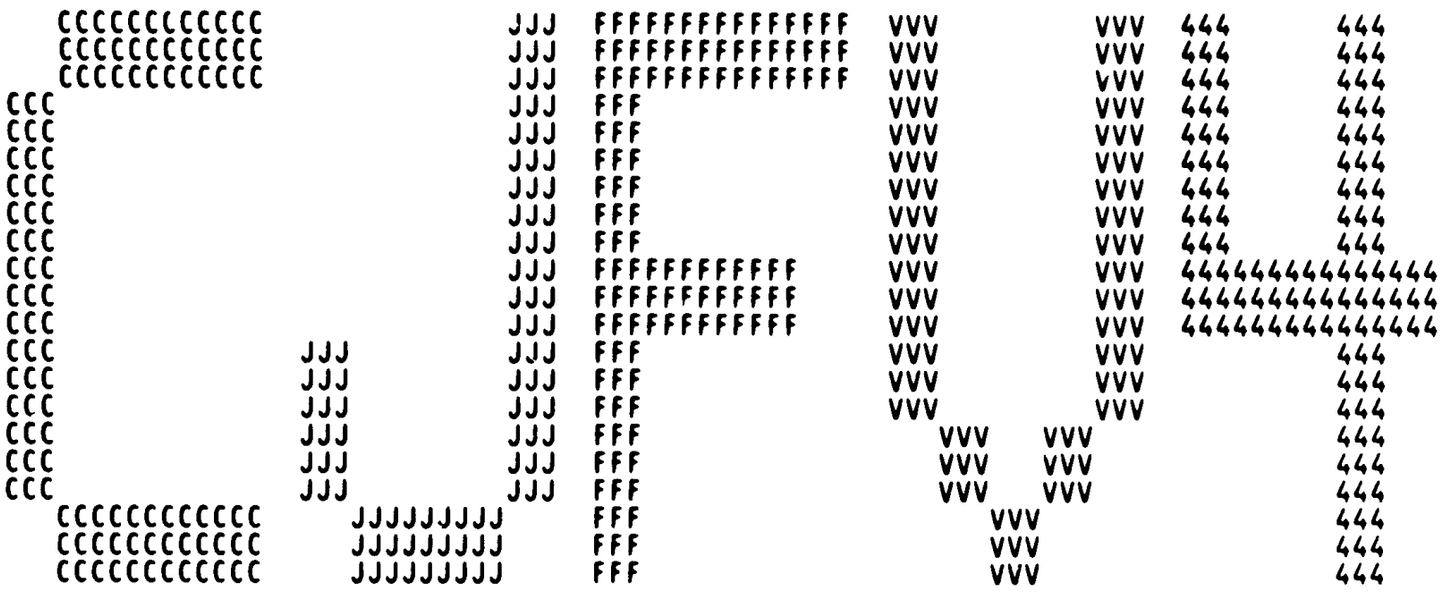
CCCCCCCCCCC JJJ FFFFFFFFFFFFFFF VVV VVV 444 444
CCCCCCCCCCC JJJ FFFFFFFFFFFFFFF VVV VVV 444 444
CCCCCCCCCCC JJJ FFFFFFFFFFFFFFF VVV VVV 444 444
CCC JJJ FFF VVV VVV 444 444
CCC JJJ FFF VVV VVV 444 444
CCC JJJ FFF VVV VVV 444 444
CCC JJJ FFF VVV VVV 444 444
CCC JJJ FFF VVV VVV 444 444
CCC JJJ FFFFFFFFFFFF VVV VVV 444444444444444
CCC JJJ FFFFFFFFFFFFF 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
CCCCCCCCCCC JJJJJJJJJ FFF VVV 444
CCCCCCCCCCC JJJJJJJJJ FFF VVV 444
CCCCCCCCCCC JJJJJJJJJ FFF VVV 444

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


      SSSSSSSS  DDDDDDD     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
```

```
{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:  JOURNALING : DEFINITION OF INTERNAL FILE FORMATS
{
{ Abstract:
{       This module contains the symbolic definitions for non-user accessible
{       file formats.
{
{       ==> PLEASE NOTE:
{           Any time new fields, structures or entries are defined, concurrent
{           changes should be made to the dump formatting utility, JCPDMP,
{           to allow it to recognize and format them.
{
{ Author:       CJF group       Creation Date: 18-FEB-1981
{
{ Modified by:
{
{       V03-027 GJA0059         Greg Awdziewicz         16-Dec-1983
{               Add some comments to JET defs.   Add index delimiter.
{
{       V03-026 JSV0349         Joost Verhofstad        13-JUL-1983
{               Ada PHASE$W_ENTLEN
{
{       V03-025 LY0377          Larry Yetto             26-MAY-1983 11:02:50
{               Correct PRA$L_CSID and all other fields that try to
{               overlay the recovery unit ID.  Add PRA$W_CSID_IDX
{               and PRA$W_CSID_SEQ whic overlay PRA$L_CSID.
{
{       V03-024 JSV0304         Joost Verhofstad        23-MAY-1983
{               Add PRA$L_CSID
{
{       V03-023 JSV0290         Joost Verhofstad        18-MAY-1983
```

