

[illegible]

CL

```

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

```


Version: 'V04-000'

COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
ALL RIGHTS RESERVED.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
TRANSFERRED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

++

FACILITY: SYSLOA - System loadable code

ABSTRACT:

This file contains the SDL source for definitions relating to
the cluster loadable code.

ENVIRONMENT:

n/a

AUTHOR: Steve Beckhardt CREATION DATE: 7-Jan-1983

MODIFIED BY:

V03-047	DWT0229	David W. Thiel	25-Jul-1984
	Update protocol level to 12 to force incompatibility with FT2 update.		
V03-046	DWT0224	David W. Thiel	27-Jun-1984
	Update protocol level to 11 for FT2 update.		
V03-045	DWT0203	David W. Thiel	25-Mar-1984
	Remove symbols CNCTSV_QUORUM and CNCTSV_TRANSITION.		

V03-044 RSH0123 R. Scott Hanna 25-Mar-1984
Replace \$CLUQBDEF with \$CLUQFDEF.

V03-043 DWT0198 David W. Thiel 23-Mar-1984
Correct previous additions to CLMPRODEF. Add
CLMSTSSQ_REFTIME to CLMSTSDEF.

V03-042 DWT0197 David W. Thiel 22-Mar-1984
Add flags field to CLMPRODEF. Remove some obsolete
bits from the flags field in CLMSTSDEF. Update
protocol level to 10.

V03-041 SRB0117 Steve Beckhardt 18-Mar-1984
Added message definitions for distributed deadlock detection.

V03-040 DWT0186 David W. Thiel 9-Mar-1984
Add CLSMMSG\$K_FAC_BLK for block transfer messages.
Add CLSMMSG\$K_FAC_TST for testing/measurement.
Add CLMBLKDEF to define special block transfer
messages.

V03-039 DWT0176 David W. Thiel 27-Feb-1984
Add CLMPRO\$W_QDVOTES to propagage number of votes
assigned to the quorum disk. Add CLMSTSSV SHUTDOWN
to support cluster shutdown and CLMSTSSV QF DYNVOTE.
Add CLMDRSSV LONG BREAK. Add CLUBTX\$L_MSGBCD.
Update CNCT\$C_PROTOCOL to 9.

V03-038 DWT0159 David W. Thiel 11-Jan-1983
Add CLSMMSG\$K_FAC_CSP for CSP facility messages.

V03-037 SRB0106 Steve Beckhardt 18-Nov-1983
Added several definitions in \$LKMSGDEF.

V03-036 DWT0151 David W. Thiel 1-Dec-1983
Add CLMSTSSW_QDVOTES to support variable number of
votes for the quorum disk. Remove unused fields
from CLMSTS message. Bump cluster protocol level
(CNCT\$K_PROTOCOL) to 8. Compute worst case cluster
message size as CLSMMSG\$K_MAXMSG and update some
message definitions to support this computation.
Remove CLMPRO\$L_DIRSYS field from CLMPRODEF.

V03-035 DWT0144 David W. Thiel 11-Nov-1983
Add CLMSTSSW_LCKDIR field to CLMSTSDEF and the
CLMPRO\$L_FMERIT and CLMPRO\$W_MEMSEQ fields to CLMPRODEF.

V03-034 RSH0076 R. Scott Hanna 09-Nov-1983
\$CLUQBDEF - Change INVALID flag to IGNORE and move it out
of the fields that contribute to the checksum. Update the
quorum block version number.

V03-033 LY0433 Larry Yetto 8-NOV-1983 17:33
fix spelling of BLKREADDATA

V03-032 DWT0141 David W. Thiel 07-Nov-1983

Correct header definition for connection manager messages.
Add definition for protocol level (CNCT\$K_PROTOCOL).
Remove obsolete definitions from \$CNCTDEF.

V03-031 LY0417 Larry Yetto 16-SEP-1983 17:02:17
Add CJMSG\$W_RD_CPLXLEN field to CJF read message

V03-030 LY0416 Larry Yetto 15-SEP-1983 14:08:14
Add CJMSG\$L_RD_IRP field to CJF read message
Add CJMSG\$K_BLKJNLRC function code

V03-029 LY0410 Larry Yetto 25-AUG-1983 14:21:22
Add CJMSG\$K_REQINIT and CJMSG\$K_BLKINIT function codes.

V03-028 RSH0057 R. Scott Hanna 23-Aug-1983
Add \$CLUQBDEF

V03-027 ROW0214 Ralph O. Weber 23-AUG-1983
Add a BTX field to save the return address for
CNX\$PARTNER_RESPOND. Previous use of another field proved
fatal to error recovery, and there are no other fields to
share this function with.

V03-026 DWT0124 David W. Thiel 22-Aug-1983
Add reply code definitions for the CLMCNX\$B_REPLY field
to support recovery from failures during state transitions.
Add CLMCNX\$K_DATA phase to values for the
CLMCNX\$B_XTN_PHASE field to tag messages following the
lock request and preceding the Phase 1 message. Improve
SDL usage in defining the flag bits in the CLMST\$B_FLAGS
field. Remove the obsolete CNCT\$B_DIRSYS field. Improve
SDL usage in defining the flags bits in the CNCT\$B_CLSSTS
CNCT\$B_CNXSTS fields. Correct structure name in CLMTOP.

V03-025 RNG0025 Rod N. Gamache 19-Aug-1983
Remove unneeded LIMSG\$ symbols, move start of GETLKI
messages to leave room for block transfer buffer handle.

V03-024 ROW0206 Ralph O. Weber 9-AUG-1983
Fix CLUBTX\$K_LENGTH to match CLUBTX\$T_MSG_BUF.

V03-023 LY0408 Larry Yetto 4-AUG-1983 10:57:59
Add yet a few more fields to assorted CJF messages, this
time they are for information needed for a cancel IO.
Add RUEBIT function code

V03-022 DWT0114 David W. Thiel 29-Jul-1983
Cleanup a bit. Add quorum disk support. Add
CLMTOPDEF messages and support for quorum change
and transition status request messages.

