```
   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
```

FILEID**RMS

```
RRRRRRRR    MM      MM    SSSSSSSS
RRRRRRRR    MM      MM    SSSSSSSS
RR      RR  MMMM  MMMM  SS
RR      RR  MMMM  MMMM  SS
RR      RR  MM  MM  MM  SS
RR      RR  MM  MM  MM  SS
RRRRRRRR    MM      MM    SSSSSS
RRRRRRRR    MM      MM    SSSSSS
RR  RR      MM      MM          SS
RR  RR      MM      MM          SS
RR    RR    MM      MM          SS      ....
RR    RR    MM      MM          SS      ....
RR      RR  MM      MM  SSSSSSSS        ....
RR      RR  MM      MM  SSSSSSSS        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1        0001  0 %title 'RMS - Main Module for ANALYZE/RMS_FILE'
;     2        0002  0         module rms      (main=anl$rms,
;     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, Main Module for ANALYZE/RMS_FILE
;    32        0032  1 !
;    33        0033  1 ! Abstract:     This is the main module for the ANALYZE/RMS_FILE command.
;    34        0034  1 !               It contains the routine that determines which mode we are
;    35        0035  1 !               to operate in.  It also contains routines that don't fit
;    36        0036  1 !               anywhere else.
;    37        0037  1 !
;    38        0038  1 !
;    39        0039  1 ! Environment:
;    40        0040  1 !
;    41        0041  1 ! Author: Paul C. Anagnostopoulos, Creation Date: 18 February 1981
;    42        0042  1 !
;    43        0043  1 ! Modified By:
;    44        0044  1 !
;    45        0045  1 !       V03-003 DGB0046         Donald G. Blair         08-May-1984
;    46        0046  1 !               Fix condition handling for ANALYZRMS so that it
;    47        0047  1 !               returns status correctly upon image exit.  Rather
;    48        0048  1 !               than always return anlrms$_notok, return the first
;    49        0049  1 !               error that occurs of the highest severity.
;    50        0050  1 !
;    51        0051  1 !       V03-002 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
;    52        0052  1 !               Change the message prefix to ANLRMS$_ to ensure that
;    53        0053  1 !               message symbols are unique across all ANALYZEs.  This
;    54        0054  1 !               is necessitated by the new merged message files.
;    55        0055  1 !
;    56        0056  1 !       V03-001 PCA1002         Paul C. Anagnostopoulos 25-Oct-1982
;    57        0057  1 !               Add the ANL$PREPARE_QUOTED_STRING routine to format a
```

```
;    58          0058  1 !                     quoted string for inclusion in an FDL specification.
;    59          0059  1 !                     Add code for /SUMMARY mode.
;    60          0060  1 !--
```

```
;    62          0061  1 %sbttl 'Module Declarations'
;    63          0062  1 !
;    64          0063  1 ! Libraries and Requires:
;    65          0064  1 !
;    66          0065  1
;    67          0066  1 library 'lib';
;    68          0067  1 require 'rmsreq';
;    69          0576  1
;    70          0577  1 !
;    71          0578  1 ! Table of Contents:
;    72          0579  1 !
;    73          0580  1
;    74          0581  1 forward routine
;    75          0582  1         anl$rms: novalue,
;    76          0583  1         anl$unwind_handler,
;    77          0584  1         anl$worst_error_handler,
;    78          0585  1         anl$internalize_number,
;    79          0586  1         anl$check_flags: novalue,
;    80          0587  1         anl$prepare_quoted_string: novalue;
;    81          0588  1
;    82          0589  1 !
;    83          0590  1 ! External References:
;    84          0591  1 !
;    85          0592  1
;    86          0593  1 external routine
;    87          0594  1         anl$check_mode,
;    88          0595  1         anl$fdl_mode,
;    89          0596  1         anl$format_error,
;    90          0597  1         anl$interactive_mode,
;    91          0598  1         cli$present: addressing_mode(general),
;    92          0599  1         lib$establish: addressing_mode(general),
;    93          0600  1         ots$cvt_ti_l: addressing_mode(general),
;    94          0601  1         ots$cvt_tz_l: addressing_mode(general);
;    95          0602  1
;    96          0603  1 !
;    97          0604  1 ! Global Variables:
;    98          0605  1 !
;    99          0606  1
;   100          0607  1 global
;   101          0608  1         anl$gb_mode: byte,      ! current mode of operation
;   102          0609  1         anl$worst_error:        ! this contains either success status or if
;   103          0610  1           initial(anlrms$_ok);  ! errors occurred, it contains the first error
;   104          0611  1                                 ! of the worst severity that occurred.
;   105          0612  1
;   106          0613  1 !
;   107          0614  1 ! Own Variables:
;   108          0615  1 !
;   109          0616  1
```

```
;   111          0617  1 %sbttl 'ANL$RMS - Main Routine for ANALYZE/RMS_FILE'
;   112          0618  1 !++
;   113          0619  1 ! Functional Description:
;   114          0620  1 !       This is the main routine, entered when the user performs an
;   115          0621  1 !       ANALYZE/RMS_FILE command.  We decide which mode of operation
;   116          0622  1 !       has been requested and do it.
;   117          0623  1 !
;   118          0624  1 ! Formal Parameters:
;   119          0625  1 !       none
;   120          0626  1 !
;   121          0627  1 ! Implicit Inputs:
;   122          0628  1 !       global data
;   123          0629  1 !
;   124          0630  1 ! Implicit Outputs:
;   125          0631  1 !       global data
;   126          0632  1 !
;   127          0633  1 ! Returned Value:
;   128          0634  1 !       none
;   129          0635  1 !
;   130          0636  1 ! Side Effects:
;   131          0637  1 !
;   132          0638  1 !--
;   133          0639  1
;   134          0640  1
;   135          0641  2 global routine anl$rms: novalue = begin
;   136          0642  2
;   137          0643  2 lib$establish(anl$worst_error_handler);
;   138          0644  2
;   139          0645  2 ! See which mode the user has requested.  The default is /CHECK.