```
{               Add OPT, CJL and RHD
{
{       V03-022 MKL0074         Mary Kay Lyons          06-May-1983
{               Add remaster entry symbol definitions. Add symbols for
{               for local sequence numbers.
{
{       V03-021 JSV0230         Joost Verhofstad        27-APR-1983
{               Add RUSYNC bits
{
{       V03-020 LY0345          Larry Yetto             6-APR-1983 11:0C:55
{               Modify NTE structure definition to remove overlays used
{               for slave node control structures only.  A new structure
{               has now been added to JNLDEFINT.SDL to take care of this
{               task.
{
{       V03-019 JSV0191         Joost Verhofstad        14-MAR-1983
{               Change PROL$L_FRSTBVBN to PROL$L_BASEVBN
{
{       V03-018 JSV0153         Joost Verhofstad        18-FEB-1983
{               Add JET$C_DCJL
{
{       V03-017 JSV0150         Joost Verhofstad        17-FEB-1983
{               Add fields: PROL$L_FRSTBVBN, PROL$L_DA1CNV,
{               PROL$L_FRSTSEQN
{
{       V03-016 JSV0145         Joost Verhofstad        14-FEB-1983
{               Add DCJL structure
{
{       V03-015 JSV0137         Joost Verhofstad        03-FEB-1983
{               Replace source, put in null packet
{
{       V03-014 LY0271          Larry Yetto             19-Jan-1983
{               Replace SPARE in CHUNKDEF, the JCP dump journal routine
{               needs it.  Set NTE journal name length back to 12.
{
{       V03-013 ROW0154         Ralph O. Weber          6-JAN-1983
{               Convert MDL to SDL and make SPARE in CHUNKDEF a fill field.
{               Completely replace NTEDEF with new structure which can be used
{               both for file name table entries and for short remote journal
{               identifiers on slave nodes before the first journal assign
{               channel service is executed.
{
{       V03-012 LY0250          Larry Yetto             05-Jan-1983
{               Remove FOF$V_PRIM flag
{
{       V03-011 JSV0103         Joost Verhofstad        3-DEC-1982
{               Add JDB
{
{       V03-010 GJA0031         Greg Awdziewicz,        1-Dec-1982  16:38
{               Removed JDB$... structure definitions.
{
{       V03-009 JSV0080         Joost Verhofstad        08-Oct-1982
{               Add JAB$L_PRCUIC
{
{       V03-008 JSV0077         Joost Verhofstad        08-Oct-1982
{               Add CJL data structure value JET$C_CJL
```

```
{
{       V03-007 LY0054          Larry Yetto             27-Jul-1982
{               Add ACMODE to NTE
{
{       V03-006 JSV0023         Joost Verhofstad        21-Jul-1982
{               misc bits and bytes added. Main one: put SEQ NO
{               in CEH and remove from PRA
{
{       V03-005 JAY0002         John A. Ywoskus         21-Jul-1982
{               Add CEH field to PHASE structure.
{
{       V03-004 JAY0001         John A. Ywoskus         20-Jul-1982
{               Restore PHASE$L_COUNT.
{
{       V03-003 LY0046          Larry Yetto     15-Jul-1982
{               Add MAXREC constant to UTE and NTE definitions
{               Add JET codes for recovery units and PRA$B_ENTATR
{               and PRA$V_COMPLETED. Change CODE field of PHASE
{               structure from $L to $B.
{
{       V03-002 JSV014          Joost Verhofstad        7-Jul-1982
{               Add PROL$L_MASK
{
{       V03-001 LY0031                  Larry Yetto  30-Jun-1982
{               Add name table entry format (NTE)
{
{--
```

```
module $JETDEF;

/+++
/*
/* JET - Journal Entry Type
/*
/*      The first byte of all control entries, after the common
/*      CEH bytes, in the journal file is the
/*      record type.  This defines the values for this byte.
/*
/*      Some of these values are used for record/entry type and
/*      some are used for block/structure types.  This could be a
/*      source of some confusion.  Entries are contained within a
/*      journal bucket structure.  The major structures always
/*      are block structures.
/*
/*      The structure types are PROL, BUFHDR, RHD, OPT, CLT, EPIL, and CJL.
/*
/*      The entry types are JAB, USERREC, CHUNK, JDB, MARK, OPJ, UICTBL,
/*      PHASE1, PHASE2, ABORT, COMPLETED, RESIDUAL, CLEANUP, RESET, RUSYNC,
/*      DCJL, and RMST.
/*
/*      JABLST and RCH are obsolete.  JABLST has been replaced by the
/*      structure type CJL for tape journals and by the entry type DCJL for
/*      disk journals.  Specific RU entry types are used instead of the
/*      nonspecific RCH type.
/*
/*      ==> Any time new structures or entries are defined, concurrent
/*          changes should be made to the dump formatting utility, JCPDMP,
/*          to allow it to recognize and format them.
/*
/*--

constant(
      PROL                                  /* prologue
    , JAB                                   /* journal assign block
    , BUFHDR                                /* buffer header
    , USERREC                               /* user record
    , CHUNK                                 /* chunk (continuation segment)
    , JDB                                   /* journal deassign block.
    , MARK                                  /* mark point
    , JABLST                                /* JAB list
    , OPJ                                   /* open-journal entry
    , RCH                                   /* RU control entry
    , RHD                                   /* Reel header (for tapes)
    , OPT                                   /* open tape entry
    , CLT                                   /* close tape
    , EPIL                                  /* epilogue
    , UICTBL                                /* UIC table modification record
    , PHASE1                                /* phase1 entry
    , PHASE2                                /* phase2 entry
    , ABORT                                 /* abort entry
    , COMPLETED                             /* entry indicating RU is rolled fw.
    , RESIDUAL                              /* residual entry
    , CLEANUP                               /* cleanup entry
    , RESET                                 /* reset entry
```