V03-021 LY0400 Larry Yetto 29-JUL-1983 11:42:21
Add new fields to CJF messages needed for block transfers

V03-020 LY0389 Larry Yetto 1-JUL-1983 10:31:01
Add new CJF dispatch codes and new fields to CJF read message

Add CJMSG\$MINF_MSG and CJMSG\$GWRBUF_MSG and CJMSG\$RWRBUF_MSG structures.

- V03-019 ROW0185 Ralph O. Weber 22-JUN-1983
Modify connection manager message format to include a buffer handle. This field is located immediately following the "standard" connection manager header. However, it is not included in the byte count for the "standard" header. This prevents messages which do not need the buffer handle (i.e. non-block transfer request messages) from having to carry it as unused space. Also add definition for the BTX (block-transfer extension), structure type CLUBTX\$.
- V03-018 LY0384 Larry Yetto 17-JUN-1983 08:17:31
Modify CJF read init message format to handle read also. move CJF dispatch function codes here from the CJF definition file
- V03-017 RNG0017 Rod N. Gamache 14-Jun-1983
Add distributed GETLKI message definitions.
- V03-016 LY0379 Larry Yetto 1-JUN-1983 16:42:39
Added new fields to generic function message and status message. Added CJF readinit message format.
- V03-015 SRB0088 Steve Beckhardt 26-May-1983
Added new lock manager messages for failover, added some fields in other messages.
- V03-014 LY0378 Larry Yetto 26-MAY-1983 15:31:30
Add some new fields to journaling write response message
- V03-012 DWT0102 David W. Thiel 25-May-1983
Add message definitions sequencing failover steps.
- V03-011 LY0371 Larry Yetto 23-MAY-1983 19:01:23
Modify common journaling message formats. Remove journaling specific function dispatch codes. Remove CLSMMSG\$W_FUNC and leave just the \$B_FUNC and \$B_FACILITY symbols
- V03-010 DWT0099 David W. Thiel 5-May-1983
Make additions to connection manager message formats.
- V03-009 SRB0078 Steve Beckhardt 15-Apr-1983
Revised LKMSG definitions.
- V03-008 DWT0091 David W. Thiel 9-APR-1983
Add REMOVED as a disconnect/reject status code. Make all disconnect/reject codes even. Adjust and redefine reconnect data. Add CLSMMSG\$K_FAC_ACK facility code.
- V03-007 DWT0089 David W. Thiel 29-MAR-1983
Add reconnect data to \$CNCTDEF. Define some connection manager messages.

Modify \$CLSMGDEF to make two-level dispatch
standard.

V03-006 DWT0081 David W. Thiel 3-MAR-1983
Correct \$CLMDRS.

V03-005 ROW0163 Ralph O. Weber 25-FEB-1983
Change journal name size from 15 to 12 characters.

V03-004 DWT0076 David W. Thiel 11-FEB-1983
Add \$CLMDRS to define cluster disconnect and reject
status values. Rearrange \$CNCTDEF.

V03-003 ROW0160 Ralph O. Weber 26-JAN-1983
Add message codes for cluster journaling access, deaccess,
write, and delete UCB's functions. Add journal-entry-in-
message extension to cluster journal message format. Change
CLMSGSL_STATUS to CLMSGSQ_STATUS.

V03-002 ROW0155 Ralph O. Weber 9-JAN-1983
Add a return status overlay to journaling message definitions.
Add specific fields for create journal message. Correct
JNLTP to be a byte field.


```

module $CLSMGDEF;
/*+
/* CLMSG - CLUSTER MESSAGE
/*
/* THIS DEFINES THE FORMAT OF THE CLUSTER MESSAGE HEADER
/*-

aggregate CLMSGSTRUCT structure prefix CLMSG$;

/*
/* This union is used to compute the size of the largest sequenced
/* message buffer that is needed for ACKMSG. Any modifications or
/* alternate uses of these structures (for example, to define
/* block transfer messages) must take this into account.
/*
CLMSGUNION union;

CLMSGDEF structure prefix CLMSG$;
  SEQNUM word unsigned; /* MESSAGE SEQUENCE NUMBER
  ACKSEQ word unsigned; /* ACKNOWLEDGE SEQUENCE NUMBER
  RSPID longword unsigned; /* RESPONSE ID
  FACILITY byte unsigned; /* FACILITY CODE
  constant ( /* FACILITY CODE VALUES
    FAC_CNX, /* CONNECTION MANAGER FACILITY
    FAC_LCK, /* LOCK MANAGER FACILITY
    FAC_CJF, /* COMMON JOURNALING FACILITY
    FAC_ACK, /* ACKNOWLEDGED MESSAGE SERVICE
    FAC_LKI, /* GETLKI SYSTEM SERVICE
    FAC_CSP, /* CSP (CLUSTER SERVER PROCESS) FACILITY
    FAC_BLK, /* BLOCK TRANSFER SERVICE
    FAC_TST /* TESTING / PERFORMANCE MEASUREMENT FACILITY
  ) equals 1 increment 1;
  constant RESPMSG equals $X80 tag M; /* RESPONSE MESSAGE FLAG
  FUNC byte unsigned; /* FACILITY SPECIFIC FUNCTION
  FILL_1 word fill; /* SPARE
  constant "LENGTH" equals . prefix CLMHDR$; /* STANDARD MESSAGE HEADER LENGTH
  constant "LENGTH" equals .;
  REQR_BUFH longword dimension 3; /* Requestor's buffer handle.
  constant BT_LENGTH equals . prefix CLMHDR$; /* Message w/ block transfer
  constant BT_LENGTH equals .; /* header length
end CLMSGDEF;

/* Block transfer message definition */
CLMBLKDEF structure prefix CLMBLK$;
  constant ( /* FACILITY SPECIFIC MESSAGE CODES
    FNC_RETRY /* REQUEST REISSUE OF REQUESTOR REQUEST
  ) equals 1 increment 1;
  FILL_1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
  RSPID longword; /* RSPID OF REQUEST TO REISSUE
end CLMBLKDEF;

/* Standard Connection Manager Message Sub-header */
CLMCNXDEF structure prefix CLMCNX$;
  constant ( /* FACILITY SPECIFIC MESSAGE CODES
    FNC_STATUS, /* STATUS MESSAGE
    FNC_ENTER, /* REQUEST CLUSTER MEMBERSHIP

