

UU	UU	TTTTTTTTTT	IIIIII	LL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	
UU	UU	TTTTTTTTTT	IIIIII	LL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UU	UU	TT	II	LL	DD	DD	FF	
UUUUUUUUUU		TT	IIIIII	LLLLLLLLLL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF
UUUUUUUUUU		TT	IIIIII	LLLLLLLLLL	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF

RRRRRRRR	EEEEEEEEEE	QQQQQQ		
RRRRRRRR	EEEEEEEEEE	QQQQQQ		
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RRRRRRRR	EEEEEEEEEE	QQ	QQ	QQ
RRRRRRRR	EEEEEEEEEE	QQ	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EE	QQ	QQ
RR	RR	EEEEEEEEEE	QQQQ	QQ
RR	RR	EEEEEEEEEE	QQQQ	QQ



Commonly used definitions for VMS modules written in BLISS

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.
*
*****

```

++
ABSTRACT:

This is the common require file for any module written
in BLISS

ENVIRONMENT:

VAX/VMS operating system.

AUTHOR: Tim Halvorsen, Feb 1980

MODIFIED BY:

V03-001 MHB0127 Mark Bramhall
Added the MOVE_QUAD macro.

5-Apr-1984

Equated symbols

```
LITERAL
  true      = 1,           ! boolean true
  false     = 0,           ! boolean false
  ok        = 1,           ! success return code
  error     = 2,           ! error return code
  quad      = 8;           ! quadword allocation definition
```

Define structure type for VMS structures

```
STRUCTURE
  bblock [o, p, s, e; n] =
    [n]
    (bblock+o)<p,s,e>;

MACRO
  move_quad (src, dst) =      ! Move a quadword
  BEGIN
    (dst)+0 = .(src)<0, 32>;
    (dst)+4 = .(src)<32, 32>;
  END%;

MACRO
  descriptor [] =             ! Generate a static string descriptor
  UPLIT (%CHARCOUNT (%STRING (%REMAINING))),
  UPLIT BYTE (%STRING (%REMAINING))) %;

MACRO
  own_descriptor [] =         ! Generate the actual static string descriptor
  %BBLOCK [8] INITIAL(%CHARCOUNT(%STRING(%REMAINING))),
  UPLIT BYTE (%STRING(%REMAINING))) %;

MACRO
  return_if_error(command) =
  BEGIN
  LOCAL
    status;

    status = command;
    IF NOT .status
    THEN
      RETURN .status;
  END%;

MACRO
  signal_if_error(command) =
  BEGIN
  LOCAL
    status;

    status = command;
```

```

IF NOT .status
THEN
  BEGIN
    SIGNAL(.status);
    RETURN .status;
  END;
END%;

```

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	-->	4
F(SEVERE_ERROR)	4	-->	7

```

MACRO
severity_level (status) =
  BEGIN
    LOCAL code: BBLOCK [LONG];
    code = status;
    .code [sts$v_severity] - (4 * .code [sts$v_success]) + 3
  END%;

```

```

MACRO
cli$external(prefix) =
  %IF %DECLARED(%QUOTE %QUOTE cli$prefix)
  %THEN UNDECLARE %QUOTE %QUOTE cli$prefix; %FI
  MACRO cli$prefix = prefix %QUOTE %;
  EXTERNAL LITERAL
  cli$external_loop(%REMAINING)%,

cli$external_loop[name] =
  %NAME(cli$prefix,name): UNSIGNED(8)%;

```

```

MACRO
$external_literal(symbol) =
  BEGIN
    %IF NOT %DECLARED(symbol) %THEN EXTERNAL LITERAL symbol
    %IF %LENGTH GTR 1 %THEN : %REMAINING %FI; %FI
  symbol
  END%;

```

```

MACRO
$fab_dev(dev bit) = ! Access FAB$L_DEV bits of FAB block
  %BYTEOFFSET(fab$L dev),
  %BITPOSITION(%NAME('dev$v_',dev_bit)),1,0%;

```

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)
```

MACRO

```
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('SHRS_', MSG_ID) + FACILITY_CODE*65536 +
  %IF %DECLARED(%NAME('STSSK ', SEVERITY))
  %THEN %NAME('STSSK ', SEVERITY)
  %ELSE SEVERITY %FI-%;
```