;   140          0646  2
;   141          0647  3 if cli$present(describe('FDL')) then (
;   142          0648  3         anl$gb_mode = anl$k_fdl;
;   143          0649  3         anl$fdl_mode();
;   144          0650  3 )
;   145          0651  3 else if cli$present(describe('INTERACTIVE')) then (
;   146          0652  3         anl$gb_mode = anl$k_interactive;
;   147          0653  3         anl$interactive_mode();
;   148          0654  3 )
;   149          0655  3 else if cli$present(describe('STATISTICS')) then (
;   150          0656  3         anl$gb_mode = anl$k_statistics;
;   151          0657  3         anl$check_mode();
;   152          0658  3 )
;   153          0659  3 else if cli$present(describe('SUMMARY')) then (
;   154          0660  3         anl$gb_mode = anl$k_summary;
;   155          0661  3         anl$check_mode();
;   156          0662  3 )
;   157          0663  3 else (
;   158          0664  3         anl$gb_mode = anl$k_check;
;   159          0665  3         anl$check_mode();
;   160          0666  2 );
;   161          0667  2
;   162          0668  2 ! if it was an interactive session, always return success.  otherwise
;   163          0669  2 ! return worst error
;   164          0670  2
;   165          0671  2 if .anl$gb_mode eql anl$k_interactive then
;   166          0672  3         $exit(code=anlrms$_ok)
;   167          0673  2 else
```

```
;  168          0674  2          $exit(code=.anl$worst_error or sts$m_inhib_msg);
;  169          0675  2
;  170          0676  1 end;


                                                                .TITLE  RMS RMS - Main Module for ANALYZE/RMS_FILE
                                                                .IDENT  \V04-000\

                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                 4C  44  46  00000 P.AAB:   .ASCII  \FDL\                                    ;
                                                             00003          .BLKB   1
                                                   00000003  00004 P.AAA:   .LONG   3                                        ;
                                                   00000000' 00008          .ADDRESS P.AAB                                  ;
                                45  56  49  54  43  41  52  45  54  4E  49  0000C P.AAD:   .ASCII  \INTERACTIVE\                ;
                                                             00017          .BLKB   1
                                                   0000000B  00018 P.AAC:   .LONG   11                                       ;
                                                   00000000' 0001C          .ADDRESS P.AAD                                  ;
                                    53  43  49  54  53  49  54  41  54  53  00020 P.AAF:   .ASCII  \STATISTICS\               ;
                                                             0002A          .BLKB   2
                                                   0000000A  0002C P.AAE:   .LONG   10                                       ;
                                                   00000000' 00030          .ADDRESS P.AAF                                  ;
                                            59  52  41  4D  4D  55  53  00034 P.AAH:   .ASCII  \SUMMARY\                    ;
                                                             0003B          .BLKB   1
                                                   00000007  0003C P.AAG:   .LONG   7                                        ;
                                                   00000000' 00040          .ADDRESS P.AAH                                  ;

                                                                .PSECT  $GLOBAL$,NOEXE,2

                                                             00000 ANL$GB_MODE::
                                                                           .BLKB   1
                                                             00001         .BLKB   3
                                                   00000000G 00004 ANL$WORST_ERROR::
                                                                           .LONG   ANLRMS$_OK                              ;

                                                                .EXTRN  ANLRMS$_OK, ANLRMS$_ALLOC
                                                                .EXTRN  ANLRMS$_ANYTHING
                                                                .EXTRN  ANLRMS$_BACKUP, ANLRMS$_BKT
                                                                .EXTRN  ANLRMS$_BKTAREA
                                                                .EXTRN  ANLRMS$_BKTCHECK
                                                                .EXTRN  ANLRMS$_BKTFLAGS
                                                                .EXTRN  ANLRMS$_BKTFREE
                                                                .EXTRN  ANLRMS$_BKTKEY, ANLRMS$_BKTLEVEL
                                                                .EXTRN  ANLRMS$_BKTNEXT
                                                                .EXTRN  ANLRMS$_BKTPTRSIZE
                                                                .EXTRN  ANLRMS$_BKTRECID
                                                                .EXTRN  ANLRMS$_BKTRECID3
                                                                .EXTRN  ANLRMS$_BKTSAMPLE
                                                                .EXTRN  ANLRMS$_BKTVBNFREE
                                                                .EXTRN  ANLRMS$_BUCKETSIZE
                                                                .EXTRN  ANLRMS$_CELL, ANLRMS$_CELLDATA
                                                                .EXTRN  ANLRMS$_CELLFLAGS
                                                                .EXTRN  ANLRMS$_CHECKHDG
                                                                .EXTRN  ANLRMS$_CONTIG, ANLRMS$_CREATION
                                                                .EXTRN  ANLRMS$_CTLSIZE
                                                                .EXTRN  ANLRMS$_DATAREC
                                                                .EXTRN  ANLRMS$_DATABKTVBN
```

```
                .EXTRN  ANLRMS$_DUMPHEADING
                .EXTRN  ANLRMS$_EOF, ANLRMS$_ERRORCOUNT
                .EXTRN  ANLRMS$_ERRORNONE
                .EXTRN  ANLRMS$_ERRORS, ANLRMS$_EXPIRATION
                .EXTRN  ANLRMS$_FILEATTR
                .EXTRN  ANLRMS$_FILEHDR
                .EXTRN  ANLRMS$_FILEID, ANLRMS$_FILEORG
                .EXTRN  ANLRMS$_FILESPEC
                .EXTRN  ANLRMS$_FLAG, ANLRMS$_GLOBALBUFS
                .EXTRN  ANLRMS$_HEXDATA
                .EXTRN  ANLRMS$_HEXHEADING1
                .EXTRN  ANLRMS$_HEXHEADING2
                .EXTRN  ANLRMS$_IDXAREA
                .EXTRN  ANLRMS$_IDXAREAALLOC
                .EXTRN  ANLRMS$_IDXAREABKTSZ
                .EXTRN  ANLRMS$_IDXAREANEXT
                .EXTRN  ANLRMS$_IDXAREANOALLOC
                .EXTRN  ANLRMS$_IDXAREAQTY
                .EXTRN  ANLRMS$_IDXARERECL