```



```

FNC_LOCK,          /* LOCK REQUEST MESSAGE
FNC_UNLOCK,        /* UNLOCK REQUEST MESSAGE
FNC_DESC,          /* NODE DESCRIPTION MESSAGE
FNC_VEC,           /* VECTOR SLOT DESCRIPTION MESSAGE
FNC_FORM,          /* NEW CLUSTER PROPOSAL (PH 1) MESSAGE
FNC_RECONFIG,      /* CLUSTER RECONFIGURATION PROPOSAL (PH 1) MESSAGE
FNC_JOIN,          /* ADD NODE PROPOSAL (PH 1) MESSAGE
FNC_PH2,           /* PHASE 2 MESSAGE
FNC_READY,         /* NODE IS READY FOR FAILOVER STEP MESSAGE
FNC_DOSTEP,        /* DO FAILOVER STEP MESSAGE
FNC_QUORUM,        /* QUORUM CHANGE (PH 1) MESSAGE
FNC_TRNSTS,        /* TRANSITION STATUS REQUEST MESSAGE
FNC_TOPOLOGY       /* TOPOLOGY EXCHANGE MESSAGE
) equals 1 increment 1;
FILL 1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
XTN_ID longword unsigned; /* TRANSACTION NUMBER
XTN_PHASE byte unsigned; /* TRANSACTION PHASE
constant (
  IDLE,            /* IDLE - NO TRANSACTION
  LOCK,            /* LOCK PHASE
  DATA,           /* DATA EXCHANGE PHASE
  PH1,             /* PHASE 1
  UNLOCK,          /* UNLOCK PHASE
  PH2              /* PHASE 2
) equals 16 increment 16;
XTN_CODE byte unsigned; /* TRANSACTION IDENTIFIER
constant ( /* TRANSACTION IDENTIFIER CODES:
  XTN_FORM,        /* FORM CLUSTER
  XTN_JOIN,        /* ADD NODE TO CLUSTER
  XTN_RECONFIG,    /* RECONFIGURE CLUSTER
  XTN_QUORUM       /* QUORUM CHANGE
) equals 1 increment 1;
ACK byte unsigned; /* SUCCESS/FAILURE FLAG
REPLY structure byte unsigned; /* REPLY CODE
RP_TRNSTS_CMT bitfield mask; /* COMMIT RESPONSE FLAG FOR TRNSTS REQUEST
constant ( /* TRNSTS RESPONSE CODES
  RP_TRNSTS_PH0,   /* TRANSACTION IN OR PRECEDES PHASE 0
  RP_TRNSTS_PH1B,  /* TRANSACTION IN PHASE 1, COORD CNX BROKEN
  RP_TRNSTS_PH1,   /* TRANSACTION IN PHASE 1, COORD CNX OK
  RP_TRNSTS_PH2    /* TRANSACTION HAS BEEN COMMITTED
) equals 1 increment 1;
end REPLY;
constant HEADER equals .; /* STANDARD MESSAGE SUB-HEADER LENGTH
end CLMCNXDEF;

/* Connection Manager Coordination Lock Request Message */
CLMLCKDEF structure prefix CLMLCK$:
  FILL 1 byte dimension CLMCNX$K_HEADER; /* SPACE FOR HEAD AND SUB_HEADER
  XTN_TIME quadword; /* TIME-STAMP FOR TRANSACTION
  constant "LENGTH" equals .; /* MESSAGE LENGTH
end CLMLCKDEF;

/* Connection Manager Node Description Message */
CLMNODDEF structure prefix CLMNOD$:
  FILL 1 byte dimension CLMCNX$K_HEADER; /* SPACE FOR HEAD AND SUB_HEADER
  SYSTEMID byte dimension (6); /* SYSTEM ID

