


```

RRRRRRRR      MM      MM      SSSSSSSS  RRRRRRRR      EEEEEEEEEEE  QQQQQQ
RRRRRRRR      MM      MM      SSSSSSSS  RRRRRRRR      EEEEEEEEEEE  QQQQQQ
RR      RR    MMMM  MMMM  SS          RR      RR    EE          QQ      QQ
RR      RR    MMMM  MMMM  SS          RR      RR    EE          QQ      QQ
RR      RR    MM   MM   MM   SS          RR      RR    EE          QQ      QQ
RRRRRRRR      MM      MM      SSSSSS    RRRRRRRR      EEEEEEEEEEE  QQ      QQ
RRRRRRRR      MM      MM      SSSSSS    RRRRRRRR      EEEEEEEEEEE  QQ      QQ
RR  RR        MM      MM                SS          RR  RR    EE          QQ  QQ
RR  RR        MM      MM                SS          RR  RR    EE          QQ  QQ
RR      RR    MM      MM                SS          RR      RR    EE          QQ  QQ
RR      RR    MM      MM                SS          RR      RR    EE          QQ  QQ
RR      RR    MM      MM      SSSSSSSS    RR      RR    EEEEEEEEEEE  QQQQ  QQ
RR      RR    MM      MM      SSSSSSSS    RR      RR    EEEEEEEEEEE  QQQQ  QQ

```

```

RRRRRRRR      EEEEEEEEEEE  QQQQQQ
RRRRRRRR      EEEEEEEEEEE  QQQQQQ
RR      RR    EE          QQ      QQ
RR      RR    EE          QQ      QQ
RR      RR    EE          QQ      QQ
RRRRRRRR      EEEEEEEEEEE  QQ      QQ
RRRRRRRR      EEEEEEEEEEE  QQ      QQ
RR  RR        EE          QQ  QQ
RR  RR        EE          QQ  QQ
RR      RR    EE          QQ  QQ
RR      RR    EE          QQ  QQ
RR      RR    EEEEEEEEEEE  QQQQ  QQ
RR      RR    EEEEEEEEEEE  QQQQ  QQ

```

ident 'V04-000'

* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
* ALL RIGHTS RESERVED. *

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
* TRANSFERRED. *

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
* CORPORATION. *

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *

♦♦ Facility: VAX/VMS Analyze Command, BLISS Require File for /RMS_FILE

Abstract: This is the BLISS require file for the ANALYZE/RMS_FILE facility. It includes various useful constructs and the definitions of all control blocks used by the facility.

Environment:

Author: Paul C. Anagnostopoulos, Creation Date: 12 February 1981

Modified By:

V03-006 DGB0044 Donald G. Blair 08-May-1984
Add "severity level" macro as part of my new
condition handling code which allows ANALYZRMS
to return the correct condition value in R0.

V03-005 RRB0002 Rowland R. Bradley 13-Dec-1983
Add support for the display of journaling information.

V03-004 PCA1019 Paul C. Anagnostopoulos 23-May-1983
Remove messages for READ_CHECK and WRITE_CHECK FDL
secondaries, as they have always been bogus.

V03-003 PCA1012 Paul C. Anagnostopoulos 6-Apr-1983
Add symbols for various new messages.

V03-002 PCA1011 Paul C. Anagnostopoulos 1-Apr-1983

Change the message prefix to ANLRMSS to ensure that message symbols are unique across all ANALYZEs. This is necessitated by the new merged message files.

V03-001 PCA1007 Paul C. Anagnostopoulos 10 Feb 1983
Add a couple of message symbols.

