

CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCC	DDD	UUU	UUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU
CCCCCCCCCCCC	DDDDDDDDDDDD	UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU

```

GGGGGGGG  EEEEEEEEE  NN      NN  RRRRRRRR  AAAAAA  LL      RRRRRRRR  EEEEEEEEE  QQQQQQ
GGGGGGGG  EEEEEEEEE  NN      NN  RRRRRRRR  AAAAAA  LL      RRRRRRRR  EEEEEEEEE  QQQQQQ
GG      EE      NN      NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      EE      NN      NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      EE      NNNN     NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      EE      NNNN     NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      EEEEEEE  NN  NN  NN  RRRRRRRR  AA      AA  LL      RRRRRRRR  EEEEEEE  QQ      QQ
GG      EEEEEEE  NN  NN  NN  RRRRRRRR  AA      AA  LL      RRRRRRRR  EEEEEEE  QQ      QQ
GG      GG      NN      NNNN  RR  RR  AAAAAAAAAA  LL      RR  RR  EE      EE      QQ  QQ  QQ
GG      GG      NN      NNNN  RR  RR  AAAAAAAAAA  LL      RR  RR  EE      EE      QQ  QQ  QQ
GG      GG      NN      NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      GG      NN      NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GG      GG      NN      NN  RR      RR  AA      AA  LL      RR      RR  EE      EE      QQ      QQ
GGGGGGGG  EEEEEEEEE  NN      NN  RR      RR  AA      AA  LLLLLLLLLL  RR      RR  EEEEEEEEE  QQQQ  QQ
GGGGGGGG  EEEEEEEEE  NN      NN  RR      RR  AA      AA  LLLLLLLLLL  RR      RR  EEEEEEEEE  QQQQ  QQ

```

```

RRRRRRRR  333333  222222
RRRRRRRR  333333  222222
RR      RR  33      33  22      22
RR      RR  33      33  22      22
RR      RR  33      33  22      22
RR      RR  33      33  22      22
RRRRRRRR  33      22
RRRRRRRR  33      22
RR  RR      33      22
RR  RR      33      22
RR      RR  33      22
RR      RR  33      22
RR      RR  333333  2222222222
RR      RR  333333  2222222222

```

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: Standard Bliss Require File
Abstract: This require file contains standard definitions and macros
          useful for all kinds of Bliss programming. This file is
          general, not specific to any facility, and embodies all of
          the constructs found to be useful over many years of Bliss
          programming.
Environment: As defined by the facility using this require file.
Author: Paul C. Anagnostopoulos
Creation: 18 November 1982
Modifications:
--

```

```
!      N E W   D A T A   T Y P E S
!      -----
```

```
! The following items define a boolean data type, which does not exist in
! the standard Bliss language.
```

```
literal
  true = 1,
  false = 0;
```

```
macro
  boolean = byte %;
```

```
! The following items define a general pointer data type.
```

```
macro
  pointer = ref block[,byte] %;
```

```
! The following literals define the most frequently used control characters.
```

```
literal
  NUL = %x'00',
  ETX = %x'03',
  BEL = %x'07',
  HT  = %x'09',
  LF  = %x'0a',
  FF  = %x'0c',
  CR  = %x'0d',
  ESC = %x'1b',
  DEL = %x'7f';
```

```
! The following items define some simple macros for generating the various
! kinds of strings we use: counted, described by descriptor, zero-delimited.
```

```
macro
  ctext(string) = uplit byte(%ascic string) %,
  dtext(string) = %ascid string %,
  ztext(string) = uplit byte(%asciz string) %;
```

```
! The following items define a descriptor data type, which does not exist
! in the Bliss language. These constructs allow simpler manipulation of
! descriptors and the data they describe.
```

```
field
  descriptor_fields = set
    len = [dsc$w_length],
    typ = [dsc$b_dtype],
    cls = [dsc$b_class],
    ptr = [dsc$a_pointer]
  tes;
```

```
macro
  descriptor =
    block[8,byte] field(descriptor_fields) %,
  dbuffer(name,length) =
```

```
name: block[8+length,byte] field(descriptor_fields) preset(
    [len] = length,
    [typ] = 0,
    [cls] = 0,
    [ptr] = name + 8
) %;

with_dbuffer(name,length)[] =
begin
local
    name: block[8+length,byte] field(descriptor_fields);
    name[len] = length;
    name[typ] = 0;
    name[cls] = 0;
    name[ptr] = name + 8;
%remaining
end %;
```

! NEW CONSTRUCTS

! Define a macro that allows reference to an external message name without
! having to declare it up top.

```
macro
  msg(ident) =
    (%if not %declared(ident) %then
      external literal
        ident;
    %fi
    ident
    ) %;
```

! Define a macro which will check a status code and signal an error.

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

! The following literal will set the bits in the \$FA0 count of a \$PUTMSG
! argument list so that the facility, severity, and identification won't
! be included.

```
literal
  nobabble = %x'00010000';
```

! Define two macros from incrementing and decrementing a variable.

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

! Define a macro for rounding an integer up to a specified multiple.

```
macro
  round_up(expression,multiple) = (((expression)+(multiple)-1) / (multiple) * (multiple)) %;
```

! Define a macro for the head of an "infinite" loop.

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

! Define a macro for building a descriptor at runtime.

```
macro
  build_descriptor(name,length,address) =
    (name[len] = length;
     name[typ] = name[cls] = 0;
     name[ptr] = address;) %;
```

GENRAL REQ R32	EXTCAL LIS
CLISDEF R32	GENCODE4 LIS
CDUMSGS LIS	CDU
CDU MAP	GENCODE1 LIS
CDUREQ R32	GENCODE3 LIS
CDUTPODEF LIS	GENCODE2 LIS