```



```

    FILL 2 word fill;          /* SPARE WORD TO ALIGN DATA
    SWINCARN quadword;         /* SOFTWARE INCARNATION NUMBER
    CSID longword unsigned;    /* TENTATIVE CSID
    constant "LENGTH" equals .; /* MESSAGE LENGTH
end CLMNODDEF;

/* Connection Manager Cluster Proposal Message */
/* General Phase 1 message */
CLMPRODEF structure prefix CLMPROS;
    FILL 1 byte dimension CLMCNXX$K_HEADER; /* SPACE FOR HEAD AND SUB_HEADER
    NEXT CSID word unsigned;                /* CSID ASSIGNMENT CONTEXT
    QUORUM word unsigned;                   /* CLUSTER QUORUM
    MEMSEQ word unsigned;                   /* MEMBERSHIP TRANSITION SEQUENCE NUMBER
    QDVOTES word unsigned;                  /* NUMBER OF VOTES ASSIGNED TO QUORUM DISK
    FMERIT longword unsigned;               /* FIGURE OF MERIT OF PROPOSED CLUSTER
    FTIME quadword;                        /* CLUSTER FORMATION TIMESTAMP
    CURTIME quadword;                      /* CURRENT TIME
    FLAGS structure byte unsigned;          /* CLUSTER STATUS BITS
    QF VOTE bitfield mask;                  /* QUORUM FILE IS A CLUSTER MEMBER
end FLAGS;
    FSYSID byte dimension (6);              /* FOUNDING NODE SYSTEMID
    NODEMAP byte dimension (32);            /* BITMAP OF NODES
    constant "LENGTH" equals .;            /* MESSAGE LENGTH
end CLMPRODEF;

/* Connection Manager Cluster Vector Entry Description */
CLMVECDEF structure prefix CLMVECS;
    FILL 1 byte dimension CLMCNXX$K_HEADER; /* SPACE FOR HEAD AND SUB_HEADER
    INDEX word unsigned;                    /* INDEX OF DESCRIBED ENTRY
    SEQUENCE word unsigned;                  /* LAST SEQUENCE NUMBER
    constant "LENGTH" equals .;            /* MESSAGE LENGTH
end CLMVECDEF;

/* Connection Manager Topology Exchange Message */
CLMTOPDEF structure prefix CLMTOPS;
    FILL 1 byte dimension CLMCNXX$K_HEADER; /* SPACE FOR HEAD AND SUB_HEADER
    NODEMAP byte dimension (32);             /* CONNECTIVITY BITMAP
    constant "LENGTH" equals .;             /* MESSAGE LENGTH
end CLMTOPDEF;

/* Connection Manager Status Message/Reply */
CLMSTSDEF structure prefix CLMSTSS;
    FILL 1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
    FLAGS structure byte unsigned;          /* CLUSTER STATUS BITS
    CLUSTER bitfield mask;                  /* NODE IS CLUSTER MEMBER
    QF ACTIVE bitfield mask;                /* QUORUM FILE READABLE -- STATIC QUORUM
    SHUTDOWN bitfield mask;                 /* CLUSTER SHUTDOWN IN PROGRESS
end FLAGS;
    FILL 2 byte fill;                      /* SPARE TO ALIGN FIELDS
    CQUORUM word unsigned;                  /* CLUSTER QUORUM
    CVOTES word unsigned;                   /* CLUSTER VOTES
    NODES word unsigned;                    /* NODES IN CLUSTER
    NQUORUM word unsigned;                  /* NODE QUORUM SETTING
    NVOTES word unsigned;                   /* NODE'S VOTES
    QDVOTES word unsigned;                  /* NUMBER OF VOTES HELD BY QUORUM DISK
    LCKDIRWT word unsigned;                 /* LOCK MANAGER DISTRIBUTED DIRECTORY WEIGHT

```



```

FTIME quadword;          /* TIME OF FOUNDING
LST_TIME quadword;       /* TIME-STAMP OF LAST COMPLETED TRANSACTION
MAX_XTN longword unsigned; /* LARGEST TRANSACTION ID SEEN
QDISK byte dimension(16); /* QUORUM DISK NAME
REFTIME quadword;       /* REFERENCE TIME FOR NODE
constant 'LENGTH' equals .; /* MESSAGE LENGTH
end CLMSTSDEF;

```

```

/* Failover sequencing message definition */

```

```

CLMSTPDEF structure prefix CLMSTP$;
  FILL_1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
  ID longword unsigned; /* FAILOVER IDENTIFICATION
  STEP longword unsigned; /* FAILOVER STEP INDEX
  constant 'LENGTH' equals .; /* MESSAGE LENGTH
end CLMSTPDEF;

```

```

/*
/* Lock manager message definitions
/*

```

```

LKMSGDEF structure prefix LKMSG$;

```

```

  constant (
    NEWLOCK, /* FACILITY SPECIFIC MESSAGE CODES
    GRANTED, /* NEW LOCK REQUEST
    DEQ, /* LOCK GRANTED MESSAGE
    RMVDIR, /* DEQUEUE MESSAGE
    BLKING, /* REMOVE DIR. ENTRY MESSAGE
    CVTLCKM, /* BLOCKING MESSAGE
    CVTLCKR, /* CONVERT LOCK MESSAGE
    REBLDLCK, /* CONVERT LOCK REQUEST
    TSRQST, /* REBUILD LOCK (FAILOVER)
    SRCHDLCK, /* TIMESTAMP REQUEST
    DLCKFND, /* SEARCH FOR DEADLOCK
    REDO_SRCH, /* DEADLOCK FOUND
    ) equals 1 increment 1; /* REDO DEADLOCK SEARCH

```

```

EXTENSION_2 union;

```

```

  LOCK_MSGS structure;

```

```

    FILL_1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
    MEMSEQ word unsigned; /* MEMBERSHIP SEQUENCE NUMBER
    HASHVAL word unsigned; /* RESOURCE HASH VALUE
    MSTLKID_OVERLAY union fill;
      EPIDNEW longword unsigned; /* EPID FOR NEW LOCK REQUESTS
      MSTLKID longword unsigned; /* MASTER LOCK ID
    end MSTLKID_OVERLAY;
    PRCLKID longword unsigned; /* PROCESS LOCK ID

```

```

  EXTENSION union;