                .EXTRN  ANLRMS$_IDXAREAUSED
                .EXTRN  ANLRMS$_IDXKEY, ANLRMS$_IDXKEYAREAS
                .EXTRN  ANLRMS$_IDXKEYBKTSZ
                .EXTRN  ANLRMS$_IDXKEYBYTES
                .EXTRN  ANLRMS$_IDXKEY1TYPE
                .EXTRN  ANLRMS$_IDXKEYDATAVBN
                .EXTRN  ANLRMS$_IDXKEYFILL
                .EXTRN  ANLRMS$_IDXKEYFLAGS
                .EXTRN  ANLRMS$_IDXKEYKEYSZ
                .EXTRN  ANLRMS$_IDXKEYNAME
                .EXTRN  ANLRMS$_IDXKEYNEXT
                .EXTRN  ANLRMS$_IDXKEYMINREC
                .EXTRN  ANLRMS$_IDXKEYNULL
                .EXTRN  ANLRMS$_IDXKEYPOSS
                .EXTRN  ANLRMS$_IDXKEYROOTLVL
                .EXTRN  ANLRMS$_IDXKEYROOTVBN
                .EXTRN  ANLRMS$_IDXKEYSEGS
                .EXTRN  ANLRMS$_IDXKEYSIZES
                .EXTRN  ANLRMS$_IDXPRIMREC
                .EXTRN  ANLRMS$_IDXPRIMRECFLAGS
                .EXTRN  ANLRMS$_IDXPRIMRECID
                .EXTRN  ANLRMS$_IDXPRIMRECLEN
                .EXTRN  ANLRMS$_IDXPRIMRECRRV
                .EXTRN  ANLRMS$_IDXPROAREAS
                .EXTRN  ANLRMS$_IDXPROLOG
                .EXTRN  ANLRMS$_IDXREC, ANLRMS$_IDXRECPTR
                .EXTRN  ANLRMS$_IDXSIDR
                .EXTRN  ANLRMS$_IDXSIDRDUPCNT
                .EXTRN  ANLRMS$_IDXSIDRFLAGS
                .EXTRN  ANLRMS$_IDXSIDRRECID
                .EXTRN  ANLRMS$_IDXSIDRPTRFLAGS
                .EXTRN  ANLRMS$_IDXSIDRPTRREF
                .EXTRN  ANLRMS$_INTERCOMMAND
                .EXTRN  ANLRMS$_INTERHDG
                .EXTRN  ANLRMS$_LONGREC
                .EXTRN  ANLRMS$_MAXRECSIZE
                .EXTRN  ANLRMS$_NOBACKUP
                .EXTRN  ANLRMS$_NOEXPIRATION
```

```
                .EXTRN  ANLRMS$_NOSPANFILLER
                .EXTRN  ANLRMS$_PERFORM
                .EXTRN  ANLRMS$_PROLOGFLAGS
                .EXTRN  ANLRMS$_PROLOGVER
                .EXTRN  ANLRMS$_PROT, ANLRMS$_RECATTR
                .EXTRN  ANLRMS$_RECFMT, ANLRMS$_RECLAIMBKT
                .EXTRN  ANLRMS$_RELBUCKET
                .EXTRN  ANLRMS$_RELEOFVBN
                .EXTRN  ANLRMS$_RELMAXREC
                .EXTRN  ANLRMS$_RELPROLOG
                .EXTRN  ANLRMS$_RELIAB, ANLRMS$_REVISION
                .EXTRN  ANLRMS$_STATHDG
                .EXTRN  ANLRMS$_SUMMARYHDG
                .EXTRN  ANLRMS$_OWNERUIC
                .EXTRN  ANLRMS$_JNL, ANLRMS$_AIJNL
                .EXTRN  ANLRMS$_BIJNL, ANLRMS$_ATJNL
                .EXTRN  ANLRMS$_ATTOP, ANLRMS$_BADCMD
                .EXTRN  ANLRMS$_BADPATH
                .EXTRN  ANLRMS$_BADVBN, ANLRMS$_DOWNHELP
                .EXTRN  ANLRMS$_DOWNPATH
                .EXTRN  ANLRMS$_EMPTYBKT
                .EXTRN  ANLRMS$_NODATA, ANLRMS$_NODOWN
                .EXTRN  ANLRMS$_NONEXT, ANLRMS$_NORECLAIMED
                .EXTRN  ANLRMS$_NORECS, ANLRMS$_NORRV
                .EXTRN  ANLRMS$_RESTDONE
                .EXTRN  ANLRMS$_STACKFULL
                .EXTRN  ANLRMS$_UNINITINDEX
                .EXTRN  ANLRMS$_FDLIDENT
                .EXTRN  ANLRMS$_FDLSYSTEM
                .EXTRN  ANLRMS$_FDLSOURCE
                .EXTRN  ANLRMS$_FDLFILE
                .EXTRN  ANLRMS$_FDLALLOC
                .EXTRN  ANLRMS$_FDLNOALLOC
                .EXTRN  ANLRMS$_FDLBESTTRY
                .EXTRN  ANLRMS$_FDLBUCKETSIZE
                .EXTRN  ANLRMS$_FDLCLUSTERSIZE
                .EXTRN  ANLRMS$_FDLCONTIG
                .EXTRN  ANLRMS$_FDLEXTENSION
                .EXTRN  ANLRMS$_FDLGLOBALBUFS
                .EXTRN  ANLRMS$_FDLMAXRECORD
                .EXTRN  ANLRMS$_FDLFILENAME
                .EXTRN  ANLRMS$_FDLORG, ANLRMS$_FDLOWNER
                .EXTRN  ANLRMS$_FDLPROTECTION
                .EXTRN  ANLRMS$_FDLRECORD
                .EXTRN  ANLRMS$_FDLSPAN
                .EXTRN  ANLRMS$_FDLCC, ANLRMS$_FDLVFCSIZE
                .EXTRN  ANLRMS$_FDLFORMAT
                .EXTRN  ANLRMS$_FDLSIZE
                .EXTRN  ANLRMS$_FDLAREA
                .EXTRN  ANLRMS$_FDLKEY, ANLRMS$_FDLCHANGES
                .EXTRN  ANLRMS$_FDLDATAAREA
                .EXTRN  ANLRMS$_FDLDATAFILL
                .EXTRN  ANLRMS$_FDLDATAKEYCOMPB
                .EXTRN  ANLRMS$_FDLDATARECCOMPB
                .EXTRN  ANLRMS$_FDLDUPS
                .EXTRN  ANLRMS$_FDLINDEXAREA
                .EXTRN  ANLRMS$_FDLINDEXCOMPB
```

```
.EXTRN  ANLRMS$_FDLINDEXFILL
.EXTRN  ANLRMS$_FDLL1INDEXAREA
.EXTRN  ANLRMS$_FDLKEYNAME
.EXTRN  ANLRMS$_FDLNORECS
.EXTRN  ANLRMS$_FDLNULLKEY
.EXTRN  ANLRMS$_FDLNULLVALUE
.EXTRN  ANLRMS$_FDLPROLOG
.EXTRN  ANLRMS$_FDLSEGLENGTH
.EXTRN  ANLRMS$_FDLSEGPOS
.EXTRN  ANLRMS$_FDLSEGTYPE
.EXTRN  ANLRMS$_FDLANALAREA
.EXTRN  ANLRMS$_FDLRECL
.EXTRN  ANLRMS$_FDLANALKEY
.EXTRN  ANLRMS$_FDLDATAKEYCOMP
.EXTRN  ANLRMS$_FDLDATARECCOMP
.EXTRN  ANLRMS$_FDLDATARECS
.EXTRN  ANLRMS$_FDLDATASPACE
.EXTRN  ANLRMS$_FDLDEPTH
.EXTRN  ANLRMS$_FDLDUPSPER
.EXTRN  ANLRMS$_FDLIDXCOMP
.EXTRN  ANLRMS$_FDLIDXFILL
.EXTRN  ANLRMS$_FDLIDXSPACE
.EXTRN  ANLRMS$_FDLIDXL1RECS
.EXTRN  ANLRMS$_FDLDATALENMEAN
.EXTRN  ANLRMS$_FDLIDXLENMEAN
.EXTRN  ANLRMS$_STATAREA
.EXTRN  ANLRMS$_STATRECL
.EXTRN  ANLRMS$_STATKEY
.EXTRN  ANLRMS$_STATDEPTH
.EXTRN  ANLRMS$_STATIDXL1RECS
.EXTRN  ANLRMS$_STATIDXLENMEAN
.EXTRN  ANLRMS$_STATIDXSPACE
.EXTRN  ANLRMS$_STATIDXFILL
.EXTRN  ANLRMS$_STATIDXCOMP
.EXTRN  ANLRMS$_STATDATARECS
.EXTRN  ANLRMS$_STATDUPSPER
.EXTRN  ANLRMS$_STATDATALENMEAN
.EXTRN  ANLRMS$_STATDATASPACE
.EXTRN  ANLRMS$_STATDATAFILL
.EXTRN  ANLRMS$_STATDATAKEYCOMP
.EXTRN  ANLRMS$_STATDATARECCOMP
.EXTRN  ANLRMS$_STATEFFICIENCY
.EXTRN  ANLRMS$_BADAREA1ST2
.EXTRN  ANLRMS$_BADAREABKTSIZE
.EXTRN  ANLRMS$_BADAREAFIT
.EXTRN  ANLRMS$_BADAREAID
.EXTRN  ANLRMS$_BADAREANEXT
.EXTRN  ANLRMS$_BADAREAROOT
.EXTRN  ANLRMS$_BADAREAUSED
.EXTRN  ANLRMS$_BADBKTAREAID
.EXTRN  ANLRMS$_BADBKTCHECK
.EXTRN  ANLRMS$_BADBKTFREE
.EXTRN  ANLRMS$_BADBKTKEYID
.EXTRN  ANLRMS$_BADBKTLEVEL
.EXTRN  ANLRMS$_BADBKTROOTBIT
.EXTRN  ANLRMS$_BADBKTSAMPLE
.EXTRN  ANLRMS$_BADCELLFIT
```

```
                                                .EXTRN  ANLRMS$_BADCHECKSUM
                                                .EXTRN  ANLRMS$_BADDATARECBITS
                                                .EXTRN  ANLRMS$_BADDATARECFIT
                                                .EXTRN  ANLRMS$_BADDATARECPS
                                                .EXTRN  ANLRMS$_BAD3IDXKEYFIT
                                                .EXTRN  ANLRMS$_BADIDXLASTKEY
                                                .EXTRN  ANLRMS$_BADIDXORDER
                                                .EXTRN  ANLRMS$_BADIDXRECBITS
                                                .EXTRN  ANLRMS$_BADIDXRECFIT
                                                .EXTRN  ANLRMS$_BADIDXRECPS
                                                .EXTRN  ANLRMS$_BADKEYAREAID
                                                .EXTRN  ANLRMS$_BADKEYDATABKT
                                                .EXTRN  ANLRMS$_BADKEYDATAFIT
                                                .EXTRN  ANLRMS$_BADKEYDATATYPE
                                                .EXTRN  ANLRMS$_BADKEYIDXBKT
