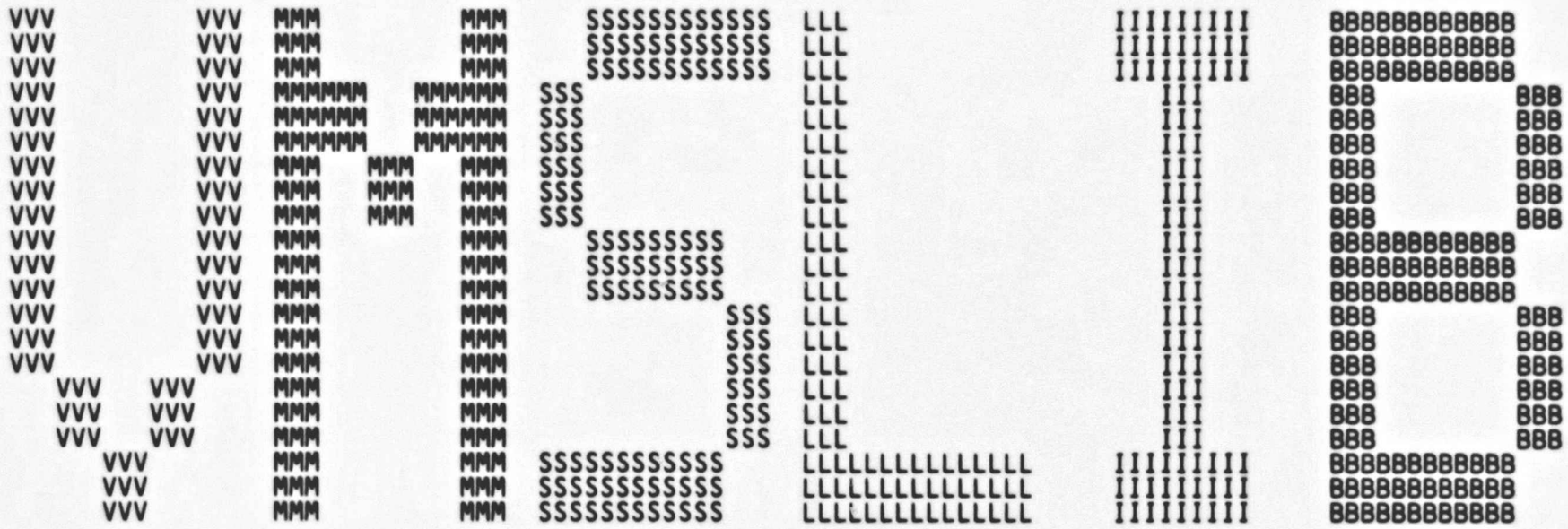
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB

