL2   R1, BLOCKS
                     52          01  D0 00042 2$:      MOVL    #1, VBN                 ; 1391
                                 39  11 00045          BRB     6$
                04   AE          52  D0 00047 3$:      MOVL    VBN, P+4                ; 1393
                                 7E  D4 0004B          CLRL    -(SP)                   ; 1394
                         04      AE  9F 0004D          PUSHAB  P
                     66          02  FB 00050          CALLS   #2, ANL$BUCKET
                     51      0C  AE  D0 00053          MOVL    P+12, R1                ; 1401
                                 53  B4 00057          CLRW    CHECKSUM                ; 1403
                                 50  D4 00059          CLRL    I                       ; 1404
                     53      6140  A0 0005B 4$:      ADDW2   (R1)[I], CHECKSUM       ; 1405
                                 50  D6 0005F          INCL    I
           000000FE  8F          50  D1 00061          CMPL    I, #254
                                 F1  1B 00068          BLEQU   4$
               01FE  C1          53  B1 0006A          CMPW    CHECKSUM, 510(R1)       ; 1407
                                 0D  13 0006F          BEQL    5$
                                 52  DD 00071          PUSHL   VBN                     ; 1408
                  00000000G      8F  DD 00073          PUSHL   #ANLRMS$_BADCHECKSUM
               0000G  CF          02  FB 00079          CALLS   #2, ANL$FORMAT_ERROR
                                 52  D6 0007E 5$:      INCL    VBN                     ; 1391
                     54          52  D1 00080 6$:      CMPL    VBN, BLOCKS
                                 C2  1B 00083          BLEQU   3$
                     7E          01  CE 00085          MNEGL   #1, -(SP)               ; 1412
                         04      AE  9F 00088          PUSHAB  P
                     66          02  FB 0008B          CALLS   #2, ANL$BUCKET
                                 04  0008E          RET                             ; 1415

; Routine Size:  143 bytes,    Routine Base:  $CODE$ + 06BD

;  922          1416  1
;  923          1417  0 end eludom

                                                        .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY

;      Name                     Bytes                          Attributes

;  $PLIT$                         312  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        1868  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                           28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                            Library Statistics

;                                    -------- Symbols --------      Pages       Processing
;      File                          Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        41         0        1000        00:01.9
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         1868 code + 340 data bytes
; Run Time:       00:32.1
; Elapsed Time:   01:42.4
; Lines/CPU Min:   2651
; Lexemes/CPU-Min: 16817
; Memory Used:  238 pages
; Compilation Complete
```

RMSINTER
LIS

RMSCHECKA
LIS

RMSFDL
LIS

RMSCHECKB
LIS

RMSINPUT
LIS

RMSMSG
LIS