```
    , RUSYNC                                    /* rusync entry
    , CJL                                       /* current JAB list for tapes
    , DCJL                                      /* current JAB list for disk
    , RMST                                      /* remaster journal entry
                                                /* Any new JET codes should be inserted before this.
    , HIGH_LIMIT )                              /* high-end delimiter for indexing the JET codes.
    equals 1 increment 1  prefix JET tag $C;


end_module $JETDEF;
```

```
module $PROLDEF;

/***
/*
/* PROL -  Journal Prologue (on disk or tape)
/*
/*         The first block of each disk journal file is used as a prologue
/*         block to contain control information. For tape journals the
/*        prologues are written as tape blocks. The fields defined for
/*        a prologue block are described here
/*
/*--

aggregate PROLDEF structure prefix PROL$;
    CEH_OVERLAY union fill;
        CEH longword unsigned dimension 5;              /* control entry header fields
        CEH_FIELDS structure fill;
            LEN longword unsigned;                      /* length (for tapes)
            JNLID longword unsigned;                    /* journal ID (for tapes)
            TSIZE word unsigned;                        /* size record (for tapes)
            STRUCT byte unsigned;                       /* structure type (for tapes)
            FILL_1 byte fill prefix PROLDEF tag $$;     /* spare (for tapes)
        end CEH_FIELDS;
    end CEH_OVERLAY;
    TYPE byte unsigned;                                 /* record type for prologue
    SIZE byte unsigned;                                 /* prologue size in blocks
    MAJVER byte unsigned;                               /* major software version number
    MINVER byte unsigned;                               /* minor software version number
    LINK longword unsigned;                             /* (in core use only) next prologue
    NAME character length 13;                           /* journal name (counted ASCII)
    FILL_2 byte dimension 3 fill prefix PROLDEF tag $$; /* spare
    JNL_TYPE byte unsigned;                             /* journal type
    COPIES byte unsigned;                               /* number of copies created (disks only)
    PROT word unsigned;                                 /* protection mask of journal
    DATCRE longword unsigned dimension 2;               /* creation date/time
    DATOPN longword unsigned dimension 2;               /* date/time last opened
    DATCNV longword unsigned dimension 2;               /* date/time a new version was created
    CRECNT longword unsigned;                           /* creation count
    UIC longword unsigned;                              /* UIC of journal
    FILEXT word unsigned;                               /* file extend size
    BUFSIZ word unsigned;                               /* buffer size (in bytes)
    MAXSIZ word unsigned;                               /* maximum journal entry size
    ACMODE byte unsigned;                               /* access mode for journal
    FILL_3 byte fill prefix PROLDEF tag $$;             /* spare
    FACCOD word unsigned;                               /* facility code
    APPLID word unsigned;                               /* application ID
    "MASK" longword unsigned;                           /* mask (for AT journals only)
    QUOTA longword unsigned;                            /* quota , for Ru journals only
    FLAGS_OVERLAY union fill;
        FLAGS longword unsigned;                        /* flags; maust be same as in JSB
        FLAGS_BITS structure fill;
            TMPJNL bitfield mask;                       /* temp journal
            SITE bitfield mask;                         /* (*) installation journal
            CREATE bitfield mask;                       /* create new file
            CREATE_IF bitfield mask;                    /* create-if
            TMPFIL bitfield mask;                       /* temporary file: delete on device deletion
```

```
        CREACP bitfield mask;                   /* create new ACP for this journal
        DIFACP bitfield mask;                   /* use different ACP (name in JSB)
        REPLACE bitfield mask;                  /* replace current journal with this
        TAPEDRIVE bitfield mask;                /* create journal tape drive (internal)
    end FLAGS_BITS;

