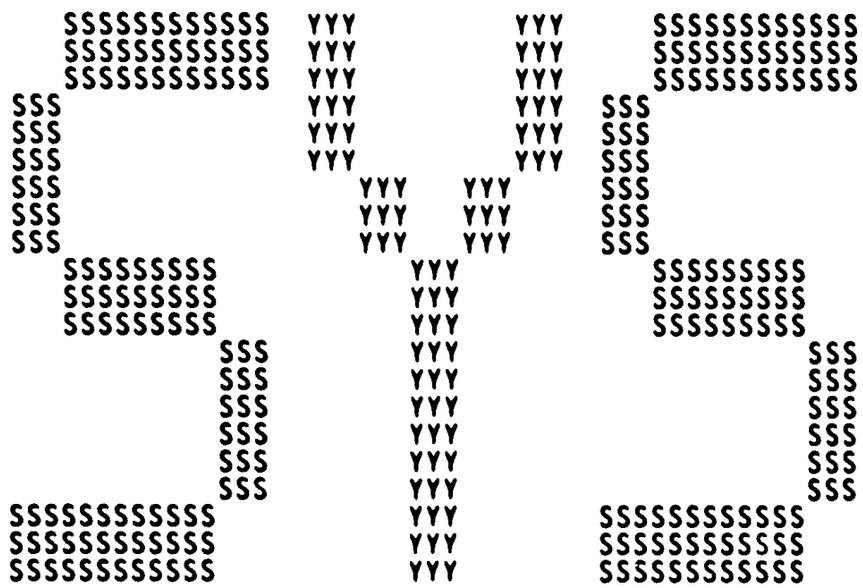

**FILE**ID**SYSDEFAE

```
  SSSSSSSS  YY      YY    SSSSSSSS  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    AAAAAA    EEEEEEEEEE
  SSSSSSSS  YY      YY    SSSSSSSS  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    AAAAAA    EEEEEEEEEE
SS          YY      YY  SS          DD      DD  EE          FF          AA      AA  EE
SS          YY      YY  SS          DD      DD  EE          FF          AA      AA  EE
SS            YY  YY    SS          DD      DD  EE          FF          AA      AA  EE
SS            YY  YY    SS          DD      DD  EE          FF          AA      AA  EE
  SSSSSS        YY        SSSSSS    DD      DD  EEEEEEEE    FFFFFFFF    AA      AA  EEEEEEEE
  SSSSSS        YY        SSSSSS    DD      DD  EEEEEEEE    FFFFFFFF    AA      AA  EEEEEEEE
        SS      YY              SS  DD      DD  EE          FF          AAAAAAAAAA  EE
        SS      YY              SS  DD      DD  EE          FF          AAAAAAAAAA  EE
        SS      YY              SS  DD      DD  EE          FF          AA      AA  EE              ....
        SS      YY              SS  DD      DD  EE          FF          AA      AA  EE              ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD    EEEEEEEEEE  FF          AA      AA  EEEEEEEEEE      ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD    EEEEEEEEEE  FF          AA      AA  EEEEEEEEEE      ....

  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:      'V04-002'
{
{*****************************************************************************
{*                                                                           *
{*  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:  VAX/VMS System Macro Libraries
{
{ ABSTRACT:
{
{       This file contains the SDL source for all operating system control
{       blocks, from A to E.  That is, all control blocks from AAA to EZZ.
{
{ ENVIRONMENT:
{
{       n/a
{
{--
{
{
{ AUTHOR:  The VMS Group          CREATION DATE:  1-Aug-1976
{
{ MODIFIED BY:
{
{       V04-002 ACG0467         Andrew C. Goldstein,    12-Sep-1984  17:28
{               Add separate read and write protection check bits to CCB
{
{       V04-001 SRB0145         Steve Beckhardt         9-Sep-1984
{               Moved CDRP$L_VAL9 into regular CDRP out of long CDRP.
{
{       V03-156 RLRBIDEFS4      Robert L. Rappaport     21-Aug-1984
{               Once more another update $BUADEF from even newer, newer specs.
{
{       V03-155 RLRBIDEFS3      Robert L. Rappaport     13-Aug-1984
```

```
{               Again update $BUADEF from even newer specs.
{
{       V03-154 RLRBIDEFS2      Robert L. Rappaport     11-Aug-1984
{               Bring $BIICDEF and $BUADEF into line with latest spec.
{
{       V03-153 KDM0106         Kathleen D. Morse       01-Aug-1984
{               Move ICCS back into the cpu-independent registers so that
{               11/730 and MicroVAX I dumps do not lose the bugcheck code,
{               when writing the EMB crash entry.
{
{       V03-152 ROW0396         Ralph O. Weber          22-JUL-1984
{               Add CDRP$V_DENSCK, a class driver flag used to signal that a
{               tape density check is required.
{
{       V03-151 WMC151          Wayne Cardoza           19-Jul-1984
{               New quorum disk flag for write-locked.
{
{       V03-150 WMC150          Wayne Cardoza           10-Jul-1984
{               Fix a typo.
{
{       V03-149 ROW0378         Ralph O. Weber           6-JUL-1984
{               Add DYN$C_CD_SHDW_WRK for the class driver shadow set work
{               buffer.  Add CDDB$V_DAPBYS, a busy flag for the DAP CDRP, and
{               CDDB$V_2PBSY, a busy flag for failover fork block.
{
{       V03-148 WMC148          Wayne Cardoza           27-Jun-1984
{               Add an error bit to CLUDCBDEF.
{
{       V03-147 DWT0221         David W. Thiel          25-Jun-1984
{               Add CLUFCB$V_WAITING bit to CLUFCB sub-block of CLUB.
{
{       V03-146 ROW0356         Ralph O. Weber          1-MAY-1984
{               Add CDDB$V_QUORLOST to CDDB$W_STATUS.  This bit will be used
{               to indicate that disk revalidation is occuring because
{               connection manager quorum has been lost.
{
{       V03-145 KTA3124         Kerbey T. Altmann       11-Apr-1984
{               Add new DYN codes.
{
{       V03-144 LMP0221         L. Mark Pilant,         7-Apr-1984  12:59
{               Remove the last vestiges of the old CHKPRO interface.  The
{               definition of the CHIP block.
{
{       V03-143 ROW0336         Ralph O. Weber          7-APR-1984
{               Remove CDDB$L_CONNQLF/BJL.  Add two more reserved longwords to
{               the CDDB definition.  Add CDRP$V_IVCMD to CDRP$L_DUTUFLAGS.
{
{       V03-142 KPL0001         Peter Lieberwirth       6-Apr-1984
{               Add several reserved longwords to the ADP to be used
{               to support volatile BI adapter information.
{
{       V03-141 EMD0073         Ellen M. Dusseault      06-Apr-1984
{               Add new field, CDRP$L_KEYDESC, corresponding to
{               in the IRP field for the address of a descriptor
{               describing an encryption key.
{
```

```
{       V03-140 DWT0209         David W. Thiel          06-Apr-1984
{               Add CLUBPWF substructure to CLUBDEF for use as a
{               fork block during power failures.
{               Add CLUB$V_NO_FORM.  Remove unused fields from
{               CLUB.
{
{       V03-139 RSH0124         R. Scott Hanna          25-Mar-1984
{               Replace $CLUDCBDEF.
{
{       V03-138 SSA0021         Stan Amway              23-Mar-1984
{               Backed out DWT0198, since it is nc longer necessary.
{
{       V03-137 DWT0198         David W. Thiel          22-Mar-1984
{               Length CLUB$S_HANG_FKB to make it look more like
{               an IRP.
{
{       V03-136 SRB0117         Steve Beckhardt         22-Mar-1984
{               Added VAL9 and VAL10 to CDRP definitions.
{
{       V03-135 DWT0196         David W. Thiel          21-Mar-1984
{               Remove CDRP$L_BTX, CDRP$K_PART_RESP, CSB$T_SYSTEMID,
{               CSB$W_WAITCNT, CSB$V_QUORUM, CSB$V_TRANSITION,
{               CSB$V_QF_DYNVOTE, CSB$L_MSGS_SENT, CSB$L_MSGS_RCVD.
{               Add CLUB$V_QF_NEWVOTE, CLUB$Q_NEWQDVOTES.
{
{       V03-134 ACG0408         Andrew C. Goldstein,    21-Mar-1984  11:22
{               Add ARB$C_HEADER
{
{       V03-133 LMP0214         L. Mark Pilant,         21-Mar-1984  10:02
{               Modify $ARBDEF to remove the temporary SDL hack.
{               Add two new structures: $CHPCTLDEF and CHPRETDEF.  These are
{               used in the new interface to EXE$CHKPRO_INT.
{
{       V03-132 RSH0121         R. Scott Hanna          21-Mar-1984
{               $CLUBDEF / Remove the QF_WRITE flags bit. Rename the
{               QF_SKIP_READ flags bit to QF_FAILED_NODE.
{
{       V03-131 EMD0065         Ellen M. Dusseault      14-Mar-198
{               Move the journal definition in $DYNDEF to the sub-
{               type region where its subtypes are defined. Ensure
{               that the first subtype of a generic function has a
{               value of 1.
{
{       V03-130 JLV0340         Jake VanNoy              9-MAR-1984
{               Remove BRK$Q_OLDPRIVS, add BRK$T_DEVNAME.
{
{       V03-129 DWT0189         David W. Thiel           9-Mar-1984
{               Add CLUB$L_RETRYCNT.  Add CDRP$K_PART_MAP.  Add
{               CSB$Q_REFTIME.
{
{       V03-128 LMP0206         L. Mark Pilant,          7-Mar-1984  11:53
{               Add additional flags to the CCB status to note that physical
{               and logical I/O access checks have been done.
{
{       V03-127 RLRBIDEFS1      Robert L. Rappaport      5-Mar-1984
{               Correct error in $BUADEF and update $BIICDEF.
```

```
{
{       V03-126 CDS0002         Christian D. Saether    28-Feb-1984
{               Add AQB$V_XQIOPROC, AQB$L_BUFCACHE,
{               DYN$C_PGD, DYN$C_PGD_FILBC.
{
{       V03-125 DWT0180         David W. Thiel          27-Feb-1984
{               Add CLUB$V_SHUTDOWN, CLUB$V_QF_DYNVOTE, CLUB$T_QDNAME,
{               CLUB$V_ADJ_QUORUM, CLUB$W_ADJ_QUORUM,
{               CSB$W_QAITCNT, CSB$L_CURRCDRP, CSB$V_SHUTDOWN,
{               CSB$V_QF_DYNVOTE, CSB$W_CNX_STS_R0, CSB$W_CNX_STS_R1.
{
{       V03-124 WHM0004         Bill Matthews           24-feb-1984
{               Remove obsolete field ACF$B_VECTOR.
{
{       V03-123 SSA0010         Stan Amway              14-Feb-1984
{               Added DYN$C_DCCB to module $DYNDEF.
{
{       V03-122 ROW0298         Ralph O. Weber          10-FEB-1984
{               Define CDRB$W_ENDMSGSIZ, a field in the class driver CDRP
{               extension which holds the size of the most recent MSCP end
{               message.
{
{       V03-121 ROW0296         Ralph O. Weber           6-FEB-1984
{               Add CDDB$V_RSTRTWAIT which when set indicates that a
{               connection is waiting to execute RESTART_NEXT_CDRP.
{
{       V03-120 WHM0003         Bill Matthews           04-Feb-1984
{               Change field ACF$B_COMBO_VECTOR to ACF$B_VECTOR and added
{               field ACF$B_COMBO_VECTOR_OFFSET to clean up support for
{               combo style devices.
{
{       V03-119 LMP0185         L. Mark Pilant,         31-Jan-1984  11:02
{               Add CCB$V_PROCHKDON to indicate that a protection check
{               has been completed on the channel (for sharable, non-mountable
{               devices).
{
{       V03-118 ROW0289         Ralph O. Weber          25-JAN-1984
{               Add three DDT dispatch fields for various driver-specific
{               flavors of mount verification:  DDT$L_MNTV_SQD for sequential
{               device mount verification, DDT$L_MNTV_FOR for foreign device
{               mount verification, and DDT$L_MNTV_SSC for shadow set state
{               change mount verification.
{
{       V03-117 ROW0280         Ralph O. Weber          14-JAN-1984
{               Rearrange bits in CDRP$L_DUTUFLAGS so that CDRP$M_CAND is the
{               low-order bit.  This provides for the fastest possible testing
{               of the bit, which must be tested on every trip through the
{               mainline code path of the disk class driver.  Add CDRP$M_PERM
{               and CDRP$M_HIRT to CDRP$L_DUTUFLAGS.  Add CDDB$M_NOCONN to
{               CDDB$W_STATUS.  Add CDDB$Q_WTUCBCTR, a counter of the number
{               of UCBs waiting for mount verification to complete before
{               single stream CDRP processing can begin.
{
{       V03-116 WHM0002         Bill Matthews           12-Jan-1984
{               Moved $ACFDEF back into this module since it can be
{               referenced by drivers.
```

```
{
{       V03-115 DWT0160         David W. Thiel          11-Jan-1984
{               Rename CSB$Q_TIMEOUT to CSB$L_TIMEOUT.  Remove
{               CSB$L_SPARE% fields.
{
{       V03-114 RSH0090         R. Scott Hanna          11-Jan-1984
{               Add QF_CSP bit to FLAGS longword in $CLUDCBDEF.
{
{       V03-113 ACG0385         Andrew C. Goldstein,    9-Jan-1984  17:04
{               Add $ALFDEF - auto-login file definitions
{
{       V03-112 ROW0275         Ralph O. Weber          7-JAN-1984
{               Add CDRP$K_PART_RESP, block transfer partner responding,
{               to connection manager extension in $CDRPDEF.  Add
{               CDRP$L_DUTUFLAGS and CDRP$W_DUTUCNTR to class driver
{               extension in $CDRPDEF.
{
{       V03-111 SSA0004         Stan Amway              29-Dec-1983
{               Added DYN$C_PMB and DYN$C_PFB to module $DYNDEF.
{
{       V03-110 WHM001          Bill Matthews           14-Dec-1983     14:12
{               Moved $ACFDEF to BOOTDEF.SDL
{
{       V03-109 LY0440          Larry Yetto             7-DEC-1983 16:12
{               Add DYN$C_JNL_DIOREAD
{
{       V03-018 LMP0177         L. Mark Pilant,         7-Dec-1983  10:06
{               Reduce the size of te local rights list in the ARB.
{               Also, move $ACLDEF to STARDEFAE.SDL.
{
{       V03-107 ACG0376         Andrew C. Goldstein,    5-Dec-1983  13:14
{               Restructure CIA block for breakin detection changes
{
{       V03-106 ROW0263         Ralph O. Weber          24-NOV-1983
{               Add DDB$L_2P_UCB which is equivalent to DDB$L_DP_UCB.  Add
{               CDDB$V_ALCLS_SET bit which provides a mechanism for limiting
{               the determination of allocation class for class driver devices
{               to once.
{
{       V03-105 DWT0151         David W. Thiel          17-Nov-1983
{               Replace CLUB$W_NEWMEMSEQ with CLUB$W_QDVOTES.
{               Add CSB$W_QDVOTES.
{
{       V03-104 SOP0001         J. R. Sopka             11-Nov-1983
{               Change definition of DDB$T_NAME and DDB$T_DRVNAME fields
{               of $DDBDEF to produce DDB$S_NAME and DDB$S_DRVNAME symbols
{               and *_LEN and *_STR symbols for referencing subfields of
{               these counted ASCII strings.
{
{       V03-103 DWT0146         David W. Thiel          11-Nov-1983
{               $CSBDEF - Add CSB$W_LCKDIRWT to support distribution of
{                         lock manager directory over a cluster.
{               $CLUBDEF - Add CLUB$W_MEMSEQ and CLUB$W_NEWMEMSEQ to
{                          provide sequence number for cluster membership
{                          transitions.
{               $DYNDEF - Add DYN$C_CLU_LCKDIR to identify lock
```

```
{                       manager directory vector.
{
{       V03-102 RSH0077         R. Scott Hanna          10-Nov-1983
{               $CLUDCBDEF - Change BUF0 and BUF1 sizes. Add FLAGS longword.
{               $CLUBDEF - Remove QF_TRANS and QF_TIMEOUT FLAGS bits. (Moved
{                          to $CLUDCBDEF). Add FMERIT longword.
{
{       V03-101 RLRBIDEFS       Robert L. Rappaport     09-Nov-1983
{               Add $BUADEF, $BIMEMDEF and $BIICDEF.
{
{       V03-100 DWT0142         David W. Thiel          07-Nov-1983
{               Define $CLUOPTDEF structure for maintaining the
{               context needed to perform optimal cluster reconfigurations.
{
{       V03-099 DWT0135         David W. Thiel          05-Oct-1983
{               Add CLUB$V_LOST_CNX bit to $CLUBDEF to more finely
{               sort out states during cluster failover.
{
{       V03-098 KDM0083         Kathleen D. Morse       20-Sep-1983
{               Fix offsets in $EMBCRDEF, which were incorrect due to
{               moving 4 IPRs from the cpu-independent area to the cpu-
{               dependent area.
{
{       V03-097 KDM0082         Kathleen D. Morse       20-Sep-1983
{               Add BTD symbols for QNA and PROM for Micro-VAX booting.
{
{       V03-096 DWT0130         David W. Thiel          15-Sep-1983
{               Add CLUB$V_RECONFIG and CLUB$V_LOSTMSG bits to the
{               CLUB$L_FLAGS field in $CLUBDEF.  Add CLUB$B_HANG_FKB
{               field to $CLUBDEF.