```

```

  /* New lock and rebuild lock message extension

```



```

NEWLOCK structure;
  FLAGS word unsigned;          /* FLAGS
  RQMODE byte unsigned;         /* REQUESTED MODE
  GRMODE byte;                  /* GRANTED MODE
  BLKASTFLG longword unsigned;  /* BLOCKING AST FLAG
  PARPRCLKID longword unsigned; /* PARENT PROCESS LOCK ID
  PARMSTLKID longword unsigned; /* PARENT MASTER LOCK ID
  GROUP word unsigned;          /* GROUP NUMBER
  RMOD byte unsigned;           /* REQUEST ACCESS MODE
  RSNLEN byte unsigned;         /* RESOURCE NAME LENGTH
  RESNAM character length 32 tag T; /* RESOURCE NAME
  DLCKPRI_NEW longword unsigned; /* DEADLOCK PRIORITY FOR NEW LOCKS

/* Extension for rebuild lock message

  RQSEQALT word unsigned;       /* ALTERNATE REQUEST SEQUENCE NUMBER
  FILL 7 word fill;             /* SPARE
  VALBKALT quadword unsigned;   /* ALTERNATE VALUE BLOCK
  FILL 2 quadword fill;         /* MORE VALUE BLOCK
  VALSEQALT longword unsigned;  /* ALTERNATE VALUE BLOCK SEQ. NUMBER
  LSTATUS byte unsigned;        /* LKB STATUS (BYTE FORM)
  RSTATUS byte unsigned;        /* RSB STATUS (BYTE FORM)
  LCKSTATE byte unsigned;       /* LOCK STATE
end NEWLOCK;

/* Resend and not queued response to new lock message

RESEND structure;
  CSID longword unsigned;       /* CSID OF SYSTEM TO RESEND TO
  STATUS word unsigned;         /* RETURN STATUS
  STATE byte unsigned;          /* RESPONSE STATE
end RESEND;

/* Lock queued response to new lock message

LOCKQED structure;
  LKBSTATUS word unsigned;       /* LKB status
  RSBSTATUS word unsigned;       /* RSB status
  FILL 3 longword fill;         /* STATUS AND STATE
  VALBCK quadword unsigned;     /* VALUE BLOCK
  FILL 5 quadword fill;         /* MORE VALUE BLOCK
  SEQNUM OVERLAY union fill;
    RQSEQNM word unsigned;       /* REQUEST SEQUENCE NUMBER
    VALSEQNUM longword unsigned; /* VALUE BLOCK SEQUENCE NUMBER
    EPIDCVT longword unsigned;   /* EPID FOR CONVERSIONS
  end SEQNUM OVERLAY;
  DLCKPRI_CVT longword unsigned; /* DEADLOCK PRIORITY FOR CONVERSIONS
end LOCKQED;

/* Dequeue message

DEQ structure;
  FILL 6 longword fill;         /* FLAGS and MODES
  VALBCKFLG longword unsigned;  /* VALUE BLOCK FLAG
end DEQ;

```


end EXTENSION;

end LOCK_MSGS;

DLCK_MSGS structure;

```

FILL_1 byte dimension CLMHDR$K_LENGTH; /* SPACE FOR HEADER
FILL_2 word fill; /* SPACE FOR MEMSEQ
TSLT byte unsigned; /* TIMESTAMP LIFETIME
FILL_3 byte fill; /* SPARE
ORIGEPID longword unsigned; /* ORIGINAL EPID
ORIGLKID longword unsigned; /* ORIGINAL LOCKID
ORIGCSID longword unsigned; /* ORIGINAL CSID
BITMAP_EXP quadword unsigned; /* BITMAP EXPIRATION TIMESTAMP
VCTMPRI longword unsigned; /* VICTIM PRIORITY
VCTMLKID longword unsigned; /* VICTIM LOCKID
VCTMCSID longword unsigned; /* VICTIM CSID
NEXTLKID longword unsigned; /* NEXT LOCKID TO SEARCH FOR
constant DLM_LENGTH equals .; /* LENGTH OF DEADLOCK MESSAGE
end DLCK_MSGS;
```

end EXTENSION_2;

end LKMSGDEF;

```

/*
/* Common journaling message definitions
/*
```

CJMSGDEF structure prefix CJMSG\$;

```

constant (
MSGCREJNL /* Create journal UCB in message
,BLKCREJNL /* Create journal block transfer the UCB
,DELJNL /* Delete journal message sent to master
,DELUCB /* Delete journal UCB
,ACCJNL /* Access journal
,DEACCJNL /* Deaccess journal
,MSGWRITE /* Write from message buffer
,BLKWRITE /* Block transfer over CI
/* and then write
,MSGREADINI /* Read init cplx buff in msg
,BLKREADINI /* Read init block xfer cplx buff
,MSGREADCPLX /* Read complex buff xfer in msg
,BLKREADCPLX /* Read block xfer complex buff
,MSGREADDATA /* Xfer read data to slave in msg
,BLKREADDATA /* Block xfer read data to slave
,CANCEL /* Cancel IO for channel
,FORCE /* Force IO for channel
,FLUSH /* Flush all writes for journal
,BCSTDELJNL /* DELJNL message broadcast to all nodes
,ACKWRITE /* ACK a write that is in a CWQ queue
,GMINFO /* Get-Master-Info from slave
,INQWRBUF /* Inquire-Write-Buffer from node
,RESUBRC /* Resubmit read-context for re-readinit
,RESUBWR /* Resubmit a given write
```



```

,WRTFOVRCPL      /* Write-Fail-Over-Complete
,GETPART         /* Get part of entry kept in buffer
,RUEBIT          /* Set/Clear bit in RUE
,BCSTDELPND      /* Delete pending message broadcast to all nodes
,REQINIT         /* Request beginning of init node function
,BLKINIT         /* Block Xfer init data to the new node
,BLKJNLRC        /* Block transfer JNLRC block
{ /* Add new codes here !
,HIGHBOUND) equals 1 increment 1;

```

JOURNAL_MESSAGES union fill;

/* Return status journal message

STATUS_MSG structure fill;

FILL_1 byte dimension CLMHDR\$K_LENGTH fill; /* SPACE FOR HEADER

WRT_JNL_STAT union fill;

STATUS quadword unsigned; /* Return status

SEQNO OVERLAY structure fill;

SEQNOFILL longword fill;

SEQNO longword unsigned; /* Return entry sequence number

end SEQNO_OVERLAY;

end WRT_JNL_STAT;

STAT_VAL4 longword unsigned; /* Extra longword of misc data

STAT_VAL5 longword unsigned; /* Extra longword of misc data

STAT_VAL6 longword unsigned; /* Extra longword of misc data

STAT_VAL7 longword unsigned; /* Extra longword of misc data

STAT_VAL8 longword unsigned; /* Extra longword of misc data

constant 'STATUS_LENGTH' equals . ; /* Status message length

end STATUS_MSG;

SEND_MSG structure fill;

FILL_1 byte dimension CLMHDR\$K_BT_LENGTH fill; /* SPACE FOR HEADER

MCSID OVERLAY union fill;