                                                .EXTRN  ANLRMS$_BADKEYFILL
                                                .EXTRN  ANLRMS$_BADKEYFIT
                                                .EXTRN  ANLRMS$_BADKEYREFID
                                                .EXTRN  ANLRMS$_BADKEYROOTLEVEL
                                                .EXTRN  ANLRMS$_BADKEYSEGCOUNT
                                                .EXTRN  ANLRMS$_BADKEYSEGVEC
                                                .EXTRN  ANLRMS$_BADKEYSUMMARY
                                                .EXTRN  ANLRMS$_BADREADNOPAR
                                                .EXTRN  ANLRMS$_BADREADPAR
                                                .EXTRN  ANLRMS$_BADSIDRDUPCT
                                                .EXTRN  ANLRMS$_BADSIDRPTRFIT
                                                .EXTRN  ANLRMS$_BADSIDRPTRSZ
                                                .EXTRN  ANLRMS$_BADSIDRSIZE
                                                .EXTRN  ANLRMS$_BADSTREAMEOF
                                                .EXTRN  ANLRMS$_BADVBNFREE
                                                .EXTRN  ANLRMS$_BKTLOOP
                                                .EXTRN  ANLRMS$_EXTENDERR
                                                .EXTRN  ANLRMS$_FLAGERROR
                                                .EXTRN  ANLRMS$_MISSINGBKT
                                                .EXTRN  ANLRMS$_NOTOK, ANLRMS$_SPANERROR
                                                .EXTRN  ANLRMS$_TOOMANYRECS
                                                .EXTRN  ANLRMS$_UNWIND, ANLRMS$_VFCTOOSHORT
                                                .EXTRN  ANLRMS$_CACHEFULL
                                                .EXTRN  ANLRMS$_CACHERELFAIL
                                                .EXTRN  ANLRMS$_FACILITY
                                                .EXTRN  ANL$CHECK_MODE, ANL$FDL_MODE
                                                .EXTRN  ANL$FORMAT_ERROR
                                                .EXTRN  ANL$INTERACTIVE_MODE
                                                .EXTRN  CLI$PRESENT, LIB$ESTABLISH
                                                .EXTRN  OTS$CVT_TI_L, OTS$CVT_TZ_L
                                                .EXTRN  SYS$EXIT

                                                .PSECT  $CODE$,NOWRT,2

                                    000C 00000  .ENTRY  ANL$RMS, Save R2,R3                     ; 0641
                              53    0000' CF  9E 00002  MOVAB   ANL$GB_MODE, R3                 ;
                              52 00000000G 00  9E 00007  MOVAB   CLI$PRESENT, R2                ;
                                    0000V CF  9F 0000E  PUSHAB  ANL$WORST_ERROR_HANDLER         ; 0643
                 00000000G    00       01  FB 00012  CALLS   #1, LIB$ESTABLISH                ;
                                    0000' CF  9F 00019  PUSHAB  P.AAA                           ; 0647
                              62       01  FB 0001D  CALLS   #1, CLI$PRESENT                  ;
                              0A       50  E9 00020  BLBC    R0, 1$                           ;
```

```
                                63              02  90 00023          MOVB    #2, ANL$GB_MODE                                 ; 0648
                          0000G CF              00  FB 00026          CALLS   #0, ANL$FDL_MODE                                ; 0649
                                                3A  11 0002B          BRB     6$                                              ; 0647
                                      0000'     CF  9F 0002D 1$:      PUSHAB  P.AAC                                           ; 0651
                                62              01  FB 00031          CALLS   #1, CLI$PRESENT                                 ;
                                0A              50  E9 00034          BLBC    R0, 2$                                          ;
                                63              03  90 00037          MOVB    #3, ANL$GB_MODE                                 ; 0652
                          0000G CF              00  FB 0003A          CALLS   #0, ANL$INTERACTIVE_MODE                        ; 0653
                                                26  11 0003F          BRB     6$                                              ; 0651
                                      0000'     CF  9F 00041 2$:      PUSHAB  P.AAE                                           ; 0655
                                62              01  FB 00045          CALLS   #1, CLI$PRESENT                                 ;
                                05              50  E9 00048          BLBC    R0, 3$                                          ;
                                63              04  90 0004B          MOVB    #4, ANL$GB_MODE                                 ; 0656
                                                12  11 0004E          BRB     5$                                              ; 0657
                                      0000'     CF  9F 00050 3$:      PUSHAB  P.AAG                                           ; 0659
                                62              01  FB 00054          CALLS   #1, CLI$PRESENT                                 ;
                                05              50  E9 00057          BLBC    R0, 4$                                          ;
                                63              05  90 0005A          MOVB    #5, ANL$GB_MODE                                 ; 0660
                                                03  11 0005D          BRB     5$                                              ; 0661
                                63              01  90 0005F 4$:      MOVB    #1, ANL$GB_MODE                                 ; 0664
                          0000G CF              00  FB 00062 5$:      CALLS   #0, ANL$CHECK_MODE                              ; 0665
                                03              63  91 00067 6$:      CMPB    ANL$GB_MODE, #3                                 ; 0671
                                                08  12 0006A          BNEQ    7$                                              ;
                                    00000000G   8F  DD 0006C          PUSHL   #ANLRMS$_OK                                     ; 0672
                                                09  11 00072          BRB     8$                                              ;
                     7E      04     A3 10000000 8F  C9 00074 7$:      BISL3   #268435456, ANL$WORST_ERROR, -(SP)              ; 0674
                 00000000G  00                  01  FB 0007D 8$:      CALLS   #1, SYS$EXIT                                    ;
                                                    04 00084          RET                                                     ; 0676

; Routine Size:  133 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  172          0677  1 %sbttl 'ANL$UNWIND_HANDLER - Unwind to Caller'
;  173          0678  1 !+++
;  174          0679  1 !
