```
   CCCCCCCCCCCC              JJJ  FFFFFFFFFFFFFFF  VVV         VVV  444      444
   CCCCCCCCCCCC              JJJ  FFFFFFFFFFFFFFF  VVV         VVV  444      444
   CCCCCCCCCCCC              JJJ  FFFFFFFFFFFFFFF  VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFF              VVV         VVV  444      444
CCC                          JJJ  FFFFFFFFFFFF     VVV         VVV  444444444444444
CCC                          JJJ  FFFFFFFFFFFF     VVV         VVV  444444444444444
CCC                          JJJ  FFFFFFFFFFFF     VVV         VVV  444444444444444
CCC              JJJ         JJJ  FFF              VVV         VVV           444
CCC              JJJ         JJJ  FFF              VVV         VVV           444
CCC              JJJ         JJJ  FFF              VVV         VVV           444
CCC              JJJ         JJJ  FFF                 VVV   VVV              444
CCC              JJJ         JJJ  FFF                 VVV   VVV              444
CCC              JJJ         JJJ  FFF                 VVV   VVV              444
   CCCCCCCCCCCC     JJJJJJJJJ     FFF                    VVV                 444
   CCCCCCCCCCCC     JJJJJJJJJ     FFF                    VVV                 444
   CCCCCCCCCCCC     JJJJJJJJJ     FFF                    VVV                 444
```

```
**FILE**ID**JNLDEFINT                                   D 10
```

```
        JJ  NN      NN  LL          DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    IIIIII    NN      NN  TTTTTTTTTT
        JJ  NN      NN  LL          DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    IIIIII    NN      NN  TTTTTTTTTT
        JJ  NN      NN  LL          DD      DD  EE          FF              II      NN      NN      TT
        JJ  NN      NN  LL          DD      DD  EE          FF              II      NN      NN      TT
        JJ  NNNN    NN  LL          DD      DD  EE          FF              II      NNNN    NN      TT
        JJ  NNNN    NN  LL          DD      DD  EE          FF              II      NNNN    NN      TT
        JJ  NN  NN  NN  LL          DD      DD  EEEEEEEE    FFFFFFFF        II      NN  NN  NN      TT
        JJ  NN  NN  NN  LL          DD      DD  EEEEEEEE    FFFFFFFF        II      NN  NN  NN      TT
JJ      JJ  NN    NNNN  LL          DD      DD  EE          FF              II      NN    NNNN      TT
JJ      JJ  NN    NNNN  LL          DD      DD  EE          FF              II      NN    NNNN      TT
JJ      JJ  NN      NN  LL          DD      DD  EE          FF              II      NN      NN      TT          ....
JJ      JJ  NN      NN  LL          DD      DD  EE          FF              II      NN      NN      TT          ....
  JJJJJJ    NN      NN  LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE  FF            IIIIII    NN      NN      TT          ....
  JJJJJJ    NN      NN  LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE  FF            IIIIII    NN      NN      TT          ....


  SSSSSSSS  DDDDDDDD    LL
  SSSSSSSS  DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
  SSSSSS    DD      DD  LL
  SSSSSS    DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
{       $begin  JNLDEFINT,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:  JOURNALING : DEFINITION OF INTERNAL SYMBOLICS
{
{ Abstract:
{       This module contains the symbolic definitions for non-user accessible
{       data structures.
{
{ Author:       Joost Verhofstad
{
{ Modified by:
{
{       V03-052 MKL0210         Mary Kay Lyons          16-DEC-1983
{               Add MCSID field to JNLMSG UCBDATA message.
{
{       V03-051 MKL0199         Mary Kay Lyons          29-NOV-1983
{               Add JNLMSG$W_JNL_PROT.
{
{       V03-050 LY0418          Larry Yetto             27-SEP-1983 15:10:06
{               Add EPID and ARB_PRIV to JNLCWQDEF
{
{       V03-049 LY0415          Larry Yetto             13-SEP-1983 10:40:11
{               Add REOCSB and some spare fields to JNLBTX structure
{
{       V03-048 MKL0168         Mary Kay Lyons          23-AUG-1983
{               Add STS field to JNLMSG UCBDATA message.
{
{       V03-047 LY0406          Larry Yetto             3-AUG-1983 08:57:57
{               Fix JNLMSGDATA structure.  Change IOSTS$V_RESUBS
{               to IOSTS$V_RESUB
{
```

```
{       V03-046 LY0405          Larry Yetto             2-AUG-1983 14:45:47
{               Add JNLMSGDATA structure
{
{       V03-045 LY0403          Larry Yetto             1-AUG-1983 15:18:18
{               Add JNLBXSTS$V_FNCTCMPL and JNLBXSTS$V_CNXBRK
{
{       V03-044 LY0399          Larry Yetto             28-JUL-1983 15:37:31
{               Add JNLBXSTS and JNLBTX structures to hold information relavant
{               to block transfer operations in progress that were initiated
{               from some other node.
{
{       V03-043 MKL0132         Mary Kay Lyons          24-JUL-1983
{               Change JNLRC to contain an offset to filter information.
{
{       V03-042 MKL0126         Mary Kay Lyons          10-JUL-1983
{               Remove JNLRM$B_JNLTYP definition.  Add JNLMSG
{               definitions for creating journaling I/O database.
{               Define JNLRC$Q_DATTIM to overlay JNLRC$O_RUID.
{               Keep the file version number in the JMT.  Remove
{               IOSTS$M_REM_WRITE and IOSTS$V_REM_WRITE.
{               Make journal names 12 bytes and various changes for send-
{               journal-message stuff.
{
{       V03-041 MKL0116         Mary Kay Lyons          22-JUN-1983
{               Add pointer to mount item list in the ADB.  Add
{               UPDATE_ADL message definitions.
{
{       V03-040 LY0383          Larry Yetto             16-JUN-1983 17:43:21
{               Move cluster message dispatch codes to [SYSLOA.SRC]CLUSTER.SDL
{
{       V03-039 PRB0196         Paul Beck               12-JUN-1983 14:20
{               Add RUE$V_NOFAC, RUE$V_NOOBJ.
{
{       V03-038 MKL0096         Mary Kay Lyons          01-Jun-1983
{               Add JNLRCDEF.
{
{       V03-037 MKL0093         Mary Kay Lyons          27-MAY-1983
{               Replace missing JNLMSGDEF.
{
{       V03-036 LY0373          Larry Yetto             24-MAY-1983 15:52:40
{               Add new BCB fields for high sequence number completely
{               in the buffer and written.  Add JNLCWQ structure.  Add
{               fields to overlay RUE$Q_RUID.
{
{       V03-035 MKL0087         Mary Kay Lyons          19-MAY-1983
{               Change JNLMSGDEF.
{
{       V03-034 JSV0289         Joost Verhofstad        18-MAY-1983
{               Reorganize and split up into:
{                       JNLDEFINT.SDL
{                       JNLSYSDEF.SDL
{                       JNLACPDEF.SDL
{                       JNLFILE.SDL
{
{       V03-033 LY0361          Larry Yetto             9-MAY-1983 12:32:19
{               Rename CJLMSG macro to CJFMSGFNC.  Add JNLACBM .
{               Add JNLLOG$V_SLVCRFAIL.  Remove JNLCB def.
{
{       V03-032 JSV0229         Joost Verhofstad        27-APR-1983
{               Add RUSYNC bits
{
{       V03-031 LY0355          Larry Yetto             20-APR-1983 10:03:27
{               Add cluster message dispatch codes CJLMSG macro and remove
{               the obsolete SCS message crap.
{               Remove ENT_TYPE codes and bit definitions from FLTR macro
{
{       V03-030 MKL0068         Mary Kay Lyons          08-APR-1983
{               Add RCB$L_LSTBLK1 and RCB$L_LSTBLK2.
{
{       V03-029 JSV0212         Joost Verhofstad        06-APR-1983
{               Change ACP filter to contain two date-time fields
{
{       V03-028 LY0346          Larry Yetto             6-APR-1983 11:03:17
{               Add the JNLCB structure.  This structure is the Journal control
{               block for slave nodes with no channels.
{
{       V03-027 MKL0062         Mary Kay Lyons          30-MAR-1983
{               Add JFTE$L_FRSTJVBN, RCB$L_LSTBLK1, and RCB$L_LSTBLK2.
{
{       V03-026 JSV190          Joost Verhofstad        14-MAR-1983
{               Add JFTE fields
{
{       V03-025 MKL0048         Mary Kay Lyons          24-FEB-1983
{               Update comments for JFTE$L_JMT and JFTE$L_DEVNAM.