MCSID longword unsigned; /* Master node CSID

MCSID_SUBF structure fill;

MCSID_SEQ word unsigned; /* Master CSID sequence number

MCSID_IDX word unsigned; /* Master CSID node index

end MCSID_SUBF;

end MCSID_OVERLAY;

MUNIT word unsigned; /* Master node unit number

JNL_TYP byte unsigned; /* Journal type

FILL_2 byte fill;

JNL_EXTENSIONS union fill;

/* Generic function message extension

FNCT_MSG structure fill;

VAL1 longword unsigned; /* Extra longword of misc data

VAL2 longword unsigned; /* Function code + word misc data

VAL3 longword unsigned; /* Sender's UCB address

VAL4 longword unsigned; /* Extra longword of misc data

VAL5 longword unsigned; /* Extra longword of misc data

VAL6 longword unsigned; /* Extra longword of misc data

VAL7 longword unsigned; /* Extra longword of misc data

VAL8 longword unsigned; /* Extra longword of misc data

constant 'FNCT_LENGTH' equals . ; /* FNCT message length


```

end FNCT_MSG ;

/* Create journal message extension
CREATE MSG structure fill;
  RMBLK_LENGTH longword unsigned; /* Length of data for MSGCREATE
  CR_BTSEQNO longword unsigned; /* Block transfer seq no
  CR_EPID longword unsigned; /* Creator's EPID
  CR_CHAN word unsigned; /* Channel index
  CR_RMOD byte unsigned; /* requestors access mode
  RMBLK character length 49; /* Beginning of remaster block for MSGCREATE
  constant 'CREATE_LENGTH' equals . ; /* Create message length
end CREATE_MSG;

/* Write message extension for message sent to the master
WRITE JNL_SND structure fill;
  IOFUNC word unsigned; /* Original I/O function
  BYTCNT word unsigned; /* Count of bytes in message
  RUID octaword unsigned; /* Recovery unit ID.
  WRUFLAGS longword unsigned; /* Write RU flags.
  LSEQNO longword unsigned; /* Local sequence number
  WRMASK longword unsigned; /* Write mask
  ASID longword unsigned; /* Assign ID for the channel
  WRT_PRIV quadword unsigned; /* ARB priv mask
  FACCOD word unsigned; /* Channel facility code
  IOSTS byte unsigned; /* I/O status (used only for writes)
  WRATR byte unsigned; /* Write attributes
  WRT_EPID longword unsigned; /* Writer's EPID
  WRT_BTSEQNO longword unsigned; /* Block transfer seq num
  WRT_STS word unsigned; /* Status from IRP
  WRT_CHAN word unsigned; /* Channel index
  WRT_RMOD byte unsigned; /* requestors access mode
  MSGBUF character length 3; /* Base of journal entry in a message
  constant 'WRITE_LENGTH' equals .; /* Write message length
end WRITE_JNL_SND;

/* Read and read init message extension
READ MSG structure fill;
  STATUS_OVERLAY union fill ;
    RESP_STRUCT structure fill;
      READ_STATUS quadword unsigned; /* Read init return status (response)
      READ_SEQNO longword unsigned; /* Seq no of last entry
      RUID_OVLY union fill;
        READ_RUID octaword unsigned; /* Recovery unit ID ( RU only )
        READ_DATTIM quadword unsigned; /* Date/time of last entry ( nonRU only )
      end RUID_OVLY;
    end RESP_STRUCT;
    SEND_STRUCT structure fill;
      IOST1 longword unsigned; /* IOST1 from IRP
      IOST2 longword unsigned; /* IOST2 from IRP
      RDFUNC word unsigned; /* Original IO function
      RD_STS word unsigned; /* Status from IRP
      RDACMODE longword unsigned; /* Access mode of caller
      ARB_UIC longword unsigned; /* UIC from reader's ARB
      ARB_PRIV quadword unsigned; /* Privilege mask from reader's ARB
    end SEND_STRUCT;
  end STATUS_OVERLAY;

```



```

NUM_DESC word unsigned;      /* # of descriptors in cplx buff
BUFF_LEN word unsigned;      /* Length of buffer
RD_BTSEQNO longword unsigned; /* Block transfer seq num
RD_IRP longword unsigned;    /* requestor's IRP address
RD_CHAN word unsigned;       /* channel index
RD_CPLXLEN word unsigned;    /* size of complex buffer
RD_RMOD byte unsigned;       /* requestors access mode
CPLX_BUFF character length 19; /* Start of complex buffer
constant 'READ_LENGTH' equals .; /* Read message length
end READ_MSG;

/* Return from Get-Master-Information message
MINF_MSG structure fill;
  MINF_STATUS longword unsigned; /* Status
  MINF_FLAGS_OVERLAY union fill; /* flags
    MINF_FLAGS longword unsigned; /* flags longword
    MINF_BITS structure fill; /* bits defined in flags longword
      MINF_REF bitfield mask; /* reference count > 0
      MINF_DMT bitfield mask; /* marked for dismount
    end MINF_BITS;
  end MINF_FLAGS_OVERLAY;
  constant 'MINF_LENGTH' equals .; /* Get-Master-Info message length
end MINF_MSG;

/* Get-Write-Buffer information
GWRBUF_MSG structure fill;
  GWRBUF_UNIT word unsigned; /* unit # new master
  GWRBUF_SPARE1 word unsigned; /* spare word
  GWRBUF_CSID longword unsigned; /* CSID new master
  GWRBUF_LAST longword unsigned; /* seq # last entry info sent for
  GWRBUF_ADDR longword unsigned; /* address info block if one exists
  constant 'GWRBUF_LENGTH' equals .; /* Get-Write-Buffer info message length
end GWRBUF_MSG;

/* Return Write Buffer message
RWRBUF_MSG structure fill;
  RWRBUF_STATUS longword unsigned; /* status
  RWRBUF_LAST longword unsigned; /* seq # last entry sent here
  RWRBUF_ADDR longword unsigned; /* address of info block on new master
  RWRBUF_BFR longword unsigned; /* start JNLMSG message block
  constant 'RWRBUF_LENGTH' equals .; /* Return Write Buffer info message length
end RWRBUF_MSG;

end JNL_EXTENSIONS;
end SEND_MSG;
end JOURNAL_MESSAGES;
end CJMSGDEF;

/*
/* GETLKI message definitions
/*
LIMSGDEF structure prefix LIMSG$;
  constant ( /* FACILITY SPECIFIC MESSAGE CODES