                                                /*  marked (*) are used in the prologue
                                                /*  NOTE: do not change JSB independently
end FLAGS_OVERLAY;
STATUS_OVERLAY union fill;                      /* journal file status
    STATUS longword unsigned;
    STATUS_BITS structure fill;
        OPEN bitfield mask;                     /* journal file is opened, not closed
        NORUS bitfield mask;                    /* if set : no RUEs in RUL when journal
                                                /*        closed (RU journals only)
    end STATUS_BITS;
end STATUS_OVERLAY;
FRSTSEQN longword unsigned;                     /* sequence number first entry in this file
BASEVBN longword unsigned;                      /* VBN first bucket in this journal file
LASTVBN longword unsigned;                      /* VBN last bucket written (only valid
                                                /*  if journal properly closed + used
                                                /*  by ACP for unopened tape journals)
LASTSEQN longword unsigned;                     /* last sequence number used
FST_VBN longword unsigned;                      /* VBN first free byte
FST_OFF word unsigned;                          /* offset first free byte
FILL_4 word fill prefix PROLDEF tag $$;         /* spare
PRV_VBN longword unsigned;                      /* VBN last chunk
PRV_OFF word unsigned;                          /* offset last chunk
FILL_5 word fill prefix PROLDEF tag $$;         /* spare
PRV_EVBN longword unsigned;                     /* VBN last entry
PRV_EOFF word unsigned;                         /* offset last entry
FILL_6 word fill prefix PROLDEF tag $$;         /* spare
LASTAID longword unsigned;                      /* last assign ID used
constant "LENGTH" equals . prefix PROL$ tag K;  /* length of structure
constant "LENGTH" equals . prefix PROL$ tag C;  /* length of structure

end PROLDEF;

end_module $PROLDEF;
```

```
module $JABDEF;

/***
/*
/* JAB - Journal Assign Block
/*
/*        One journal assign block is placed in the journal file
/*        whenever the journal is assigned by the $ASSJNL or $CREJNL
/*        service.
/*
/*--

aggregate JABDEF structure prefix JAB$;
    CEH longword unsigned dimension 5;                  /* control entry header fields
    TYPE byte unsigned;                                 /* record type to indicate JAB
    FILL_1 byte fill prefix JABDEF tag $$;              /* spare
    JNLID word unsigned;                                /* journal ID for multi-journal tapes
    ASSSEQ longword unsigned;                           /* assign sequence number (unique per $ASSJNL per journal)
    FACCOD word unsigned;                               /* facility code
    NODE byte unsigned dimension 6;                     /* host node number of user
    UIC longword unsigned;                              /* UIC used as UIC for entries written
    PID longword unsigned;                              /* PID of user
    PRCNAM character length 16;                         /* Process name
    TIME quadword unsigned;                             /* time (standard 64-bit format)
    PROT word unsigned;                                 /* protection mask of $ASSJNL
    REFCNT word unsigned;                               /* (inclusive) reference count of
                                                        /* writers currently assigned to this
                                                        /* journal.
    ACMODE byte unsigned;                               /* access mode of $ASSJNL
    FILL_2 byte dimension 3 fill prefix JABDEF tag $$;  /* spare
    PRCUIC longword unsigned;                           /* process UIC
    constant "LENGTH" equals . prefix JAB$ tag K;       /* length of structure
    constant "LENGTH" equals . prefix JAB$ tag C;       /* length of structure
end JABDEF;

end_module $JABDEF;
```

```
module $JDBDEF;

/+++
/*
/* JDB - Journal Deassign Block
/*
/*        One journal deassign block is placed in the journal file
/*        whenever the journal is deassigned by the $DEASJNL or $DELJNL
/*        service.  JAB$... field definitions are used for the JDB.
/*
/*
/*--

aggregate JDBDEF structure prefix JDB$;
    CEH longword unsigned dimension 5;                  /* control entry header fields
    TYPE byte unsigned;                                 /* record type to indicate JAB
    FILL_1 byte fill prefix JDBDEF tag $$;              /* spare
    JNLID word unsigned;                                /* journal ID for multi-journal tapes
    ASSSEQ longword unsigned;                           /* assign sequence number (unique per $ASSJNL per journal)
    FACCOD word unsigned;                               /* facility code
    NODE byte unsigned dimension 6;                     /* host node number of user
    UIC longword unsigned;                              /* UIC used as UIC for entries written
    PID longword unsigned;                              /* PID of user
    PRCNAM character length 16;                         /* Process name
    TIME longword unsigned dimension 2;                 /* time (standard 64-bit format)
    PROT word unsigned;                                 /* protection mask of $ASSJNL
    REFCNT word unsigned;                               /* (inclusive) reference count of
                                                        /* writers currently assigned to this
                                                        /* journal.
    ACMODE byte unsigned;                               /* access mode of $ASSJNL
    FILL_2 byte dimension 3 fill prefix JDBDEF tag $$;  /* spare
    PRCUIC longword unsigned;                           /* process UIC
    constant "LENGTH" equals . prefix JDB$ tag K;       /* length of structure
    constant "LENGTH" equals . prefix JDB$ tag C;       /* length of structure
end JDBDEF;

end_module $JDBDEF;
```

```
module $PHASEDEF;

/+++
/*
/* PHASE - Phase Marker
/*
/*        When the user does a RUF$PH1_END, RUF$PH2_END, RUF$CANCEL,
/*        RUF$RESET or a RUF$MARKPOINT, a phase marker is written
/*        to the RU journal by the ENDRU routine.
/*
/*--