{
{       V03-095 ACG0354         Andrew C. Goldstein,    9-Sep-1983  19:11
{               Remove unused fields from CHIP$ block, rearrange
{               for more efficient access
{
{       V03-094 ROW0215         Ralph O. Weber          25-AUG-1983
{               Add CDDB$B_FOVER_CTR and some reserved fields to the CDDB.
{               Also correct comments on CDDB$W_RSTRTCNT.
{
{       V03-093 RSH0056         R. Scott Hanna          23-Aug-1983
{               Add $CLUDCBDEF. Add DYN$C_CLU_CLUDCB to $DYNDEF.
{
{       V03-092 KDM0073         Kathleen D. Morse       22-Aug-1983
{               Add BQO$L_UMR_TMPL, BQO$B_UMR_DP, BQO$B_CPUTYPE, BQO$L_CPUDATA,
{               BQO$L_TENUSEC, and BQO$L_UBDELAY.
{
{       V03-091 GAS0168         Gerry Smith             22-Aug-1983
{               Add definitions for the Compound Intrusion Analysis block,
{               as well as DYN$C_CIA, to identify the block type.
{
{       V03-090 DWT0120         David W. Thiel          19-Aug-1983
{               Improve use of SDL in $CLUBDEF and $CSBDEF.  Add $CLUBFKB
{               subblock and other fields to $CLUBDEF.
{
{       V03-089 LMP0136         L. Mark Pilant,         9-Aug-1983  13:15
{               Correctly align the protection fields in $CHIPDEF.
```

```
{
{       V03-088 CDS0001         Christian D. Saether    2-Aug-1983
{               Remove type definition for obsolete RVX structure.
{
{       V03-087 LY0404          Larry Yetto             2-AUG-1983 14:42:03
{               Add DYN$C_JNL_MSGDATA
{
{       V03-086 DWT0115         David W. Thiel          1-Aug-1983
{               Add CLUB$V_BACKOUT bit to $CLUBDEF.
{
{       V03-085 BLS0231         Benn Schreiber          31-Jul-1983
{               Correct EMBCR definition
{
{       V03-084 DWT0113         David W. Thiel          29-Jul-1983
{               Add quorum disk support to CLUB and CSB.  Add
{               CLUB$V_INIT bit to CLUB to synchronize with SYSINIT.
{
{       V03-083 MLJ0115         Martin L. Jack          29-Jul-1983
{               Add DJI$K_FILE_SPECIFICATION.
{
{       V03-082 LY0402          Larry Yetto              29-JUL-1983 14:27:42
{               Add DYN$C_JNL_BXSTS
{
{       V03-081 PRB0229         Paul Beck                29-JUL-1983 13:40
{               Add CLUB$L_CSPFL, CLUB$L_CSPBL, CLUB$L_CSPIPID.
{
{       V03-080 NPK3029         N. Kronenberg           29-Jul-1983
{               Add performance counters to $CDTDEF.
{
{       V03-079 KDM0062         Kathleen D. Morse       28-Jul-1983
{               Move ICCS, ICR, ACCS, and TODR to cpu-dependent registers
{               in $EMBCRDEF.
{
{       V03-078 JLV0276         Jake VanNoy             27-JUL-1983
{               Change CRB$x_TT symbols to CRB$x_DZ.
{
{       V03-077 RLREMB          Robert L. Rappaport     27-Jul-1983
{               Add EMB$C_INVSTS, EMB$C_INVATT, EMB$C_NOUNIT_DG, and
{               EMB$C_LOGMSCP.
{
{       V03-076 JSV0366         Joost Verhofstad        27-JUL-1983
{               Add DYN$C_JNL_MSG
{
{       V03-075 LY0395          Larry Yetto             25-JUL-1983 13:42:30
{               Add DYN$C_JNLWCB and CLUB$L_JNL_FAIL
{
{       V03-074 RNG0074         Rod Gamache             25-Jul-1983
{               Add CXB$Q_STATION overlay to $CXBDEF.
{
{       V03-073 JLV0275         Jake VanNoy             25-JUL-1983
{               Add $BRKTDEF, used by $BRKTHRU and cluster broadcast module.
{               Remove obsolete $BRDDEF.
{
{       V03-072 LMP0125         L. Mark Pilant,         26-Jun-1983  21:35
{               Twiddle the $CHIPDEF structure definition to make the
{               symbol CHIP$L_PROTECTION available in MACRO.
```

```
{
{       V03-071 DWT0107         David W. Thiel          23-Jun-1983
{               Correct previous entry.  Remove the CLUB$W_LOCKCNT
{               field and add the CLUB$L_TQE field to $CLUBDEF.
{
{       V03-070 RPG0069         Bob Grosso              23-Jun-1983
{               Add structure type codes for new Known file structures
{               and remove KFI and KFH from DYNDEF.
{
{       V03-069 RLRCDDB1        Robert L. Rappaport     23-Jun-1983
{               Added CDDB$B_DAPCOUNT.
{
{       V03-068 KTA060          Kerbey T. Altmann       23-Jun-1983
{               Added BQO$L_UNIT_DISC and BQO$L_DEVNAME.
{
{       V03-067 ADE0001         Alan D. Eldridge        22-Jun-1983
{               Added CXB$L_END_ACTION, CXB$W_BOFF, CXB$W_BCNT.
{               Removed CXB$W_UQUO, CXB$W_JQUO, CXB$B_ASTCNT CXB$L_SSB and
{               CXB$L_ENDACTION.
{
{       V03-066 ROW0185         Ralph O. Weber          21-JUN-1983
{               Delete CSB$L_SELQFL and CSB$L_SELQBL and replace that queue
{               header with CSB$L_PARTNERQFL and CSB$L_PARTNERQBL, the queue
{               header for the queue of active block-transfer partner BTX
{               blocks.  Add block transfer fields to the connection manager
{               CDRP extension.   Add NO_JOIN bit in CLUB.
{
{       V03-065 RLRCDDB         Robert L. Rappaport     17-Jun-1983
{               Add CDDB$L_DAPCDRP and CDDB$L_CDDBLINK.
{
{       V03-064 LMP0120         L. Mark Pilant,         16-Jun-1983  10:11
{               Add subfields to the protection vector in the CHIP
{               block.
{
{       V03-063 MKL0095         Mary Kay Lyons          01-Jun-1983
{               Add DYN$C_JNL_RC subtype field for read context structure.
{
{       V03-062 DWT0102         David W. Thiel          27-May-1983
{               Add CLUFCB sub-block and CLUB$L_LOCAL_CSID to $CLUBDEF.
{
{       V03-061 RLRALOCLS       Robert L. Rappaport     26-May-1983
{               Add CDDB$L_ALLOCLS.
{
{       V03-060 RLRDPATH        Robert L. Rappaport     25-May-1983
{               Add DDB$L_DP_UCB, secondary UCB link for dual path
{               controllers.
{
{       V03-059 LY0376          Larry Yetto             24-MAY-1983 16:31:35
{               Add DYN$C_JNL_CWQ subtype field for journal cluster write Q
{               entry.
{
{       V03-058 DWT0100         David W. Thiel          23-May-1983
{               Revise $CLUBDEF and $CSBDEF to support N node clusters.
{
{       V03-057 JSV0294         Joost Verhofstad        20-MAY-1983
{               Add DYN$ subtype values for journaling, add DYN$C_JNL
```

```
{                and remove other journaling DYN$ type values
{
{       V03-056 KDM0044         Kathleen D. Morse       3-May-1983
{               Change EXE$GL_ARCHFLAGS to EXE$GL_ARCHFLAG.
{               Also, add new sub-types for loadable instruction emulation code.
{
{       V03-55  EAD0055         Elliott A. Drayton      1-May-1983
{               Removed one of two lines which defined CLU_CLUB in $DYNDEF.
{
{       V03-54  DWT0097         David W. Thiel          29-Apr-1983
{               Add pointer to system block to $CSBDEF.
{
{       V03-53  KDM0043         Kathleen D. Morse       29-Apr-1983
{               Add $ARCDEF, architectural bit field definitions for
{               EXE$GL_ARCHFLAGS.
{
{       V03-052 LMP0109         L. Mark Pilant,         29-Apr-1983  12:47
{               Add item to $CHIPDEF for returning the access rights mask.
{
{       V03-051 JLV0246         Jake VanNoy             29-APR-1983
{               Move $BRKDEF to STARLET.
{
{       V03-050 TCM0005         Trudy C. Matthews       28-Apr-1983
{               Add new field to $DDBDEF, DDB$L_ALLOCLS.
{
{       V03-049 SRB0082         Steve Beckhardt         28-Apr-1983
{               Reroved message queue from $CDRPDEF
{
{       V03-048 GAS0128         Gerry Smith             28-Apr-1983
{               Add a new type for $DYNDEF, DYN$C_RIGHTSLIST.
{
{       V03-047 MLJ0112         Martin L. Jack, 27-Apr-1983  16:48
{               Add $DJIDEF.
{
{       V03-046 ROW0185         Ralph O. Weber          24-APR-1983
{               Add CLU subtype for the block transfer extension, BTX.
{
{       V03-45  LMP0095         L. Mark Pilant,         14-Apr-1983  16:19
{               Add a hack definition in $CHIPDEF until the new SDL comes along.
{
{       V03-044 ROW0181         Ralph O. Weber          14-APR-1983
{               Add CDRP$L_VAL7 and CDRP$L_VAL8.  Eventually, these fields
{               should replace currently used fields.  However, the current
{               fields cannot be deleted yet.  Therefore, the CDRP and the
{               IRP will be bigger than we want for a few weeks.
{
{       V03-043 MMD0143         Meg Dumont,     14-Apr-1983  9:15
{               Add $EO4DEF, EOF4 definitions.
{
{       V03-42  LMP0099         L. Mark Pilant,         14-Apr-1983  8:32
{               Add return length addr storage to $CHIPDEF.
{
{       V03-041 DWT0094         David W. Thiel          12-Apr-1983
{               More miscellaneous $CLUBDEF and $CSBDEF changes.
{
{       V03-040 TCM0004         Trudy C. Matthews       12-Apr-1983
```

```
{               Add error log types EMB$C_EMM (Environmental Monitor logs),
{               EMB$C_HLT (processor error halt logs), and EMB$C_CRBT
{               (console reboot logs) to $EMBETDEF.
{
{       V03-039 LY0350          Larry Yetto             11-APR-1983 07:48:25
{               Remove DYN$C_NTE and replace it with DYN$C_JNLCB
{
{       V03-038 DWT0092         David W. Thiel          6-Apr-1983
{               Add fields to $CSBDEF and $CLUBDEF
{
{       V03-037 DWT0088         David W. Thiel          29-Mar-1983
{               Add fields to $CSBDEF and $CLUBDEF.
{
{       V03-036 JWH0204         Jeffrey W. Horn         24-Mar-1983
{               Add DYN$C_NON_PAGED and DYN$C_PAGED as subtypes of
{               DYN$C_LOADCODE.
{
{       V03-035 LMP0086         L. Mark Pilant,         11-Mar-1983  9:25
{               Longword align the $CHIPDEF structure.
{
{       V03-034 DWT0084         David W. Thiel          10-Mar-1983
{               Add $CLUBDEF to define cluster block.
{
{       V03-033 LMP0084         L. Mark Pilant,         1-Mar-1983  16:05
{               Add $CHIPDEF, the internal interface definition to the
{               $CHKPRO system service.
{
{       V03-032 JLV0235         Jake VanNoy             1-MAR-1983
{               Add $BRKDEF, for $BRKTHRU system service.
{
{       V03-031 RLRDDBB         Robert L. Rappaport     1-Mar-1983
{               Also added CDDB$L_ORIGUCB, pointer to UCB created by SYSGEN.
{
{       V03-030 RLRDDBA         Robert L. Rappaport     1-Mar-1983
{               Also added CDDB$L_UCBCHAIN to link all UCB's
{               on a connection into a chain.
{
{       V03-029 RLRDDB          Robert L. Rappaport     1-Mar-1983
{               Add DDB$L_CONLINK, (Connection Link) to allow linking
{               of all DDB's that service one Disk or Tape Class Connection.
{
{       V03-028 JLV0232         Jake VanNoy             24-FEB-1983
{               Add CCB$V_IMGTMP flag to allow an image temporary,
{               kernel mode channel.
{
{       V03-027 ROW0162         Ralph O. Weber          23-FEB-1983
{               Add CANCEL type for associated mailbox.  This will be used
{               when the a mailbox driver's cancel I/O routine is called as
{               the result of a channel deassign which disassociates a mailbox
{               causing the mailbox reference count to go to zero.  In this
{               case the mailbox is about to be deleted and the driver is
{               required to cleanup preparatory to that event.
{
{       V03-026 DWT0076         David W. Thiel          22-Feb-1983
{               Add DYN$C_CLU as the major type for all cluster
{               related control blocks.  Make CSB to be a subtype.
```

```
{               Add fields to $CSBDEF.
{               Add DYN$C_SCS_SPNB to $DYNDEF.
{
{       V03-025 DWT0075         David W. Thiel          11-Feb-1983
{               Correct previous entry.  Add fields to $CSBDEF.
{
{       V03-024 DWT0066         David W. Thiel          20-Jan-1983
{               Add DYN$C_SCS_SPPB control block subtype.
{
{       V03-023 MIR0022         Michael I. Rosenblum    19-Jan-1983
{               Move TTDRIVER local CRB and IDB definitions into
{               the main definitions.
{
{       V03-022 ROW0156         Ralph O. Weber          12-JAN-1983
{               Remove hard coded filler offsets in IRP to be symbolic.
{               Reorder connection manager extension to CDRP so that the VAL1
{               through VAL6 fields overlay the fields in the block transfer
{               CDRP extension.  Add DYN$C_NTE for journaling memory-format
{               name table entries which will be produced on slave nodes.
{
{       V03-021 SRB0060         Steve Beckhardt         7-Jan-1983
{               Added some new data structure definitions in $DYNDEF.
{               Added $CSBDEF (Cluster System Blocks).  Added connection
{               manager extension to $CDRPDEF.
{
{       V03-020 WMC0020         Wayne Cardoza           05-JAN-1982
{               New machine check error codes in EMBETDEF.
{
{       V03-019 KTA3026         Kerbey T. Altmann       03-Jan-1983
{               Add GETDONE flag to $ACFDEF.
{
{       V03-018 ACG0307         Andrew C. Goldstein,    30-Dec-1982  17:11
{               Add rights list to ARB
{
{       V03-017 ACG0303         Andrew C. Goldstein,    9-Dec-1982  15:11
{               Add FILL attribute to extraneous field names
{
{       V03-016 DMW4015         DMWalp                  9-Dec-1982
{               Added DYN structure type for LNM blocks
{
{       V03-015 MLJ0101         Martin L. Jack, 17-Nov-1982  13:56
{               Fix AVECTOR definition.
{
{       V03-014 KTA3019         Kerbey T. Altmann       08-Nov-1982
{               Add new field to DDB for system block address.
{
{       V03-013 TCM0003         Trudy C. Matthews       29-Oct-1982
{               Move definition of ADP$L_AVECTOR (see TCM0002) into
{               common portion of ADP structure.  Add $CONDEF, which
{               defines console function codes.
{
{       V03-012 RLRPOLL         Robert L. Rappaport     8-Oct-1982
{               Add CDDB$M_POLLING bit to CDDB$W_STATUS.
{
{       V03-011 ROW0131         Ralph O. Weber          7-OCT-1982
{               Increment DPT$C_VERSION to indicate significant change in
```

```
{               driver data structures.  This will cause SYSGEN to abort, with
{               an error message, attempts to load V3.x drivers on post X1NR
{               systems and vice versa.
{
{       V03-010 ROW0125         Ralph O. Weber          19-SEP-1982
{               Add DDT$L_CLONEDUCB to driver entry points listed in $DDTDEF,
{               the offset deffinitions for the driver dispatch table.
{
{       V03-009 TCM0002         Trudy C. Matthews       10-Aug-1982
{               Add new field ADP$L_AVECTOR, the address of the 1st SCB
{               vector for this adaptor, to $ADPDEF
{
{       V03-008 LMP0036         L. Mark Pilant,         29-Jun-1982  13:00
{               Add the Access Control List (ACL) data structure.  Also,
{               add the DYN$C_ACL data structure type code.
{
{       V03-007 LY0026          Larry Yetto             29-Jun-1982
{               Add DYN$C_NDL to data structure type definitions
{
{       V03-006 TMH0006         Tim Halvorsen           14-Jun-1982
{               Add WQE and XWB structure codes for DECnet.
{
{       V03-005 JSV007          Joost Verhofstad        10-Jun-1982
{               Add DYN$C_ADL, DYN$C_JNL_BUF, DYN$C_VCL
{
{       V03-004 RLRV3A3         Robert L. Rappaport     15-Apr-1982
{               Add EMB$C_ACPTH to log Attention messages.
{
{       V03-003 RLRV3A2         Robert L. Rappaport     6-Apr-1982
{               Add EMB$C_AVATN, EMB$C_DUPUN so as to log Attention
{               messages from MSCP controllers. Also add EMB$C_IVCMD.
{
{       V03-002 KTA0090         Kerbey T. Altmann       29-Mar-1982
{               Add new field to BOOT QIO vector for microcode address.
{
{       V03-001 RLRV3A1         Robert L. Rappaport     23-Mar-1982
{               Add two state bits to CDDB definition and add field
{               to EMBSPDEF (Error Log Software Parameter block).
