


```

EEEEEEEEEE RRRRRRRR FFFFFFFFFF DDDDDDDD EEEEEEEEEE FFFFFFFFFF
EEEEEEEEEE RRRRRRRR FFFFFFFFFF DDDDDDDD EEEEEEEEEE FFFFFFFFFF
EE          RR      RR FF          DD      DD EE          FF
EE          RR      RR FF          DD      DD EE          FF
EE          RR      RR FF          DD      DD EE          FF
EE          RR      RR FF          DD      DD EE          FF
EEEEEEEEEE RRRRRRRR FFFFFFFFFF DD      DD EEEEEEEEEE FFFFFFFFFF
EEEEEEEEEE RRRRRRRR FFFFFFFFFF DD      DD EEEEEEEEEE FFFFFFFFFF
EE          RR  RR  FF          DD      DD EE          FF
EE          RR  RR  FF          DD      DD EE          FF
EE          RR      RR FF          DD      DD EE          FF
EEEEEEEEEE RR      RR FF          DDDDDDDD EEEEEEEEEE FF
EEEEEEEEEE RR      RR FF          DDDDDDDD EEEEEEEEEE FF

```

```

....
....
....
....

```

```

RRRRRRRRR EEEEEEEEEE QQQQQQ
RRRRRRRRR EEEEEEEEEE QQQQQQ
RR      RR EE          QQ      QQ
RR      RR EE          QQ      QQ
RR      RR EE          QQ      QQ
RR      RR EE          QQ      QQ
RRRRRRRRR EEEEEEEEEE QQ      QQ
RRRRRRRRR EEEEEEEEEE QQ      QQ
RR  RR  EE          QQ  QQ  QQ
RR  RR  EE          QQ  QQ  QQ
RR      RR EE          QQ      QQ
RR      RR EE          QQ      QQ
RR      RR EEEEEEEEEE QQQQ  QQ
RR      RR EEEEEEEEEE QQQQ  QQ

```

Require file for the ERF facility
Version '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: ERF, Errorlog Report Formatter

ABSTRACT:

This file contains definitions of general applicability to the error log report formatter (ERF) facility.

ENVIRONMENT:

VAX/VMS operating system. unprivileged user mode.

AUTHOR: Elliott A. DRAYTON

7-Feb-1983

Modified by:

V03-006	SAR0283	Sharon A.Reynolds	2-Jul-1984
	Added build_fao_string and format string macros.		
V03-005	EAD0144	Elliott A. Drayton	12-Apr-1984
	Changed library to LIB.		
V03-004	SAR0247	Sharon A. Reynolds	9-Apr-1984
	Added cstring and store_strings macros.		
V03-003	JMG0004	Joel M. Gringorten	29-Dec-1983

Added literals for histogram support.

V03-002 SAR0012 Sharon Reynolds, 18-Apr-1983
Added literals for /summary support.

V03-001 EAD0091 Elliott A. Drayton 13-Apr-1983
Add literals for report modes.

SWITCHES ADDRESSING_MODE(EXTERNAL=GENERAL, NONEXTERNAL=GENERAL);

Psect global = EMB (pic, global, overlay, share, noexecute, read, write);
Psect global = SYECOM (pic, global, overlay, share, noexecute, read, write);

PSECT

CODE = \$code (PIC, ADDRESSING_MODE(GENERAL)),
OWN = \$OWN\$ (PIC, ADDRESSING_MODE(GENERAL)),
GLOBAL = \$GLOBAL\$ (PIC, ADDRESSING_MODE(GENERAL)),
PLIT = \$plit (PIC, ADDRESSING_MODE(GENERAL));

LIBRARY 'SYSS\$LIBRARY:LIB'; ! VAX/VMS common definitions

LITERAL

All_summ_out = 1,	
Brief_rep = 1,	Brief report type
Dev_summ_out = 2,	Output device summary type
Dev_summ_upd = 3,	Update device summary type
Entry_summ_out = 4,	Output entry summary type
Entry_summ_upd = 5,	Update entry summary type
Full_rep = 2,	Full report type
Memory_summ_out = 6,	Output memory summary type
Reg_dump_rep = 3,	Register dump report type
Volume_summ_out = 7,	Output volume summary type
Histo_summ_upd = 8,	Update histogram summary type
Histo_summ_out = 9,	Output histogram summary type
True = 1,	
False = 0,	
DC\$_ZERO_CLASS = 0;	

FIELD

Desc_fields = SET
Desc_one = [0,0,32,0],
Desc_two = [2,0,32,0]
TES;

MACRO

Define the SD macro, which will generate static string descriptors.