;  175          0680  1 ! Functional Description:
;  176          0681  1 !       This condition handler is established at various points
;  177          0682  1 !       in analyzrms and allows the stack to be unwound
;  178          0683  1 !       and execution continued after any of an assortment
;  179          0684  1 !       of serious errors occurs.
;  180          0685  1 !
;  181          0686  1 ! Formal Paramters:
;  182          0687  1 !       signal_args = Address of signal argument list
;  183          0688  1 !       mechanism_args = Address of mechanism argument list
;  184          0689  1 !
;  185          0690  1 ! Implicit Inputs:
;  186          0691  1 !       none
;  187          0692  1 !
;  188          0693  1 ! Returned Value:
;  189          0694  1 !       ss$_resignal    This was not an anlrms$_unwind condition.
;  190          0695  1 !       ss$_continue
;  191          0696  1 !
;  192          0697  1 ! Side Effects:
;  193          0698  1 !       anl$worst_error is updated with highest severity error.
;  194          0699  1 !
;  195          0700  1 !---
;  196          0701  1
;  197          0702  2 global routine anl$unwind_handler (signal_args, mechanism_args) = begin
;  198          0703  2
;  199          0704  2 map
;  200          0705  2     signal_args:        ref bblock,     ! Address of signal argument list
;  201          0706  2     mechanism_args:     ref bblock;     ! Address of mechanism argument list
;  202          0707  2
;  203          0708  2 local
;  204          0709  2     code:               bblock [long],  ! Condition code (longword)
;  205          0710  2     status;
;  206          0711  2
;  207          0712  2 code = .signal_args [chf$l_sig_name];   ! Get condition code
;  208          0713  2
;  209          0714  2 ! If the condition is not anlrms$unwind, then we resignal it.
;  210          0715  2
;  211          0716  2 if .code nequ anlrms$_unwind then
;  212          0717  2     return ss$_resignal
;  213          0718  2
;  214          0719  2 ! It's a drastic structure error.  We can no longer continue what we
;  215          0720  2 ! were doing.  In interactive mode, we want to return to user command
;  216          0721  2 ! level.  In check mode, we want to quit analyzing this file and
;  217          0722  2 ! go on to the next file.  In FDL mode, since there can only be 1
;  218          0723  2 ! file spec, we just quit.  In all cases, we put out an error message
;  219          0724  2 ! and back out by unwinding to the frame of the caller of the routine
;  220          0725  2 ! that called lib$establish.  Note that since we do not resignal
;  221          0726  2 ! the error to allow anl$worst_error_handler to save the anl$worst_error,
;  222          0727  2 ! we must save it here.
;  223          0728  2
;  224          0729  3 else (
;  225          0730  3     if   severity_level (.code) gtr
;  226          0731  4          severity_level (.anl$worst_error)      ! If higher than watermark
;  227          0732  3     then anl$worst_error = .code;               !  -then set new worst error
;  228          0733  3
```

```
;  229          0734  3      anl$format_error(anlrms$_unwind);
;  230          0735  3      status=$unwind();
;  231          0736  3      check (.status,.status);
;  232          0737  3
;  233          0738  3      return ss$_continue;
;  234          0739  2 );
;  235          0740  2
;  236          0741  1 end;


                                                                .EXTRN  SYS$UNWIND

                                                001C 00000      .ENTRY  ANL$UNWIND_HANDLER, Save R2,R3,R4               ; 0702
                                    54 00000000G  8F  D0 00002  MOVL    #ANLRMS$_UNWIND, R4
                                    50       04   AC  D0 00009  MOVL    SIGNAL_ARGS, R0                             ; 0712
                                    53       04   A0  D0 0000D  MOVL    4(R0), CODE
                                    54            53  D1 00011  CMPL    CODE, R4                                    ; 0716
                                                  06  13 00014  BEQL    1$
                                    50     0918   8F  3C 00016  MOVZWL  #2328, R0                                  ; 0729
                                                  04  0001B  RET
                                    50            53  D0 0001C 1$:  MOVL    CODE, TMP_CODE                          ; 0730
                51                  50      03    00  EF 0001F  EXTZV   #0, #3, TMP_CODE, R1
                50                  50      01    00  EF 00024  EXTZV   #0, #1, TMP_CODE, R0
                                    50            04  C4 00029  MULL2   #4, R0
                                    51            50  C2 0002C  SUBL2   R0, R1
                                    51            03  C0 0002F  ADDL2   #3, R1
                                    50     0000'  CF  D0 00032  MOVL    ANL$WORST_ERROR, TMP_CODE                   ; 0731
                52                  50      03    00  EF 00037  EXTZV   #0, #3, TMP_CODE, R2
                50                  50      01    00  EF 0003C  EXTZV   #0, #1, TMP_CODE, R0
                                    50            04  C4 00041  MULL2   #4, R0
                                    52            50  C2 00044  SUBL2   R0, R2
                                    50       03   A2  9E 00047  MOVAB   3(R2), R0
                                    50            51  D1 0004B  CMPL    R1, R0
                                                  05  15 0004E  BLEQ    2$
                                          0000'  CF             53  D0 00050  MOVL    CODE, ANL$WORST_ERROR           ; 0732
                                                                54  DD 00055 2$:  PUSHL   R4                      ; 0734
                                          0000G  CF             01  FB 00057  CALLS   #1, ANL$FORMAT_ERROR
                                                                7E  7C 0005C  CLRQ    -(SP)                       ; 0735
                                    00000000G    00           02  FB 0005E  CALLS   #2, SYS$UNWIND
                                                 09           50  E8 00065  BLBS    STATUS, 3$                  ; 0736
                                                              50  DD 00068  PUSHL   STATUS
                                    00000000G    00           01  FB 0006A  CALLS   #1, LIB$SIGNAL
                                                 50           01  D0 00071 3$:  MOVL    #1, R0                      ; 0738
                                                              04  00074  RET                                       ; 0741

; Routine Size:  117 bytes,    Routine Base:  $CODE$ + 0085
```

```
;   238          0742  1 %sbttl 'ANL$WORST_ERROR_HANDLER - Baddest error handler in the West'
;   239          0743  1 !++
;   240          0744  1 ! Functional Description:
;   241          0745  1 !       This condition handler is established by the main routine in
;   242          0746  1 !       analyzrms.  It gains control when any error is signaled except
;   243          0747  1 !       for ANLRMS$_UNWIND, which is handled specially by the
;   244          0748  1 !       ANL$UNWIND_HANDLER.  If the error signaled is more severe than any
;   245          0749  1 !       which has preceded it, save the error status as the
;   246          0750  1 !       anl$worst_error.  The resignal the error so the last-chance
;   247          0751  1 !       condition handler can get a crack at the error.