{
{**
```

```
module $ACMDEF;
/*+
/* ACMDEF - ACCOUNTING MANAGER DEFINITIONS
/*-



aggregate ACMDEF  union prefix ACM$;
    ACMDEF_BITS structure fill;
        PROCESS bitfield;                               /* PROCESS ACCOUNTING ENABLED
        IMAGE bitfield;                                 /* IMAGE ACCOUNTING ENABLED
        INTERACTIVE bitfield;                           /* INTERACTIVE ACCOUNTING ENABLED
        LOGFAIL bitfield;                               /* LOGIN FAILURE ACCOUNTING ENABLED
        SUBPROCESS bitfield;                            /* SUBPROCESS ACCOUNTING ENABLED
        DETACHED bitfield;                              /* DETACHED PROCESS ACCOUNTING ENABLED
        BATCH bitfield;                                 /* BATCH ACCOUNTING ENABLED
        NETWORK bitfield;                               /* NETWORK PROCESS ACCOUNTING ENABLED
        PRINT bitfield;                                 /* PRINT JOB ACCOUNTING ENABLED
        USER_DATA bitfield;                             /* USER_DATA ACCOUNTING ENABLED
        ACM_FUNC bitfield;                              /* ACM FUNCTION ACCOUNTING ENABLED
        SYS_FUNC bitfield;                              /* SYSTEM FUNCTION ACCOUNTING ENABLED
    end ACMDEF_BITS;

end ACMDEF;

    aggregate ACMDEF1 structure prefix ACM$ origin TYPE;
    MSGSTS word unsigned;                               /* MSG STATUS IN MAILBOX IOSB (JOBCTL SPECIFIC)
    MSGLEN word unsigned;                               /* MSG LENGTH IN MAILBOX IOSB (JOBCTL SPECIFIC)
    PROCID longword unsigned;                           /* PROCESS ID IN MAILBOX IOSB (JOBCTL SPECIFIC)
    TYPE word unsigned;                                 /* MESSAGE TYPE
    MAILBOX word unsigned;                              /* MAILBOX UNIT NUMBER
    PRVMSK quadword unsigned;                           /* PROCESS PRIV MASK
    UIC_OVERLAY union fill;
        UIC longword unsigned;                          /* PROCESS UIC
        UIC_FIELDS structure fill;
            MEM word unsigned;                          /* MEMBER UIC
            GRP word unsigned;                          /* GROUP UIC
        end UIC_FIELDS;
    end UIC_OVERLAY;
    USERNAME character length 12;                       /* USERNAME
    ACCOUNT character length 8;                         /* ACCOUNT NAME
    PROCPRI byte unsigned;                              /* PROCESS BASE PRIORITY
    FILL_1 byte dimension 3 fill prefix ACMDEF tag $$;  /* SPARE BYTES (LONGWORD ALIGNMENT)
    PID longword unsigned;                              /* PROCESS ID
    STS longword unsigned;                              /* PROCESS STATUS
    OWNER longword unsigned;                            /* OWNER PROCESS ID (0 => NONE)
    TERMINAL character length 8;                        /* TERMINAL NAME (COUNTED ASCII STRING)
    SYSTIME quadword unsigned;                          /* CURRENT SYSTEM TIME
/*
/* SEND TO ACCOUNTING MANAGER FIELDS
/*
    end ACMDEF1;

aggregate ACMDEF2 structure prefix ACM$;
    FILL_2 byte dimension 68 fill prefix ACMDEF tag $$;
    USERREQ word unsigned;                              /* USER REQUEST TYPE
```

```
    DATA character length 256;                          /* USER DATA
/*
/* PROCESS/IMAGE DELETE/PURGE FIELDS
/*
end ACMDEF2;

aggregate ACMDEF3 structure prefix ACM$;
    FILL_3 byte dimension 68 fill prefix ACMDEF tag $$;
    LOGIN quadword unsigned;                            /* PROCESS/IMAGE START TIME
    FINALSTS longword unsigned;                         /* PROCESS FINAL STATUS
    IMGCNT longword unsigned;                           /* IMAGE EXECUTION COUNT
    CPUTIME longword unsigned;                          /* CPU USAGE
    PAGEFLTS longword unsigned;                         /* PAGEFAULT COUNT
    PGFLTIO longword unsigned;                          /* PAGEFAULT I/O
    WSPEAK longword unsigned;                           /* WORKING SET PEAK
    PGFLPEAK longword unsigned;                         /* PAGE FILE PEAK
    DIOCNT longword unsigned;                           /* DIRECT I/O COUNT
    BIOCNT longword unsigned;                           /* BUFFERED I/O COUNT
    VOLUMES longword unsigned;                          /* VOLUME MOUNT COUNT
    NODEADDR word unsigned;                             /* MESSAGE OFFSET TO REMOTE NODE ADDRESS
    NODENAME word unsigned;                             /* MESSAGE OFFSET TO REMOTE NODE NAME
    REMOTEID word unsigned;                             /* MESSAGE OFFSET TO REMOTE ID
    IMAGENAME word unsigned;                            /* MESSAGE OFFSET TO IMAGE NAME
    constant PROCLEN equals . prefix ACM$ tag K;        /* MIN. PROCESS/IMAGE TERMINATION MESSAGE LENGTH
    constant PROCLEN equals . prefix ACM$ tag C;        /* MIN. PROCESS/IMAGE TERMINATION MESSAGE LENGTH
end ACMDEF3;

end_module $ACMDEF;
```

```
module $ACBDEF;
/**
/* AST CONTROL BLOCK DEFINITIONS
/*
/* AST CONTROL BLOCKS EXIST AS SEPARATE STRUCTURES AND AS SUBSTRUCTURES
/* WITHIN LARGER CONTROL BLOCKS SUCH AS I/O REQUEST PACKETS AND TIMER
/* QUEUE ENTRIES.
/*
/*-



aggregate ACBDEF structure prefix ACB$;
    ASTQFL longword unsigned;                          /*AST QUEUE FORWARD LINK
    ASTQBL longword unsigned;                          /*AST QUEUE BACKWARD LINK
    SIZE word unsigned;                                /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                /*STRUCTURE TYPE CODE
    RMOD_OVERLAY union fill;
        RMOD byte unsigned;                            /*REQUEST ACCESS MODE
        RMOD_BITS structure fill;
            MODE bitfield length 2;                    /*MODE FOR FINAL DELIVERY
            FILL_1 bitfield length 2 fill prefix ACBDEF tag $$;/*SPARE
            PKAST bitfield mask;                       /*PIGGY BACK SPECIAL KERNEL AST
            NODELETE bitfield mask;                    /*DON'T DELETE ACB ON DELIVERY
            QUOTA bitfield mask;                       /*ACCOUNT FOR QUOTA
            KAST bitfield mask;                        /*SPECIAL KERNEL AST
        end RMOD_BITS;
    end RMOD_OVERLAY;
    PID longword unsigned;                             /*PROCESS ID OF REQUEST
    AST longword unsigned;                             /*AST ROUTINE ADDRESS
    ASTPRM longword unsigned;                          /*AST PARAMETER
    KAST longword unsigned;                            /*INTERNAL KERNEL MODE XFER ADDRESS
    constant "LENGTH" equals . prefix ACB$ tag K;      /* Length of block.
    constant "LENGTH" equals . prefix ACB$ tag C;      /* Length of block.

end ACBDEF;

end_module $ACBDEF;
```

```
module $ACFDEF;
/*
/* CONFIGURATION CONTROL BLOCK OFFSET DEFINITIONS
/*


aggregate ACFDEF structure prefix ACF$;
    ADAPTER longword unsigned;                  /*ADDRESS OF ADAPTER CONTROL BLOCK
    CONFIGREG longword unsigned;                /*ADDRESS OF CONFIGURATION STATUS REGISTER
    AVECTOR word unsigned;                      /*OFFSET TO ADAPTER INTERRUPT VECTOR (SCB)
    AUNIT byte unsigned;                        /*ADAPTER UNIT NUMBER
    AFLAG_OVERLAY union fill;
        AFLAG byte unsigned;                    /*ADAPTER GENERATION CONTROL FLAGS
        AFLAG_BITS structure fill;
            RELOAD bitfield mask;               /* RELOAD DRIVER
            CRBBLT bitfield mask;               /* CRB AND IDB ARE BUILT
            SCBVEC bitfield mask;               /* CVECTOR IS OFFSET INTO SCB
            NOLOAD_DB bitfield mask;            /* DON'T LOAD DATABASE, ONLY LOAD DRIVER
            SUPPORT bitfield mask;              /* DEVICE IS SUPPORTED
            GETDONE bitfield mask;              /* GET OF IO DATABASE ALREADY DONE
        end AFLAG_BITS;
    end AFLAG_OVERLAY;
    CONTRLREG longword unsigned;                /*ADDRESS OF CONTROL REGISTER
    CVECTOR word unsigned;                      /*OFFSET TO CONTROLLER INTERRUPT VECTOR (TABLE)
    CUNIT word unsigned;                        /*CONTROLLER UNIT NUMBER
    DEVNAME longword unsigned;                  /*ADDRESS OF DEVICE NAME COUNTED STRING
    DRVNAME longword unsigned;                  /*ADDRESS OF DRIVER NAME COUNTED STRING
    MAXUNITS word unsigned;                     /*MAXIMUM UNITS THAT CAN BE CONNECTED
    CNUMVEC byte unsigned;                      /*NUMBER OF CONTROLLER VECTORS
    COMBO_VECTOR_OFFSET byte;                   /*OFFSET TO START OF VECTORS FOR A COMBO STYLE DEVICE
    COMBO_CSR_OFFSET byte;                      /*OFFSET TO START OF CONTROL REGISTERS FOR A COMBO DEVICE
    NUMUNIT byte unsigned;                      /*NUMBER OF UNITS TO CONFIGURE
    FILL_1 word fill prefix ACFDEF tag $$;      /*(SPARE)
    DLVR_SCRH longword unsigned;                /*SCRATCH FOR DELIVER ROUTINES
    constant "LENGTH" equals . prefix ACF$ tag K;   /*LENGTH OF DEVICE DESCRIPTOR ARGUMENT LIST
    constant "LENGTH" equals . prefix ACF$ tag C;   /*LENGTH OF DEVICE DESCRIPTOR ARGUMENT LIST

end ACFDEF;

end_module $ACFDEF;
```

```
module $ADPDEF;

/**
/* ADAPTER CONTROL BLOCK DEFINITIONS
/*
/* THERE IS ONE ADP FOR EACH SYSTEM INTERCONNECT ADAPTER THAT IS
/* USED FOR ANY TYPE OF I/O.  FOR EXAMPLE: MASBUSS ADAPTER, UNIBUS
/* ADAPTER.  THERE IS NO ADAPTER CONTROL BLOCK FOR MAIN MEMORY ADAPTERS.
/*-


aggregate ADPDEF structure prefix ADP$;
    CSR longword unsigned;                          /*ADAPTER CONFIGURATION STATUS REGISTER ADDRESS
    LINK longword unsigned;                         /*ADDRESS OF NEXT ADAPTER CONTROL BLOCK
    SIZE word unsigned;                             /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                             /*STRUCTURE TYPE CODE
    NUMBER byte unsigned;                           /*ORDINAL ADAPTER NUMBER
    TR word unsigned;                               /*CONFIGURATION TR NUMBER
    ADPTYPE word unsigned;                          /*SOFTWARE ADAPTER TYPE
    VECTOR_OVERLAY union fill;
        VECTOR longword unsigned;                   /*UBA - ADDRESS OF VECTOR JUMP TABLE
        CRB longword unsigned;                      /*MBA OR DR32 - ADDRESS OF ADAPTER'S CRB
    end VECTOR_OVERLAY;
    DPQFL_OVERLAY union fill;
        DPQFL longword unsigned;                    /*UBA - DATAPATH WAIT QUEUE FORWARD LINK
        PRQQFL_OVERLAY union fill;
            PRQQFL longword unsigned;               /*MPM - INTER-PROCESSOR REQUEST WAIT QUEUE FLINK
            MBASCB longword unsigned;               /*MBA - SCB VECTOR VALUE FOR MBA NEXUS
        end PRQQFL_OVERLAY;
    end DPQFL_OVERLAY;
    DPQBL_OVERLAY union fill;
        DPQBL longword unsigned;                    /*UBA - DATAPATH WAIT QUEUE BACKWARD LINK
        PRQQBL_OVERLAY union fill;
            PRQQBL longword unsigned;               /*MPM - INTER-PROCESSOR REQUEST WAIT QUEUE BLINK
            MBASPTE longword unsigned;              /*MBA - SPTE VALUE WHICH MAPS MBA ADDRESS SPACE
        end PRQQBL_OVERLAY;
    end DPQBL_OVERLAY;
    AVECTOR longword unsigned;                      /* ADDR OF 1ST SCB VECTOR FOR THIS ADAPTOR
    BI_ONLY longword unsigned dimension 4;          /*BI ADAPTER VOLATILE INFO
    constant MBAADPLEN equals . prefix ADP$ tag K;  /*LENGTH OF ADP FOR MASSBUS ADAPTER
    constant MBAADPLEN equals . prefix ADP$ tag C;  /*LENGTH OF ADP FOR MASSBUS ADAPTER
    constant DRADPLEN equals . prefix ADP$ tag K;   /*LENGTH OF ADP FOR DR32
    constant DRADPLEN equals . prefix ADP$ tag C;   /*LENGTH OF ADP FOR DR32
    constant CIADPLEN equals . prefix ADP$ tag K;   /*LENGTH OF ADP FOR CI
    constant CIADPLEN equals . prefix ADP$ tag C;   /*LENGTH OF ADP FOR CI
    MRQFL_OVERLAY union fill;
        MRQFL longword unsigned;                    /*UBA - MAP REGISTER WAIT QUEUE FORWARD LINK
        SHB longword unsigned;                      /*MPM - SHARED MEMORY CONTROL BLOCK ADDR
    end MRQFL_OVERLAY;
    MRQBL_OVERLAY union fill;
        MRQBL longword unsigned;                    /*UBA - MAP REGISTER WAIT QUEUE BACKWARD LINK
        PORT byte unsigned;                         /*MPM - PORT NUMBER
    end MRQBL_OVERLAY;
    INTD longword unsigned dimension 3;             /*UBA - INTERRUPT TRANSFER VECTOR
    constant MPMADPLEN equals . prefix ADP$ tag K;  /*LENGTH OF ADP FOR MULTI-PORT MEMORY
```

```
    constant MPMADPLEN equals . prefix ADP$ tag C;      /*LENGTH OF ADP FOR MULTI-PORT MEMORY
    UBASCB longword unsigned dimension 4;               /*UBA - SCB VECTOR VALUE FOR 4 UBA VECTORS
    UBASPTE longword unsigned dimension 2;              /*UBA - SPTE VALUES FOR MAPPING UBA ADDRESSES
    MRACTMDRS longword unsigned;                        /* UBA - ! active map register descriptors
    DPBITMAP word unsigned;                             /* UBA - Datapath Allocation Bitmap
    MRNFENCE word unsigned;                             /* Fence preceeding array. Init'ed to -1
    MRNREGARY word unsigned dimension 124;              /* ! map regs in an extent array
    MRFFENCE word unsigned;                             /* Fence precedding array. Init'ed to -1
    MRFREGARY word unsigned dimension 124;              /* 1st reg in extent array.
    UMR_DIS word unsigned;                              /* Num of Map Registers to disable
                                                        /* NOTE** - UNIBUS ADP must be integral
                                                        /*  number of longwords long so that 780
                                                        /*  interrupt vectors are longword aligned.
    constant UBAADPLEN equals . prefix ADP$ tag K;      /*LENGTH OF ADP FOR UNIBUS ADAPTER
    constant UBAADPLEN equals . prefix ADP$ tag C;      /*LENGTH OF ADP FOR UNIBUS ADAPTER
    constant NUMDATAP   equals 16  prefix ADP tag $C;   /*UBA - NUMBER OF DATAPATHS

end ADPDEF;

end_module $ADPDEF;
```

```
module $AIBDEF;
/*+
/* FORMAT OF ACP I/O BUFFER PACKET. THIS PACKET CONTAINS ALL THE DATA
/* TRANSMITTED FROM THE USER TO THE ACP AND BACK FOR AN ACP FUNCTION.
/* NOTE THAT THE DESCRIPTORS IN THE PACKET ARE TREATED BY BLISS CODE
/* AS A BLOCKVECTOR.