{
{       V03-024 JSV0151         Joost Verhofstad        17-FEB-1983
{               Add JMT$L_BASEVBN and JMT$L_LTVBN and SFT$L_BASEVBN
{
{       V03-023 JSV0144         Joost Verhofstad        14-FEB-1983
{               Add BCB$M_NWVPR
{
{       V03-022 JSV0141         Joost Verhofstad        09-FEB-1983
{               Add JFTE$L_NEXTVER
{
{       V03-021 JSV0137         Joost Verhofstad        03-FEB-1983
{               replace source, put in null packet
{
{       V03-020 LY0245          Larry Yetto             10-JAN-1983
{               Move RUS structure to JNLDEF.SDL
{
{       V03-19  JSV0116         Joost Verhofstad        04-Jan-1983
{               Remove PROCNAME, BINARY, PROCNODE, PROCGROUP,
{               PROCRUNTIME fields from FLTR structure
{
{       V03-18  JSV0107         Joost Verhofstad        04-Jan-1983
{               Fix RCB fields + commentary
{
{       V03-17  JSV0106         Joost Verhofstad        30-Dec-1982
{               Add RCB fields
{
{       V03-16  JSV0105         Joost Verhofstad        12-Dec-1982
{               Add JFTE field
```

```
{
{       V03-15  JSV0097         Joost Verhofstad        23-Nov-1982
{               Add OPCHDR data structure
{
{       V03-14  JSV0087         Joost Verhofstad        28-Oct-1982
{               Add TBUF data structure and GTB, RCB, RHD
{               and JFTE symbols.
{
{       V03-13  JSV0078         Joost Verhofstad        08-Oct-1982
{               Add CJL data structure
{
{       V03-12  JSV0064         Joost Verhofstad        22-Sep-1982
{               Add a few GTB, JMT, JFTE fields for tape reading
{
{       V03-011 JSV0054         Joost Verhofstad        26-Aug-1982
{               Add FLTR$V_OUTRANGE and FLTR$S_OUTRANGE
{
{       V03-010 JAY0007         John A. Ywoskus         02-Aug-1982
{               Generate $M's for status bits in RUE and RUS.
{
{       V03-009 JAY0006         John A. Ywoskus         21-Jul-1982
{               Add INDEX field to RUS.
{
{       V03-008 JSV0024         Joost Verhofstad        21-Jul-1982
{               Add JNLLOG bits
{
{       V03-007 JAY0005         John A. Ywoskus         21-Jul-1982
{               Make RUE$W_JNLCNT be a longword. Add this field to RUS.
{
{       V03-006 JAY0004         John A. Ywoskus         15-Jul-1982
{               Change RUS structure. Delete WRFLG and add entry
{               attributes. Add COUNT field to NDL.
{
{       V03-005 JAY0003         John A. Ywoskus         12-Jul-1982
{               Add JNLCNT field to RUE.
{
{       V03-004 JSV00           Joost Verhofstad        7-Jul-1982
{               BUFFER$W_JNLID => BUFFER$L_JNLID
{
{       V03-003 JAY0002         John A. Ywoskus         06-Jul-1982
{               Rename RULIST structure to RUS. Change 'RESIDUAL'
{               status to RESID_FOR and RESID_BCK in RUE and RUS.
{               Add an 'INDEX' field to RUE.
{
{       V03-002 LY0028          Larry Yetto             29-Jun-1982
{               Added Name table Device List (NDL) definition
{
{       V03-001 JAY0001         John A. Ywoskus         17-Jun-1982
{               Added JNLDB, message structures for cluster journaling.
{               Delete RUDEF structure, replace with a version of RULIST.
{
{
```

```
module $CJFFLGDEF;
/*++
/*
/* CJFFLG - Flags that can be returned from SENSEMODE
/*
/*--


aggregate CJFFLGDEF  union fill prefix CJFFLG$;
    CJFFLGDEF_BITS structure fill;
        TAPE bitfield mask;                         /* this is a tape based journal
        SPOOL bitfield mask;                        /* the tape is being spooled at present
    end CJFFLGDEF_BITS;
end CJFFLGDEF;

end_module $CJFFLGDEF;
```

```
module $JNLBTXDEF;

/*++
/*
/* JNLBTX -     Journal block transfer
/*              This structure is used to define the offsets in the
/*              buffer allocated by CNX for our use with a block transfer.
/*
/*--


aggregate JNLBTXDEF structure fill prefix JNLBTX$;
    JNLBXSTS longword unsigned;                 /* Address if BXIP for this request
    RMBLK longword unsigned;                    /* Address of Remaster block
    REQCSB longword unsigned;                   /* Address of requestor's CSB
    SPARE1 longword unsigned;                   /* Spare longword
    SPARE2 longword unsigned;                   /* Spare longword
    SPARE3 longword unsigned;                   /* Spare longword
    constant LENGTH equals . tag K ;            /* Structure size
    constant LENGTH equals . tag C ;            /* Structure size

end JNLBTXDEF;

end_module $JNLBTXDEF;
```

```
module $JNLDMTDEF;
/*++
/*
/* JNLDMT - codes for the parameters passed with dismount journal
/*          medium. These codes are used to identify the parameters
/*          to the Journal ACP, when passed in the complex buffer.
/*
/*--
constant DNAM   equals 1  prefix JNLDMT tag $C;         /* device name parameter code
constant DGRPN  equals 2  prefix JNLDMT tag $C;         /* group name parameter code
constant FLAGS  equals 3  prefix JNLDMT tag $C;         /* flags value parameter code

end_module $JNLDMTDEF;
```

```
module $IOSTSDEF;
/*++
/*
/* IO status masks. These masks are in the third byte of IRP$L_IOSTS1
/* and are used during a write operation to indicate
/* the properties of the part (chunk) of the entry being written at the time,
/* and the status of the IO request at certain times.
/* The driver is the only one to use this I/O status field.
/*
/*--


aggregate IOSTSDEF  union fill prefix IOSTS$;
    IOSTSDEF_BITS structure fill;
        FSTCH bitfield mask;                            /* First entry
        MULCH bitfield mask;                            /* Multiple entries
        SEQNOVF bitfield mask;                          /* sequence number overflow
        WAITFIO bitfield mask;                          /* this IRP is waiting for buffer
                                                        /*  write to complete.