--

```
! Here we will define "extensions" to the BLISS language.
! First we need values for boolean variables.
literal
    false      = 0,
    true       = 1;

! Define structure type for VMS structures
structure
    bblock [o,p,s,e;n] =
        [n]
        (bblock+o)<p,s,e>;

! Now we will define macros to generate various things associated with
! string descriptors.
field_descriptor_fields = set
    len      = [0,0,16,0],
    ptr      = [4,0,32,0]
tes;

macro descriptor =
    block[8,byte] field(descriptor_fields) %;

macro describe[] =
    uplit long(%charcount(%remaining), uplit byte(%remaining)) %;

macro build_descriptor(name,length,address) =
    (name[0,0,32,0] = length;
     name[ptr] = address;)
%;

! Now we define two macros that can generate described buffers. The
! first is for OWN buffers and the second for LOCAL buffers. Note that
! the local buffer must be defined last in the declarations.

macro own_described_buffer(name,length) =
    name: block[8+length,byte] field(descriptor_fields)
        initial(length,name+8)
%;

macro local_described_buffer(name,length) =
    name: block[8+length,byte] field(descriptor_fields);
    name[0,0,32,0] = length;
    name[ptr] = name+8
%;

! Now we define macros to increment and decrement a variable.

macro increment (var) =
    (var = .var + 1) %,
decrement (var) =
    (var = .var - 1) %;
```

```
! We need an "infinite" loop construct. We also need a more elegant construct
! for terminating a loop.
```

```
macro loop =
  while 1 do %;
```

```
macro exitif[] =
  if %remaining then exitloop %;
```

```
! Define a macro that can check statuses from routines.
```

```
macro check(status)[] =
  (if not status then
   signal(%remaining);)
%;
```

```
! Macro to implement a function (f) of the message severity level that
! maps the various severity levels such that arithmetic comparisons of the
! mapped values ( f(severity) ) yield a order of precedence that is
! intuitively acceptable:
```

ERROR NAME	OLDVAL		NEWVAL
F(SUCCESS)	1	-->	0
F(INFORMATIONAL)	3	-->	2
F(WARNING)	0	-->	3
F(ERROR)	2	-->	5
F(SEVERE_ERROR)	4	-->	7

```
macro
severity_level (tmp_status) =
  BEGIN
  LOCAL tmp_code: BBLOCK [LONG];
  tmp_code = tmp_status;
  .tmp_code [sts$v_severity] - (4 * .tmp_code [sts$v_success]) + 3
  END%;
```

```
! Define literals for useful control characters.
```

```
literal
bell           = %x'07';
backspace     = %x'08';
tab           = %x'09';
linefeed     = %x'0a';
formfeed     = %x'0c';
creturn      = %x'0d';
ctrl_u       = %x'15';
ctrl_w       = %x'17';
ctrl_z       = %x'1a';
escape       = %x'1b';
delete       = %x'7f';
```

! The following literals identify the various modes we can be in.
! Some routines require knowledge of the mode in order to operate correctly.

```
literal
  anl$K_check           = 1,      ! ANALYZE/RMS_FILE/CHECK
  anl$K_fdl             = 2,      ! ANALYZE/RMS_FILE/FDL
  anl$K_interactive     = 3,      ! ANALYZE/RMS_FILE/INTERACTIVE
  anl$K_statistics      = 4,      ! ANALYZE/RMS_FILE/STATISTICS
  anl$K_summary         = 5;      ! ANALYZE/RMS_FILE/SUMMARY
```

! The following macro is used by the low-level structure analysis routines
! to call the statistics callback routines. See module RMSSTATS.

```
macro statistics_callback[] =
  if .anl$gb_mode equl anl$K_fdl or .anl$gb_mode equl anl$K_statistics then (
    %remaining
  )
%;
```

! The following data structure is called a Bucket Structure Descriptor.
! It contains all the information necessary to identify a particular
! data structure within a bucket.

```
field bsd_fields = set
  bsd$w_type      = [ 0,0,16,0],  ! Type of structure (used in RMSINTER
                                ! module).
  bsd$w_size      = [ 2,0,16,0],  ! Size of bucket in blocks.
  bsd$l_vbn       = [ 4,0,32,0],  ! VBN of bucket.
  bsd$l_offset    = [ 8,0,32,0],  ! Offset of structure within bucket.
  bsd$l_bufptr    = [12,0,32,0],  ! Address of bucket buffer.
  bsd$l_endptr    = [16,0,32,0],  ! Address of byte past end of buffer.
  bsd$l_work      = [20,0,32,0],  ! Work longword. Used to specify
                                ! additional context vis-a-vis the
                                ! structure.
```

```
tes;
literal
  bsd$c_size      = 24;           ! Size of a BSD.
```

```
macro bsd = block[bsd$c_size,byte] field(bsd_fields) %;
```