/*-


aggregate AIBDEF structure prefix AIB$;
    DESCRIPT longword unsigned;                 /* ADDRESS OF START OF DESCRIPTORS
    FILL_1 longword fill prefix AIBDEF tag $$;  /* SPARE LONGWORD
    SIZE word unsigned;                         /* SIZE OF PACKET
    TYPE byte unsigned;                         /* PACKET TYPE CODE
    FILL_2 byte fill prefix AIBDEF tag $$;      /* SPARE
    constant "LENGTH" equals . prefix AIB$ tag K;   /* LENGTH OF PACKET HEADER
    constant "LENGTH" equals . prefix AIB$ tag C;   /* LENGTH OF PACKET HEADER

end AIBDEF;

end_module $AIBDEF;
```

```
module $ABDDEF;


aggregate ABDDEF structure prefix ABD$;
    TEXT word unsigned;                                 /* WORD OFFSET TO DATA TEXT
    COUNT word unsigned;                                /* BYTE COUNT OF TEXT
    USERVA longword unsigned;                           /* USER VIRTUAL ADDRESS OF TEXT
    constant "LENGTH" equals . prefix ABD$ tag K;       /* SIZE OF DESCRIPTOR
    constant "LENGTH" equals . prefix ABD$ tag C;       /* SIZE OF DESCRIPTOR
    constant WINDOW     equals 0  prefix ABD tag $C;    /* DESCRIPTOR FOR WINDOW ADDRESS
    constant FIB        equals 1  prefix ABD tag $C;    /* DESCRIPTOR FOR FIB
    constant NAME       equals 2  prefix ABD tag $C;    /* DESCRIPTOR FOR NAME STRING
    constant RESL       equals 3  prefix ABD tag $C;    /* DESCRIPTOR FOR RESULT LENGTH
    constant RES        equals 4  prefix ABD tag $C;    /* DESCRIPTOR FOR RESULT STRING
    constant ATTRIB     equals 5  prefix ABD tag $C;    /* FIRST ATTRIBUTE DESCRIPTOR

end ABDDEF;

end_module $ABDDEF;
```

```
module $ALFDEF;
/**
/*
/* $ALFDEF - structure for auto-login file.
/*
/*-


aggregate ALFDEF structure prefix ALF$;
    DEVNAME character length 63;                /* Terminal device name
    USERNAME character length 32;               /* Associated username
    FILL_1 byte dimension 33 fill;
    constant "LENGTH" equals . tag C;
    constant "LENGTH" equals . tag K;
end ALFDEF;

end_module $ALFDEF;
```

```
module $AQBDEF;
/**
/* DEFINITION OF ACP QUEUE HEADER
/*-



aggregate AQBDEF structure prefix AQB$;
    ACPQFL longword unsigned;                       /* QUEUE FORWARD LINK
    ACPQBL longword unsigned;                       /* QUEUE BACK LINK
    SIZE word unsigned;                             /* CONTROL BLOCK SIZE IN BYTES
    TYPE byte unsigned;                             /* BLOCK TYPE CODE
    MNTCNT byte unsigned;                           /* ACP MOUNT COUNT
    ACPPID longword unsigned;                       /* ACP PROCESS PID
    LINK longword unsigned;                         /* AQB LIST LINKAGE
    STATUS_OVERLAY union fill;
        STATUS byte unsigned;                       /* STATUS BYTE
        STATUS_BITS structure fill;
            UNIQUE bitfield mask;                   /* ACP IS UNIQUE TO THIS DEVICE
            DEFCLASS bitfield mask;                 /* ACP IS DEFAULT FOR THIS CLASS
            DEFSYS bitfield mask;                   /* ACP IS DEFAULT FOR THE SYSTEM
            CREATING bitfield mask;                 /* ACP IS CURRENTLY BEING CREATED
            XQIOPROC bitfield mask;                 /* eXtended QIO PROCessor is being used.
        end STATUS_BITS;
    end STATUS_OVERLAY;
    ACPTYPE byte unsigned;                          /* ACP TYPE CODE
/*
/* ***** The following ACP type codes are now a user visible interface
/* ***** and the values may not be changed.  There are parallel definitions
/* ***** in the $DVIDEF macro that define symbols of the form:
/* *****
/* *****                   DVI$C_ACP_F11V1
/* *****                   DVI$C_ACP_F11V2
/* *****                   DVI$C_ACP_MTA
/* *****                   ...
/* *****
/* ***** All new ACP type values must be added at the end and the names
/* ***** must be 5 characters or less to keep the DVI form of the name
/* ***** 15 characters or less.  Any additions must also be made in $DVIDEF
/* ***** and in the list of ASSUMES in the module SYSGETDEV in [SYS.SRC]
/*
    constant(
          UNDEFINED                                 /* UNDEFINED ACP
        , F11V1                                     /* FILES-11 STRUCTURE LEVEL 1
        , F11V2                                     /* FILES-11 STRUCTURE LEVEL 2
        , MTA                                       /* MAGTAPE
        , NET                                       /* NETWORKS
        , REM                                       /* REMOTE I/O
        , JNL                                       /* JOURNAL

        ) equals 0 increment 1  prefix AQB tag $K;
    CLASS byte unsigned;                            /* ACP CLASS CODE
    FILL_1 byte fill prefix AQBDEF tag $$;          /* RESERVED
    BUFCACHE longword unsigned;                     /* POINTER TO BUFFER CACHE
    constant 'LENGTH' equals . prefix AQB$ tag K;   /* SIZE OF AQB
    constant 'LENGTH' equals . prefix AQB$ tag C;   /* SIZE OF AQB
```

```
end AQBDEF;
end_module $AQBDEF;
```

```
module $ARBDEF;
/**
/*
/* Access Rights Block - structure defining process access rights and
/* privileges. Currently part of the PCB (meaning that the size of the
/* ARB declared here must track in the PCB).
/*
/*-

aggregate ARBDEF structure prefix ARB$;
    PRIV quadword unsigned;             /* Privilege mask
    FILL_1 longword fill tag $$;        /* Spare to allow for JIB type,size
    CLASS structure;                    /* Security classification mask
        FILL_2 longword dimension 5 fill tag $$;
    end CLASS;
    RIGHTSLIST longword unsigned dimension 4; /* Rights list descriptors
    constant HEADER equals . prefix ARB$ tag C; /* Length of header
    constant HEADER equals . prefix ARB$ tag K; /* Length of header
    RIGHTSDESC structure;               /* Descriptor for local rights list
        FILL_3 longword dimension 2 fill tag $$;
    end RIGHTSDESC;
    LOCALRIGHTS structure;              /* Process local rights list
        UIC longword unsigned;          /* User identification code.
        FILL_4 longword dimension 15 fill tag $$;
    end LOCALRIGHTS;
    constant "LENGTH" equals . prefix ARB$ tag K;       /* Structure length
    constant "LENGTH" equals . prefix ARB$ tag C;       /* Structure length
end ARBDEF;

end_module $ARBDEF;
```

```
module $ARCDEF;

/*+
/*
/* Bit definitions for EXE$GL_ARCHFLAG - flags for VAX architecture differences
/*
/*-


aggregate ARCDEF  union prefix ARC$;
    ARCDEF_BITS structure;
        FILL_1 bitfield length 4 fill prefix ARCDEF tag $$;/*
        CHAR_EMUL bitfield mask;                        /* Char Str Ins Emul
        DCML_EMUL bitfield mask;                        /* Decimal String Emul
        EDPC_EMUL bitfield mask;                        /* EDITPC Instr Emul
        CRC_EMUL bitfield mask;                         /* CRC Instr Emul
        DFLT_EMUL bitfield mask;                        /* D-flt Data Type Emul
        FFLT_EMUL bitfield mask;                        /* F-flt Data Type Emul
        GFLT_EMUL bitfield mask;                        /* G-flt Data Type Emul
        HFLT_EMUL bitfield mask;                        /* H-flt Data Type Emul
        FILL_2 bitfield length 20 fill prefix ARCDEF tag $$;/*
    end ARCDEF_BITS;
end ARCDEF;

end_module $ARCDEF;
```

```
module $BBSDEF;
/**
/*
/* Structure of message from disk ACP to bad block scan utility.
/*
/*-



aggregate BBSDEF structure prefix BBS$;
    MSGTYPE byte unsigned;                          /* message type code (MSG$C_SCANBAD)
    FILL_1 byte dimension 3 fill prefix BBSDEF tag $$;  /* unused
    SEQUENCE word unsigned;                         /* message sequence number
    FILL_2 word fill prefix BBSDEF tag $$;          /* unused
    UCB longword unsigned;                          /* UCB address of device
    FID word unsigned dimension 3;                  /* file ID of file
    constant "LENGTH" equals . prefix BBS$ tag K;
    constant "LENGTH" equals . prefix BBS$ tag C;

end BBSDEF;

end_module $BBSDEF;
```

```
module $BIICDEF;
/*+
/* BI Interface Chip Register Offset Definitions
/*-

aggregate BIICDEF structure prefix BIIC$;

/*+
/* BI Required Registers
/*-

    DTREG_OVERLAY union fill;
        DTREG longword unsigned;                /*Device Type Register
        DEVTYPE_FIELD_OVERLAY union fill;
            DEVTYPE word unsigned;              /* Device Type Field
            DEVTYPE_BITS structure fill;
                FILL_1  bitfield length 8 fill prefix BIICDEF tag $$;
                                                /* Lo order devtype bits
                MEMNODE bitfield length 7;      /* If zero, then memory
                NONDEC  bitfield mask;          /* If set, non-DEC node
                REVCODE word unsigned;          /* Revision code
            end DEVTYPE_BITS;
        end DEVTYPE_FIELD_OVERLAY;
    end DTREG_OVERLAY;

    BICSR_OVERLAY union fill;
        BICSR longword unsigned;                /*BI Control/Status Register
        BICSR_BITS structure fill;
            NODE_ID  bitfield length 4;         /* Node ID
            ARBCNTL  bitfield length 2;         /* Arbitration Control
            SEIE     bitfield mask;             /* Soft Error interrupt enable
            HEIE     bitfield mask;             /* Hard Error interrupt enable
            UWP      bitfield mask;             /* Unlock Write Pending
            FILL_2   bitfield length 1 fill prefix BIICDEF tag $$;
            SST      bitfield mask;             /* Start Self test
            STS      bitfield mask;             /* Self test Status
            BROKE    bitfield mask;             /* Broke bit
            INIT     bitfield mask;             /* Init bit
            SES      bitfield mask;             /* Soft error summary
            HES      bitfield mask;             /* Hard error summary
            BIICTYPE bitfield length 8;         /* BIIC type
            BIICREVN bitfield length 8;         /* BIIC Revision Number
        end BICSR_BITS;
    end BICSR_OVERLAY;

    BER_OVERLAY union fill;
        BER longword unsigned;                  /*Bus Error Register
        BER_BITS structure fill;
            NPE      bitfield mask;             /* Null Bus Parity Error
            CRD      bitfield mask;             /* Corrected Read Data
            IPE      bitfield mask;             /* ID Parity Error
            UPEN     bitfield mask;             /* User Parity Enabled
            FILL_3   bitfield length 12 fill prefix BIICDEF tag $$;
            ICE      bitfield mask;             /* Illegal Confirmation Error
            NEX      bitfield mask;             /* Non-existent Address
            BTO      bitfield mask;             /* Bus Timeout
```

```
            STO     bitfield mask;              /* Stall Timeout
            RTO     bitfield mask;              /* Retry Timeout
            RDS     bitfield mask;              /* Read Data Substitute
            SPE     bitfield mask;              /* Slave Parity Error
            CPE     bitfield mask;              /* Command Parity Error
            IVE     bitfield mask;              /* IDENT Vector Error
            TDF     bitfield mask;              /* Transmitter During Fault
            ISE     bitfield mask;              /* Interlock Sequence Error
            MPE     bitfield mask;              /* Master Parity Error
            CTE     bitfield mask;              /* Control Transmit Error
            MTCE    bitfield mask;              /* Master Transmit Check Error
            NMR     bitfield mask;              /* No Ack to Multi-Responder Command
        end BER_BITS;
    end BER_OVERLAY;

    EICR_OVERLAY union fill;
        EICR longword unsigned;                 /*Error Interrupt Control Register
        EICR_BITS structure fill;
            FILL_4   bitfield length 2 fill prefix BIICDEF tag $$;
            EIVECTOR bitfield length 12;        /* Vector
            FILL_5   bitfield length 2 fill prefix BIICDEF tag $$;
            LEVEL    bitfield length 4;         /* Interrupt Level
            EIFORCE  bitfield mask;             /* Force
            EISENT   bitfield mask;             /* INTR command sent
            FILL_6   bitfield length 1 fill prefix BIICDEF tag $$;
            EIINTC   bitfield mask;             /* Interrupt Complete
            EIINTAB  bitfield mask;             /* Interrupt Abort
        end EICR_BITS;
    end EICR_OVERLAY;

    IDR longword unsigned;                      /* Interrupt Destination
                                                /*  decoded ID in Lo order
/**
/* BIIC Specific Device Registers
/*-

    IPIMR longword unsigned;                    /* IP Interrupt Mask
                                                /*  decoded ID in Hi order

    IPIDR longword unsigned;                    /* IP Interrupt Destination
                                                /*  decoded ID in Lo order

    IPISR longword unsigned;                    /* IP Interrupt Source
                                                /*  decoded ID in Hi order

                                                /*Note: following two
                                                /* registers have lo order
                                                /* 18 bits MBZ. This means
                                                /* memories are multiples
                                                /* of 256KB.
    SAR   longword unsigned;                    /* Starting Address Register
    EAR   longword unsigned;                    /* Ending Address Register

    BCICR_OVERLAY union fill;
        BCICR longword unsigned;                /*BCI Control Register
        BCICR_BITS structure fill;
```

```
            FILL_7   bitfield length 3 fill prefix BIICDEF tag $$;
            RTOEVEN  bitfield mask;              /* RTO EV Enable
            PNXTEN   bitfield mask;              /* Pipeline NXT Enable
            IPINTREN bitfield mask;              /* IP Interrupt Enable
            INTREN   bitfield mask;              /* Interrupt Enable
            BICSREN  bitfield mask;              /* BIIC CSR Space Enable
            UCSREN   bitfield mask;              /* User CSR Space Enable
            WINVALEN bitfield mask;              /* Write Invalidate Enable
            INVALEN  bitfield mask;              /* INVAL Enable
            IDENTEN  bitfield mask;              /* IDENT Enable
            RESEN    bitfield mask;              /* Reserved Enable
            STOPEN   bitfield mask;              /* STOP Enable
            BDCSTEN  bitfield mask;              /* Broadcast Enable
            MSEN     bitfield mask;              /* Multicast Space Enable
            IPINTRF  bitfield mask;              /* IP Interrupt Force
            BURSTEN  bitfield mask;              /* Burst Enable
        end BCICR_BITS;
    end BCICR_OVERLAY;

    WSR_OVERLAY union fill;
        WSR longword unsigned;                   /*Write Status Register
        WSR_BITS structure fill;
            FILL_8   bitfield length 28 fill prefix BIICDEF tag $$;
            GPR0     bitfield mask;              /* These bits indicate
            GPR1     bitfield mask;              /*  that the corresponding
            GPR2     bitfield mask;              /*  General Purpose Register
            GPR3     bitfield mask;              /*  has been written to.
        end WSR_BITS;
    end WSR_OVERLAY;

    IPISTPF_OVERLAY union fill;
        IPISTPF longword unsigned;               /*IPINTR/STOP Force CMD Reg
        IPISTPF_BITS structure fill;
            FILL_9   bitfield length 11 fill prefix BIICDEF tag $$;
            MIDEN    bitfield mask;              /* Determines whether Master ID
                                                 /*  transmitted on BI D<31:16>.
            CMD      bitfield length 4;          /* Command (IPINTR or STOP).
        end IPISTPF_BITS;
    end IPISTPF_OVERLAY;

    FILL_10 longword fill;                       /*Unused
    FILL_11 longword fill;                       /*Unused
    FILL_12 longword fill;                       /*Unused


    UICR_OVERLAY union fill;
        UICR longword unsigned;                  /*UserInterrupt Control Register
        UICR_BITS structure fill;
            FILL_13  bitfield length 2 fill prefix BIICDEF tag $$;
            UIVECTOR bitfield length 12;         /* Vector
            FILL_14  bitfield length 1 fill prefix BIICDEF tag $$;
            EXVECTOR bitfield mask;              /* External Vector
            UIFORCE  bitfield length 4;          /* Force (1 for each level)
            UISENT   bitfield length 4;          /* INTR command sent(1 for each level)
            UIINTC   bitfield length 4;          /* Interrupt Complete(1 for each level)
            UIINTAB  bitfield length 4;          /* Interrupt Abort(1 for each level)
```

```
        end UICR_BITS;
    end UICR_OVERLAY;

    FILL_15 byte dimension 172 fill prefix BIICDEF tag $$;

/*+
/* BIIC General Purpose Device Registers
/*-

    GPR0 longword unsigned;                 /*General Purpose Register 0
    GPR1 longword unsigned;                 /*General Purpose Register 1
    GPR2 longword unsigned;                 /*General Purpose Register 2
    GPR3 longword unsigned;                 /*General Purpose Register 3

end BIICDEF;

end_module $BIICDEF;
```

```
module $BIMEMDEF;

/*+
/* BI Memory Node Registers
/*-

aggregate BIMEMDEF structure prefix BIMEM$;

    FILL_15 byte dimension 256 fill prefix BIMEMDEF tag $$;

    CSR1_OVERLAY union fill;
        CSR1 longword unsigned;                         /*CSR 1
        CSR1_OVERLAY union fill;
            CSR1_BITS structure fill;
                DIAGBTS bitfield length 7;              /* Used during ECC diag cycles
                FILL_1  bitfield length 1 fill prefix BIMEMDEF tag $$;
                INTLV   bitfield mask;                  /* 1=> internally interleaved
                CNTLERR bitfield mask;                  /* Controller error
                MWRITER bitfield mask;                  /* RDS on masked write
                FILL_2  bitfield length 1 fill prefix BIMEMDEF tag $$;
                BROKE   bitfield mask;                  /* Broke bit
                INTLK   bitfield mask;                  /* Interlock flag
                MEMVAL  bitfield mask;                  /* Memory contents valid
                INHCRD  bitfield mask;                  /* Inhib. CRD reporting
                RAMTYPE bitfield length 2;              /* 00=>64Ks, 01=>256Ks
                MEMSIZE bitfield length 11;             /* Size in 256KB increments
                ECCDIS  bitfield mask;                  /* Used with following bit
                ECCDIAG bitfield mask;                  /*
                ERRSUM  bitfield mask;                  /* Error summary(includes CSR2)
            end CSR1_BITS;
        end CSR1_OVERLAY;
    end CSR1_OVERLAY;

    CSR2_OVERLAY union fill;
        CSR2 longword unsigned;                         /*CSR 2
        CSR2_OVERLAY union fill;
            CSR2_BITS structure fill;
                ERRSYND bitfield length 7;              /* Error syndrome
                FILL_3  bitfield length 1 fill prefix BIMEMDEF tag $$;
                INTLVAD bitfield mask;                  /* Interleave Address
                ERRADDR bitfield length 15;             /* Internal addr of error
                FILL_4  bitfield length 4 fill prefix BIMEMDEF tag $$;
                ADRSERR bitfield mask;                  /* Internal address parity error
                CRDLOGR bitfield mask;                  /* CRD Error Log REQ
                HIERATE bitfield mask;                  /* Hi Error Rate
                RDSLOGR bitfield mask;                  /* RDS Error Log REQ
            end CSR2_BITS;
        end CSR2_OVERLAY;
    end CSR2_OVERLAY;

end BIMEMDEF;
end_module $BIMEMDEF;
```

```
module $BOODEF;
/**
/* BOO - Boot Control Block
/*
/* A boot control block is produced by SYSBOOT and placed in non-paged
/* pool.  It is pointed to by the cell EXE$GL_BOOTCB and contains
/* the mapping information for SYS.EXE, SYSDUMP.DMP, SYSPARAM portion
/* of SYS.EXE, and non-resident BUGCHECK code.