;   248          0752  1 !
;   249          0753  1 ! Formal Paramters:
;   250          0754  1 !       signal_args = Address of signal argument list
;   251          0755  1 !       mechanism_args = Address of mechanism array
;   252          0756  1 !
;   253          0757  1 ! Implicit Inputs:
;   254          0758  1 !       none
;   255          0759  1 !
;   256          0760  1 ! Returned Value:
;   257          0761  1 !       ss$_resignal    Continue to search call frames.
;   258          0762  1 !
;   259          0763  1 ! Side Effects:
;   260          0764  1 !       anl$worst_error is updated with highest severity error.
;   261          0765  1 !
;   262          0766  1 !---
;   263          0767  1
;   264          0768  2 global routine anl$worst_error_handler (signal_args, mechanism_args) = begin
;   265          0769  2
;   266          0770  2 map
;   267          0771  2     signal_args:        ref bblock,     ! Address of signal argument list
;   268          0772  2     mechanism_args:     ref bblock;     ! Address of mechanism argument list
;   269          0773  2
;   270          0774  2 local
;   271          0775  2     code:               bblock [long];  ! Condition code (longword)
;   272          0776  2
;   273          0777  2 code =  .signal_args [chf$l_sig_name];          ! Get condition code
;   274          0778  2 if   severity_level (.code) gtr
;   275          0779  3     severity_level (.anl$worst_error)          ! If higher than watermark
;   276          0780  2 then anl$worst_error = .code;                  !  -then set new worst error
;   277          0781  2
;   278          0782  2 return ss$_resignal;
;   279          0783  2
;   280          0784  1 end;


                                        000C 00000          .ENTRY  ANL$WORST_ERROR_HANDLER, Save R2,R3          ; 0768
                                50      04   AC   D0 00002  MOVL   SIGNAL_ARGS, R0                             ; 0777
                                53      04   A0   D0 00006  MOVL   4(R0), CODE                                 ;
                                50           53   D0 0000A  MOVL   CODE, TMP_CODE                             ; 0778
                51              50      03   00   EF 0000D  EXTZV  #0, #3, TMP_CODE, R1                       ;
                50              50      01   00   EF 00012  EXTZV  #0, #1, TMP_CODE, R0                       ;
                                50           04   C4 00017  MULL2  #4, R0                                     ;
                                51           50   C2 0001A  SUBL2  R0, R1                                    ;
                                51           03   C0 0001D  ADDL2  #3, R1                                    ;
```

```
                                50      0000'   CF  D0 00020         MOVL    ANL$WORST_ERROR, TMP_CODE                       ; 0779
          52            50      03              00  EF 00025         EXTZV   #0, #3, TMP_CODE, R2                            ;
          50            50      01              00  EF 0002A         EXTZV   #0, #1, TMP_CODE, R0                            ;
                                50              04  C4 0002F         MULL2   #4, R0                                          ;
                                52              50  C2 00032         SUBL2   R0, R2                                          ;
                                50        03    A2  9E 00035         MOVAB   3(R2), R0                                       ;
                                50              51  D1 00039         CMPL    R1, R0                                          ;
                                                05  15 0003C         BLEQ    1$                                              ;
                        0000'   CF              53  D0 0003E         MOVL    CODE, ANL$WORST_ERROR                           ; 0780
                                50      0918    8F  3C 00043 1$:     MOVZWL  #2328, R0                                       ; 0782
                                                    04 00048         RET                                                     ; 0784

; Routine Size:  73 bytes,    Routine Base:  $CODE$ + 00FA
```

```
;  282          0785  1 %sbttl 'ANL$INTERNALIZE_NUMBER - Convert String to Longword'
;  283          0786  1 !++
;  284          0787  1 ! Functional Description:
;  285          0788  1 !       This routine will convert the ASCII representation of a decimal
;  286          0789  1 !       or hexadecimal number to a longword.  It is here for lack of a
;  287          0790  1 !       better place.
;  288          0791  1 !
;  289          0792  1 ! Formal Parameters:
;  290          0793  1 !       string          Address of descriptor of ASCII number.  Decimal
;  291          0794  1 !                       numbers are just digits, while hexadecimal numbers
;  292          0795  1 !                       begin with %X or are enclosed in %X'...'.
;  293          0796  1 !       longword        Address of longword in which to return value.
;  294          0797  1 !
;  295          0798  1 ! Implicit Inputs:
;  296          0799  1 !       global data
;  297          0800  1 !
;  298          0801  1 ! Implicit Outputs:
;  299          0802  1 !       global data
;  300          0803  1 !
;  301          0804  1 ! Returned Value:
;  302          0805  1 !       True if number was valid, false if invalid.
;  303          0806  1 !
;  304          0807  1 ! Side Effects:
;  305          0808  1 !
;  306          0809  1 !--
;  307          0810  1
;  308          0811  1
;  309          0812  2 global routine anl$internalize_number(string,longword) = begin
;  310          0813  2
;  311          0814  2 bind
;  312          0815  2         string_dsc = .string: descriptor;
;  313          0816  2
;  314          0817  2 local
;  315          0818  2         status: long,
;  316          0819  2         sp: ref block[,byte],
;  317          0820  2         hex_dsc: descriptor;
;  318          0821  2
;  319          0822  2
;  320          0823  2 ! If the string is null, then it's invalid.
;  321          0824  2
;  322          0825  2 if .string_dsc[len] eqlu 0 then
;  323          0826  2         return false;
;  324          0827  2
;  325          0828  2 ! Split up depending upon whether it's a decimal or hex number.
;  326          0829  2
;  327          0830  3 if ch$eql(minu(.string_dsc[len],2),.string_dsc[ptr], 2,uplit byte('%X'), ' ') then (
;  328          0831  3
;  329          0832  3         ! We have a hex number.  Build a descriptor of the actual digits.
;  330          0833  3         ! If the third character is an apostrophe, then we must find the
;  331          0834  3         ! matching apostrophe.
;  332          0835  3
;  333          0836  4         if ch$rchar(.string_dsc[ptr]+2) eqlu '''' then (
;  334          0837  4                 sp = ch$find_ch(.string_dsc[len]-3,.string_dsc[ptr]+3, '''');
;  335          0838  4                 if .sp eqlu 0 then
;  336          0839  4                         return false;
;  337          0840  4                 build_descriptor(hex_dsc,.sp-.string_dsc[ptr]-3,.string_dsc[ptr]+3);
;  338          0841  3         ) else
```

```
;  339     0842  3                 build_descriptor(hex_dsc,,string_dsc[len]-2,,string_dsc[ptr]+2);
;  340     0843  3         status = ots$cvt_tz_l(hex_dsc,,longword,4,%b'1');
;  341     0844  3
;  342     0845  2 ) else
;  343     0846  2
;  344     0847  2         ! We have a decimal number.  Convert it and return the status.