FILEID**UTLDEFB

```
UU      UU  TTTTTTTTTT  LL          DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  BBBBBBBB
UU      UU  TTTTTTTTTT  LL          DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  BBBBBBBB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EEEEEEEE    FFFFFFFF    BBBBBBBB
UU      UU      TT      LL          DD      DD  EEEEEEEE    FFFFFFFF    BBBBBBBB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB      ....
UU      UU      TT      LL          DD      DD  EE          FF          BB      BB      ....
UUUUUUUUUU      TT      LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE  FF          BBBBBBBB        ....
UUUUUUUUUU      TT      LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE  FF          BBBBBBBB        ....

BBBBBBBB      333333      222222
BBBBBBBB     333333      222222
BB      BB  33      33  22      22
BB      BB  33      33  22      22
BB      BB          33          22
BB      BB          33          22
BBBBBBBB        33          22
BBBBBBBB        33          22
BB      BB          33      22
BB      BB          33      22
BB      BB  33      33      22
BB      BB  33      33      22
BBBBBBBB      333333      2222222222
BBBBBBBB      333333      2222222222
```

```
Version:      'V04-000'
```



```
++

UTLDEF.B32 - UTILITY DEFINITION MACROS FOR BLISS PROCESSING
OF STARLET DEFINITION MACROS.

MODIFIED BY:

        V03-005 GRR3005         Greg Robert     12-Nov-1982
                Added an allocation check to $ITMLST_INIT and
                modified to produce better code.

        V03-004 SBL3004         Steve Lionel    8-Nov-1982
                Change error message in $ASSUME so as to not try to
                display the macro arguments, since BLISS can't handle
                expressions in %ERRORMACRO.

        V03-003 GRR3003         Greg Robert     22-Oct-1982
                Add $ITMLST macros and structure definition to
                define and initialized item lists.

        V03-002 SBL3002         Steve Lionel    21-Oct-1982
                Add $ASSUME macro to verify compiletime assumptions.

        V03-001 BLS0181         Benn Schreiber  27-Jul-1982
                Add $init_dyndesc macro to initialize a dynamic
                descriptor

--
```

```
!
! MACROS TO EXTRACT OFFSETS, FIELD WIDTHS, ETC., FROM FIELD EXTRACTION MACROS.
!
MACRO   $BYTEOFFSET (OFFSET, POSITION, WIDTH, SIGN) = OFFSET%;

MACRO   $BITPOSITION (OFFSET, POSITION, WIDTH, SIGN) = POSITION%;

MACRO   $FIELDWIDTH (OFFSET, POSITION, WIDTH, SIGN) = WIDTH%;

MACRO   $EXTENSION (OFFSET, POSITION, WIDTH, SIGN) = SIGN%;

MACRO   $FIELDMASK (OFFSET, POSITION, WIDTH, SIGN) =
        (1^(POSITION+WIDTH) - 1^POSITION)%;

!
! MACRO TO GENERATE EQULST CONSTRUCTS.
!
MACRO
        $EQULST(P,G,I,S)[A]=
            %NAME(P,GET1ST_ A) =
                %IF NUL2ND_ A
                %THEN (I) + %COUNT*(S)  ! ASSUMES I, S ALWAYS GENERATED BY CONVERSION PROGRAM
                %ELSE GET2ND_ A
                %FI %,

        GET1ST_(A,B)=
             A %,
        GET2ND_(A,B)=
             B %,           ! KNOWN NON-NULL
        NUL2ND_(A,B)=
             %NULL(B) %;

!
! MACRO TO GENERATE A STRING DESCRIPTOR PLIT
!

MACRO
        $DESCRIPTOR[]=
                UPLIT(%CHARCOUNT(%REMAINING),UPLIT BYTE(%REMAINING))%;
!
! $SHR_MSGDEF - a macro which defines facility-specific message codes
!       which are based on the system-wide shared message codes.
!
!       $SHR_MSGDEF( name, code, scope, (msg,severity), ... )
!
!       where:
!               name    is the name of the facility (e.g., COPY)
!               code    is the corresponding facility code (e.g., 103)
!               scope   is GLOBAL for global definitions (anything else gets
!                               LOCAL)
!               msg     is the name of the shared message (e.g., BEGIN)