! The following macro initializes a new BSD.

```
macro init_bsd(b) =
  ch$fill(%x'00', bsd$c_size,b)
%;
```

! The following macro will copy a bucket by building a matching BSD and
! then reading the bucket. If the "to" BSD (which must at least be
! initialized) describes a previous bucket, ANL\$BUCKET will free it up.

```
macro copy_bucket(from_bsd,to_bsd) =
  (bInd
    f = from_bsd: bsd,
    t = to_bsd: bsd;

    t[bsd$w_type] = .f[bsd$w_type];
```

```
t[bsd$w_size] = .f[bsd$w_size];  
t[bsd$l_vbn] = .f[bsd$l_vbn];  
t[bsd$l_offset] = .f[bsd$l_offset];  
t[bsd$l_work] = .f[bsd$l_work];  
anl$bucket(t,0);  
)
```

z:

E
V

! Include the definitions of all the ridiculous message status codes.

external literal

anlrms\$-ok,
anlrms\$-alloc,
anlrms\$-anything,
anlrms\$-backup,
anlrms\$-bkt,
anlrms\$-bktarea,
anlrms\$-bktcheck,
anlrms\$-bktflags,
anlrms\$-bktfree,
anlrms\$-bktkey,
anlrms\$-bktlevel,
anlrms\$-bktnext,
anlrms\$-bktptrsize,
anlrms\$-bktrecid,
anlrms\$-bktrecid\$,
anlrms\$-bktsample,
anlrms\$-bktvbnfree,
anlrms\$-bucketsize,
anlrms\$-cell,
anlrms\$-celldata,
anlrms\$-cellflags,
anlrms\$-checkhdg,
anlrms\$-contig,
anlrms\$-creation,
anlrms\$-ctlsize,
anlrms\$-datarec,
anlrms\$-databktvbn,
anlrms\$-dumpheading,
anlrms\$-eof,
anlrms\$-errorcount,
anlrms\$-errornone,
anlrms\$-errors,
anlrms\$-expiration,
anlrms\$-fileattr,
anlrms\$-filehdr,
anlrms\$-fileid,
anlrms\$-fileorg,
anlrms\$-filespec,
anlrms\$-flag,
anlrms\$-globalbufs,
anlrms\$-hexdata,
anlrms\$-hexheading1,
anlrms\$-hexheading2,
anlrms\$-idxarea,
anlrms\$-idxareaalloc,
anlrms\$-idxareabktsz,
anlrms\$-idxareanext,
anlrms\$-idxareanoalloc,
anlrms\$-idxareaqty,
anlrms\$-idxarearecl,
anlrms\$-idxareaused,
anlrms\$-idxkey,
anlrms\$-idxkeyareas.