/*-

aggregate BOODEF structure prefix BOO$;
    CHECKSUM longword unsigned;                 /* Checksum
    PARAM_MAP longword unsigned;                /* Address of map for SYSPARAM
    SIZE word unsigned;                         /* Size of fixed portion of BOOTCB
    TYPE byte unsigned;                         /* Type of control block
    SUBTYP byte unsigned;                       /* Sub-type
    SYS_VBN longword unsigned;                  /* SYS.EXE starting VBN
    SYS_SIZE longword unsigned;                 /* SYS.EXE size in blocks
                                                /* from starting VBN to end of executable image
    SYS_MAP longword unsigned;                  /* Adr of map for SYS.EXE
    DMP_VBN longword unsigned;                  /* Starting VBN for dump file
    DMP_SIZE longword unsigned;                 /* Size in blocks of dump file
                                                /* from starting VBN to end of file
    DMP_MAP longword unsigned;                  /* Adr of map for SYSDUMP.DMP
    BUG_MAP longword unsigned;                  /* Adr of map for non-resident BUGCHECK code
    constant "LENGTH" equals . prefix BOO$ tag K;
    constant "LENGTH" equals . prefix BOO$ tag C;
end BOODEF;

end_module $BOODEF;
```

```
module $BQODEF;
/*+
/*
/* Offsets into the IO vector of the BOOT driver.
/*
/*-



aggregate BQODEF structure prefix BQO$;
    QIO longword unsigned;                  /* QIO entry
    MAP longword unsigned;                  /* Mapping entry
    SELECT longword unsigned;               /* Selection entry
    DRIVRNAME longword unsigned;            /* Offset to driver name
    VERSION word unsigned;                  /* Version number of VMB
    VERCHECK word unsigned;                 /* Check field
    RESELECT longword unsigned;             /* Reselection entry
    MOVE longword unsigned;                 /* Move driver entry
    UNIT_INIT longword unsigned;            /* Unit initialization entry
    AUXDRNAME longword unsigned;            /* Offset to auxiliary driver name
    UMR_DIS longword unsigned;              /* UNIBUS Map Registers to disable
    UCODE longword unsigned;                /* Absolute address of booting microcode
    UNIT_DISC longword unsigned;            /* Unit disconnecting entry
    DEVNAME longword unsigned;              /* Offset to boot device name
    UMR_TMPL longword unsigned;             /* UNIBUS map register template
    UMR_DP byte unsigned;                   /* UNIBUS map register data path
    CPUTYPE byte unsigned;                  /* Cpu type from SID
    CPUDATA longword unsigned;              /* Cpu data from SID
    TENUSEC longword unsigned;              /* TIMEDWAIT loop delay counter
    UBDELAY longword unsigned;              /* TIMEDWAIT loop delay counter
end BQODEF;

end_module $BQODEF;

MODULE  $BRKTDEF;           /*
/* +
/*
/*  Structure of breakthru message descriptor block.
/*
/* -

AGGREGATE $BRKTDEF STRUCTURE;

BRKTHRU_OVERLAY UNION;          /* set up overlay

BRKTHRU_1 STRUCTURE PREFIX BRK$;
/*
/*  Common Storage
/*
    PRIVS       quadword unsigned;          /*  privs to set
    SIZE        word unsigned;              /*  block size
    OUTCNT      word unsigned;              /*  outstanding I/O count
    DEVNAME     character length 16;        /*  device name for $ASSIGN
    PCB         longword unsigned;          /*  Address of PCB
    IOSB        longword unsigned;          /*  Address of return IOSB
    ASTADR      longword unsigned;          /*  Address of AST routine
```

```
    ASTPRM      longword unsigned;          /*  Value of AST parameter
    TIMEOUT     quadword unsigned;          /*  Timeout value
    CARCON      longword unsigned;          /*  carriage control
    FLAGS       longword unsigned;          /*  flags
    SENDNAME    character length 16;        /*  username/terminal name
    SENDTYPE    word unsigned;              /*  send descriptor type
    SECONDS     word unsigned;              /*  Timeout in seconds
    REQID       longword unsigned;          /*  send requestor ID
/*
/*  miscellaneous context
/*
    PIDCTX      longword unsigned;          /*  Last PID in user search
    UCBCTX      longword unsigned;          /*  Last UCB in TTY search
    DDBCTX      longword unsigned;          /*  Last DDB in TTY search
    QIOCTX      longword unsigned;          /*  per QIO context address

    EFN         word unsigned;              /*  user event flag *BYTE***?
    STS_OVERLAY UNION FILL;
        STS     byte unsigned;              /*  status flags
        STS_BITS STRUCTURE FILL;            /* status flags bit definition
            LOCKED      bitfield mask;      /* I/O dataabse locked
            DONE        bitfield mask;      /* done looking for terminals
            CHKPRIV     bitfield mask;      /* check privilege
        END STS_BITS;
    END STS_OVERLAY;
    PRVMODE     byte unsigned;              /*  previous mode

    SCRMSGLEN   longword unsigned;          /*  screen message length
    SCRMSG      longword unsigned;          /*  screen message address
/*
/*  status block
/*
    STATUS      word unsigned;              /*  status
    SUCCESSCNT  word unsigned;              /*  Success count
    TIMEOUTCNT  word unsigned;              /*  Timeout count
    REFUSEDCNT  word unsigned;              /*  Refused count
/*
/*  start of mailbox message
/*
    TRMMSG      word unsigned;              /*  mailbox message code
    TRMUNIT     word unsigned;              /*  tty unit number
    TRMNAME     character length 16;        /*  terminal name
/*
/*  real message starts here
/*
    MSGLEN      word unsigned;              /*  length of msgbuf
    MSGBUF      character length 0;         /*  start of message
/*
/* Length
/*
    CONSTANT LENGTH EQUALS . TAG C;

END BRKTHRU_1;

BRKTHRU_2 STRUCTURE PREFIX brk2$;
```

```
/*
/*  Per QIO storage
/*

    COMMON      longword unsigned;          /*  address of common area
    IOSB        quadword unsigned;          /*  iosb for QIO
    CHAN        word unsigned;              /*  channel

/*
/* Length of Per QIO context
/*
    constant LENGTH equals . tag C;

END BRKTHRU_2;
END BRKTHRU_OVERLAY;

end $BRKTDEF;
end_module $BRKTDEF;
```

```
module $BTDDEF;
/*+
/*
/* Boot device codes
/*
/*-


                                                        /* "$K_" added, 8/30/79, CHP
constant MB      equals 0  prefix BTD tag $K;           /* Massbus device
                                                        /* Types 1-31. reserved for unibus
                                                        /* devices
constant DM      equals 1  prefix BTD tag $K;           /* RK06/7
constant DL      equals 2  prefix BTD tag $K;           /* RL02
constant DQ      equals 3  prefix BTD tag $K;           /* RB02/RB80
constant QNA     equals 7  prefix BTD tag $K;           /* QNA
constant PROM    equals 8  prefix BTD tag $K;           /* PROM
constant UDA     equals 17  prefix BTD tag $K;          /* UDA
                                                        /* End of unibus devices
constant HSCCI   equals 32  prefix BTD tag $K;          /* HSC on CI
constant CONSOLE         equals 64  prefix BTD tag $K;  /* Console block storage device

end_module $BTDDEF;
```

```
module $BUADEF;
/++
/* BI Bus UNIBUS Adapter Register Offset Definitions
/*-

aggregate BUADEF structure prefix BUA$;
    FILL_1 byte dimension 240 fill prefix BUADEF tag $$; /* Value is  F0 (Hex)

    GPR0_OVERLAY union fill;                    /* BIIC GPR0 used by BUA.
        GPR0 longword unsigned;
        GPR0FIELDS structure fill;
            UBPUP    bitfield mask;             /* UNIBUS Power Up (RO)
            FILL_2   bitfield length 15 fill prefix BUADEF tag $$;/*SPARE
            IEN_COPY bitfield length 16;        /* Internal Error Number copied
                                                /*  here from BUACSR.
        end GPR0FIELDS;
    end GPR0_OVERLAY;

    FILL_3 byte dimension 1580 fill prefix BUADEF tag $$;/* Cummulative Value
                                                        /*  is 720 (Hex)
/* This register holds the error summaries and error interrupt enable for BUA.

    CSR_OVERLAY union fill;
        CSR longword unsigned;          /* Control and Status Register
        CSR_BITS structure fill;
            IEN    bitfield length 8;   /* Instruction Error Number (RO)
                                        /*  Self Test failure code

            FILL_4 bitfield length 8 fill prefix BUADEF tag $$;/* Reserved Field

            REGDMP bitfield mask;       /* uDiagnostic Register Dump bit.  (WO)
                                        /*  When 1 is written, causes uEngine to
                                        /*  dump stored internal registers.
            UPI    bitfield mask;       /* UNIBUS Power Initialization bit. (WO)
                                        /*  When 1 written, causes power up init
                                        /*  on UNIBUS.

            FILL_5 bitfield length 2 fill prefix BUADEF tag $$;/* Reserved Field

            EIE    bitfield mask;       /* BUA Error Interrupt Enable (R/W)

            FILL_6 bitfield length 3 fill prefix BUADEF tag $$;/* Reserved Field

            BADBDP bitfield mask;       /* Bit set if BDP 6 or 7 selected (W1C).
            IMR    bitfield mask;       /* Invalid Map Register (W1C)
            UIE    bitfield mask;       /* Bit set if DATO(B) does not follow
                                        /*  DATIP on UNIBUS (W1C)
            USSTO  bitfield mask;       /* UNIBUS SSYNC timeout (W1C)
            BIF    bitfield mask;       /* UNIBUS to BI failure (W1C)

            FILL_7 bitfield length 2 fill prefix BUADEF tag $$;/* Reserved Field

            ERR    bitfield mask;       /* Logical OR of error bits in CSR (RO)
        end CSR_BITS;
    end CSR_OVERLAY;
```

```
/* BUA Vector Offset Register - BITS [13:09] of the VOR register are
/*  concatenated with the incoming UNIBUS vector to form a 14 bit BI vector.

    VOR_OVERLAY union fill;
        VOR longword unsigned;          /* Vector Offset Register
        VOR_BITS structure fill;

            FILL_8 bitfield length 9 fill prefix BUADEF tag $$;/* Reserved Field

            VECOFF bitfield length 5;   /* Vector Offset (R/W)
        end VOR_BITS;
    end VOR_OVERLAY;

/* Failed UNIBUS Address Register (FUBAR)

    FUBAR_OVERLAY union fill;
        FUBAR longword unsigned;        /* Failed UNIBUS Address Register
        FUBAR_BITS structure fill;
            FUBAR_ADR bitfield length 16;/* Failed UNIBUS Address (Hi 16 bits)
        end FUBAR_BITS;
    end FUBAR_OVERLAY;

    FILL_9 byte dimension 4 fill prefix BUADEF tag $$;

    BDP1_OVERLAY union fill;
        BDP1 longword unsigned;                         /* BDP1
        BDPFIELDS structure fill;
            STATUS bitfield length 16;                  /* Bit for each byte
            ADDR   bitfield length 16;                  /* UNIBUS addr of octaword
        end BDPFIELDS;
    end BDP1_OVERLAY;
        BDP2 longword unsigned;                         /* BDP2
        BDP3 longword unsigned;                         /* BDP3
        BDP4 longword unsigned;                         /* BDP4
        BDP5 longword unsigned;                         /* BDP5

    FILL_10 byte dimension 12 fill prefix BUADEF tag $$;/* Cummulative Value

    DPCSR0_OVERLAY union fill;
        DPCSR0 longword unsigned;                       /* Datapath 0 CSR
        DPCSR_BIT structure fill;
            PURGE bitfield mask;                        /* Purge (WO) bit
            FILL_11 bitfield length 20 fill prefix BUADEF tag $$;/*SPARE
            DPSEL bitfield length 3;                    /* Data Path #
        end DPCSR_BIT;
    end DPCSR0_OVERLAY;

        DPCSR1 longword unsigned;                       /* Datapath 1 CSR
        DPCSR2 longword unsigned;                       /* Datapath 2 CSR
        DPCSR3 longword unsigned;                       /* Datapath 3 CSR
        DPCSR4 longword unsigned;                       /* Datapath 4 CSR
        DPCSR5 longword unsigned;                       /* Datapath 5 CSR

    FILL_12 byte dimension 8 fill prefix BUADEF tag $$;/* Cummulative Value
```

```
    FILL_13 byte dimension 144 fill prefix BUADEF tag $$;

    MAP_OVERLAY union fill;
        MAP longword unsigned dimension 496;    /* Map Registers
        MAP_BITS structure fill;
            MAP_ADDR bitfield length 21;        /* PFN
            MAP_DPD bitfield length 3;          /* Datapath Designator

            FILL_14 bitfield length 1 fill prefix BUADEF tag $$;/* Reserved field

            MAP_BO  bitfield mask;              /* Byte Offset
            LWAEN   bitfield mask;              /* Long Word Access Enable

            FILL_15 bitfield length 3 fill prefix BUADEF tag $$;/* Reserved field

            PPIE    bitfield mask;              /* Reserved for use on BUA's
                                                /*  with PDP-11 on UNIBUS.
            MAP_VALID bitfield mask;            /* Map Register Valid
        end MAP_BITS;
        constant MAXDP  equals 5  prefix BUA tag $C;    /*MAXIMUM DATAPATH !

    end MAP_OVERLAY;
end BUADEF;

end_module $BUADEF;
```

```
module $CADEF;

/**
/* CONDITIONAL ASSEMBLY PARAMETER DEFINITIONS
/*
/*        A NONZERO PARAMETER VALUE INDICATES PRESENCE OF THE FEATURE.
/*        A ZERO PARAMETER VALUE INDICATES ABSENCE OF THE FEATURE
/*
/*        ALL PARAMETERS MUST BE DEFINED
/*-

constant SIMULATOR      equals 1  prefix CA tag $;      /*INCLUDE SIMULATOR SUPPORT CODE
constant MEASURE        equals 2  prefix CA tag $;      /*INCLUDE PERFORMANCE MEASUREMENT HOOKS
constant MEASURE_IOT    equals 4  prefix CA tag $;      /*INCLUDE I/O TRANSACTION DATA COLLECTION

end_module $CADEF;
```

```
module $CANDEF;

/**
/* CAN - DEFINE DRIVER CANCEL ROUTINE REASON CODES
/*
/* THESE CODES ARE PASSED TO THE CANCEL ROUTINE OF A DRIVER SO THAT
/* THE ROUTINE CAN DISTINGUISH BETWEEN CALLS FROM $DASSGN AND $CANCEL.
/*
/*-


constant(
      CANCEL                                        /*CANCEL INVOKED DUE TO $CANCEL SERVICE
    , DASSGN                                        /*CANCEL INVOKED DUE TO $DASSGN SERVICE
    , AMBXDGN                                       /*CANCEL INVOKED DUE TO MB DISASSOCIATION
    ) equals 0 increment 1 prefix CAN tag $C;

end_module $CANDEF;
```

```
module $CDRPDEF;
/++
/* CDRP - CLASS DRIVER I/O REQUEST PACKET
/*
/* This structure contains within it, at negative offsets, a full IRP.
/* For this reason all IRP fields must be at the same relative offsets
/* as the corresponding fields in the IRP.