!               severity is the desired message severity (e.g., 1, 0, 2, or
!                               WARNING, SUCCESS, INFO, ERROR, SEVERE)
!
```

```
MACRO
    $SHR_MSGDEF( FACILITY_NAME, FACILITY_CODE, SCOPE) =
        %IF %IDENTICAL(%STRING(SCOPE),'GLOBAL')
            %THEN GLOBAL LITERAL
        $SHR_MSGIDS( FACILITY_NAME, FACILITY_CODE, %REMAINING );
            %ELSE LITERAL
        $SHR_MSGIDS( FACILITY_NAME, FACILITY_CODE, %REMAINING );
        %FI%,

    $SHR_MSGIDS( FACILITY_NAME, FACILITY_CODE) [ VALUE ] =
        $SHR_MSGCALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,

    $SHR_MSGCALC( FACILITY_NAME, FACILITY_CODE, MSG_ID, SEVERITY ) =
        %NAME(FACILITY_NAME,'$ ',MSG_ID) = %NAME('SHR$_',MSG_ID) + FACILITY_CODE*65536 +
            %IF %DECLARED(%NAME('STS$K_',SEVERITY))
                %THEN %NAME('STS$K_',SEVERITY)
                %ELSE SEVERITY %FI %;
!
! Define VMS block structures (equivalent to BLOCK[,BYTE])
!
STRUCTURE
        $BBLOCK [O, P, S, E; N] =
                [N]
                ($BBLOCK+O)<P,S,E>;
!
! Macro to initialize a dynamic descriptor
!
MACRO
        $init_dyndesc(d) =
            begin
                d[dsc$w_length] = 0;
                d[dsc$b_class] = dsc$k_class_d;
                d[dsc$b_dtype] = dsc$k_dtype_t;
                d[dsc$a_pointer] = 0;
                end%;
```

```
DVI

DVI
CYL

; F
DVI
FRE

; L
DVI
LOG

;
;NA
;

; V
DVI
VOL

; V
DVI
VOL

; R
DVI
ROO

;
;NA
;

; N
DVI
NEX

; T
DVI
TRA

; M
DVI
MOU

;
;NA
;

; C
DVI
CLU

; M
DVI
MAX

; S
```

```
!++
!
! FUNCTIONAL DESCRIPTION:
!       These macros facilitate the allocation and initialization
!       of item lists in Bliss.  The lists are suitable for use
!       with GETDVI, GETSYI, GETJPI etc.
!
! MACROS:
!       $ITMLST_DECL    - allocates storage and declares the structure
!       $ITMLST_INIT    - dynamically initializes an item list
!       $ITMLST_UPLIT   - generates an UPLIT to a static (read-only) item list
!
! INPUT PARAMETERS:
!       ITEMS   - Number of items in the list (default=1)
!       ITMLST  - Address of the item list
!       ITMCOD  - Item to be obtained
!       BUFSIZ  - Size of the buffer to receive the item (default=4)
!       BUFADR  - Address of the buffer to receive the item
!       RETLEN  - Address of word to receive the resultant item size (default=0)
!
! EXAMPLE:
!
!    LOCAL
!       LIST: $ITMLST_DECL (ITEMS=4),
!       DEVCLASS,
!       DEVTYPE,
!       DEVDEPEND,
!       DEVNAM: VECTOR [64, BYTE],
!       DEVNAMSIZ;
!
!    BEGIN
!    $ITMLST_INIT (ITMLST=LIST,
!       (ITMCOD=DVI$K_DEVCLASS, BUFADR=DEVCLASS),
!       (ITMCOD=DVI$K_DEVTYPE, BUFADR=DEVTYPE),
!       (ITMCOD=DVI$K_DEVDEPEND, BUFADR=DEVDEPEND),
!       (ITMCOD=DVI$K_DEVNAM, BUFADR=DEVNAM, BUFSIZ=64, RETLEN=DEVNAMESIZ)
!       );
!
!    $GETDVI (ITMLST=LIST, DEVNAM=$DESCRIPTOR('SYS$OUTPUT'));
!    END;
!
! STRUCTURES:
!       $ITMBLK [items, item_size, allocation_unit]
!
!       This structure defines an item list as a blockvector with
!       a trailing longword used to terminate the list.
!
!       You must specify the number of items in the list.  The size of
!       each item defaults to 12 and the allocation unit defaults to BYTE.
!
!       Fields in an item list declared with this structure can be
!       referenced in the following way:
!
!               item_list_address [item_number, field_specifier]
!
!       For example, ITMLST [3, ITM$W_ITMCOD] references the item code
```

```
!       field of the third item in the item list.
!


!
!       First define macro's to access item fields
!

MACRO
    ITM$S_ITEM = 12 %,                  ! Item list member size
    ITM$W_BUFSIZ = 0,0,16,0 %,          ! Target buffer size
    ITM$W_ITMCOD = 2,0,16,0 %,          ! Item code