        REMOTE bitfield mask;                           /* This is an internal IRP and the
                                                        /* operation was started from a remote node
        RESUB bitfield mask;                            /* this request has not
                                                        /*  been resubmitted yet if set
    end IOSTSDEF_BITS;
end IOSTSDEF;

end_module $IOSTSDEF;
```

```
module $JNLMSGDEF;
/***
/*
/* JNLMSG - JNLACP - Driver Cluster Message Definitions
/*
/*--

aggregate JNLMSGDEF structure fill prefix JNLMSG$;
    FLINK longword unsigned;                               /* forward link
    BLINK longword unsigned;                               /* backward link
    SIZE word unsigned;                                    /* size of structure
    TYPE byte unsigned;                                    /* structure type
    SUBTYPE byte unsigned;                                 /* structure sub-type
    MSG_TYPE byte unsigned;                                /* message type
    FILL_1 byte dimension 3 fill prefix JNLMSGDEF tag $$;
    CSID longword unsigned;                                /* originator's CSID

    constant "HDRLEN" equals . prefix JNLMSG$ tag K;      /* header length
    constant "HDRLEN" equals . prefix JNLMSG$ tag C;      /* header length

    constant    (
        WRTBUFINF                 /* Write buffer information
        ,ALLDEV                   /* Add allocated device to ADL
        ,DEALLDEV                 /* Delete allocated device from ADL
        ,MNTDEV                   /* Add mounted device to ADL
        ,DMNTDEV                  /* Delete mounted device from ADL
        ,CRESLVDS                 /* Create slave data structures
        ) equals 1 increment 1 tag C;

end JNLMSGDEF;

/*
/* MESSAGE DEPENDENT EXTENSIONS
/*
/* MESSAGE 1 - Write buffer information
/*

aggregate JNLMSGDEF1 structure fill prefix JNLMSG$;
    FILL_1 byte dimension JNLMSG$C_HDRLEN fill prefix JNLMSGDEF1 tag $$;
    JNL_SEQN longword unsigned;         /* Highest jnl seq # written to disk
    LSEQNO longword unsigned;           /* Lowest local seq # outstanding
    SEQN_TCNT word unsigned;            /* total # writes in CWQ for which jnl seq # have been
                                        /*  assigned (1 seq # per follows)
    SEQN_CCNT word unsigned;            /* current count of writes in CWQ for which jnl seq # have been
                                        /*  assigned (1 seq # per follows)
    FILL_2 word unsigned fill prefix JNLMSGDEF1 tag $$;  /* spare

    constant MSG1_LEN equals . prefix JNLMSG$ tag K;    /* Size of fixed part MSG1
    constant MSG1_LEN equals . prefix JNLMSG$ tag C;    /* Size of fixed part MSG1

end JNLMSGDEF1;

aggregate JNLMSGDEF1_SEQN structure fill prefix JNLMSG$;
/*
/* there is one of these JNLMSGDEF1_SEQN pieces per entry in the CWQ for
/* which a journal seq # has been assigned, in the message
```

```
/*
    SEQ_NUM longword unsigned;          /* Entry journal sequence number
    FLAGS_OVERLAY union fill;
        FLAGS longword unsigned;        /* flags longword
        FLAGS_BITS structure fill;
            NEWVER bitfield mask;       /* Last write on a new version request
            PARTIAL bitfield mask;      /* Only part of the entry saved
        end FLAGS_BITS;
    end FLAGS_OVERLAY;

    constant "SEQENTLEN" equals . prefix JNLMSG$ tag K; /* length of sequence
    constant "SEQENTLEN" equals . prefix JNLMSG$ tag C; /* number information

end JNLMSGDEF1_SEQN;

/*
/* MESSAGE DEPENDENT EXTENSIONS
/*
/* MESSAGE 2, 3, 4, 5, - Update the ADL
/*

aggregate JNLMSGDEF2 structure fill prefix JNLMSG$;
    FILL_1 byte dimension JNLMSG$C_HDRLEN fill prefix JNLMSGDEF2 tag $$;
    STATUS word unsigned;               /* status of device
    ITMLSTLEN word unsigned;            /* Item list length (mount only)
    ITMLSTOFF word unsigned;            /* Offset to item list (mount only)
    DEVNUM word unsigned;               /* # of dev names which follow
    NAMELEN byte unsigned;              /* device name length
    DEVNAM byte unsigned dimension 15;  /* device name (ASCII)

    constant MSG2_LEN equals . prefix JNLMSG$ tag K;    /* Size of fixed part MSG2
    constant MSG2_LEN equals . prefix JNLMSG$ tag C;    /* Size of fixed part MSG2

end JNLMSGDEF2;

/*
/* MESSAGE DEPENDENT EXTENSIONS
/*
/* MESSAGE 6 - Create slave data structures
/* Each one byte item code in the message is followed by a longword which
/* is either the value or the offset to the information indicated.
/*

aggregate JNLMSGDEF6 structure fill prefix JNLMSG$;


    constant    (
        BLDUCB                      /* Build UCB - item value = journal type
        ,UCBDATA                    /* offset to slave UCB data
        ,JNLNAM                     /* offset to ASCIC journal name
        ,BLDJNLRM                   /* Build a remaster block - no item
        ,RMFLGS                     /* JNLRM flags
        ,ACPNAM                     /* offset to ASCIC ACP name
        ,TAPGRP                     /* offset to ASCIC tape group name
        ,DSKINF                     /* offset to ASCIC disk name
        ,BLDADL                     /* Build an ADL - no item
        ,BLDRUL                  /* Build an RUL - no item
        ,MAXDSICOD               /* Maximum value
        ) equals 1 increment 1 tag C;

    ITEMCODE byte unsigned;             /* Item code
    ITEM longword unsigned;             /* item information (value or offset)

    constant IENTLEN equals . prefix JNLMSG$ tag C;  /* Size of item entry

end JNLMSGDEF6;

aggregate JNLMSGDEF6_UCBDATA structure fill prefix JNLMSG$;

    OWNUIC longword unsigned;           /* Owner UIC
    MCSID longword unsigned;            /* Master CSID
    DEVCHAR longword unsigned;          /* Device characteristics
    DEVCHAR2 longword unsigned;         /* Device characteristics 2
    JNL_SEQNO longword unsigned;        /* Journal sequence number
    JNL_QUOT longword unsigned;         /* Quota for RU journals
    JNL_MASK longword unsigned;         /* Mask for AT journals
    VPROT word unsigned;                /* protection
    JNL_PROT word unsigned;             /* protection
    JNL_ID word unsigned;               /* Journal ID
    JNL_MXENT word unsigned;            /* Maximum entry size
    JNL_MUNIT word unsigned;            /* Master unit number
    DEVSTS word unsigned;               /* Device status
    STS word unsigned;                  /* bits that need duplication on slave
    AMOD byte unsigned;                 /* Access mode

    constant UCBDATALEN equals . prefix JNLMSG$ tag C;  /* Size of entry

end JNLMSGDEF6_UCBDATA;

end_module $JNLMSGDEF;
```

```
module $JNLMSGDATADEF;

/*++
/*
/* JNLMSGDATA -
/*
/*--

aggregate JNLMSGDATA structure fill prefix JNLMSGDATA$;

    FLINK longword unsigned;                    /* Forward link
    BLINK longword unsigned;                    /* Backward link
    SIZE  word unsigned;                        /* structure size
    TYPE  byte unsigned;                        /* structure type code
    SUBTYPE byte unsigned;                      /* structure sub type field
    VAL1 longword unsigned;                     /* misc longword of data
    VAL2 longword unsigned;                     /* misc longword of data
    VAL3 longword unsigned;                     /* misc longword of data
    VAL4 longword unsigned;                     /* misc longword of data
    VAL5 longword unsigned;                     /* misc longword of data
    constant "LENGTH" equals . prefix JNLMSGDATA$ tag C;
    constant "LENGTH" equals . prefix JNLMSGDATA$ tag K;

end JNLMSGDATA;

end_module $JNLMSGDATADEF;
```

```
module $WBLDEF;
/*++
/*
/* WBL - Wait Block List
/*
/* When a thread is being rescheduled all its state is saved in a WBL
/*
/*--

aggregate WBLDEF structure fill prefix WBL$;
    WBLQFL longword unsigned;                              /* forward q link
    WBLQBL longword unsigned;                              /* backward q link
    SIZE word unsigned;                                    /* size of structure
    FILL_1 word fill prefix WBLDEF tag $$;                 /* spare
    STATUS longword unsigned;                              /* status
    ASTBLK longword unsigned;                              /* address AST Block for rescheduling
    IRP longword unsigned;                                 /* IRP address
    USTSIZE word unsigned;                                 /* user stack save block size
    FILL_2 word fill prefix WBLDEF tag $$;                 /* descriptor type field
    UST longword unsigned;                                 /* address user stack save block
    USTADDR longword unsigned;                             /* original start address user stack
    KSTSIZE word unsigned;                                 /* kernel stack save block size
    FILL_3 word fill prefix WBLDEF tag $$;                 /* descriptor type field
    KST longword unsigned;                                 /* address kernel stack save block
    KSTADDR longword unsigned;                             /* original start address kernel stack
    OWNSIZE word unsigned;                                 /* own save block size
    FILL_4 word fill prefix WBLDEF tag $$;                 /* descriptor type field
    OWN longword unsigned;                                 /* address own space save block
    FILL_5 longword fill prefix WBLDEF tag $$;             /* spare
    GBLSIZE word unsigned;                                 /* global save block size
    FILL_6 word fill prefix WBLDEF tag $$;                 /* descriptor type field
    GBL longword unsigned;                                 /* address global space save block
    FILL_7 longword fill prefix WBLDEF tag $$;             /* spare
    constant "LENGTH" equals . prefix WBL$ tag K;          /* length structure
    constant "LENGTH" equals . prefix WBL$ tag C;          /* length structure

end WBLDEF;

end_module $WBLDEF;
```

```
module $OPCHDRDEF;
/*++
/*
/* OPCHDR - OPCOM message header
/*
/* This structure defines the fields in the common OPCOM message
/* header. This data structure is defined in [SYS SRC]SYSSNDMSG.MAR
/* in the commentary at the top. If this data structure ever changes in that
/* source module, then we need to change it here also.