```



```

STDINFO,          /* GET STANDARD LOCK INFORMATION
BLKING,           /* REQUEST LIST OF BLOCKING LOCKS
BLKBY,            /* REQUEST LIST OF BLOCKED BY LOCKS
LOCKS             /* REQUEST LIST OF LOCKS ON RESOURCE
) equals 1 increment 1;

```

```

FILL_1 byte dimension CLMHDR$K_BT_LENGTH; /* SPACE FOR HEADER

```

```

MSTLKID longword unsigned; /* MASTER LOCK ID
PRCLKID longword unsigned; /* PROCESS LOCK ID
STATUS byte unsigned;      /* RETURN STATUS
    constant (
        RSPSUCCESS,        /* SUCCESS
        RSPIVLKID,         /* INVALID LOCKID
        RSPMORE             /* MORE DATA TO COME
    ) equals 1 increment 1;
FILL_2 byte unsigned;      /* UNUSED BYTE

```

```

EXTENSION union;

```

```

/* Standard lock information message

```

```

STDINFO structure;
    FILL_3 word unsigned; /* UNUSED WORD
    STATE longword unsigned; /* LOCK STATE
    RSBREFCNT longword unsigned; /* SUB-RESOURCE REFERENCE COUNT
    LCKCOUNT longword unsigned; /* COUNT OF LOCKS ON RESOURCE
    VALBLK quadword unsigned; /* RESOURCE VALUE BLOCK
    ALTVALBLK quadword fill; /* MORE VALUE BLOCK
    constant "STDINFO_LEN" equals .; /* Standard info message length

```

```

end STDINFO;

```

```

/* All list return messages

```

```

LIST structure;
    LISTSIZE word unsigned; /* NUMBER OF BYTES IN LIST
end LIST;

```

```

end EXTENSION;

```

```

end LIMSGDEF;

```

```

end CLSMMSGUNION;
constant MAXMSG equals .; /* SIZE OF LARGEST CLUSTER MESSAGE
end CLSMMSGSTRUCT;

```

```

END_MODULE $CLSMMSGDEF;

```

```

/*
/*
/*

```

```

/*
/*
/*

```

```

/*
/*
/*

```

```

/*
/*
/*

```



```
module $CLMDRSDEF;
```

```
/**
/* CLMDRS - CLUSTER DISCONNECT/REJECT STATUS FORMAT
/*
/* THIS DEFINES THE CLUSTER DISCONNECT/REJECT STATUS CODES
/*-
```

```
aggregate CLMDRSDEF structure prefix CLMDRSS;
```

```
CODE byte unsigned; /* DISCONNECT/REJECT REASON CODE
constant ( /* DISCONNECT/REJECT CODES:
    RESOURCE, /* RESOURCE ERROR
    PROTOCOL, /* PROTOCOL ERROR
    VERSION, /* INCOMPATIBLE PROTOCOL VERSION
    BUSY, /* INCOMPATIBLE ACTIVITY IN PROGRESS
    REMOVED /* NODE HAS BEEN REMOVED FROM CLUSTER
) equals 2 increment 2 tag C;
FATAL bitfield length 1 MASK; /* RECIPIENT SHOULD BUGCHECK
CLUSFTL bitfield length 1 MASK; /* RECIPIENT CLUSTER SHOULD BUGCHECK
DEADCNX bitfield length 1 MASK; /* CONNECTION IS DEAD
LONG_BREAK bitfield length 1 MASK; /* DECLARE LONG BREAK
FILL_1 bitfield length 3 fill; /* PAD TO WORD BOUNDARY
DRS Bitfield length 1 MASK; /* MARK THIS AS A CLMDRS code
constant "LENGTH" equals . tag C; /* LENGTH
constant "LENGTH" equals . tag K; /* LENGTH
end CLMDRSDEF;
```

```
END_MODULE $CLMDRSDEF;
```



```
module $CNCTDEF;
/**
/* CNCT - CONNECT MESSAGE FORMAT
/*
/* THIS DEFINES THE FORMAT OF THE CONNECT MESSAGE
/*-

aggregate CNCTDEF structure prefix CNCT$:
    ECOLVL byte unsigned; /* PROTOCOL ECO LEVEL
    VERNUM byte unsigned; /* PROTOCOL VERSION NUMBER
    constant PROTOCOL equals 12 tag K; /* PROTOCOL LEVEL
    TYPE byte unsigned; /* CONNECT TYP
    constant INITIAL equals 1; /* INITIAL CONNECTION
    constant RECONNECT equals 2; /* REMAKE BROKEN CONNECTION
    ACKLIM byte unsigned; /* ACK LIMIT
    RECONNECT DATA structure fill; /* RECONNECT/REACCEPT DATA
        QUORUM word unsigned; /* CLUSTER QUORUM
        VOTES word unsigned; /* CLUSTER VOTES
        NODES word unsigned; /* NODES IN CLUSTER
        CLSSTS structure byte unsigned; /* CLUSTER STATUS FLAGS
            CLUSTER bitfield mask; /* SENDER NODE IS CLUSTER MEMBER
        end CLSSTS;
        CNXSTS structure byte unsigned; /* CONNECTION STATUS FLAGS
            LONG BREAK bitfield mask; /* LONG BREAK IN CONNECTION
            MEMBER bitfield mask; /* RECEIVER IS MEMBER OF SENDING CLUSTER
            REMOVED bitfield mask; /* RECEIVER REMOVED FROM SENDING CLUSTER
        end CNXSTS;
    end RECONNECT DATA;
    RCVDSQNM word unsigned; /* LAST SEQUENCE NUMBER RECEIVED
    FILL_3 word fill; /* SPARE
    constant 'LENGTH' equals . tag K;
end CNCTDEF;

END_MODULE $CNCTDEF;
```



```
module $CLUBTXDEF;
```

```
/*+
/* CLUBTX - CLUSTER BLOCK TRANSFER CDRP EXTENSION
/*
/* This defines the format of a block transfer extension to the CDRP used
/* by the cluster acknowledged message services. Such CDRPs have a
/* CDRP$L_LBUFH_AD (or CDRP$L_VAL1) which points to CLUBTX$L_LBUFHNDL.
/* On such nodes, CDRP$L_LBUFH_AD points to the buffer handle within
/* the BTX.
/*-
```

```
aggregate CLUBTXDEF structure prefix CLUBTX$;
```