SD[A] = Global bind %name(A, '_DESC') = \$DESCRIPTOR(A) %,

```
! Define message vector offsets
```

```
!
msg$w_arg_cnt = 0,0,16,1 %,
msg$w_msg_flg = 2,0,16,1 %,
msg$l_msg_id = 4,0,32,0 %,
msg$w_new_flg = 8,0,16,1 %,
msg$w_fao_cnt = 10,0,16,1 %,
msg$l_fao_arg1 = 12,0,32,0 %,
msg$l_fao_arg2 = 16,0,32,0 %,
msg$l_fao_arg3 = 20,0,32,0 %,
driver_type = 18,0,0,0 %.
```

```
! CLI PARSING MACROS --
```

```
! Determine if a command line entity is present or get its value
```

```
! GET_PRESENT (STRING) =
```

```
! BEGIN
! EXTERNAL ROUTINE CLIP$PRESENT: ADDRESSING_MODE (GENERAL);
! CLIP$PRESENT (AD (STRING))
! END%,
```

```
! GET_VALUE (STRING, DESC) =
```

```
! BEGIN
! EXTERNAL ROUTINE CLIG$GET VALUE: ADDRESSING_MODE (GENERAL);
! CLIG$GET_VALUE (AD (STRING), DESC)
! END%,
```

```
! CALL_FUNCTION MACRO --
```

```
! This macro executes COMMAND and test the returned status.
```

```
! CALL_FUNCTION (command) =
```

```
! BEGIN
! LOCAL
! status;
!
! status = command;
! IF NOT .status
! THEN
! RETURN .status;
! END%,
```

```
! A) Macro to describe a string
```

```
! B) Macro to generate a quadword string descriptor
```

```
! C) Macro to generate the address of a string descriptor
```

```
! D) Macro to abbreviate last macro
```

```
! PRIMDESC [] = %CHARCOUNT (%STRING (%REMAINING)),
! UPLIT (%STRING (%REMAINING))%,
! INITDESC [] = %BLOCK [DSC%_S_BLN]
```

```

ADDRDESC [] = INITIAL (PRIMDESC (%REMAINING))%,
AD [] = UPLIT (PRIMDESC (%REMAINING))%,
AD [] = ADDRDESC (%REMAINING)%,

```

Define a macro that generates pointers to counted ascii strings.

```
CSTRING (string) = Uplit byte (%ASCIC string) %,
```

Define a macro (PUT_STRING) that will describe a string.

Define a macro (STORE_STRINGS) that will allocate a block of quadword string descriptors and initialize them with the specified text.

call format:

```

STORE_STRINGS (name of descriptor table (to be appended to '_desc_tbl'),
               'string1',
               'string2') ;

```

```

PUT_STRING [string] = %CHARCOUNT(%STRING(string)),
                    Uplit (%STRING(string)) % ,

```

```
STORE_STRINGS (name) =
```

```

  Literal
  %name(%string(name, '_number')) = (%length - 1) ;

```

```

  OWN
  %name(%string(name, ' desc_tbl')) :
    BLOCKVECTOR [%name(name, ' number'), 2]
    Initial (PUT_STRING(%REMAINING)) ; %,

```

Define a macro that will build an fao control string.

repeat_cnt = number of times to repeat the string.

string = the string.