/*
/*--


aggregate OPCHDRDEF structure fill prefix OPCHDR$;
    TYPE word unsigned;                                 /* message type
    RMBX word unsigned;                                 /* reply mailbox channel number
    PRIV quadword unsigned;                             /* sender's privilege mask
    UIC longword unsigned;                              /* sender's UIC
    USRNAM byte unsigned dimension 12;                  /* sender's USERNAME, 12 bytes blank filled
    ACCNT byte unsigned dimension 8;                    /* sender's ACCOUNT, 8 bytes blank filled
    BPRIO byte unsigned;                                /* sender's base priority
    FILL_1 byte fill prefix OPCHDRDEF tag $$;           /* unused
    constant "LENGTH" equals . prefix OPCHDR$ tag K;    /* length structure
    constant "LENGTH" equals . prefix OPCHDR$ tag C;    /* length structure
end OPCHDRDEF;

end_module $OPCHDRDEF;


{              JNLSYSDEF : The following modules need to go into SYSDEF
{
{*****************************************************************************
{*                                                                           *
{*  Copyright (c) 1980                                                       *
{*  by DIGITAL Equipment Corporation, Maynard, Mass.                         *
{*                                                                           *
{*  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:  JOURNALING : DEFINITION OF INTERNAL SYMBOLICS
```

```
{
{ Abstract:
{       This module contains the symbolic definitions for non-user accessible
{       data structures.
{
{ Author:       Joost Verhofstad               18-MAY-1983
{
{ Modified by:
{
{--
```

```
module $ABEDEF;
/***
/*
/* ABE - AI-BI List element
/*
/* For each AI or BI journal written to from inside an RU, the journal
/* name is in the AI-List or BI-list (for AI and BI journals resp)
/* This structure is the slot in the list, as used for one journal
/*
/*--


aggregate ABEDEF structure fill prefix ABE$;
    JNLNAME character;                                      /* length name
    NAME byte unsigned dimension 18;                        /* journal name
    STATUS_OVERLAY union fill;
        STATUS word unsigned;                               /* status
        STATUS_BITS structure fill;
            PURGED bitfield mask;                           /* slot not used
        end STATUS_BITS;
    end STATUS_OVERLAY;
    FILL_1 byte fill prefix ABEDEF tag $$;                  /* spare
    constant "LENGTH" equals . prefix ABE$ tag K;           /* length structure
    constant "LENGTH" equals . prefix ABE$ tag C;           /* length structure
end ABEDEF;

end_module $ABEDEF;
```

```
module $ABLDEF;
/*++
/*
/* ABL - AI-BI List
/*
/* For each AI or BI journal written to from inside an RU, the journal
/* name is in the AI-List or BI-list (for AI and BI journals resp)
/*
/*--


aggregate ABLDEF structure fill prefix ABL$;
    NEXT longword unsigned;                               /* next ABL
    SLOTS word unsigned;                                  /* number of slots in list
    JNLS word unsigned;                                   /* number of journals in list
    SIZE word unsigned;                                   /* size structure
    TYPE_OVERLAY union fill;
        STRUCT byte unsigned;                             /* structure type
        TYPE byte unsigned;                               /* data type field
    end TYPE_OVERLAY;
    SUBTYPE byte unsigned;                                /* CJF subtype field
    constant FIXED_LEN equals . prefix ABL$ tag K;       /* length structure
    constant FIXED_LEN equals . prefix ABL$ tag C;       /* length structure
end ABLDEF;

end_module $ABLDEF;
```

```
module $ADBDEF;
/*++
/*
/* ADB - Allocated Device Block
/*
/* For each disk or tape device allocated by a Journal ACP, the
/* ADL off the UCB for the ACP Control Journal contains a ADB
/* (Allocated Device Block). The ADB contains the device name
/* and some control information
/*
/*--


aggregate ADBDEF structure fill prefix ADB$;
    LINK longword unsigned;                             /* link to next ADB in same volume set
    STATUS_OVERLAY union fill;
        STATUS word unsigned;                           /* status of device and this ADB
        STATUS_BITS structure fill;
            MNTALLOC bitfield mask;                     /* allocated during MOUNT
            MOUNTED bitfield mask;                      /* device is mounted
            PURGED bitfield mask;                       /* this ADB is available
        end STATUS_BITS;
    end STATUS_OVERLAY;
    FILL_1 word fill prefix ADBDEF tag $$;              /* spare
    FILL_2 longword fill prefix ADBDEF tag $$;          /* spare
    NAMELEN byte unsigned;                              /* device name length
    DEVNAM byte unsigned dimension 15;                  /* device name (ASCII)
    constant "LENGTH" equals . prefix ADB$ tag K;       /* length structure
    constant "LENGTH" equals . prefix ADB$ tag C;       /* length structure
end ADBDEF;

end_module $ADBDEF;
```

```
module $ADLDEF;
/*++
/*
/* ADL - Allocated Device List
/*
/* For each disk or tape device allocated by a Journal ACP, the
/* ADL off the UCB for the ACP Control Journal contains a ADB
/* (Allocated Device Block).
/*
/*--


aggregate ADLDEF structure fill prefix ADL$;
    LINK longword unsigned;                         /* link to next ADL for this ACP (only
                                                    /*  for first ADL, not for extensions)
    UCB longword unsigned;                          /* backpointer to UCB
    SIZE word unsigned;                             /* size of list (ADL+ADBs in this ADL)
    TYPE byte unsigned;                             /* data structure type
    SUBTYPE byte unsigned;                          /* CJF subtype field
    EXTEND longword unsigned;                       /* next ADL extension
    DEVCNT word unsigned;                           /* device count: ! of devices allocated
                                                    /*  in this ADL
    ADBCNT word unsigned;                           /* number of ADBs in this ADL
    FSTADB word unsigned;                           /* offset first ADB, from this location
    FILL_2 word fill prefix ADLDEF tag $$;          /* spare
    constant FIXED_LEN equals . prefix ADL$ tag K;  /* length fixed portion
    constant FIXED_LEN equals . prefix ADL$ tag C;  /* length fixed portion
    constant START_ADB equals . prefix ADL$ tag K;  /* Start of list.
    constant START_ADB equals . prefix ADL$ tag C;  /* Start of list.
end ADLDEF;

end_module $ADLDEF;
```

```
module $BCBDEF;
/*++
/*
/* BCB - Buffer Control Block
/*
/*       For each mounted journal there are two buffers pointed to by the
/*       BCB which is pointed to by the journal UCB. The BCB always describes
/*       the characteristics and status of these buffers
/*
/*--


aggregate BCBDEF structure fill prefix BCB$;
    ADDR1 longword unsigned;                            /* address of buffer 1
    ADDR2 longword unsigned;                            /* address of buffer 2
    SIZE word unsigned;                                 /* structure size
    TYPE byte unsigned;                                 /* structure type code
    SUBTYPE byte unsigned;                              /* subtype field for CJF
    STS_OVERLAY union fill;
        STS byte unsigned;                              /* status code
        STS_BITS structure fill;
            CUR bitfield mask;                          /* current buffer indicator
        end STS_BITS;
    end STS_OVERLAY;
    FILL_1 word fill prefix BCBDEF tag $$;              /* SPARE
    FILL_2 byte fill prefix BCBDEF tag $$;              /* SPARE
    UCB longword unsigned;                              /* UCB address of journal
    BSIZ1 word unsigned;                                /* size of buffer 1 in bytes
    BSIZ2 word unsigned;                                /* size of buffer 2 in bytes
    STS1_OVERLAY union fill;
        STS1 word unsigned;                             /* status of buffer 1
        STS1_BITS structure fill;
            IOPR bitfield mask;                         /* I/O in progress bit
            WRPR bitfield mask;                         /* write in progress bit
            WRPEN bitfield mask;                        /* write pending bit
            REPR bitfield mask;                         /* read in progress bit
            REAPEN bitfield mask;                       /* read pending bit
            EXTPR bitfield mask;                        /* extend in progress
            EXTPEN bitfield mask;                       /* extend pending
            RECLE bitfield mask;                        /* buffer read and cleared bit
            SETPEN bitfield mask;                       /* "set-buffer-to-next-one" pending
            NWVPR bitfield mask;                        /* create new version in progress
        end STS1_BITS;
    end STS1_OVERLAY;
    STS2 word unsigned;                                 /* status of buffer 2
    WRCNT1 word unsigned;                               /* write count for first buffer
    WRCNT2 word unsigned;                               /* write count for second buffer
    RDCNT1 word unsigned;                               /* read count for first buffer
    RDCNT2 word unsigned;                               /* read count for second buffer
    OFFS1 word unsigned;                                /* offset first free byte in buffer 1
    OFFS2 word unsigned;                                /* offset first free byte in buffer 2
    VBN1 longword unsigned;                             /* first VBN buffer 1
    VBN2 longword unsigned;                             /* first VBN buffer 2
    PRVVBN longword unsigned;                           /* VBN bucket in which previous chunk is
    PRVEVBN longword unsigned;                          /* VBN bucket in which previous entry is
    PRVOFF word unsigned;                               /* offset of previous chunk written
    PRVEOFF word unsigned;                                  /* offset of previous entry written
    LOWSN longword unsigned;                                /* lowest seq.no in current buffer
    HISN longword unsigned;                                 /* highest seq.no of any entry written
                                                            /* into the buffers