XQFL longword unsigned;	/* A queue forward link
XQBL longword unsigned;	/* A queue backward link
SIZE word unsigned;	/* Structure size
TYPE byte unsigned;	/* Structure type
SUBTYPE byte unsigned;	/* Structure subtype
LBUFHNDL longword unsigned dimension 3;	/* Local buffer handle
CDRP longword unsigned;	/* Address of owning CDRP
CSID longword unsigned;	/* CSID of requestor
ERRADDR longword unsigned;	/* Error action routine address
USER_BUF longword unsigned;	/* Address of user pool space base
SAVED_PC longword unsigned;	/* Saved caller's PC
MSGBLD longword unsigned;	/* Auxiliary message build routine

```
/* End of requestor BTX
```

```
/*
/* Note: although the requestor BTX is much shorter, it is still allocated
/* based upon the base size of the (longer) partner BTX. Since a SRP is
/* allocated in either case, this is unimportant.
```

```
{ This ends the defined BTX. Following the defined area is a copy of
{ the incoming message buffer. CLUBTX$T_MSG_BUF is the offset from the
{ BTX base to the beginning of the copied message.
```

```
constant "LENGTH" equals .;
```

```
MSG_BUF character varying;          /* Base of copied message buffer
```

```
end CLUBTXDEF;
```

```
end_module $CLUBTXDEF;
```



```
module $CLUQFDEF;
```

```
/*+
/* CLUQF - Cluster Quorum File offset definitions
/*
/* This module defines the format of the data in the cluster quorum file.
/* The quorum file consists of the owner and activity blocks. The owner
/* block contains information obtained from a node in the cluster currently
/* using the quorum file. The activity block contains a counter which is
/* incremented by members of the cluster currently using the quorm file.
/*-
```

```
aggregate CLUQF structure prefix CLUQF$;
```

```
/* Start of owner block
```

```
IDENT character length 12;
VERSION word unsigned;
FLAGS UNION union fill;
    FLAGS word unsigned;
    FLAG BITS structure fill;
        QUORUM bitfield mask;
    end FLAG BITS;
end FLAGS UNION;
```

```
FOU_TIME quadword unsigned;
LST_TIME quadword unsigned;
QF_TIME quadword unsigned;
SWINCARN quadword unsigned;
CSID UNION union fill;
    CSID longword unsigned;
    CSID_FIELDS structure fill;
        CSID_IDX word unsigned;
        CSID_SEQ word unsigned;
    end CSID_FIELDS;
end CSID UNION;
```

```
QUORUM word unsigned;
VOTES word unsigned;
SYSID byte unsigned dimension 6;
FSYSID byte unsigned dimension 6;
```

```
/*
/* Note that CLUQF$L_CHECKSUM must be longword aligned
/*
```

```
CHECKSUM longword unsigned;
```

```
constant CHECK_LENGTH equals . tag C;
constant CHECK_LENGTH equals . tag K;
```

```
IGNORE byte unsigned;
```

```
/* End of owner block
```

```
constant OWNER_LENGTH equals . tag C;
constant OWNER_LENGTH equals . tag K;
```

```
SPARE byte dimension 512-..;
```

```
/* Quorum block identification area
/* Quorum block version number
```

```
/* Flags word
```

```
/* Cluster has dynamic quorum
```

```
/* Founding time
```

```
/* Last completed transaction time-stamp
```

```
/* Quorum block time-stamp
```

```
/* Software incarnation number
```

```
/* Cluster system ID
```

```
/* Slot index
```

```
/* Sequence number
```

```
/* Cluster quorum
```

```
/* Cluster votes
```

```
/* System ID
```

```
/* Founding nodes SYSID
```

```
/* Quorum block checksum
```

```
/* Length of the quorum block fields
/* that contribute to the checksum
```

```
/* If non-zero, data in quorum
/* file should be ignored
```

```
/* Quorum file owner block length
```

```
/* Quorum file owner block length
```

```
/* Start activity block on next block boundary
```



```
/* Start of activity block
```

```
ACT_COUNT longword unsigned;
```

```
/* End of activity block
```

```
constant ACT_LENGTH equals .-512 tag C;  
constant ACT_LENGTH equals .-512 tag K;
```

```
constant "LENGTH" equals . tag C;  
constant "LENGTH" equals . tag K;
```

```
constant BLOCKS equals 2 tag C;  
constant BLOCKS equals 2 tag K;
```

```
constant VERSION equals 2 tag C;  
constant VERSION equals 2 tag K;
```

```
/* Activity counter
```

```
/* Quorum file activity block length  
/* Quorum file activity block length
```

```
/* Quorum file length  
/* Quorum file length
```

```
/* Number of blocks in quorum file  
/* Number of blocks in quorum file
```

```
/* Quorum block version number  
/* Quorum block version number
```

```
end CLUQF;
```

```
end_module $CLUQFDEF;
```


0391 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