anlrms\$ _idxkeybktsz,
anlrms\$ _idxkeybytes,
anlrms\$ _idxkeyltype,
anlrms\$ _idxkeydatavbn,
anlrms\$ _idxkeyfill,
anlrms\$ _idxkeyflags,
anlrms\$ _idxkeykeysz,
anlrms\$ _idxkeyname,
anlrms\$ _idxkeynext,
anlrms\$ _idxkeyminrec,
anlrms\$ _idxkeynull,
anlrms\$ _idxkeyposs,
anlrms\$ _idxkeyrootlvl,
anlrms\$ _idxkeyrootvbn,
anlrms\$ _idxkeysegs,
anlrms\$ _idxkeysizes,
anlrms\$ _idxprimrec,
anlrms\$ _idxprimrecflags,
anlrms\$ _idxprimrecid,
anlrms\$ _idxprimreclen,
anlrms\$ _idxprimrecrrv,
anlrms\$ _idxproareas,
anlrms\$ _idxprolog,
anlrms\$ _idxrec,
anlrms\$ _idxrecptr,
anlrms\$ _idxsidr,
anlrms\$ _idxsidrdupcnt,
anlrms\$ _idxsidrflags,
anlrms\$ _idxsidrrecid,
anlrms\$ _idxsidrptrflags,
anlrms\$ _idxsidrptrref,
anlrms\$ _intercommand,
anlrms\$ _interhdg,
anlrms\$ _longrec,
anlrms\$ _maxrecsize,
anlrms\$ _nobackup,
anlrms\$ _noexpiration,
anlrms\$ _nospanfiller,
anlrms\$ _perform,
anlrms\$ _prologflags,
anlrms\$ _prologver,
anlrms\$ _prot,
anlrms\$ _recattr,
anlrms\$ _recfmt,
anlrms\$ _reclaimbkt,
anlrms\$ _relbucket,
anlrms\$ _releofvbn,
anlrms\$ _relmaxrec,
anlrms\$ _relprolog,
anlrms\$ _reliab,
anlrms\$ _revision,
anlrms\$ _stathdg,
anlrms\$ _summaryhdg,
anlrms\$ _owneruic,
anlrms\$ _jnl,
anlrms\$ _ajnl,

anlrms\$_bijnl,
anlrms\$_atjnl,
anlrms\$_atjnl,
anlrms\$_atjnl,
anlrms\$_badcmd,
anlrms\$_badpath,
anlrms\$_badvbn,
anlrms\$_downhelp,
anlrms\$_downpath,
anlrms\$_emptybkt,
anlrms\$_nodata,
anlrms\$_nodown,
anlrms\$_nonext,
anlrms\$_noreclaimed,
anlrms\$_norecs,
anlrms\$_norrv,
anlrms\$_restdone,
anlrms\$_stackfull,
anlrms\$_uninitindex,
anlrms\$_fdlident,
anlrms\$_fdlssystem,
anlrms\$_fdlsource,
anlrms\$_fdlfile,
anlrms\$_fdlalloc,
anlrms\$_fdlnoalloc,
anlrms\$_fdlbesttry,
anlrms\$_fdlbucketsize,
anlrms\$_fdlclustersize,
anlrms\$_fdlcontig,
anlrms\$_fdlextension,
anlrms\$_fdlglobalbufs,
anlrms\$_fdlmaxrecord,
anlrms\$_fdlfilename,
anlrms\$_fdlorg,
anlrms\$_fdlowner,
anlrms\$_fdlprotection,
anlrms\$_fdlrecord,
anlrms\$_fdlspan,
anlrms\$_fdlcc,
anlrms\$_fdlvfcsiz,
anlrms\$_fdlformat,
anlrms\$_fdlsize,
anlrms\$_fdlarea,
anlrms\$_fdlkey,
anlrms\$_fdlchanges,
anlrms\$_fdldataarea,
anlrms\$_fdldatafill,
anlrms\$_fdldatakeycompb,
anlrms\$_fdldatareccompb,
anlrms\$_fdldups,
anlrms\$_fdlindexarea,
anlrms\$_fdlindexcompb,
anlrms\$_fdlindexfill,
anlrms\$_fdlllindexarea,
anlrms\$_fdlkeyname,
anlrms\$_fdlnorecs,
anlrms\$_fdlnullkey,