    CRCTBL longword unsigned;                               /* address of CRC table
    HISN_CMPL longword unsigned;                            /* High sequence number completely in a buffer
    HISN_WRT longword unsigned;                             /* High sequence number written
                                                            /*    to secondary storage
    constant "LENGTH" equals . prefix BCB$ tag K;           /* length of structure
    constant "LENGTH" equals . prefix BCB$ tag C;           /* length of structure
end BCBDEF;

end_module $BCBDEF;
```

```
module $JNLACBMDEF;
/*++
/*
/* JNLACBM - Journal access bit map
/*
/*      This bit map will contain a single bit for each node in
/*      the cluster.  When ever a slave node assigns his first
/*      journal to the journal or deassigns his last journal channel
/*      the node bit will be adjusted.  This bit map will be indexed
/*      via the node index portion of the node's CSID
/*--


aggregate JNLACBMDEF structure fill prefix JNLACBM$;
    FLINK longword unsigned;                    /* Forward link
    BLINK longword unsigned;                    /* Backward link
    SIZE  word unsigned;                        /* structure size
    TYPE  byte unsigned;                        /* structure type code
    SUBTYPE byte unsigned;                      /* structure sub type field
    MAPSIZE word unsigned;                      /* Bit map size
    BITMAP  character length 0 tag X ;          /* Bit map start
    constant LENGTH equals .;                   /* Size of JNLACBM header
end JNLACBMDEF ;

end_module $JNLACBMDEF ;
```

```
module $JNLBUFDEF;
/*++
/*
/* JNLBUF - Buffer of which there are two for each journal
/*
/* The BCB pointed to by the journal UCB points to the two buffers
/*
/*--


aggregate JNLBUFDEF structure fill prefix JNLBUF$;
    LEN word unsigned;                                  /* total length of buffer header minus
                                                        /*  length of this word (RMS seq. record)
    LEN2 word unsigned;                                 /* second word of length (only for tape)
    TYPE_OVERLAY union fill;
        TYPE byte unsigned;                             /* record type to indicate control entry
        TYPE_BITS structure fill;
            USER bitfield mask;                         /* user entry
            CONTR bitfield mask;                        /* control entry
        end TYPE_BITS;
    end TYPE_OVERLAY;
    BUFHDR byte unsigned;                               /* buffer header length
    FILL_1 word fill prefix JNLBUFDEF tag $$;           /* SPARE (to match other records)
    BUFSIZ word unsigned;                               /* buffer size : this MUST be 1st word
                                                        /*        in 3rd longword
    DTYPE_OVERLAY union fill;
        STRUCT byte unsigned;                           /* data structure type value : this MUST
                                                        /*        be 3rd byte in 3rd longword
        DTYPE byte unsigned;                            /* data type field
    end DTYPE_OVERLAY;
    STYPE_OVERLAY union fill;
        ENTTYP byte unsigned;                           /* entry type
        SUBTYPE byte unsigned;                          /* data subtype field
    end STYPE_OVERLAY;
    VBN longword unsigned;                              /* journal block number (of 1st. bl in bucket)
    LSTENO word unsigned;                               /* last entry/chunk in bucket - offset
    FILL_2 word fill prefix JNLBUFDEF tag $$;           /* spare
    JNLID longword unsigned;                            /* journal ID
    LOWSN longword unsigned;                            /* lowest sequence number of all entries
                                                        /*  in this bucket
    HISN longword unsigned;                             /* highest sequence number of all entries
                                                        /*  in this bucket
    CDPTR word unsigned;                                /* current data pointer (! of data bytes
                                                        /*  written for BI,AI,AT and next byte
                                                        /*  to write for RU jnl)
    STS_OVERLAY union fill;
        STS word unsigned;                              /* buffer status
        STS_BITS structure fill;
            UPDATE bitfield mask;                       /* this buffer has been updated
        end STS_BITS;
    end STS_OVERLAY;
    CHKSUM longword unsigned;                           /* CRC of bucket
    constant HDRLEN equals . prefix JNLBUF$ tag K;      /* length header
    constant HDRLEN equals . prefix JNLBUF$ tag C;      /* length header
    constant STDAT equals . prefix JNLBUF$ tag K;       /* first longword of data
    constant STDAT equals . prefix JNLBUF$ tag C;       /* first longword of data
```

```
end JNLBUFDEF;
end_module $JNLBUFDEF;
```

```
module $JNLBXSTSDEF;

/*++
/*
/* JNLBXSTS -   Journal block transfer in procress queue entry
/*              This structure is used to keep track of all pertenant
/*              information concerning an IRP that has been initiated
/*              on the local node via a block transfer request from
/*              some other node.  If the connection between the two
/*              nodes breaks before the local node has sent the response
/*              then the the message may be retransmitted and we must be
/*              able to deal with that.  Hopefully this structure will
/*              contain all the information we will need.
/*
/*--


aggregate JNLBXSTSDEF structure fill prefix JNLBXSTS$;
    FLINK   longword unsigned;                  /* Forward link
    BLINK   longword unsigned;                  /* Backward link
    SIZE    word unsigned;                      /* size data structure
    TYPE    byte unsigned;                      /* type of structure
    SUBTYPE byte unsigned;                      /* subtype of structure
    STS_OVERLAY union fill;
        STS longword unsigned;                  /* block Xfer status
        STS_BITS structure fill;
            READCMPL bitfield mask;             /* The block read is complete
            READINP bitfield mask;              /* The block read is in progress
            WRITECMPL bitfield mask;            /* The block write is complete
            WRITEINP bitfield mask;             /* The block write is in progress
            RESPSENT bitfield mask;             /* The response has been sent
            FNCTCMPL bitfield mask;             /* The function is complete (no response sent)
            CNXBRK bitfield mask;               /* The connection has broken
        end STS_BITS;
    end STS_OVERLAY;
    REQ_CSID_OVERLAY union fill ;
        REQ_CSID longword unsigned;             /* CSID of node which originated
                                                /*  the message (requestor)
        REQ_CSID_SUBF structure fill;
            REQ_CSID_SEQ word unsigned;         /* CSID sequence number
            REQ_CSID_IDX word unsigned;         /* CSID node index
        end REQ_CSID_SUBF ;
    end REQ_CSID_OVERLAY ;
    BTXSEQNO longword unsigned;                 /* Block transfer sequence #
    CURR_IRP longword unsigned;                 /* Address of the current IRP
    RTX_IRP longword unsigned;                  /* Address of IRP from last retransmit
    SPARE1 longword unsigned;
    SPARE2 longword unsigned;
    SPARE3 longword unsigned;
    constant LENGTH equals . tag K ;            /* Structure size
    constant LENGTH equals . tag C ;            /* Structure size

end JNLBXSTSDEF;

end_module $JNLBXSTSDEF;
```

```
module $JNLCWQDEF;

/*++
/*
/* JNLCWQ -      Journal cluster write queue entry
/*               This structure is used to keep track of all
/*               writes that have been sent from a slave to the master
/*               node but have not yet been written to secondary storage.
/*               During fail over of a node this information is necessary
/*               to resubmit the write's for the user.  Once we have
/*               told the user that the write is complete we must make
/*               sure that it makes it out to the file unless the node
/*               it was issued from crashes
/*
/*--


aggregate JNLCWQDEF structure fill prefix JNLCWQ$;
    FLINK   longword unsigned;                    /* Forward link
    BLINK   longword unsigned;                    /* Backward link
    SIZE    word unsigned;                        /* size data structure
    TYPE    byte unsigned;                        /* type of structure
    SUBTYPE byte unsigned;                        /* subtype of structure
    UCB     longword unsigned;                    /* Back pointer to the UCB
    FOVSTS_OVERLAY union fill;
        FOVRSTAT longword unsigned;                       /* fail-over status
        FOVSTS_BITS structure fill;
            RESUB bitfield mask;                          /* this entry must be resubmitted if set
        end FOVSTS_BITS;
    end FOVSTS_OVERLAY;
    SEND_CSID_OVERLAY union fill ;
        SEND_CSID longword unsigned;              /* CSID of node we originally
                                                  /*  sent the message to

        SEND_CSID_SUBF structure fill;
            SEND_CSID_SEQ word unsigned;          /* CSID sequence number
            SEND_CSID_IDX word unsigned;          /* CSID node index
        end SEND_CSID_SUBF ;
    end SEND_CSID_OVERLAY ;
    SEND_UNIT word unsigned;                      /* Unit number of original
                                                  /*  master journal device
    IOFUNC word unsigned;                         /* Original I/O function
    IRP  longword unsigned;                       /* Address of the IRP.  We may
                                                  /*  still have to post it at failover
    SEQNO  longword unsigned;                     /* Entry's sequence # (0 in not ACK'd)
    LSEQNO longword unsigned;                     /* Entry's local sequence #
    BEGIN_OFFSET longword unsigned;               /* Beginning offset of remaining
                                                  /*  portion of a partial write
    BYTCNT_REM word unsigned;                     /* Bytes remaining for partial write
    BYTCNT_ORG word unsigned;                     /* Original count of bytes in message
    RUID octaword unsigned;                       /* Recovery unit ID.