aggregate PHASEDEF structure prefix PHASE$;
    CEH longword unsigned dimension 5;                  /* Control entry header
    CODE byte unsigned;                                 /* Phase Code
    ENTLEN word unsigned;                               /* length variable portion
    FILL_1 byte dimension 1 fill prefix PHASEDEF tag $$;/* Spare
    constant RID equals . prefix PHASE$ tag K;          /* offset for RU-ID
    constant RID equals . prefix PHASE$ tag C;          /* offset for RU-ID
    RUID character length 16;                           /* Recovery Unit ID (all cases)
    COUNT longword unsigned;                            /* Journal count
    MARKPT longword unsigned;                           /* Markpoint ID if RUF$MARKPOINT
    FLAGS_OVERLAY union fill;
        FLAGS longword unsigned;                        /* flags longword
        FLAGS_BITS structure fill;
            RUSYNC bitfield mask;                       /* RUSYNC expected later
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    constant "LENGTH" equals . prefix PHASE$ tag K;     /* Length of fixed portion
    constant "LENGTH" equals . prefix PHASE$ tag C;     /* Length of fixed portion
    NAME_AREA byte unsigned;                            /* Start of area containing names (always at the end!)
end PHASEDEF;

end_module $PHASEDEF;
```

```
module $OPJDEF;

/***
/*
/* OPJ - open journal entry
/*
/*         An open-journal entry is written each time the journal file
/*        if opened for a journal-creation
/*
/*--

aggregate OPJDEF structure prefix OPJ$;
    CEH longword unsigned dimension 5;                  /* control entry header
    TYPE byte unsigned;                                 /* type field
    FILL_1 byte fill prefix OPJDEF tag $$;              /* spare
    FILL_2 word fill prefix OPJDEF tag $$;              /* spare
    constant "LENGTH" equals . prefix OPJ$ tag K;       /* length of structure
    constant "LENGTH" equals . prefix OPJ$ tag C;       /* length of structure
end OPJDEF;

end_module $OPJDEF;
```

```
module $RMSTDEF;

/+++
/*
/* RMST - remaster journal entry
/*
/*        A remaster journal entry is written each time the journal file
/*        is opened due to the journal being remastered.
/*
/*--

aggregate RMSTDEF structure prefix RMST$;
    CEH longword unsigned dimension 5;                  /* control entry header
    TYPE byte unsigned;                                 /* type field
    FILL_1 byte fill prefix RMSTDEF tag $$;             /* spare
    FILL_2 word fill prefix RMSTDEF tag $$;             /* spare
    constant "LENGTH" equals . prefix RMST$ tag K;      /* length of structure
    constant "LENGTH" equals . prefix RMST$ tag C;      /* length of structure
end RMSTDEF;

end_module $RMSTDEF;
```

```
module $PRADEF;

/+++
/*
/* PREAMBLE - Journal record preamble
/*
/*        Each user record in the journal file is preceded by a preamble
/*        that describes the entry.
/*
/*        This preamble MUST always have an even length
/*
/*--

aggregate PRADEF structure prefix PRA$;
    LEN word unsigned;                          /* total length entry - this word
    LEN2 word unsigned;                         /* second word of lengtn; for tapes only
    TYPE_OVERLAY union fill;
        TYPE byte unsigned;                     /* record type to indicate user reocrd
        TYPE_BITS structure fill;
            USER bitfield mask;                 /* user entry
            CONTR bitfield mask;                /* control entry
            MULCHNKS bitfield mask;             /* multiple chunks
            FRSTCHNK bitfield mask;             /* first chunk
            LSTCHK bitfield mask;               /* last chunk
            NULL bitfield mask;                 /* null entry
        end TYPE_BITS;
    end TYPE_OVERLAY;
    PRALEN byte unsigned;                       /* preamble length
    PRVOFF word unsigned;                       /* offset previous record in its bucket
                                                /*  for RU jnls: offset previous entry by
                                                /*               same RU
    PRVVBN longword unsigned;                   /* VBN of bucket with previous entry
                                                /*  for RU jnls: VBN previous entry by
                                                /*               same RU
    SEQNO longword unsigned;                    /* sequence number
    LSEQNO longword unsigned;                   /* local sequence number
    PRATYPE byte unsigned;                      /* type field for entry
    ENTATR byte unsigned;                       /* entry attribute
    ENTLEN word unsigned;                       /* Total length user written entry.
    constant RID equals . prefix PRA$ tag K;    /* RID starts here.
    constant RID equals . prefix PRA$ tag C;    /* RID starts here.
    RIDVAL_OVERLAY union fill;
        RIDVAL quadword unsigned dimension 2;   /* RU ID (for RU journals only)
                                                /* The next subfields are for other than
                                                /*  RU journals
        RID_OVERLAY structure fill;
            "MASK" longword unsigned;           /* mask that user specified on $QIO
            RUID_LW2 longword unsigned;         /* second longword of RUID
            CSID_UNION union fill ;
                CSID longword unsigned;         /* CSID portion of RUID,
                                                /*  also used for other journals (Not just RU)
                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 RID_OVERLAY;
end RIDVAL_OVERLAY;
TIME quadword unsigned;                         /* time (standard 64-bit format)
ASSSEQ longword unsigned;                       /* assign sequence number (of related JAB)
FLAGS_OVERLAY union fill;
    FLAGS word unsigned;                        /* flags byte
    FLAGS_BITS structure fill;
        USERENT bitfield mask;                  /* user written entry
        MULTIPLE bitfield mask;                 /* multiple entry
        SEQNOVF bitfield mask;                  /* seq no overflow
                                                /*
                                                /* The following flags are for RU
                                                /* journal entries only
                                                /*
        RUJNL bitfield mask;                    /* this is a RU journal entry
        ROLLFW bitfield mask;                   /* roll forward entry
        ROLLBW bitfield mask;                   /* roll back entry
        FIRST bitfield mask;                    /* first entry for this RU
        PHASE1 bitfield mask;                   /* phase 1 done
        PHASE2 bitfield mask;                   /* phase 2 done
        ABORT bitfield mask;                    /* RU has been aborted
        COMPLETED bitfield mask;                /* completed RU
        P2$AB$2 bitfield mask;                  /* phase2 or abort entry to be
                                                /* encountered twice before deletion
        RUSYNCEX bitfield mask;                 /* rusync expected later
        RUSYNCWR bitfield mask;                 /* rusync written
    end FLAGS_BITS;
end FLAGS_OVERLAY;
FACCOD word unsigned;                           /* facility code
constant "LENGTH" equals . prefix PRA$ tag K;   /* length of structure
constant "LENGTH" equals . prefix PRA$ tag C;   /* length of structure
end PRADEF;