    ITM$L_BUFADR = 4,0,32,0 %,          ! Target buffer address
    ITM$L_RETLEN = 8,0,32,0 %;          ! Address of word to receive length


!
!       Define an item list structure
!

STRUCTURE
        $ITMBLK [I, O, P, S, E; N, BS=ITM$S_ITEM, UNIT=1] =
                [N*BS*UNIT+4]
                ($ITMBLK+(I*BS+O)*UNIT)<P,S,E>;


!
!       Define the allocation macro
!

KEYWORDMACRO
    $ITMLST_DECL (ITEMS=1) = $ITMBLK [ITEMS] % ;


!
!       Define the list initialization macro
!

MACRO
    $ITMLST_INIT (ITMLST) [ITEM_VALUES] =
        %IF %COUNT EQL 0
        %THEN
            $$ITM_INITIATE (ITMLST, NUMITM = %LENGTH - 1)
        %FI

        $$ITM_INIT (%REMOVE (ITEM_VALUES))

        %IF %COUNT EQL %LENGTH - 2
        %THEN
            $$ITMBLKPTR [0, 0, 32, 0] = 0;
            $$ITMBLKPTR = $$ITMBLKPTR + 4
            END
        %FI
    % ;
```

```
!
!       Define the list initiation macro
!
KEYWORDMACRO
        $$ITM_INITIATE (ITMLST, NUMITM) =
            %IF %ALLOCATION (ITMLST) LSSU ((NUMITM) * ITM$S_ITEM + 4)
            %THEN
                %ERRORMACRO ('initialization data exceeds allocation of ', ITMLST)
            %FI

            BEGIN
            LOCAL $$ITMBLKPTR: REF $BBLOCK;
            $$ITMBLKPTR = (ITMLST);
        % ;


!
!       Define the item initialization macro
!
KEYWORDMACRO
        $$ITM_INIT (
            BUFSIZ=4,           ! Size of return buffer
            ITMCOD,             ! Item code
            BUFADR,             ! Address of return buffer
            RETLEN=0            ! Address of word to receive resultant length
        ) =

        %IF %NULL (ITMCOD) OR %NULL (BUFADR)
        %THEN
            %ERRORMACRO ('ITMCOD and BUFADR must be specified')
        %FI

        $$ITMBLKPTR [ITM$W_BUFSIZ] = (BUFSIZ);
        $$ITMBLKPTR [ITM$W_ITMCOD] = (ITMCOD);
        $$ITMBLKPTR [ITM$L_BUFADR] = (BUFADR);
        $$ITMBLKPTR [ITM$L_RETLEN] = (RETLEN);
        $$ITMBLKPTR = .$$ITMBLKPTR + ITM$S_ITEM;
        % ;


!
!       Define the static list macro
!
MACRO
        $ITMLST_UPLIT [] =
            UPLIT ( $$ITMLST_UPLIT_REPEAT (%REMAINING) , LONG (0)) % ;

!
!       Define the repetition macro
!
MACRO   $$ITMLST_UPLIT_REPEAT [ITEM] =
                $$ITMLST_UPLIT_ITEM ( %REMOVE (ITEM) )  % ;
```

```
!
!       Define the static item macro
!

KEYWORDMACRO
        $$ITMLST_UPLIT_ITEM (
            BUFSIZ=4,           ! Size of return buffer
            ITMCOD,             ! Item code
            BUFADR,             ! Address of return buffer
            RETLEN=0            ! Address of word to receive resultant length
        ) =

        %IF %NULL (ITMCOD) OR %NULL (BUFADR)
        %THEN
            %ERRORMACRO ('ITMCOD and BUFADR must be specified')
        %FI

            WORD (BUFSIZ, ITMCOD), LONG (BUFADR, RETLEN)
        % ;
```

```
!
! $ASSUME - Test compile-time relation assumptions
!
! Format:
!
!       $ASSUME (val-1, cond, val-2)
!
! val-1 - A compile-time constant expression to be compared with val-2.
! cond  - The relation that is to hold between val-1 and val-2.  For
!         example, EQL, LSSU, etc.
! val-2 - A compile-time constant expression to be compared with val-1.
!
! Result:
!       The empty sequence.  If the given relation does not hold, an
!       error message is issued.
!

MACRO
    $ASSUME (V1, COND, V2) =
        %IF NOT (V1 COND V2)
        %THEN %ERROR ('$ASSUME relationship does not hold') %FI %;
```

DVI
DVI
TT_
DVI
TT_
DVI
TT_
DVI
TT_
;
;NA
;
DVI
TT_
DVI
TT_
DVI
TT_
;
;
;
;
;
;NA
;
DVI
TT_
DVI
TT_
DVI
TT_
DVI
TT_
DVI
TT_
DVI
TT_
;
;NA
;

CLIMAC
REQ

TPAMAC
REQ

UTLDEFB
B32

STARMISC
MAR

EQDEFM
MAR

TPAMAR
MAR

CALBYNAME
LIS

SCRPROLOG
REQ

STARLET
SDL

SCRTCB
REQ

OFFSET
MAR

SYITABLE
MAR

EQDEFB
B32

JPITABLE
MAR

DVITABLE
MAR

UTLDEFM
MAR

B32MSG
LIS

SCRTERM
REQ

RSXLBLDF
MAR

C74MSG
LIS