    WRUFLAGS longword unsigned;                   /* Write RU flags.
    WRMASK longword unsigned;                     /* Write mask
    IRPESTATUS longword unsigned;                 /* status field kept in IRPE
    ASID longword unsigned;                       /* Assign ID for the channel
    FACCOD word unsigned;                         /* Channel facility code
    IOSTS byte unsigned;                          /* I/O status (used only for writes)
    WRATR byte unsigned;                        /* Write attributes
    EPID longword unsigned;                     /* Process EPID
    ARB_PRIV quadword unsigned;                 /* Priv mask from ARB
    MSGBUF character length 0;                  /* Base of journal entry in a message
    constant FIXED_LEN equals . tag C ;         /* Fixed size
    constant FIXED_LEN equals . tag K ;         /* Fixed size

end JNLCWQDEF;

end_module $JNLCWQDEF;
```

```
module $JNLDBDEF;

/*++
/*
/* JNLDB - off of each CDT is hung a data block that serves as
/*          a queue listhead for remote IRP's waiting on a response
/*          for a connection, a queue listhead for the slave UCB's
/*          that access the master node via that CDT, and a pointer
/*          to a buffer that contains entries written to the master
/*          (via the CDT) but whose QIOs have not yet been ACK'd
/*          by the master. This structure is used for master
/*          failover recovery.
/*--


aggregate JNLDBDEF structure fill prefix JNLDB$;
    IRPQFL longword unsigned;                           /* IRP queue forward link
    IRPQBL longword unsigned;                           /* IRP queue backward link
    SIZE   word unsigned;                               /* size data structure
    TYPE   byte unsigned;                               /* type of structure
    SUBTYPE   byte unsigned;                            /* subtype of structure
    UCBQFL longword unsigned;                           /* UCB queue forward link
    UCBQBL longword unsigned;                           /* UCB queue backward link
    BUFFER longword unsigned;                           /* Pointer to write buffer
    FILL_1 longword fill prefix JNLDBDEF tag $$;        /* Spare
    constant "LENGTH" equals . prefix JNLDB$ tag K;
    constant "LENGTH" equals . prefix JNLDB$ tag C;
end JNLDBDEF;

end_module $JNLDBDEF;
```

```
module $JNLLOGDEF;
/*++
/*
/* JNLLOG - Journal error log function bits
/*
/* This structure defines the bits indicating to SYE the error
/* being logged
/*
/*--


aggregate JNLLOGDEF  union fill prefix JNLLOG$;
    JNLLOGDEF_BITS structure fill;
        RUEXT bitfield mask;                    /* RU journal extended
        RUNEXT bitfield mask;                   /* RU journal could not be extended
        SLVCRFAIL bitfield mask;                /* Failure on slave node while
                                                /*  attempting a create
    end JNLLOGDEF_BITS;
end JNLLOGDEF;

end_module $JNLLOGDEF;
```

```
module $JNLRCDEF;
/*++
/*
/* JNLRC - Journaling Read Context
/*
/* The JNLRC holds the information necessary for read failover.
/*
/*--


aggregate JNLRCDEF structure fill prefix JNLRC$;
    FILL_1 longword fill prefix JNLRCDEF tag $$;        /* unused - forward link
    FILL_2 longword fill prefix JNLRCDEF tag $$;        /* unused - back link
    SIZE word unsigned;                                  /* size of structure
    TYPE byte unsigned;                                  /* data structure type
    SUBTYPE byte unsigned;                               /* CJF subtype
    SEQNO longword unsigned;                             /* seq # previous entry
    RUID_UNION union fill;
        RUID quadword unsigned octaword;                 /* Recovery unit ID (RU only)
        RUID_OVERLAY structure fill;
            DATTIM quadword unsigned;                    /* date/time prev. entry (NONRU ONLY)
            CSID_UNION union fill ;
                CSID longword unsigned;                  /* CSID portion of RUID,
                CSID_OVERLAY structure fill;
                    CSID_SEQ word unsigned;              /* CSID sequence number
                    CSID_IDX word unsigned;              /* CSID node index
                end CSID_OVERLAY;
            end CSID_UNION;
            RUID_LW4 longword unsigned;                  /* Forth longword of RUID
        end RUID_OVERLAY;
    end RUID_UNION;
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                             /* Flags
        FLAGS_BITS structure fill;
            READDIR bitfield mask;                       /* Read direction
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    FILL_3 byte dimension 3 fill prefix JNLRCDEF tag $$; /* spare
    FLTRS longword unsigned;                             /* Offset to filters

    constant "LENGTH" equals . prefix JNLRC$ tag K;     /* length fixed part
    constant "LENGTH" equals . prefix JNLRC$ tag C;     /* length fixed part

end JNLRCDEF;

end_module $JNLRCDEF;
```

```
module $JNLRMDEF;
/*++
/*
/* JNLRM - Journaling Remaster Block
/*
/* The JNLRM is used by the CSP to construct a JSB for remastering a journal.
/*
/*--


aggregate JNLRMDEF structure fill prefix JNLRM$;
    FILL_1 longword fill prefix JNLRMDEF tag $$;       /* unused - forward link
    FILL_2 longword fill prefix JNLRMDEF tag $$;       /* unused - backward link
    SIZE word unsigned;                                /* size of structure
    TYPE byte unsigned;                                /* data structure type
    SUBTYPE byte unsigned;                             /* subtype for CJF data structure
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                           /* flags word
        FLAGS_BITS structure fill;
            DSKJNL bitfield mask;                      /* Disk journal
            TAPJNL bitfield mask;                      /* Tape journal
            TMPFIL bitfield mask;                      /* Temp file
            DIFACP bitfield mask;                      /* Different ACP
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    COPIES byte unsigned;                              /* number of copies
    FILL_3 byte fill prefix JNLRMDEF tag $$;           /* fill
    CONBLK longword unsigned;                          /* address of the 1st connect block
    ACPNAMOFF word unsigned;                           /* offset to ACP name
    ACPNAMLEN word unsigned;                           /* ACP name length

    constant "LENGTH" equals . prefix JNLRM$ tag K;    /* length fixed part
    constant "LENGTH" equals . prefix JNLRM$ tag C;    /* length fixed part

    constant "DSKJNLLST" equals . prefix JNLRM$ tag K; /* start info for disk jnls
    constant "DSKJNLLST" equals . prefix JNLRM$ tag C; /* - dev names, ver #'s

    TAPGRPOFF word unsigned;                           /* offset to tape group name
    TAPGRPLEN word unsigned;                           /* tape group name length

    constant "TAPJNLLEN" equals . prefix JNLRM$ tag K; /* length for tape journal
    constant "TAPJNLLEN" equals . prefix JNLRM$ tag C; /* length for tape journals


end JNLRMDEF;

aggregate JNLRM1DEF structure fill prefix JNLRM$;

    DEVNAMOFF word unsigned;                           /* offset to device name
    DEVNAMLEN word unsigned;                           /* device name length
    FILVEROFF word unsigned;                           /* offset to file version
    FILVERLEN word unsigned;                           /* file version length

    constant "DSKENTLEN" equals . prefix JNLRM$ tag K; /* length of disk journal
    constant "DSKENTLEN" equals . prefix JNLRM$ tag C; /* information
```

```
end JNLRM1DEF;

end_module $JNLRMDEF;
```

```
module $JNLSFTDEF;
/*+
/* JNLSFT -- Spool File Table
/*
/*         The JNLSFT describes the physical storage medium for the journal spool
/*         file. Spool files are used for tape groups only.
/*         The JNLSFTs for a given tape group are linked together in a list.