dst_desc = address of destination descriptor.

```
BUILD_FAO_STRING (repeat_cnt, string, dst_desc) [] =
```

```
  Begin
```

```
  Bind str_desc = %ASCID string : $BBLOCK ;
```

```
  Incr I from 1 to repeat_cnt do
```

```
    Begin
```

```
    CH$MOVE (.str_desc[dsc$w_length],
            .str_desc[dsc$a_pointer],
            .dst_desc[dsc$a_pointer] + .dst_desc[dsc$w_length]) ;
```

```
    Dst_desc[dsc$w_length] = .dst_desc[dsc$w_length] +
                            .str_desc[dsc$w_length] ;
```

```
    End ;
```

```
  BUILD_FAO_STRING(%REMAINING) ;
```

```
  End %,
```

Define a macro that will call \$FAOL to format a string.

```
FORMAT_STRING (fao_str, dst_desc, arg_lst) =
```

```

Begin
If NOT (status = $FAOL( CTRSTR = .fao_str,
                        OUTBUF = dst_desc,
                        OUTLEN = dst_desc,
                        PRMLST = arg_lst) )
Then
Signal_stop (.status) ;
End % ,

```

SSHR_MESSAGES - a macro which defines facility-specific message codes which are based on the system-wide shared message codes.

```
SSHR_MESSAGES( name, code, (msg,severity), ... )
```

where:

```

"name" is the name of the facility (e.g., COPY)
"code" is the corresponding facility code (e.g., 103)
"msg" is the name of the shared message (e.g., BEGIN)
"severity" is the desired message severity (e.g., 1, 0, 2)

```

```
SSHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
```

```

LITERAL
SHRMSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,

```

```
SHRMSG_IDS( FACILITY_NAME, FACILITY_CODE ) [ VALUE ] =
SHRMSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,

```

```
SHRMSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG ID, SEVERITY ) =
%NAME( FACILITY_NAME, '$ ', MSG ID ) = %NAME( 'SHR$', MSG_ID ) + FACILITY_CODE*65536 +
%IF %DECLARED(%NAME( 'STSS$', SEVERITY ))
%THEN %NAME( 'STSS$', SEVERITY )
%ELSE SEVERITY %FI %;

```

: :
DEFINE MESSAGE CODES : :

\$SHR_MESSAGES (MSG,8, ! COMMON I/O AND MISC. MESSAGES

: :

* NAME * * SEVERITY * * DESCRIPTION *

(SEARCHFAIL, ERROR), ! Error while searching for file
(OPENIN, ERROR), ! Unable to open or connect to input
(READERR, ERROR), ! Error while reading input
(CLOSEIN, ERROR), ! Unable to close output
(OPENOUT, ERROR), ! Unable to create, open, or connect
to output
(WRITEERR, ERROR), ! Error while writing output
(CLOSEOUT, ERROR), ! Unable to close output
(CREATED, INFO), ! File (or directory) created
(EXISTS, INFO), ! File (or directory) already exists
(SYNTAX, SEVERE), ! Parse failure
(INVQUAVAL, SEVERE), ! Invalid qualifier value
(CONFQUAL, INFO), ! Conflicting qualifiers
);

This page contains a grid of 100 small technical diagrams and tables, arranged in 10 rows and 10 columns. Each cell contains a different type of diagram or table, often with a title. The titles include:

- ERFPROC4 MAP
- EMBETDEF REQ
- DEVERR FOR
- MODES FOR
- SSMESSAGE FOR
- BUGCHK LIS
- ERFPROC3 MAP
- ERFDEF REQ
- MSGHDR FOR
- SYECOM FOR
- BSDRIVER LIS
- ERFRTIM MAP
- ERFTAPE MAP
- PARSERDAT SDL
- DEV CLASS FOR
- DR32COM FOR
- OPCODES FOR
- VOLMOUNT FOR
- ERFSUMM MAP
- EMBLMDEF FOR
- QIOCOMMON FOR
- ERFPROC5 MAP
- DEVERRDEF REQ
- RECSELDEF REQ
- BRIEF LIS
- EMBSPDEF FOR

The diagrams and tables are small-scale technical drawings, likely representing hardware configurations, data structures, or system requirements. Many of the tables appear to be lists or matrices of data, while others are more schematic in nature.