/*
/*-



aggregate CDRPDEF structure prefix CDRP$ origin FQFL;
    IOQFL longword unsigned;                    /*I/O QUEUE FORWARD LINK
    IOQBL longword unsigned;                    /*I/O QUEUE BACKWARD LINK
    IRP_SIZE word unsigned;                     /*SIZE OF IRP IN BYTES
    IRP_TYPE byte unsigned;                     /*STRUCTURE TYPE FOR IRP
    RMOD byte unsigned;                         /*ACCESS MODE OF REQUEST
{    RMOD subfields [defined in IRPDEF]
{       bitfield MODE length 2;                 /* MODE SUBFIELD
    PID longword unsigned;                      /*PROCESS ID OF REQUESTING PROCESS
    AST longword unsigned;                      /*ADDRESS OF AST ROUTINE
    ASTPRM longword unsigned;                   /*AST PARAMETER
    WIND longword unsigned;                     /*ADDRESS OF WINDOW BLOCK
    UCB longword unsigned;                      /*ADDRESS OF DEVICE UCB
    FUNC word unsigned;                         /*I/O FUNCTION CODE AND MODIFIERS
{    FUNC subfields [defined in IRPDEF]
{       bitfield FCODE length 6;                /* FUNCTION CODE FIELD
{       bitfield FMOD length 10;                /* FUNCTION MODIFIER FIELD
    EFN byte unsigned;                          /*EVENT FLAG NUMBER AND EVENT GROUP
    PRI byte unsigned;                          /*BASE PRIORITY OF REQUESTING PROCESS
    IOSB longword unsigned;                     /*ADDRESS OF I/O STATUS DOUBLE LONGWORD
    CHAN word unsigned;                         /*PROCESS I/O CHANNEL NUMBER
    STS word unsigned;                          /*REQUEST STATUS
{    STS subfields [defined in IRPDEF]
{       bitfield BUFIO;                         /* BUFFERED I/O FLAG   /*THESE BITS
{       bitfield FUNC;                          /* 1=>READ FUNCTION    /*MUST BE ADJACENT
{       bitfield PAGIO;                         /* PAGING I/O FLAG     /*AND IN ORDER
{       bitfield COMPLX;                        /* COMPLEX BUFFERED I/O
{       bitfield VIRTUAL;                       /* VIRTUAL I/O FUNCTION
{       bitfield CHAINED;                       /* CHAINED BUFFERED I/O OPERATION
{       bitfield SWAPIO;                        /* SWAP I/O OPERATION
{       bitfield DIAGBUF;                       /* DIAGNOSTIC BUFFER ALLOCATED
{       bitfield PHYSIO;                        /* PHYSICAL I/O
{       bitfield TERMIO;                        /* TERMINAL I/O (FOR SELECTING PRIORITY INC)
{       bitfield MBXIO;                         /* MAILBOX BUFFERED READ
{       bitfield EXTEND;                        /* AN IRPE IS LINKED TO THIS IRP
{       bitfield FILACP;                        /* FILE ACP I/O (BOTH DIOCNT AND BIOCNT)
{       bitfield MVIRP;                         /* MOUNT VERIFICATION IRP
{       bitfield KEY;                           /* ENCRYPTION KEY
    SVAPTE longword unsigned;                   /*SYSTEM VIRTUAL ADDRESS OF FIRST PTE
    BOFF word unsigned;                         /*BYTE OFFSET IN FIRST PAGE
    BCNT_OVERLAY union fill;
        BCNT longword unsigned;                 /*BYTE COUNT OF TRANSFER
        BCNT word unsigned;                     /* OLD WORD DEFINITION FOR COMPATIBILITY
    end BCNT_OVERLAY;
```

```
    FILL_1 word fill prefix CDRPDEF tag $$;             /* ROUND UP TO NEXT LONGWORD
    IOSTT_OVERLAY union fill;
        IOST1 longword unsigned;                        /*FIRST I/O STATUS LONGWORD (FOR I/O POST)
        MEDIA longword unsigned;                        /*MEDIA ADDRESS
    end IOST1_OVERLAY;
    IOST2_OVERLAY union fill;
        IOST2 longword unsigned;                        /*SECOND I/O STATUS LONGWORD
        TT_TERM_OVERLAY union fill;
            TT_TERM longword unsigned;                  /*ADDRESS OF READ TERMINATORS MASK
            CARCON byte unsigned;                       /*CARRIAGE CONTROL
        end TT_TERM_OVERLAY;
    end IOST2_OVERLAY;
    NT_PRVMSK_OVERLAY union fill;
        NT_PRVMSK quadword unsigned;                    /* PRIVILEGE MASK FOR DECNET
        NT_PRVMSK_FIELDS structure fill;
            ABCNT_OVERLAY union fill;
                ABCNT longword unsigned;                /* ACCUMULATED BYTES TRANSFERED
                ABCNT_OVERLAY1 union;
                    ABCNT word unsigned;                /* OLD WORD DEFINITION FOR COMPATIBILITY
                    TT_PRMPT word unsigned;             /* PROMPT SIZE
                end ABCNT_OVERLAY1;
            end ABCNT_OVERLAY;
            OBCNT_OVERLAY union fill;
                OBCNT longword unsigned;                /* ORIGINAL TRANSFER BYTE COUNT
                OBCNT word unsigned;                    /* OLD WORD DEFINITION FOR COMPATIBILITY
            end OBCNT_OVERLAY;
        end NT_PRVMSK_FIELDS;
    end NT_PRVMSK_OVERLAY;
    SEGVBN_OVERLAY union fill;
        SEGVBN longword unsigned;                       /*VIRTUAL BLOCK NUMBER OF CURRENT SEGMENT
        JNL_SEQNO longword unsigned;                    /* SEQUENCE NUMBER IN JOURNAL
    end SEGVBN_OVERLAY;
    DIAGBUF longword unsigned;                          /* DIAGNOSTIC BUFFER ADDRESS
    SEQNUM longword unsigned;                           /* SEQUENCE NUMBER
    EXTEND longword unsigned;                           /* ADDRESS OF IRPE
    ARB longword unsigned;                              /* ACCESS RIGHTS BLOCK ADDRESS
    KEYDESC longword unsigned;                          /* ADDRESS OF ENCRYPTION KEY DESCRIPTOR
/*
/*
    constant CDRPBASE equals . prefix CDRP$ tag K;
    constant CDRPBASE equals . prefix CDRP$ tag C;
    FQFL longword unsigned;                             /* Fork Queue FLINK
    FQBL longword unsigned;                             /* Fork Queue Blink
    CDRPSIZE word unsigned;                             /* Size field for positive section only
    CD_TYPE byte unsigned;                              /* Type, always of inte est
    FIPL byte unsigned;                                 /* Fork IPL
    FPC longword unsigned;                              /* Fork PC
    FR3 longword unsigned;                              /* Fork R3
    FR4 longword unsigned;                              /* Fork R4
    SAVD_RTN longword unsigned;                         /* Saved return address from level 1 JSB
    MSG_BUF longword unsigned;                          /* Address of allocated MSCP buffer
    RSPID longword unsigned;                            /* Allocated Request ID
    CDT longword unsigned;                              /* Address of Connection Descriptor Table
    RWCPTR longword unsigned;                           /* RWAITCNT pointer
    constant "LENGTH" equals . prefix CDRP$ tag K;
    constant "LENGTH" equals . prefix CDRP$ tag C;
```

```
/*      CDRP extensions

    CDRP_EXTENSIONS union fill;

/*      Block Transfer Extension

        BLK_XFER_EXTENSION structure fill;
            LBUFH_AD longword unsigned;             /* Local BUFfer Handle ADress
            LBOFF longword unsigned;                /* Local Byte OFFset
            RBUFH_AD longword unsigned;             /* Remote BUFfer Handle ADress
            RBOFF longword unsigned;                /* Remote Byte OFFset
            XCT_LEN longword unsigned;              /* Transfer length in bytes
            constant BT_LEN equals .;
            constant BT_LEN equals . tag C;
        end BLK_XFER_EXTENSION;

/*      Class Driver Extension

        CLS_DRV_EXTENSION structure fill;
            FILL_3 longword fill;                   {  Skip local buffer handle address (above)
            LBUFHNDL character length 12;           /* Local buffer handle
            UBARSRCE longword unsigned;             /* UNIBUS mapping resources allocated
            DUTUFLAGS structure longword unsigned;  /* Class driver status flags:
                CAND bitfield mask;                 /*  canceled I/O request
                CANIO bitfield mask;                /*  cancel operation I/O request
                ERLIP bitfield mask;                /*  error log in progress
                PERM bitfield mask;                 /*  CDDB permanent IRP/CDRP
                HIRT bitfield mask;                 /*  HIRT permanent IRP/CDRP
                DENSCK bitfield mask;               /*  Tape density check required
                filler bitfield length 2 fill;      {   Byte align IVCMD
                IVCMD bitfield mask;                /*  Invalid command processing in progress
            end DUTUFLAGS;
            DUTUCNTR word unsigned;                 /* General purpose counter
            ENDMSGSIZ word unsigned;                /* Size of most recent MSCP end message
            constant CD_LEN equals .;
            constant CD_LEN equals . tag C;
        end CLS_DRV_EXTENSION;

/*      Connection management extension

        CON_MGT_EXTENSION structure fill;
            CNX_WORK_AREA  union fill;
                CNX_CLIENT_DATA structure fill;
                    VAL1 longword unsigned;         /* data value 1
                    VAL2 longword unsigned;         /* data value 2
                    VAL3 longword unsigned;         /* data value 3
                    VAL4 longword unsigned;         /* data value 4
                    VAL5 longword unsigned;         /* data value 5
                    VAL6 longword unsigned;         /* data value 6
                    VAL7 longword unsigned;         /* data value 7
                    VAL8 longword unsigned;         /* data value 8
                end CNX_CLIENT_DATA;
                CNX_BLOCK_XFER structure fill;
                    FILL_LBUFH_AD longword fill;    { filler for CDRP$L_LBUFH_AD
                    FILL_VAL longword dimension 4;  { filler for VAL2 through VAL5
```

```
                CNXSVAPTE longword unsigned;        /* Block SVAPTE
                CNXBOFF word unsigned;              /* Block buffer offset
                CNXBCNT longword unsigned;          /* Block xfer length
                CNXRMOD byte unsigned;              /* Block access mode
                CLTSTS byte unsigned;               /* A client's status field
            end CNX_BLOCK_XFER;
        end CNX_WORK_AREA;
        MSGBLD longword unsigned;                   /* Address of MSG BUILD routine
        SAVEPC longword unsigned;                   /* Caller's saved PC
        SENDSEQNM word unsigned;                    /* Message sequence number
        CNXSTATE byte unsigned;                     /* CNX message state
            constant (                              /* Possible states:
                  NORMAL                            /* The standard case (particulary no block xfer)
                , REQUESTOR                         /* Block transfer requestor
                , PARTNER                           /* Block transfer partner, active
                , PART_IDLE                         /* Block transfer partner, idle
                , REQ_MAP                           /* Block transfer requestor, waiting for buffer handle
                , PART_MAP                          /* Block transfer partner, waiting for buffer handle
                ) equals 0 increment 1;
        FILL_5 byte fill;
        RETRSPID longword unsigned;                 /* RSPID to return
        VAL9 longword unsigned;                     /* data value 9
        constant CM_LENGTH equals .;
        /*
        /* The following fields are only valid
        /* for long connection manager CDRPs.
        /*
        VAL10 longword unsigned;                    /* data value 10
        constant CM_LONG_LENGTH equals.;
    end CON_MGT_EXTENSTION;

end CDRP_EXTENSIONS;

end CDRPDEF;
end_module $CDRPDEF;
```

```
module $CINDEF;
/++
/*
/* Connect to interrupt definitions for QIO parameters
/*
/*-



aggregate CINDEF  union prefix CIN$;
    CINDEF_BITS structure fill;
        EFN bitfield mask;                              /* Set event flag on interrupt.
        USECAL bitfield mask;                           /* Use CALL interface.
        REPEAT bitfield mask;                           /* Do repeated interrupt service.
        AST bitfield mask;                              /* Queue AST on interrupt.
        INIDEV bitfield mask;                           /* Device initialization to do.
        START bitfield mask;                            /* Start I/O routine.
        ISR bitfield mask;                              /* ISR to execute.
        CANCEL bitfield mask;                           /* Cancel I/O routine.
        FILL_1 bitfield length 8 fill prefix CINDEF tag $$;/* Spare bits.
        EFNUM bitfield mask length 16;                  /* Event flag number.
    end CINDEF_BITS;

end CINDEF;

    aggregate CINDEF1 structure prefix CIN$;
    INIDEV longword unsigned;                           /* Offset to device init routine.
    START longword unsigned;                            /* Offset to start device routine.
    ISR longword unsigned;                              /* Offset to interrupt service routine.
    CANCEL longword unsigned;                           /* Offset to cancel I/O routine.


    end CINDEF1;

aggregate CINDEF2 structure prefix CIN$;
    SPTCOUNT longword unsigned;                         /* Number of SPTs allocated.
    STARTVPN_OVERLAY union fill;
        STARTVPN longword unsigned;                     /* Starting VPN allocated.
        STARTBIT longword unsigned;                     /* Starting bit in bitmap.


    end STARTVPN_OVERLAY;
end CINDEF2;

end_module $CINDEF;
```

```
module $CCBDEF;

/**
/* CCB - CHANNEL CONTROL BLOCK
/*
/* THERE IS ONE CHANNEL CONTROL BLOCK FOR EACH SOFTWARE CHANNEL THAT A
/* PROCESS MAY INITIATE I/O REQUESTS ON. THE NUMBER OF SUCH I/O CHANNELS
/* IS DETERMINED BY THE FIXED NUMBER ASSIGNED TO A PROCESS PLUS ANY
/* ADDITIONAL CHANNELS REQUIRED BY THE IMAGE CURRENTLY BEING EXECUTED
/* BY THE PROCESS.
/*
/* **** WARNING ****
/*        THE CHANNEL CONTROL BLOCK IS ASSUMED TO BE FOUR LONG WORDS
/* THROUGHOUT THE EXEC.  ITS SIZE MAY BE CHANGED BUT ONLY BY POWERS OF 2.
/*-


aggregate CCBDEF structure prefix CCB$;
    UCB longword unsigned;                          /*ADDRESS OF ASSIGNED DEVICE UCB
    WIND longword unsigned;                         /*ADDRESS OF WINDOW BLOCK
    STS_OVERLAY union fill;
        STS byte unsigned;                          /*CHANNEL STATUS
        STS_BITS structure fill;
            AMB bitfield mask;                      /* MAILBOX ASSOCIATED WITH CHANNEL
            IMGTMP bitfield mask;                   /* IMAGE TEMPORARY
            RDCHKDON bitfield mask;                 /* READ PROTECTION CHECK COMPLETED
            WRTCHKDON bitfield mask;                /* WRITE PROTECTION CHECK COMPLETED
            LOGCHKDON bitfield mask;                /* LOGICAL I/O ACCESS CHECK DONE
            PHYCHKDON bitfield mask;                /* PHYSICAL I/O ACCESS CHECK DONE
        end STS_BITS;
    end STS_OVERLAY;
    AMOD byte unsigned;                             /*ACCESS MODE THAT ASSIGNED CHANNEL
    IOC word unsigned;                              /*NUMBER OF OUTSTANDING I/O REQUESTS ON CHANNEL
    DIRP longword unsigned;                         /*DEACCESS I/O REQUEST PACKET ADDRESS
    constant "LENGTH" equals . prefix CCB$ tag K;   /*LENGTH OF CCB
    constant "LENGTH" equals . prefix CCB$ tag C;   /*LENGTH OF CCB
end CCBDEF;

end_module $CCBDEF;
```

```
module $CDDBDEF;
/**
/* CDDB - Class Driver Data Block
/*
/* Auxiliary data block pointed at by the CRB$L_AUXSTRUC of an MSCP speaking
/* intelligent disk or tape controller.  There is one CDDB per such intelligent
/* controller.
/*
/*-


aggregate CDDBDEF structure prefix CDDB$;
    CDRPQFL longword unsigned;                          /*Active CDRP Q FLINK
    CDRPQBL longword unsigned;                          /*Active CDRP Q BLINK
    SIZE word unsigned;                                 /*Size of CDDB in bytes
    TYPE byte unsigned;                                 /*Major structure type for Class Driver
    SUBTYPE byte unsigned;                              /* CDDB structure subtype field
    SYSTEMID byte unsigned dimension 6;                 /*48 bit system ID.