end_module $PRADEF;
```

```
module $CEHDEF;

/***
/*
/* CONTROL ENTRY HEADER - header of control entries (written using WRITELBLK)
/*
/*        Each control entry in the journal file is preceded by a header
/*        that describes the entry.
/*        The preamble is longer than this header and its first few fields
/*         are identical to a control entry header
/*
/*        This header MUST always have an even length
/*
/*--

aggregate CEHDEF structure prefix CEH$;
    LEN word unsigned;                          /* total length entry - this word
    LEN2 word unsigned;                         /* second word of length; for tapes only
    TYPE_OVERLAY union fill;
        TYPE byte unsigned;                     /* record type to indicate user reocrd
        TYPE_BITS structure fill;
            USER bitfield mask;                 /* user entry
            CONTR bitfield mask;                /* control entry
            MULCHNKS bitfield mask;             /* multiple chunks
            FRSTCHNK bitfield mask;             /* first chunk
            LSTCHK bitfield mask;               /* last chunk
            NULL bitfield mask;                 /* null entry
        end TYPE_BITS;
    end TYPE_OVERLAY;
    CEHLEN byte unsigned;                       /* header length
    PRVOFF word unsigned;                       /* offset previous entry in its bucket
                                                /*  for RU jnls: offset previous entry by
                                                /*               same RU
    PRVVBN longword unsigned;                   /* VBN of bucket with previous entry
                                                /*  for RU jnls: VBN previous entry by
                                                /*               same RU
    SEQNO longword unsigned;                    /* sequence number
    LSEQNO longword unsigned;                   /* local sequence number
    constant "LENGTH" equals . prefix CEH$ tag K;   /* length of structure
    constant "LENGTH" equals . prefix CEH$ tag C;   /* length of structure
end CEHDEF;

end_module $CEHDEF;
```

```
module $CHUNKDEF;

/***
/*
/* CHUNK HEADER - header of a chunk (not the first chunk of an entry)
/*
/* An entry is broken up in chunks if it doesn't fit in one bucket
/* or if it is an RUjnl entry that doesn't fit between two valid entries
/* The first chunk gets the regular preamble/control entry header, in
/* which the type field specifies that it is the first of multiple
/* chunks. Subsequent chunks get the chunk header
/*
/* The fields in the CHUNK structure are in the same position as for preambles
/*
/*        This header MUST always have an even length
/*
/*--

aggregate CHUNKDEF structure prefix CHUNK$;
    LEN word unsigned;                                  /* total length entry - this word
    LEN2 word unsigned;                                 /* second word of length; for tapes only
    TYPE_OVERLAY union fill;
        TYPE byte unsigned;                             /* record type to indicate user reocrd
        TYPE_BITS structure fill;
            USER bitfield mask;                         /* user entry
            CONTR bitfield mask;                        /* control entry
            MULCHNKS bitfield mask;                     /* multiple chunks
            FRSTCHNK bitfield mask;                     /* first chunk
            LSTCHK bitfield mask;                       /* last chunk
            NULL bitfield mask;                         /* null entry
        end TYPE_BITS;
    end TYPE_OVERLAY;
    LEN byte unsigned;                                  /* header length
    PRVOFF word unsigned;                               /* offset previous chunk in its bucket
                                                        /*  for RU jnls: offset previous chunk by
                                                        /*               same RU
    PRVVBN longword unsigned;                           /* VBN of bucket with previous chunk
                                                        /*  for RU jnls: VBN previous chunk by
                                                        /*               same RU
    SEQNO longword unsigned;                            /* sequence number of entry
    LSEQNO longword unsigned;                           /* local sequence number of entry
    CHTYPE byte unsigned;                               /* to indicate that this is a chunk header
/*    SPARE byte unsigned dimension 3 fill;             /* spare bytes
    SPARE byte unsigned dimension 3 tag B;              /* spare bytes
    RIDVAL quadword unsigned dimension 2;               /* RU ID
    constant "LENGTH" equals . prefix CHUNK$ tag K;     /* length of structure
    constant "LENGTH" equals . prefix CHUNK$ tag C;     /* length of structure
end CHUNKDEF;

end_module $CHUNKDEF;
```

```
module $UTEDEF;