/*         The first JNLSFT is pointed to by each JMT for each tape in the group
/*
/*-


aggregate JNLSFTDEF structure fill prefix JNLSFT$;
    FORJNLLNK longword unsigned;                        /* Forward link for JMT's for this journal
    BACJNLLNK longword unsigned;                        /* Backward link for JMT's for this journal
    SIZE word unsigned;                                 /* size of JNLSFT
    TYPE byte unsigned;                                 /* structure type of JNLSFT
    SUBTYPE byte unsigned;                              /* structure subtype of JNLSFT
    constant ACPQB equals . prefix JNLSFT$ tag K;               /* label for ACP queue block
    constant ACPQB equals . prefix JNLSFT$ tag C;               /* label for ACP queue block

    FORACPLNK longword unsigned;                        /* Forward link to next JMT for this ACP
    BACACPLNK longword unsigned;                        /* Backward link to next JMT for this ACP
    JMT longword unsigned;                              /* First JMT in list of JMTs for group
                                                        /*  for which this is a spool file
    SPL_COP byte unsigned;                              /* number of spool files in list
    FILL_2 byte dimension 3 fill prefix JNLSFTDEF tag $$; /* spare

    MAX_JNLS word unsigned;                             /* max ! of journals for this spool file
    COPY_NUM word unsigned;                             /* number of spool file (zero relative)
    WRCNT word unsigned;                                /* write count
    RDCNT word unsigned;                                /* read count
    STATUS_OVERLAY union fill;
        STATUS longword unsigned;                       /* journal media status
        STATUS_BITS structure fill;
            HEAD_SFT bitfield mask;                     /* first JNLSFT (copy) for this group
            ACTIVE bitfield mask;                       /* spool file not empty: being used
        end STATUS_BITS;
    end STATUS_OVERLAY;
    BASEVBN longword unsigned;                          /* Base VBN: to be substracted from bucket
                                                        /*  VBN to get VBN of block in file
    SPL_WCB longword unsigned;                          /* pointer to journal spool file WCB
    SPL_UCB longword unsigned;                          /* pointer to journal spool file UCB
    SPL_MXVBN longword unsigned;                        /* max VBN in journal disk spool file
    SPL_STVBN longword unsigned;                        /* first VBN in journal disk spool file
    SPL_NUM word unsigned;                              /* journal spool file file ID number
    SPL_SEQ word unsigned;                              /* journal spool file file ID sequence number
    SPL_RVN word unsigned;                              /* journal spool file file ID rel vol num
    FILL_3 word fill prefix JNLSFTDEF tag $$;           /* spare
    VOLLAB byte unsigned dimension 12;                  /* volume label disk on which file is
    SPL_VBN longword unsigned;                          /* next VBN for next bucket to write to
    constant "LENGTH" equals . prefix JNLSFT$ tag K;    /* length
    constant "LENGTH" equals . prefix JNLSFT$ tag C;    /* length
                                                        /*  spool file. (spool file is used as
                                                        /*  tape, but we must keep track of VBN)
```

```
end JNLSFTDEF;

end_module $JNLSFTDEF;
```

```
module $JMTDEF;
/*+
/* JMT -- Journal Merge Table
/*
/*         The JMT describes the physical storage medium for the journal copy.
/*         The JMT is pointed to by each VCB.  When multiple journals are
/*         kept on the same storage medium (ie multiple journals on one
/*         tape), there exists one JMT for the tape, and many VCB's may
/*         point to it.
/*
/* All bits marked (*) are set in the head JMT (first in list) only
/*  in the current version.
/*
/*-


aggregate JMTDEF structure fill prefix JMT$;
    FORJNLLNK longword unsigned;                        /* Forward link for JMT's for this journal
    BACJNLLNK longword unsigned;                        /* Backward link for JMT's for this journal
    SIZE word unsigned;                                 /* size of JMT
    TYPE byte unsigned;                                 /* structure type of JMT
    SUBTYPE byte unsigned;                              /* structure subtype of JMT
    constant ACPQB equals . prefix JMT$ tag K;          /* label for ACP queue block
    constant ACPQB equals . prefix JMT$ tag C;          /* label for ACP queue block
    FORACPLNK longword unsigned;                        /* Forward link to next JMT for this ACP
    BACACPLNK longword unsigned;                        /* Backward link to next JMT for this ACP

    ACP_PRI byte unsigned;                              /* ACP's priority (priority for I/O)
    FILL_2 byte dimension 3 fill prefix JMTDEF tag $$;  /* spare
    ACP_ARB longword unsigned;                          /* pointer to ACP access rights block
    AQB longword unsigned;                              /* address of AQB for owner ACP

    MAX_JNLS word unsigned;                             /* max ! of journals for this JMT
    FILL_3 word fill prefix JMTDEF tag $$;              /* spare
    COPY_NUM word unsigned;                             /* copy number (zero relative)
    JNLIDCTR word unsigned;                             /* journal ID counter
    WRCNT word unsigned;                                /* write count
    RDCNT word unsigned;                                /* read count

    SPOOLING_OVERLAY union fill;
        SPOOLING byte unsigned;                         /* spool byte: if any of these bits is
                                                        /*  set, spooling must be done.
        SPOOLING_BITS structure fill;
            REPR bitfield mask;                         /* read in progress
            EOTPR bitfield mask;                        /* EOT processing going on (*)
        end SPOOLING_BITS;
    end SPOOLING_OVERLAY;
    FILL_4 byte dimension 3 fill prefix JMTDEF tag $$;  /* spare

end JMTDEF;

aggregate JMTDEF1 structure fill prefix JMT$;
    FILL_10 byte dimension 44 fill prefix JMTDEF tag $$;
    STATUS_OVERLAY union fill;
        STATUS longword unsigned;                       /* journal media status
        STATUS_BITS structure fill;
            SPLBYTE bitfield mask length 8;               /* spool byte
                WRPR bitfield mask;                       /* write in progress (currently unused)
                NOWRJNL bitfield mask;                    /* cannot write to journal now (not
                                                          /*     even spool file)
                HEAD_JMT bitfield mask;                   /* first JMT (copy) for this journal
                SPOOLED bitfield mask;                    /* device is spooled (*)
                SPOOLSYNC bitfield mask;                  /* all io to journal file (incl spool
                                                          /*     file) must wait: switching back or
                                                          /*     forth between tape and spool file
                                                          /*     (*)
                STARTSP bitfield mask;                    /* start spooling (*)
                STOPSP bitfield mask;                     /* stop spooling (*)
                CANCELIO bitfield mask;                   /* cancel IO to tape (*)
                DMT bitfield mask;                        /* this copy is marked for dismount
                AVL bitfield mask;                        /* this copy is available
                SYNCHCAN bitfield mask;                   /* synchronize with CANCELIO on tape (*)
                REPEN bitfield mask;                      /* read pending
                INFPEN bitfield mask;                     /* inform ACP pending (*)
                NOWRTP bitfield mask;                     /* do not write to tape: ACP stops driver
        end STATUS_BITS;
    end STATUS_OVERLAY;
    JMTSFT longword unsigned;                             /* the JMT or SFT on which an error
                                                          /*  ocurred (*)

    SPARE1 longword unsigned;
    SPARE2 longword unsigned;
    SPARE3 longword unsigned;
    SPARE4 longword unsigned;
    OWNUIC longword unsigned;                             /* owner UIC
    PROT word unsigned;                                   /* protection mask
    FILL_5 word fill prefix JMTDEF tag $$;                /* SPARE

    BASEVBN longword unsigned;                            /* base VBN first bucket (add to file VBN to get bucket VBN)
    FIL_WCB longword unsigned;                            /* pointer to journal file WCB
    FIL_UCB longword unsigned;                            /* pointer to journal file UCB
    FIL_MXVBN longword unsigned;                          /* max VBN in journal disk file
    FIL_STVBN longword unsigned;                          /* first VBN in journal disk file
    FIL_LTVBN longword unsigned;                          /* last VBN for this file
    FIL_NUM word unsigned;                                /* journal file file ID number
    FIL_SEQ word unsigned;                                /* journal file file ID sequence number
    FIL_RVN word unsigned;                                /* journal file file ID rel vol num
    FILL_6 word fill prefix JMTDEF tag $$;                /* spare
    VOLLAB character length 13;                           /* volume label disk/tape on which file is
    FILL_7 byte fill prefix JMTDEF tag $$;                /* spare
    GRPNAM character length 13;                           /* group name
    FILL_8 byte fill prefix JMTDEF tag $$;                /* spare
    GTB longword unsigned;                                /* address of corresponding GTB in ACP
                                                          /*        virtual memory
    JFTE longword unsigned;                               /* address of corresponding JFTE in ACP
                                                          /*        virtual memory
    SFT longword unsigned;                                /* first SFT (spool file table)
    SPL_VBN longword unsigned;                            /* next VBN for next bucket to write to
                                                          /*  spool file. (spool file is used as
                                                          /*  tape, but we must keep track of VBN)

    VCB_COUNT word unsigned;                              /* number of VCB's pointing to JMT
```

```
                                                        /*     (not including VCB_CNTRL)
    FILL_9 word fill prefix JMTDEF tag $$;              /* spare
    VCB_CNTRL longword unsigned;                        /* address of control VCB (tape only)
    WQFL longword unsigned;                             /* wait Q forward link
    WQBL longword unsigned;                             /* wait Q backward link
    VCL longword unsigned;                              /* list of addresses associated VCB's
    FILVER character length 6;                          /* file version number
    constant "LENGTH" equals . prefix JMT$ tag K;       /* length lzbel
    constant "LENGTH" equals . prefix JMT$ tag C;       /* length label
end JMTDEF1;

end_module $JMTDEF;
```

```
module $NDLDEF;
/*++
/*
/* NDL - Name table Device List
/*
/* This structure has a fixed header size but the tail end is a variable
/* length depending on how many name table device names are in it.