;  345     0848  2
;  346     0849  2         status = ots$cvt_ti_l(string_dsc,,longword,4,%b'11');
;  347     0850  2
;  348     0851  2 return .status;
;  349     0852  2
;  350     0853  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                        58  25  00044 P.AAI:  .ASCII  \%X\                                          ;


                                                        .PSECT  $CODE$,NOWRT,2

                                          003C  00000           .ENTRY  ANL$INTERNALIZE_NUMBER, Save R2,R3,R4,R5    ; 0812
                                    5E      08    C2  00002     SUBL2   #8, SP                                      ;
                                    55  04    AC    D0  00005     MOVL    STRING, R5                                  ; 0815
                                              65    B5  00009     TSTW    (R5)                                        ; 0825
                                              6E    13  0000B     BEQL    6$
                                    50        65    3C  0000D     MOVZWL  (R5), R0                                    ; 0830
                                    02        50    B1  00010     CMPW    R0, #2
                                              03    1B  00013     BLEQU   1$
                                    50        02    D0  00015     MOVL    #2, R0
                                    54  04    A5    D0  00018 1$: MOVL    4(R5), R4
                              02        20    64    50  2D  0001C CMPC5   R0, (R4), #32, #2, P.AAI
                                                0000' CF  00021
                                              44    12  00024     BNEQ    5$
                                    27  02    A4    91  00026     CMPB    2(R4), #39                                  ; 0836
                                              21    12  0002A     BNEQ    3$
                                    50        65    3C  0002C     MOVZWL  (R5), R0                                    ; 0837
                                    50        03    C2  0002F     SUBL2   #3, R0
                              03    A4        50    27  3A  00032 LOCC    #39, R0, 3(R4)
                                              02    12  00037     BNEQ    2$
                                              51    D4  00039     CLRL    R1
                                              51    D5  0003B 2$: TSTL    SP                                          ; 0838
                                              3C    13  0003D     BEQL    6$
                                    51        54    C2  0003F     SUBL2   R4, R1                                      ; 0840
                                    6E  FD    A1    9E  00042     MOVAB   -3(R1), HEX_DSC
                                04  AE  03    A4    9E  00046     MOVAB   3(R4), HEX_DSC+4
                                              0B    11  0004B     BRB     4$                                          ; 0836
                                    6E        65    3C  0004D 3$: MOVZWL  (R5), HEX_DSC                               ; 0842
                                    6E        02    C2  00050     SUBL2   #2, HEX_DSC
                                04  AE  02    A4    9E  00053     MOVAB   2(R4), HEX_DSC+4
                                              01    DD  00058 4$: PUSHL   #1                                          ; 0843
                                              04    DD  0005A     PUSHL   #4
                                          08  AC    DD  0005C     PUSHL   LONGWORD
                                          0C  AE    9F  0005F     PUSHAB  HEX_DSC
                                00000000G 00        04  FB  00062 CALLS   #4, OTS$CVT_TZ_L                           ;
```

```
                                 04 00069                RET                                             ; 0830
                              03 DD 0006A 5$:            PUSHL   #3                                      ; 0849
                              04 DD 0006C                PUSHL   #4                                      ;
                           08 AC DD 0006E                PUSHL   LONGWORD                                ;
                              55 DD 00071                PUSHL   R5                                      ;
                00000000G  00 04 FB 00073                CALLS   #4, OTS$CVT_TI_L                        ;
                                 04 0007A                RET                                             ; 0851
                              50 D4 0007B 6$:            CLRL    R0                                      ; 0853
                                 04 0007D                RET                                             ;

; Routine Size:  126 bytes,    Routine Base:  $CODE$ + 0143
```

```
;  352          0854  1 %sbttl 'ANL$CHECK_FLAGS - Check Flag Usage'
;  353          0855  1 !++
;  354          0856  1 ! Functional Description:
;  355          0857  1 !       This routine is called to check the usage of flags in a flag
;  356          0858  1 !       byte/word/longword.  This routine is here for no better place.
;  357          0859  1 !
;  358          0860  1 ! Formal Parameters:
;  359          0861  1 !       vbn             VBN of the bucket containing the flags.
;  360          0862  1 !       flags           A longword containing the flags to be checked.
;  361          0863  1 !       flag_def        A longword vector defining the valid flags.  The
;  362          0864  1 !                       zeroth longword contains the bit number of the
;  363          0865  1 !                       last valid flag.  The remaining longwords contain
;  364          0866  1 !                       zero if the flag is unused, non-zero otherwise.
;  365          0867  1 !
;  366          0868  1 ! Implicit Inputs:
;  367          0869  1 !       global data
;  368          0870  1 !
;  369          0871  1 ! Implicit Outputs:
;  370          0872  1 !       global data
;  371          0873  1 !
;  372          0874  1 ! Returned Value:
;  373          0875  1 !       none
;  374          0876  1 !
;  375          0877  1 ! Side Effects:
;  376          0878  1 !
;  377          0879  1 !--
;  378          0880  1
;  379          0881  1
;  380          0882  2 global routine anl$check_flags(vbn,flags,flag_def): novalue = begin
;  381          0883  2
;  382          0884  2 bind
;  383          0885  2         flags_vector = flags: bitvector[],
;  384          0886  2         flag_def_vector = .flag_def: vector[,long];
;  385          0887  2
;  386          0888  2 local
;  387          0889  2         i: long;
;  388          0890  2
;  389          0891  2
;  390          0892  2 ! We will simply sit in a loop scanning the flag bits.  If any flag is
;  391          0893  2 ! set but undefined, we will issue an error message.
;  392          0894  2
;  393          0895  3 incru i from 0 to 31 do (
;  394          0896  3         if .flags_vector[.i] then
;  395          0897  4                 if .i lequ .flag_def_vector[0] then (
;  396          0898  4                         if .flag_def_vector[.i+1] eqlu 0 then
;  397          0899  4                                 anl$format_error(anlrms$_flagerror,,.vbn,,.i)
;  398          0900  3                 ) else
;  399          0901  3                         anl$format_error(anlrms$_flagerror,,.vbn,,.i);
;  400          0902  2 );
;  401          0903  2
;  402          0904  2 return;
;  403          0905  2
;  404          0906  1 end;
```

```
                                 0004 00000         .ENTRY  ANL$CHECK_FLAGS, Save R2                        ; 0882
                              52 D4 00002         CLRL    I                                               ; 0895
                    20     08 AC    52 E1 00004 1$: BBC     I, FLAGS_VECTOR, 3$                             ; 0896
                           0C BC    52 D1 00009     CMPL    I, @FLAG_DEF                                    ; 0897
                                    0A 1A 0000D     BGTRU   2$
                              50 0C BC42 DE 0000F     MOVAL   @FLAG_DEF[I], R0                            ; 0898
                           04 A0    D5 00014        TSTL    4(R0)
                                    10 12 00017     BNEQ    3$
                              52 DD 00019 2$:       PUSHL   I                                               ; 0901
                           04 AC    DD 0001B        PUSHL   VBN
                 00000000G 8F       DD 0001E        PUSHL   #ANLRMS$_FLAGERROR
               0000G CF             03 FB 00024     CALLS   #3, ANL$FORMAT_ERROR
                              52 D6 00029 3$:       INCL    I                                               ; 0895
                           1F 52    D1 0002B        CMPL    I, #31
                              D4 1B 0002E        BLEQU   1$
                                 04 00030        RET                                                     ; 0906

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 01C1
```

```
;  406          0907  1 %sbttl 'ANL$PREPARE_QUOTED_STRING - Prepare a Quoted String'
;  407          0908  1 !++
;  408          0909  1 ! Functional Description:
;  409          0910  1 !       This routine is called to prepare a quoted string for inclusion in
;  410          0911  1 !       an FDL specification, or perhaps in a formatted message.  Preparing
;  411          0912  1 !       the string includes stripping trailing whitespace, doubling any
;  412          0913  1 !       quotation marks, and enclosing it in quotation marks.