/*++
/*
/* UTE - UIC table entry
/*
/*        The following defines the UIC table record format
/*
/*--

aggregate UTEDEF structure prefix UTE$;
    RECSIZ_OVERLAY union fill;
        RECSIZ word unsigned;                           /* Variable length record length
        RECSIZ_FIELDS structure fill;
            TYPE byte unsigned;                         /* record type to indicate UTE
            "FILL" byte unsigned;                       /* filler
        end RECSIZ_FIELDS;
    end RECSIZ_OVERLAY;
    VERSION word unsigned;                              /* record version ! field
    constant VERSION    equals 1  prefix UTE tag $C;    /* Record version ! constant
    UIC_OVERLAY union fill;
        UIC longword unsigned;                          /* uic for journal device
        UIC_FIELDS structure fill;
            UIC_MBM word unsigned;                      /* UIC member number
            UIC_GRP word unsigned;                      /* UIC group number
        end UIC_FIELDS;
    end UIC_OVERLAY;
    JNLNAM_OVERLAY union fill;
        JNLNAM character length 13;                     /* journal name (counted ASCII)
        JNLNAMLEN byte unsigned;                        /* journal name length subfield
    end JNLNAM_OVERLAY;
    JNLTYP byte unsigned;                               /* journal type
    FILL_1 word fill prefix UTEDEF tag $$;              /* spare
    FLAGS longword unsigned;                            /* flag longword
    ENTTIME quadword unsigned;                          /* Time entry was made
    CRETIME quadword unsigned;                          /* Time UIC table file was first created
    constant "LENGTH" equals . prefix UTE$ tag K;       /* length of structure
    constant "LENGTH" equals . prefix UTE$ tag C;       /* length of structure
    constant BLKSIZ     equals 512  prefix UTE tag $C;  /* UIC table block size
    constant MAXREC     equals 512  prefix UTE tag $C;  /* Maximum record size
end UTEDEF;

end_module $UTEDEF;
```

```
module $NTEDEF;

/+++
/*
/* NTE - Name table entry
/*
/*      The following defines the Name table record format.
/*
/*      A close approximation of this format is used to describe journals
/*      created remotly in a cluster to which no channel assignments have
/*      ever been made on this node.  Should a channel assignment ever occur,
/*      this node would become a slave node for the journal.  Such entries are
/*      hung off of the CRB for each journal type in a singly linked list.
/*
/*--

#JNLNAMSIZ = 12;

aggregate NTEDEF structure prefix NTE$;
    RECSIZ word unsigned;                       /* Variable length record length
    VERSION word unsigned;                      /* record version field
    constant VERSION equals 1 tag C;            /* Record version constant
    CRETIME quadword unsigned;                  /* Time name table file first created
    UIC_OVERLAY union fill;
        UIC longword unsigned;                  /* uic for journal device
        UIC_FIELDS structure fill;
            UIC_MBM word unsigned;              /* UIC member number
            UIC_GRP word unsigned;              /* UIC group number
        end UIC_FIELDS;
    end UIC_OVERLAY;
    FLAGS longword unsigned;                    /* flag longword
    JNLNAM_OVERLAY union fill;
        JNLNAMLEN byte unsigned;                /* journal name length subfield
        JNLNAM character length #JNLNAMSIZ+1;   /* journal name (counted ASCII)
    end JNLNAM_OVERLAY;
    PROT word unsigned;                         /* Journal device protection
    ACMODE byte unsigned;                       /* Journal device access mode
    JNLDEV byte unsigned;                       /* Device for journal files
    COPIES byte unsigned;                       /* Number of device name strings
    SPARE_2 byte fill;
    JNLTYP byte unsigned;                       /* journal type
    SPARE_3 byte fill;
    ENTTIME quadword unsigned;                  /* Time entry was made
    ACPNAM_OVERLAY union fill;
        ACPNAMLEN byte unsigned;                /* ACP name length subfield
        ACPNAM character length 16;             /* Jrnl ACP process name (counted ASCII)
    end ACPNAM_OVERLAY;
    constant FIXEDLEN equals .;                 /* fixed header length
    constant FIXEDLEN equals . tag C;           /* fixed header length
    constant BLKSIZ equals 512 tag C;           /* name table block size
    constant MAXREC equals 512 tag C;           /* Maximum record size
end NTEDEF;

end_module $NTEDEF;
```

```
module $FOFDEF;

/***
/*
/* FOF - File open flags
/*
/*        The following flag definitions will be used by the open
/*        routine to determine what action to take.  These flags
/*        were previously part of the UTE structure but since they
/*        are now also needed by name table routines they have been
/*        moved.
/*
/*--

aggregate FOFDEF  union prefix FOF$;
    FOFDEF_BITS structure fill;
        FILL_1 bitfield fill prefix FOFDEF tag $$;
        CREATE bitfield mask;                           /* Create new file
        READ bitfield mask;                             /* Read access only.
        NOREAD bitfield mask;                           /* Do not allow others read access
        NOWRITE bitfield mask;                          /* Open with write lock
        NAMTBL bitfield mask;                           /* Perform name table open instead
                                                        /*   of UIC table
    end FOFDEF_BITS;
end FOFDEF;

end_module $FOFDEF;
```

```
module $DCJLDEF;