anlrms\$_fdlnullvalue,
anlrms\$_fdlprolog,
anlrms\$_fdlseglength,
anlrms\$_fdlsegpos,
anlrms\$_fdlsegtype,
anlrms\$_fdlanalarea,
anlrms\$_fdlrecl,
anlrms\$_fdlanalkey,
anlrms\$_fdldatakeycomp,
anlrms\$_fdldatareccomp,
anlrms\$_fdldatarecs,
anlrms\$_fdldataspace,
anlrms\$_fdldepth,
anlrms\$_fdldupsper,
anlrms\$_fdlidxcomp,
anlrms\$_fdlidxfill,
anlrms\$_fdlidxspace,
anlrms\$_fdlidxlrecs,
anlrms\$_fdldataalenmean,
anlrms\$_fdlidxlenmean,
anlrms\$_statarea,
anlrms\$_statrecl,
anlrms\$_statkey,
anlrms\$_statdepth,
anlrms\$_statidxlrecs,
anlrms\$_statidxlenmean,
anlrms\$_statidxspace,
anlrms\$_statidxfill,
anlrms\$_statidxcomp,
anlrms\$_statdatarecs,
anlrms\$_statdupsper,
anlrms\$_statdataalenmean,
anlrms\$_statdataspace,
anlrms\$_statdatafill,
anlrms\$_statdatakeycomp,
anlrms\$_statdatareccomp,
anlrms\$_statefficiency,
anlrms\$_badarealst2,
anlrms\$_badareabktsize,
anlrms\$_badareafit,
anlrms\$_badareaid,
anlrms\$_badareanext,
anlrms\$_badarearoot,
anlrms\$_badareaused,
anlrms\$_badbktareaid,
anlrms\$_badbktcheck,
anlrms\$_badbktfree,
anlrms\$_badbktkeyid,
anlrms\$_badbktlevel,
anlrms\$_badbktrootbit,
anlrms\$_badbktsample,
anlrms\$_badcellfit,
anlrms\$_badchecksum,
anlrms\$_baddatarecbits,
anlrms\$_baddatarecfit,
anlrms\$_baddatarecps,

```

anlrms$_bad3idxkeyfit,
anlrms$_badidxlastkey,
anlrms$_badidxorder,
anlrms$_badidxrecbits,
anlrms$_badidxrecfit,
anlrms$_badidxrecps,
anlrms$_badkeyareaaid,
anlrms$_badkeydatabkt,
anlrms$_badkeydatafit,
anlrms$_badkeydatatype,
anlrms$_badkeyidxbkt,
anlrms$_badkeyfill,
anlrms$_badkeyfit,
anlrms$_badkeyrefid,
anlrms$_badkeyrootlevel,
anlrms$_badkeysegcount,
anlrms$_badkeysegvec,
anlrms$_badkeysummary,
anlrms$_badreadnopar,
anlrms$_badreadpar,
anlrms$_badsidrdupt,
anlrms$_badsidrptrfit,
anlrms$_badsidrptrsz,
anlrms$_badsidrsize,
anlrms$_badstreameof,
anlrms$_badvbnfree,
anlrms$_bktloop,
anlrms$_extenderr,
anlrms$_flagerror,
anlrms$_missingbkt,
anlrms$_notok,
anlrms$_spanerror,
anlrms$_toomanyrecs,
anlrms$_unwind,
anlrms$_vfctooshort,
anlrms$_cachefull,
anlrms$_cacherelfail,
anlrms$_facility;

```

```

! We use a few of the message in the shareable message file SHRMSG.
! Define status codes for these which include our facility code and
! the message severity.

```

```
literal
```

```

anlrms$_closein = shr$_closein + 177*65536 + sts$_k_error,
anlrms$_closeout = shr$_closeout + 177*65536 + sts$_k_error,
anlrms$_openin = shr$_openin + 177*65536 + sts$_k_error,
anlrms$_openout = shr$_openout + 177*65536 + sts$_k_severe,
anlrms$_readerr = shr$_readerr + 177*65536 + sts$_k_error,
anlrms$_writeerr = shr$_writeerr + 177*65536 + sts$_k_severe;

```


The image displays a grid of 100 small terminal window screenshots, arranged in a 10x10 grid. Each window shows a different VAX/VMS command and its output. The windows are arranged in a grid, with some windows being more prominent than others. The following table lists the labels for the windows that are clearly visible and have larger text overlaid on them:

Row	Column	Label
1	1	SETSHOAL
1	5	OB EXREQ REQ
1	7	EXEFLXP LIS
2	5	ANALYZRMS MAP
2	7	EXESTUFF LIS
3	1	SHOWAL
3	7	EXEINPUT LIS
4	1	ANALYZ
4	7	EXEDTIVE LIS
4	8	RMSREQ REQ
5	1	ANALYZOB MAP