    STATUS_OVERLAY union fill;
        STATUS word unsigned;                           /*Status word
        STATUS_BITS structure fill;
            SNGLSTRM bitfield mask;                     /* Single stream mode after VC crash
            IMPEND bitfield mask;                       /* IMmediate command PENDing
            INITING bitfield mask;                      /* Currently initializing CONNECTION
            RECONNECT bitfield mask;                    /* Currently re-CONNECTING to MSCP server
            RESYNCH bitfield mask;                      /* re_CONNECT initiated by Class Driver
            POLLING bitfield mask;                      /* Polling for units
            ALCLS_SET bitfield mask;                    /* Allocation class has been set
            NOCONN bitfield mask;                       /* CDDB currently has no connection
            RSTRTWAIT bitfield mask;                    /* Waiting to RESTART_NEXT_CDRP
            QUORLOST bitfield mask;                     /* CNXMAN quorum lost processing
            DAPBSY bitfield mask;                       /* DAP CDRP is busy
            "2PBSY" bitfield mask;                      /* Failover fork block is busy
        end STATUS_BITS;
    end STATUS_OVERLAY;
    PDT longwora unsigned;                              /*Port Descriptor Table address
    CRB longword unsigned;                              /*CRB address
    DDB longword unsigned;                              /*DDB address
    CNTRLID_OVERLAY union fill;
        CNTRLID quadword unsigned;                      /*Controller ID returned by MSCP END PACKET
        CNTRLID_FIELDS structure fill;
            FILL_2 byte dimension 6 fill prefix CDDBDEF tag $$;
            CNTRLMDL byte unsigned;                     /* Controller model ! (byte 6 of controller id)
            CNTRLCLS byte unsigned;                     /* Controller class (byte 7 of controller id)
        end CNTRLID_FIELDS;
    end CNTRLID_OVERLAY;
    CNTRLFLGS word unsigned;                            /*Controller flags also returned by END PACKET
    CNTRLTMO word unsigned;                             /*Controller timeout also returned by END PACKET
    OLDRSPID longword unsigned;                         /*RSPID of oldest outstanding MSCP command
    OLDCMDSTS longword unsigned;                        /*Latest MSCP command status for this command
    RSTRTCDRP longword unsigned;                        /*Addr of only active CDRP after VC re-establish
    RETRYCNT byte unsigned;                             /*# retries remaining for CDRP after VC reset
    DAPCOUNT byte unsigned;                             /*# DUSTMR loops until DAP_THREAD
    RSTRTCNT word unsigned;                             /*# of resynch or connection error since boot
    RSTRTQFL longword unsigned;                         /*Queue wherein we accumulate, sort and select
```

```
    RSTRTQBL longword unsigned;                     /* for re-submission following VC re-establish
    SAVED_PC longword unsigned;                     /*Saved PC on internal subroutine calls
    UCBCHAIN longword unsigned;                     /*Chain of UCBs on connection
    ORIGUCB longword unsigned;                      /*Ptr to Orig. UCB if unchained
    ALLOCLS longword unsigned;                      /*Device Allocation Class
    DAPCDRP longword unsigned;                      /*Ptr to Deter.Acc.Path CDRP
    CDDBLINK longword unsigned;                     /*Link in CDDB chain
    FOVER_CTR byte unsigned;                        /* counter of reconnect intervals per failover try
    RSVDB byte unsigned;                            /* reserved byte
    WTUCBCTR word unsigned;                         /* counter of UCBs waiting for mount ver. to finish
                                                    {  so that single stream CDRP processing may begin
    RSVD1 longword unsigned;                        /* reserved longword
    RSVD2 longword unsigned;                        /* reserved longword
    RSVD3 longword unsigned;                        /* reserved longword
    RSVD4 longword unsigned;                        /* reserved longword
    constant "LENGTH" equals . prefix CDDB$ tag K;  /*Standard length of CDDB
    constant "LENGTH" equals . prefix CDDB$ tag C;  /*Standard length of CDDB
    PERMCDRP longword unsigned;                     /*Beginning of a permanent CDRP allocated
                                                    /*  contiguous to CDDB

end CDDBDEF;

end_module $CDDBDEF;
```

```
module $CDLDEF;
/*+
/* CDL - SCS CONNECTION DESCRIPTOR LIST
/*
/* THERE IS A SYSTEM WIDE LIST OF CONNECTION DESCRIPTORS POINTED
/* TO BY THE CDL.
/*-


aggregate CDLDEF structure prefix CDL$ origin BASE;
    MAXCONIDX word unsigned;                        /*MAXIMUM ! OF CDT'S
    FILL_1 word fill prefix CDLDEF tag $$;          /*RESERVED WORD
    FREECDT longword unsigned;                      /*ADDR OF 1ST FREE CDT
    SIZE word unsigned;                             /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                             /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                           /*SCS STRUCT SUBTYPE FOR CDL
    FILL_2 longword fill prefix CDLDEF tag $$;      /*RESERVED LONGWORD
    BASE longword unsigned;                         /*BASE OF THE TABLE
    constant "LENGTH"  equals 16  prefix CDL tag $C;   /*LENGTH OF NEG PORTION OF STRUCTURE

end CDLDEF;

end_module $CDLDEF;
```

```
module $CDTDEF;
/++
/* CDT - SCS CONNECTION DESCRIPTOR TABLE
/*
/* THESE DESCRIPTORS ARE POINTED TO BY THE SYSTEM WIDE CONNECTION
/* DESCRIPTOR LIST (CDL).  ONE CDT IS USED PER SCS VIRTUAL CIRCUIT
/* OR LISTENING CONNECTION.
/*-



aggregate CDTDEF structure prefix CDT$;
    MSGINPUT_OVERLAY union fill;
        MSGINPUT longword unsigned;                 /*ADDR OF MSG INPUT DISPATCHER
        LINK longword unsigned;                     /* OR LINK TO NEXT FREE CDT
    end MSGINPUT_OVERLAY;
    DGINPUT longword unsigned;                      /*ADDR TO CALL ON DG RECEIVED
    SIZE word unsigned;                             /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                             /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                           /*SCS STRUCT SUBTYPE FOR CDT
    ERRADDR longword unsigned;                      /*ADDR TO CALL FOR ERROR NOTIFICATION
    PDT longword unsigned;                          /*ADDR OF ASSOC PORT DESC TABLE
    RCONID longword unsigned;                       /*REMOTE CONNECTION ID
    LCONID longword unsigned;                       /*LOCAL CONNECTION ID
    PB longword unsigned;                           /*ADDR OF ASSOC PATH BLOCK
    RSTATION byte unsigned dimension 6;             /*REMOTE STATION ADDR
    REASON word unsigned;                           /*REJECT/DISCONNECT REASON
    STATE word unsigned;                            /*CONNECTION STATE
                                                    /*STATE VALUES:
                                                    /* 0 ORIGIN, INCREMENTS OF 1:
    constant(
          CLOSED                                    /* CLOSED
        , LISTEN                                    /* LISTENING FOR CONNX REQUESTS
        , OPEN                                      /* OPEN
        , DISC_ACK                                  /* DISCONNECT ACKNOWLEDGED
        , DISC_REC                                  /* DISCONNECT REQ RECEIVED
        , DISC_SENT                                 /* DISCONNECT SENT
        , DISC_MTCH                                 /* DISCONNECT MATCH
        , CON_SENT                                  /* CONNECT REQ SENT
        , CON_ACK                                   /* CONNECT REQ SENT AND ACK'ED
        , CON_REC                                   /* CONNECT REQ RECEIVED
        , ACCP_SENT                                 /* ACCEPT REQ SENT
        , REJ_SENT                                  /* REJECT SENT
        , VC_FAIL                                   /* VIRTUAL CIRCUIT FAILED
    ) equals 0 increment 1  prefix CDT tag $C;      /*
BLKSTATE word unsigned;                             /*SCS SEND BLOCKED STATE
                                                    /*STATE VALUES:
                                                    /* 1 ORIGIN, INCREMENTS OF 1:
constant(
          CON_PEND                                  /* WAITING TO SEND CONNECT REQ
        , ACCP_PEND                                 /* WAITING TO SEND ACCEPT REQ
        , REJ_PEND                                  /* WAITING TO SEND REJECT REQ
        , DISC_PEND                                 /* WAITING TO SEND DISCONNECT REQ
        , CR_PEND                                   /* WAITING TO SEND CREDIT
        , DCR_PEND                                  /* WAITING TO SEND CREDIT IN
                                                    /*  PREPARATION FOR DISCONNECT
```

```
    ) equals 1 increment 1  prefix CDT tag $C;      /*
    SCSMSG longword unsigned;                       /*ADDR OF SCS RECEIVE BUFFER
    WAITQFL longword unsigned;                      /*SEND SCS MSG WAIT QUEUE FLINK
    WAITQBL longword unsigned;                      /*SEND SCS MSG WAIT QUEUE BLINK
    CRWAITQFL longword unsigned;                    /*SEND CREDIT WAIT QUEUE FLINK
    CRWAITQBL longword unsigned;                    /*SEND CREDIT WAIT QUEUE BLINK
    SEND word unsigned;                             /*CURRENT SEND CREDIT
    REC word unsigned;                              /*RECEIVE CREDIT (SEND CREDIT
                                                    /* HELD BY REMOTE
    MINREC word unsigned;                           /*MINIMUM RECEIVE CREDIT (MIN
                                                    /* SEND REQUIRED BY REMOTE)
    PENDREC word unsigned;                          /*RECEIVE CREDIT NOT YET EXTENDED
                                                    /* TO REMOTE
    INITLREC word unsigned;                         /*INITIAL RECEIVE CREDIT
    MINSEND word unsigned;                          /*MINIMUM SEND CREDIT
    DGREC word unsigned;                            /*DATAGRAMS QUEUED FOR RECEIVE
    PRIORITY byte unsigned;                         /*BLOCK TRANSFER PRIORIY
    FILL_1 byte fill prefix CDTDEF tag $$;          /*RESERVED
    RPROCNAM longword unsigned;                     /*ADDR OF REMOTE PROCESS NAME
    LPROCNAM longword unsigned;                     /*ADDR OF LOCAL PROCESS NAME
    CONDAT longword unsigned;                       /*ADDR OF CONNECT DATA
    AUXSTRUC longword unsigned;                     /*ADDR OF AUXILARY DATA STRUCTURE
    BADRSP longword unsigned;                       /*ADDR IN SYSAP TO CALL WITH
                                                    /* BAD RESPONSE(UNIMPLEMENTED)
    FPC longword unsigned;                          /*SAVED FORK PROCESS PC
    FR5 longword unsigned;                          /*SAVED FORK PROCESS R5
    CDTLST longword unsigned;                       /*LINK FOR CDT LIST FROM PB
    DGSENT longword unsigned;                       /*# APPLICATION DGS SENT
    DGRCVD longword unsigned;                       /*# APPLICATION DGS REC'D
    DGDISCARD longword unsigned;                    /*# DGS DISCARDED BY DRIVER
    MSGSENT longword unsigned;                      /*# APPLICATION MSGS SENT
    MSGRCVD longword unsigned;                      /*# APPLICATION MSGS REC'D
    SNDDATS longword unsigned;                      /*# SEND DATAS INITIATED
    BYTSENT longword unsigned;                      /*# BYTES SENT VIA SEND DATAS
    REQDATS longword unsigned;                      /*#REQ DATAS INITIATED
    BYTREQD longword unsigned;                      /*BYTES REC'D VIA REQ DATAS
    BYTMAPD longword unsigned;                      /*TOTAL BYTES MAPPED
    QCR_CNT word unsigned;                          /*# TIMES QUEUED FOR SEND CREDIT
    QBDT_CNT word unsigned;                         /*# TIMES QUEUED FOR BDT
    FILL_2 longword fill prefix CDTDEF tag $$;      /*RESERVED
    constant "LENGTH" equals . prefix CDT$ tag K;   /*LENGTH OF CDT
    constant "LENGTH" equals . prefix CDT$ tag C;   /*LENGTH OF CDT

end CDTDEF;

end_module $CDTDEF;
```

```
module $CEBDEF;
/*+
/* COMMON EVENT BLOCK
/*-


aggregate CEBDEF structure prefix CEB$;
    CEBFL_OVERLAY union fill;
        CEBFL longword unsigned;                    /*POINTER TO NEXT COMMON EVENT BLOCK
        CEBFL_BITS structure fill;
            VALID bitfield mask;                    /*SHMEM MASTER CEB, SET IF VALID ENTRY
            LOCKED bitfield mask;                   /*SHMEM MASTER CEB, SET IF ENTRY LOCKED
            REFCNTLCK bitfield mask;                /*SHMEM MASTER CEB, LOCKED FOR REFCNT CHG
        end CEBFL_BITS;
    end CEBFL_OVERLAY;
    CEBBL longword unsigned;                        /*POINTER TO PREVIOUS COMMON EVENT BLOCK
    SIZE word unsigned;                             /*SIZE OF COMMON EVENT BLOCK IN BYTES
    TYPE byte unsigned;                             /*STRUCTURE TYPE CODE FOR CEB
    STS_OVERLAY union fill;
        STS byte unsigned;                          /*STATUS FLAGS FOR CEB
        STS_BITS structure fill;
            NOQUOTA bitfield;                       /*NO QUOTA UPDATE
            PERM bitfield;                          /*PERMANENT CLUSTER
        end STS_BITS;
    end STS_OVERLAY;
    PID longword unsigned;                          /*PID OF CREATOR
    EFC longword unsigned;                          /*EVENT FLAGS (32 BIT VECTOR)
    WQFL longword unsigned;                         /*HEAD OF WAIT QUEUE
    WQBL longword unsigned;                         /*TAIL OF WAIT QUEUE
    WQCNT_OVERLAY union fill;
        WQCNT word unsigned;                        /*WAIT QUEUE COUNT(LENGTH)
                                                    /*SHMEM FIELDS IN THIS WORD
        WQCNT_FIELDS structure fill;
            LOCK byte unsigned;                     /*SHMEM MASTER CEB, ! OF PORT OWNING LOCK
            PROCCNT byte unsigned;                  /*SHMEM MASTER CEB, MAX ! OF PROCESSORS
        end WQCNT_FIELDS;
end WQCNT_OVERLAY;
STATE_OVERLAY union fill;
    STATE word unsigned;                            /*CEF WAIT STATE NUMBER
                                                    /*SHMEM FIELDS IN THIS WORD
    STATE_FIELDS structure fill;
        CREATPORT byte unsigned;                    /*SHMEM MASTER CEB, ! OF CREATOR PORT
        DELETPORT byte unsigned;                    /*SHMEM MASTER CEB, ! OF DELETER PORT
    end STATE_FIELDS;
end STATE_OVERLAY;
UIC_OVERLAY union fill;
    UIC longword unsigned;                          /*USER IDENT OF CEB CREATOR
    UIC_FIELDS structure fill;
        FILL_2 byte dimension 2 fill prefix CEBDEF tag $$;
        GRP word unsigned;                          /*GROUP NUMBER OF OWNER
    end UIC_FIELDS;
end UIC_OVERLAY;
PROT word unsigned;                                 /*PROTECTION MASK
REFC word unsigned;                                 /*REFERENCE COUNT FOR CEB
EFCNAM character length 16;                         /*EVENT CLUSTER TEXT NAME
constant "LENGTH" equals . prefix CEB$ tag K;       /*LENGTH OF NORMAL COMMON EVENT BLOCK
```

```
    constant "LENGTH" equals . prefix CEB$ tag C;       /*LENGTH OF NORMAL COMMON EVENT BLOCK

/*
/* THE FOLLOWING FIELDS ARE DEFINED FOR SHARED MEMORY COMMON EVENT BLOCKS.
/* CEB$L_SHB, CEB$W_INDX, AND CEB$L_MASTER ARE CONTAINED IN THE SLAVE CEB WHILE
/* CEB$L_VASLAVE1 IS THE OFFSET IN THE MASTER CEB TO THE FIRST SLAVE CEB.
/*
    SHB_OVERLAY union fill;
        SHB longword unsigned;                          /*SHMEM SLAVE CEB, SHMEM CTL BLK ADR
        VASLAVE1 longword unsigned;                     /*SHMEM MASTER CEB, PTR TO 1ST SLAVE CEB
    end SHB_OVERLAY;
    INDX word unsigned;                                 /*SHMEM SLAVE CEB, INDEX TO MASTER CEB
    FILL_1 word fill prefix CEBDEF tag $$;              /*SHMEM SLAVE CEB,
                                                        /*SHMEM MASTER CEB, FIELDS IN NEXT N
                                                        /* LONGWORDS ARE PROCESSOR REFCNTS
                                                        /* (ONE WORD FOR EACH PROCESSOR)
                                                        /* (OFFSET IS COMPUTED AT RUN-TIME)
    MASTER longword unsigned;                           /*SHMEM SLAVE CEB, VA OF MASTER CEB
    constant SLAVLNG equals . prefix CEB$ tag K;        /*LENGTH OF SHMEM SLAVE COMMON EVENT BLK
    constant SLAVLNG equals . prefix CEB$ tag C;        /*LENGTH OF SHMEM SLAVE COMMON EVENT BLK

end CEBDEF;

end_module $CEBDEF;
```

```
module $CHPCTLDEF;
/**
/*
/* CHeck Protection ConTroL block definition.  This block contains the
/* information concerning the type of access check being made.
/*
/*-

aggregate CHPCTL structure prefix CHPCTL$;
    ACCESS longword unsigned;                   /* Type of access desired
    FLAGS structure longword unsigned;          /* Control flags
        READ bitfield mask;                     /* Read access
        WRITE bitfield mask;                    /* Write access
        USEREADALL bitfield mask;               /* Try for read access via READALL
    end FLAGS;
    MODE byte unsigned;                         /* Access mode of request
    FILL_1 byte dimension 3 fill prefix CHPCTLDEF tag $$;
    constant 'LENGTH' equals . prefix CHPCTL$ tag K;
    constant 'LENGTH' equals . prefix CHPCTL$ tag C;
end CHPCTL;

end_module $CHPCTLDEF;
```

```
module $CHPRETDEF;
/*+
/*
/* CHeck Protection ConTroL RETurn argument block.  This block contains
/* the information needed to return arguments from the protection check.