/*
/*--


aggregate NDLDEF structure fill prefix NDL$;
    NDLQFL longword unsigned;                           /* forward q link
    NDLQBL longword unsigned;                           /* backward q link
    SIZE word unsigned;                                 /* size of structure
    TYPE byte unsigned;                                 /* structure type for NDL
    SUBTYPE byte unsigned;                              /* structure subtype
    COUNT byte unsigned;                                /* count
    FILL_1 word fill prefix NDLDEF tag $$;              /* spare
    FILL_2 byte fill prefix NDLDEF tag $$;              /* spare
    constant FIXEDLEN equals . prefix NDL$ tag K;       /* fixed size length
    constant FIXEDLEN equals . prefix NDL$ tag C;       /* fixed size length
end NDLDEF;

end_module $NDLDEF;
```

```
module $RUEDEF;
/*++
/*
/* RUE - Recovery Unit list Element
/* The Recovery Unit list contains one of these elements per recovery
/* unit active on the RU journal. The RUEs follow the RUL, which is pointed
/* to by the RU-journal's UCB. When the journal device is created a fixed
/* size list is allocated: for the RUL and a number of RUEs. When the list needs
/* to be extended, it is replaced by a longer one.
/*
/*--


aggregate RUEDEF structure fill prefix RUE$;
    RUID_UNION union fill;
        RUID quadword unsigned dimension 2;             /* RU ID
        RUID_OVERLAY structure fill;
            RUID_LW1 longword unsigned;                 /* First longword of RUID
            RUID_LW2 longword unsigned;                 /* second longword of RUID
            CSID_UNION union fill ;
                CSID longword unsigned;                 /* CSID portion of RUID,
                CSID_OVERLAY structure fill;
                    CSID_SEQ word unsigned;             /* CSID sequence number
                    CSID_IDX word unsigned;             /* CSID node index
                end CSID_OVERLAY;
            end CSID_UNION;
            RUID_LW4 longword unsigned;                 /* Forth longword of RUID
        end RUID_OVERLAY;
    end RUID_UNION;
    LSTVBN longword unsigned;                           /* VBN of bucket with last entry written
    LSTOFF word unsigned;                               /* offset of last entry written
    JNLCNT word unsigned;                               /* count of journals touched by RU
    INDEX longword unsigned;                            /* unique index for this RUE
    SEQNO longword unsigned;                            /* sequence number last entry written
    FSTEVBN longword unsigned;                          /* VBN of first entry written
    FSTVBN longword unsigned;                           /* VBN of first roll forw. entry written
    QUOTA longword unsigned;                            /* remaining number of bytes allowed to write
    STATUS_OVERLAY union fill;
        STATUS longword unsigned;                       /* status
        constant "LENGTH" equals . prefix RUE$ tag K;   /* length of RUE
        constant "LENGTH" equals . prefix RUE$ tag C;   /* length of RUE
        STATUS_BITS structure fill;
            PURGED bitfield mask;                       /* entry is free indicator
            ROLL_BACK bitfield mask;                    /* there is at least one roll back entry
            ROLL_FORW bitfield mask;                    /* there is at least one roll forward entry
            NOT_FLSHD bitfield mask;                    /* there is at least one entry not flushed
            OVER_QUOTA bitfield mask;                   /* quota exceeded
            PHASE1 bitfield mask;                       /* phase1 done
            PHASE2 bitfield mask;                       /* phase2 done
            ABORT bitfield mask;                        /* abort done
            P2$AB$2 bitfield mask;                      /* phase2 or abort entry to be encountered 2*
                                                        /*  before RU deletion
            RESIDUAL bitfield mask;                     /* this is a residual RU in journal
            COMPLETED bitfield mask;                    /* RU has been completed (rolled forward)
            CLEANUP bitfield mask;                      /* vestigial entry for RU can be ignored
            FROZEN bitfield mask;                       /* frozen RU
            RUSYNCEX bitfield mask;                    /* RUSYNC entry expected
            RUSYNCWR bitfield mask;                    /* RUSYNC entry written
            NOFAC bitfield mask;                       /* Frozen due to missing facility
            NOOBJ bitfield mask;                       /* Frozen due to missing object
        end STATUS_BITS;
    end STATUS_OVERLAY;
end RUEDEF;

end_module $RUEDEF;
```

```
module $RULDEF;
/*++
/*
/* RUL - Recovery Unit List
/*
/* This data structure forms the header of the list with the recovery
/* units that are currently active on the RU-journal for which this
/* list is used. The UCB of a RU journal points to the RUL for it.
/*
/*--


aggregate RULDEF structure fill prefix RUL$;
    NUM_RUES word unsigned;                              /* number of RUEs in the list
    FILL_1 word fill prefix RULDEF tag $$;               /* spare
    FILL_2 longword fill prefix RULDEF tag $$;           /* spare
    SIZE word unsigned;                                  /* size of total list (RUL+all RUEs)
    TYPE byte unsigned;                                  /* data structure type
    SUBTYPE byte unsigned;                               /* data structure subtype
    constant FIXED_LEN equals . prefix RUL$ tag K;       /* length of RUL fixed portion
    constant FIXED_LEN equals . prefix RUL$ tag C;       /* length of RUL fixed portion
end RULDEF;

end_module $RULDEF;
```

```
module $VCLDEF;
/*+
/* VCL - VCB List
/*
/* The VCL contains the VCB addresses of VCBs of journals that have been
/* created for a given tape group. The JMT of the head-JMT for that group
/* points to this VCL.
/*
/*-


aggregate VCLDEF structure fill prefix VCL$;
    JMT longword unsigned;                          /* JMT back pointer
    NUM_VLES word unsigned;                         /* number of VLEs in VCL
    COUNT word unsigned;                            /* number of VCB addresses in VCL
    SIZE word unsigned;                             /* size of structure
    TYPE byte unsigned;                             /* type of data structure
    SUBTYPE byte unsigned;                          /* subtype of data structure
    constant FIXED_LEN equals . prefix VCL$ tag K;
    constant FIXED_LEN equals . prefix VCL$ tag C;
end VCLDEF;

end_module $VCLDEF;
```

```
module $VLEDEF;
/*+
/* VLE - VCB List element
/*
/* The VCL contains the VCB addresses of VCBs of journals that have been
/* created for a given tape group. The JMT of the head-JMT for that group
/* points to this VCL. The VCL contains VLEs, each of which, when in use,
/* points to a VCB.
/*
/*-


aggregate VLEDEF structure fill prefix VLE$;
    STATUS_OVERLAY union fill;
        STATUS word unsigned;                            /* status
        STATUS_BITS structure fill;
            PURGED bitfield mask;
        end STATUS_BITS;
    end STATUS_OVERLAY;
    FILL_1 word fill prefix VLEDEF tag $$;               /* spare
    VCB longword unsigned;                               /* VCB address
    constant "LENGTH" equals . prefix VLE$ tag K;
    constant "LENGTH" equals . prefix VLE$ tag C;
end VLEDEF;

end_module $VLEDEF;
```

0045 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UNLBUSR
R32

UNLDEFINT
SDL

CJFV4.

UNLPREFIX
R32

CJFRUFMAC
SDL

RUFUSR
SDL

UNLFILE
SDL

UPGRADE
LIS

BOPTIONS
R32

UNLDEF
SDL