;  413          0914  1 !
;  414          0915  1 ! Formal Parameters:
;  415          0916  1 !       input_dsc       Descriptor of buffer with input string.
;  416          0917  1 !       output_dsc      Descriptor of buffer to receive output string.
;  417          0918  1 !                       The length is set correctly.
;  418          0919  1 !
;  419          0920  1 ! Implicit Inputs:
;  420          0921  1 !       global data
;  421          0922  1 !
;  422          0923  1 ! Implicit Outputs:
;  423          0924  1 !       global data
;  424          0925  1 !
;  425          0926  1 ! Returned Value:
;  426          0927  1 !       none
;  427          0928  1 !
;  428          0929  1 ! Side Effects:
;  429          0930  1 !
;  430          0931  1 !--
;  431          0932  1
;  432          0933  1
;  433          0934  1 global routine anl$prepare_quoted_string(input_dsc: ref descriptor,
;  434          0935  1                                         output_dsc: ref descriptor):
;  435          0936  2                                                     novalue = begin
;  436          0937  2
;  437          0938  2 bind
;  438          0939  2         input_vector = .input_dsc[ptr]: vector[,byte],
;  439          0940  2         output_vector = .output_dsc[ptr]: vector[,byte];
;  440          0941  2
;  441          0942  2 local
;  442          0943  2         i: signed long,
;  443          0944  2         trimmed_length: long;
;  444          0945  2
;  445          0946  2
;  446          0947  2 ! Begin by scanning the input string from the end in order to eliminate
;  447          0948  2 ! any trailing whitespace.  We actually eliminate all control characters
;  448          0949  2 ! so that we'll catch NULs too.
;  449          0950  2
;  450          0951  3 i =     (decru i from .input_dsc[len]-1 to 0 do
;  451          0952  2                 if .input_vector[.i] gtru ' ' then exitloop .i);
;  452          0953  2 trimmed_length = .i + 1;
;  453          0954  2
;  454          0955  2 ! Put the opening quotation mark in the output buffer.
;  455          0956  2
;  456          0957  2 output_vector[0] = '"';
;  457          0958  2 output_dsc[len] = 1;
;  458          0959  2
;  459          0960  2 ! Scan the input string from the beginning, moving each character into the
;  460          0961  2 ! output buffer.  Quotation marks must be doubled.
;  461          0962  2
;  462          0963  3 incr i from 0 to .trimmed_length-1 do (
```

```
;  463          0964  4          if .input_vector[.i] eqlu '"' then (
;  464          0965  4                  output_vector[.output_dsc[len]] = '"';
;  465          0966  4                  increment(output_dsc[len]);
;  466          0967  3          );
;  467          0968  3          output_vector[.output_dsc[len]] = .input_vector[.i];
;  468          0969  3          increment(output_dsc[len]);
;  469          0970  2 );
;  470          0971  2
;  471          0972  2 ! Add the closing quotation mark to the output buffer.
;  472          0973  2
;  473          0974  2 output_vector[.output_dsc[len]] = '"';
;  474          0975  2 increment(output_dsc[len]);
;  475          0976  2
;  476          0977  2 return;
;  477          0978  2
;  478          0979  1 end;


                                    003C 00000          .ENTRY  ANL$PREPARE_QUOTED_STRING, Save R2,R3,R4,R5 ; 0934
                          52    04   AC  7D 00002          MOVQ    INPUT_DSC, R2                               ; 0939
                          54    04   A3  D0 00006          MOVL    4(R3), R4                                   ; 0940
                          50         62  3C 0000A          MOVZWL  (R2), I                                     ; 0951
                                     07  11 0000D          BRB     2$                                          ;
                          20    04 B240  91 0000F 1$:      CMPB    @4(R2)[I], #32                              ; 0952
                                     04  1A 00014          BGTRU   3$                                          ;
                                     50  D7 00016 2$:       DECL    I                                           ;
                                     F5  11 00018          BRB     1$                                          ;
                                     50  D6 0001A 3$:       INCL    TRIMMED_LENGTH                              ; 0953
                               64         22  90 0001C          MOVB    #34, (R4)                               ; 0957
                               63         01  B0 0001F          MOVW    #1, (R3)                                ; 0958
                               51         01  CE 00022          MNEGL   #1, I                                   ; 0963
                                     1B  11 00025          BRB     6$                                          ;
                               22    04 B241  91 00027 4$:      CMPB    @4(R2)[I], #34                          ; 0964
                                     09  12 0002C          BNEQ    5$                                          ;
                               55         63  3C 0002E          MOVZWL  (R3), R5                                ; 0965
                             6544         22  90 00031          MOVB    #34, (R5)[R4]                           ;
                                     63  B6 00035          INCW    (R3)                                        ; 0966
                               55         63  3C 00037 5$:       MOVZWL  (R3), R5                                ; 0968
                             6544    04 B241  90 0003A          MOVB    @4(R2)[I], (R5)[R4]                     ;
                                     63  B6 00040          INCW    (R3)                                        ; 0969
                          E1         51  F2 00042 6$:       AOBLSS  TRIMMED_LENGTH, I, 4$                       ; 0963
                               50         63  3C 00046          MOVZWL  (R3), R0                                ; 0974
                             6044         22  90 00049          MOVB    #34, (R0)[R4]                           ;
                                     63  B6 0004D          INCW    (R3)                                        ; 0975
                                         04  0004F          RET                                                 ; 0979

; Routine Size:  80 bytes,    Routine Base:  $CODE$ + 01F2

;  479          0980  1
;  480          0981  0 end eludom
```

```
                                                        .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $GLOBAL$                        8  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         70  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        578  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                              Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        17         0        1000        00:01.8



;                                   COMMAND QUALIFIERS

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

; Size:         578 code + 78 data bytes
; Run Time:       00:16.8
; Elapsed Time:   00:54.3
; Lines/CPU Min:   3497
; Lexemes/CPU-Min: 16260
; Memory Used:  170 pages
; Compilation Complete
```

OB MISC
LIS

RMS21DX
LIS

RMS31DX
LIS

RMS
LIS

OBJUTIR
LIS