/*++
/*
/* DISK CURRENT JAB LIST - List of current JABs for disk journal
/*
/*
/*      When a new version of a file is created a list of all JABs
/*      is written out to the journal file. This allows us to continue
/*      to write entries for existing channels to the journal, so
/*      on reading we don't need to get JABs from the previous version.
/*
/*--

aggregate DCJLDEF structure prefix DCJL$;
    STRUCT byte unsigned;                                   /* structure type field
    SPARE1 byte unsigned;                                   /* spare
    NUM word unsigned;                                      /* number of JABs
    constant "FIXED_LEN" equals . prefix DCJL$ tag K;       /* fixed length portion of structure
    constant "FIXED_LEN" equals . prefix DCJL$ tag C;       /* fixed length portion of structure
end DCJLDEF;

end_module $DCJLDEF;
```

```
module $CJLDEF;
/*++
/*
/* CJL - Current JAB list (for tapes)
/*
/* An CJL is written when the tape is mounted again, or next reel is
/* mounted. This entry contains a full list of all JABs for all journals
/* going to this tape group
/*
/*--


aggregate CJLDEF structure fill prefix CJL$;
    LEN longword unsigned;                          /* length (in ASCII)
    FILL_1 longword fill prefix CJLDEF tag $$;      /* spare
    TSIZE word unsigned;                            /* entry size
    STRUCT byte unsigned;                           /* structure type
    FILL_2 byte fill prefix CJLDEF tag $$;          /* spare
    SECTIONS word unsigned;                         /* number of pieces this entry is broken
                                                    /*  up in
    SECNUM word unsigned;                           /* section number for this section
    constant FIXED_LEN equals . prefix CJL$ tag K;  /* end of fixed length portion
    constant FIXED_LEN equals . prefix CJL$ tag C;  /* end of fixed length portion
                                                    /*  (1 based)
end CJLDEF;

end_module $CJLDEF;
```

```
module $OPTDEF;
/***
/*
/* OPT - Open tape entry
/*
/* An open-tape entry is written when the tape is mounted again.
/*
/*--


aggregate OPTDEF structure fill prefix OPT$;
    LEN longword unsigned;                          /* length (in ASCII)
    FILL_1 longword fill prefix OPTDEF tag $$;      /* spare
    TSIZE word unsigned;                            /* entry size
    STRUCT byte unsigned;                           /* structure type
    FILL_2 byte fill prefix OPTDEF tag $$;          /* spare
    TIME quadword unsigned;                         /* time
    constant "LENGTH" equals . prefix OPT$ tag K;   /* end of fixed length portion
    constant "LENGTH" equals . prefix OPT$ tag C;   /* end of fixed length portion
end OPTDEF;

end_module $OPTDEF;
```

```
module $RHDDEF;
/+++
/*
/* RHD - Reel header
/*
/* A reel header is written to the start of each reel
/* A new reel may first get a few buffers that were still being written
/* to the previous reel while EOT was passed.
/*
/*--


aggregate RHDDEF structure fill prefix RHD$;
    LEN longword unsigned;                              /* length (ASCII)
    JNLID longword unsigned;                            /* journal ID
    TSIZE word unsigned;                                /* entry size
    STRUCT byte unsigned;                               /* structure type
    FILL_1 byte fill prefix RHDDEF tag $$;              /* spare
    COPIES word unsigned;                               /* total number in group
    NUMBER word unsigned;                               /* number of this copy in group
    REELNUM longword unsigned;                          /* number of reel in volume set
    UIC longword unsigned;                              /* UIC of owner of group
    PROT word unsigned;                                 /* protection mask for group
    FILL_2 word fill prefix RHDDEF tag $$;              /* spare
    MAX_JNLS word unsigned;                             /* maximum number of journals allowed
    TYPES_OVERLAY union fill;
        TYPES word unsigned;                            /* journal types allowed on tape
        TYPES_BITS structure fill;
            BI bitfield mask;
            AI bitfield mask;
            AT bitfield mask;
        end TYPES_BITS;
    end TYPES_OVERLAY;
    GRPNAM character length 13;                         /* group name (counted ASCII)
    FILL_3 byte dimension 3 fill prefix RHDDEF tag $$;  /* spare
    STATUS_OVERLAY union fill;
        STATUS longword unsigned,                       /* status
        STATUS_BITS structure fill;
            SPOOL bitfield mask;                        /* a spool file is used for this group.
        end STATUS_BITS;
    end STATUS_OVERLAY;
    INITTIME quadword unsigned;                         /* time at which tape reel was first written
    constant "LENGTH" equals . prefix RHD$ tag K;       /* end of fixed length portion
    constant "LENGTH" equals . prefix RHD$ tag C;       /* end of fixed length portion
end RHDDEF;

end_module $RHDDEF;
```

JNLBUSR
R32

JNLDEFINT
SDL

CJFV4.

JNLPREFIX
R32

CJFRUFMAC
SDL

RUFUSR
SDL

UPGRADE
LIS

JNLFILE
SDL

BOPTIONS
R32

JNLDEF
SDL