/*
/*-

aggregate CHPRET structure prefix CHPRET$;
    AUDITLEN word unsigned;                 /* Size of the audit ACE buffer
    FILL_1 word fill prefix CHPRETDEF tag $$;
    AUDIT longword unsigned;                /* Address of the audit ACE buffer
    AUDITRET longword unsigned;             /* Address of word to get ACE length
    ALARMLEN word unsigned;                 /* Size of the alarm ACE buffer
    FILL_2 word fill prefix CHPRETDEF tag $$;
    ALARM longword unsigned;                /* Address of the alarm ACE buffer
    ALARMRET longword unsigned;             /* Address of word to get ACE length
    MATCHED_ACELEN word unsigned;           /* Size of the matched ACE buffer
    FILL_3 word fill prefix CHPRETDEF tag $$;
    MATCHED_ACE longword unsigned;          /* Address of the matched ACE buffer
    MATCHED_ACERET longword unsigned;       /* Address of word to get ACE length
    PRIVS_USED longword unsigned;           /* Address of longword to get privileges used
    constant "LENGTH" equals . prefix CHPRET$ tag K;
    constant "LENGTH" equals . prefix CHPRET$ tag C;
end CHPRET;

end_module $CHPRETDEF;
```

```
module $CIADEF;
/**
/* CIA - Compound Intrusion Analysis block
/*
/* Contains information about suspected and known intruders
/*-

aggregate CIADEF structure prefix CIA$;
    FLINK longword unsigned;                    /* Forward link to next block
    BLINK longword unsigned;                    /* Backward link to previous block
    SIZE word unsigned;                         /* Size of block
    TYPE byte unsigned;                         /* Structure type
    SUBTYPE BYTE unsigned;                      /* Structure subtype
    constant (                                  /* Source of breakin attempt
          TERMINAL                              /* Unknown user at terminal
        , TERM_USER                             /* Known username at terminal
        , NETWORK                               /* Network source
        , USERNAME                              /* Username of parent process
        ) equals 1 increment 1 tag K;
    FLAGS structure word unsigned;              /* Breakin type flags
        INTRUDER bitfield mask;                 /* Entry is an intruder
    end FLAGS;
    COUNT word unsigned;                        /* Count of attempts
    TIME quadword unsigned;                     /* Expiration time of entry
    DATA character length 56;                   /* Data area
    constant "LENGTH" equals . tag K;           /* Length of CIA block
    constant "LENGTH" equals . tag C;           /* Length of CIA block

end CIADEF;

end_module $CIADEF;
```

```
module $CIBDTDEF;
/**
/* CIBDT - CI BUFFER DESCRIPTOR TABLE
/*
/* THIS TABLE IS SHARABLE AMONG ALL CI PORTS ON A SYSTEM.  BUFFER
/* DESCRIPTORS (BD'S) ARE ALLOCATED FOR CI BLOCK TRANSFERS.
/*-



aggregate CIBDTDEF structure prefix CIBDT$ origin FILL_2;
    WAITFL longword unsigned;                           /*BD WAIT QUEUE FWD LINK
    WAITBL longword unsigned;                           /*BD WAIT QUEUE BACK LINK
    SIZE word unsigned;                                 /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                 /*CI STRUCTURE TYPE
    SUBTYP byte unsigned;                               /*CI STRUCT SUBTYPE FOR CI BDT
    FREEBD longword unsigned;                           /*ADDR OF FIRST FREE BD
    MAXIDX longword unsigned;                           /*MAX INDEX INTO BUFFER DESCRIPTORS
    FILL_1 longword fill prefix CIBDTDEF tag $$;        /*RESERVED LONGWORD
    constant BDLIST equals . prefix CIBDT$ tag K;       /*START OF BUFFER DESCRIPTORS
    constant BDLIST equals . prefix CIBDT$ tag C;       /*START OF BUFFER DESCRIPTORS
    constant "LENGTH"  equals 24  prefix CIBDT tag $C; /*LENGTH OF NEGATIVE PORTION OF STRUCT
/*
    FILL_2 byte fill prefix CIBDTDEF tag $$;
end CIBDTDEF;

end_module $CIBDTDEF;
```

```
module $CIBDDEF;
/*+
/* BD - CI BUFFER DESCRIPTOR FORMAT
/*-



aggregate CIBDDEF structure prefix CIBD$;
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                                /*FLAGS WORD
        FLAGS_BITS structure fill;
            BOFF bitfield length 9;                         /* BYTE OFFSET OF START OF BUFFER
            FILL_1 bitfield length 3 fill prefix CIBDDEF tag $$;/* 3 SPARE BITS
            AC bitfield mask;                               /* ACCESS MODE CHECK ENABLED IF SET
            ACMOD bitfield length 2;                        /* ACCESS MODE REQ'D IN PTE'S
            V bitfield mask;                                /* VALID BIT
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    KEY word unsigned;                                      /*SEQUENCE NUMBER
    BLEN longword unsigned;                                 /*LENGTH OF MAPPED BUFFER
    SVAPTE longword unsigned;                               /*SVA OF PTE MAPPING START OF BUFFER
    CDRP_OVERLAY union fill;
        CDRP longword unsigned;                             /*ADDR OF ASSOCIATED CDRP
        constant 'LENGTH' equals . prefix CIBD$ tag K;      /*LENGTH OF A BUFFER DESCRIPTOR
        constant 'LENGTH' equals . prefix CIBD$ tag C;      /*LENGTH OF A BUFFER DESCRIPTOR
        LINK longword unsigned;                             /* OR ADDR OF NEXT FREE DESCRIPTOR

    end CDRP_OVERLAY;
end CIBDDEF;

end_module $CIBDDEF;
```

```
module $CIBHANDEF;
/++
/* CIBHAN - CI BUFFER HANDLE FORMAT
/*-


aggregate CIBHANDEF structure prefix CIBHAN$;
    BOFF longword unsigned;                          /*BYTE OFFSET IN LOCAL BUFFER
    BNAME longword unsigned;                         /*NAME OF LOCAL BUFFER
    RCONID longword unsigned;                        /*REMOTE CONNECTION ID
    constant "LENGTH" equals . prefix CIBHAN$ tag K; /*LENGTH OF CI BUFFER HANDLE
    constant "LENGTH" equals . prefix CIBHAN$ tag C; /*LENGTH OF CI BUFFER HANDLE

end CIBHANDEF;

end_module $CIBHANDEF;
```

```
module $CIFQDTDEF;
/*+
/* CIFQDT - CI FREE MESSAGE/DATAGRAM QUEUE DESCRIPTOR TABLE
/*
/* THIS DATA STRUCTURE AND THE QUEUES IT HAS HEADERS FOR MAY BE
/* SHARED AMONG ALL CI'S ON THE SYSTEM.
/*-



aggregate CIFQDTDEF structure prefix CIFQDT$;
    DGSIZ word unsigned;                                /*DATAGRAM SIZE (INCL PORT HEADER)
    MSGSIZ word unsigned;                               /*MESSAGE SIZE (INCL PORT HEADER)
    FILL_1 longword fill prefix CIFQDTDEF tag $$;       /*RESERVED LONGWORD
    SIZE word unsigned;                                 /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                 /*CI STRUCTURE TYPE
    SUBTYP byte unsigned;                               /*CI STRUCT SUBTYPE FOR CI FQDT
    DGCNT word unsigned;                                /*SUM OF INITL DG CREDITS FOR ALL CONNX
    MSGCNT word unsigned;                               /*SUM OF INITL MSG CREDITS FOR ALL CONNX
    DGFL longword unsigned;                             /*DG FREE QUEUE FWD LINK
    DGBL longword unsigned;                             /*DG FREE QUEUE BACK LINK
    MSGFL longword unsigned;                            /*MSG FREE QUEUE FWD LINK
    MSGBL longword unsigned;                            /*MSG FREE QUEUE BACK LINK
    constant "LENGTH" equals . prefix CIFQDT$ tag K;    /*LENGTH OF CI FQDT
    constant "LENGTH" equals . prefix CIFQDT$ tag C;    /*LENGTH OF CI FQDT

end CIFQDTDEF;

end_module $CIFQDTDEF;
```

```
module $CLUBDEF;
/**
/* CLUB - CLUSTER BLOCK.
/*
/*      THERE IS ONE CLUB IN A VMS SYSTEM THAT IS PART OF A CLUSTER.
/*      THE CLUB DEFINES THE STATE OF THE THE CLUSTER AS KNOWN TO
/*      THE LOCAL SYSTEM.
/*-

/*
/*      THE CLUB FORK BLOCK (CLUBFKB) IS A SUBBLOCK OF THE CLUB THAT IS
/*      USED WHEN IT NECESSARY TO WAIT IN ORDER TO ALLOCATE MEMORY OR
/*      WHEN IT IS DESIRABLE TO FORK TO ALLOW OTHER FORK PROCESSES A
/*      CHANCE TO RUN.

aggregate CLUBFKBDEF structure prefix CLUBFKB$;
    FORK_BLOCK byte dimension (24);             /* FORK BLOCK TO WAIT IN
    PC2 longword unsigned;                      /* SAVED PC
    STATUS structure longword unsigned;         /* CLUSTER FAILOVER STATUS FLAGS
        FKB_BUSY bitfield mask;                 /*  FORK BLOCK IN USE FLAG
    end STATUS;
    constant "LENGTH" equals . tag C;           /* LENGTH OF CLUBFKB
    constant "LENGTH" equals . tag K;           /* LENGTH OF CLUBFKB
end CLUBFKBDEF;

/*
/*      THE CLUB POWERFAIL FORK BLOCK (CLUBPWF) IS A SUBBLOCK OF THE CLUB
/*      THAT IS USED TO FORK FROM IPL 31 TO IPL SCS DURING POWER RECOVERY.

aggregate CLUBPWFDEF structure prefix CLUBPWF$;
    FORK_BLOCK byte dimension (24);             /* FORK BLOCK TO WAIT IN
    STATUS structure longword unsigned;         /* BLOCK STATUS FLAGS
        BUSY bitfield mask;                     /*  FORK BLOCK IN USE FLAG
    end STATUS;
    constant "LENGTH" equals . tag C;           /* LENGTH OF CLUBPWF
    constant "LENGTH" equals . tag K;           /* LENGTH OF CLUBPWF
end CLUBPWFDEF;

/*
/*      THE CLUSTER FAILOVER CONTROL BLOCK (CLUFCB) IS A SUBBLOCK OF
/*      THE CLUB THAT IS USED TO SEQUENCE FAILOVER ACTIONS IN A CLUSTER.
/*

aggregate CLUFCBDEF structure prefix CLUFCB$;
    FORK_BLOCK byte dimension (24);             /* FORK BLOCK TO WAIT IN
    STEP longword unsigned;                     /* CURRENT FAILOVER STEP INDEX
    ID longword unsigned;                       /* FAILOVER INSTANCE IDENTIFICATION
    STATUS structure longword unsigned;         /* CLUSTER FAILOVER STATUS FLAGS
        ACTIVE bitfield mask;                   /*  FAILOVER ROUTINE ACTIVE
        PENDING bitfield mask;                  /*  FAILOVER PENDING
        SYNC_NODE bitfield mask;                /*  LOCAL NODE IS SYNCHRONIZER
        FKB_BUSY bitfield mask;                 /*  FORK BLOCK IN USE FLAG
        WAITING bitfield mask;                  /*  WAITING FOR NODES TO RESPOND
    end STATUS;
    SYNC_CSB longword unsigned;                 /* ADDRESS OF CSB OF SYNCHRONIZING SYSTEM
    NODEMAP byte dimension (32);                /* BITMAP OF ALL INVOLVED NODES
```

```
    RESPMAP byte dimension (32);            /* BITMAP OF NODES READY FOR A STEP
    constant "LENGTH" equals . tag C;       /* LENGTH OF CLUFCB
    constant "LENGTH" equals . tag K;       /* LENGTH OF CLUFCB
end CLUFCBDEF;

aggregate CLUBDEF structure prefix CLUB$;
    CSBQFL longword unsigned;               /* CSB QUEUE FORWARD LINK
    CSBQBL longword unsigned;               /* CSB QUEUE BACKWARD LINK
    SIZE word unsigned;                     /* SIZE OF CLUB IN BYTES
    TYPE byte unsigned;                     /* STRUCTURE TYPE
    SUBTYPE byte unsigned;                  /* STRUCTURE SUBTYPE
    POLL_CTX longword unsigned;             /* SCS POLLER CONTEXT
    LOCAL_CSB longword unsigned;            /* ADDRESS OF THE CSB FOR LOCAL SYSTEM
    JNL_DISPT longword unsigned;            /* DISPATCH FOR INCOMING JNL MESSAGES
    JNL_FAIL longword unsigned;             /* JOURNALING FAIL OVER ENTRY POINT
    FLAGS structure longword unsigned;      /* CLUSTER STATUS FLAGS
        CLUSTER bitfield mask;              /*  THIS NODE IS MEMBER OF CLUSTER
        QF_ACTIVE bitfield mask;            /*  QUORUM FILE IS READABLE, CONTRIBUTE TO STATIC QUORUM
        SHUTDOWN bitfield mask;             /*  NODE READY FOR CLUSTER SHUTDOWN
        FILL_0 bitfield length 5 fill;      /*  PAD TO BYTE BOUNDARY
        STS_PPHASE bitfield mask;           /*  STATUS ANALYZER POLLING PHASE
        STS_PH0 bitfield mask;              /*  STATUS ANALYZER, PHASE 0 SEEN
        STS_PH1B bitfield mask;             /*  STATUS ANALYZER, PHASE 1 (COORD CNX BROKEN) SEEN
        STS_PH1 bitfield mask;              /*  STATUS ANALYZER, PHASE 1 (COORD CNX OK) SEEN
        STS_PH2 bitfield mask;              /*  STATUS ANALYZER, PHASE 2 SEEN
        FILL_01 bitfield length 3 fill;     /*  PAD TO BYTE BOUNDARY
        FKB_BUSY bitfield mask;             /*  FORK BLOCK IN USE
        UNLOCK bitfield mask;               /*  UNLOCK REQUESTED
        NO_FORM bitfield mask;              /*  PROHIBIT NODE FROM FORMING A NEW CLUSTER
        INIT bitfield mask;                 /*  READY FOR CLUSTER JOIN/FORMATION
        BACKOUT bitfield mask;              /*  MUST EVENTUALLY BACK-OUT TRANSITION
        FILL_1 bitfield fill;               /*  FILLER
        FILL_2 bitfield fill;               /*  FILLER
        LOST_CNX bitfield mask;             /*  CONNECTION TO CLUSTER MEMBER HAS BEEN LOST
        QF_FAILED_NODE bitfield mask;       /*  A NODE HAS BEEN FAILED OUT
        QF_VOTE bitfield mask;              /*  QUORUM DISK IS CONTRIBUTING A (STATIC) VOTE
        QF_NEWVOTE bitfield mask;           /*  STAGING FOR QF_VOTE
        ADJ_QUORUM bitfield mask;           /*  QUORUM ADJUSTMENT REQUESTED
        QUORUM bitfield mask;               /*  CLUSTER IS IN QUORUM
        TRANSITION bitfield mask;           /*  STATE TRANSITION IN PROGRESS
        QF_DYNVOTE bitfield mask;           /*  QUORUM FILE CAN CONTRIBUTE TO DYNAMIC QUORUM
    end FLAGS;
    QUORUM word unsigned;                   /* CLUSTER QUORUM
    VOTES word unsigned;                    /* CLUSTER VOTES
    NODES word unsigned;                    /* NODES IN CLUSTER
    FSYSID byte dimension (6);              /* FOUNDING NODE'S SYSID
    FTIME quadword;                         /* FOUNDING TIME
    LST_XTN longword unsigned;              /* LAST COMPLETED TRANSACTION NUMBER
    LST_COORD longword unsigned;            /* LAST COMPLETED TRANSACTION COORDINATOR CSID
    LST_TIME quadword;                      /* LAST COMPLETED TRANSACTION TIME-STAMP
    LST_CODE byte unsigned;                 /* LAST COMPLETED TRANSACTION CODE
    LST_PHASE byte unsigned;                /* LAST COMPLETED TRANSACTION CODE
    NEWQDVOTES word unsigned;               /* STAGING FOR QDVOTES
    CUR_XTN longword unsigned;              /* CURRENT TRANSACTION NUMBER
    CUR_COORD longword unsigned;            /* CURRENT TRANSACTION COORDINATOR CSID
    CUR_TIME quadword;                      /* CURRENT TRANSACTION TIME-STAMP
```

```
    CUR_CODE byte unsigned;                     /* TRANSACTION CODE
    CUR_PHASE byte unsigned;                    /* TRANSACTION PHASE
    MSGCNT word unsigned;                       /* OUTSTANDING/WAITING MESSAGE COUNT
    COORD longword unsigned;                    /* COORDINATOR'S CSB ADDRESS
    LOCAL_CSID structure longword unsigned;     /* LOCAL SYSTEM CSID
        LOCAL_CSID_IDX word unsigned;           /* SLOT INDEX
        LOCAL_CSID_SEQ word unsigned;           /* SEQUENCE NUMBER
    end LOCAL_CSID;
    NEXT_CSID word unsigned;                    /* INDEX OF NEXT CSID TO ASSIGN
    FIRST_INDEX word unsigned;                  /* INDEX OF FIRST CSID ASSIGNED
    MAX_XTN longword unsigned;                  /* LARGEST TRANSACTION ID SEEN
    RETRYCNT longword unsigned;                 /* RESOURCE ALLOCATION RETRIES AVAILABLE
    CTX0 longword unsigned;                     /* LEVEL 0 CONTEXT AREA
    RET1 longword unsigned;                     /* LEVEL 1 SUBROUTINE RETURN
    CTX1 longword unsigned;                     /* LEVEL 1 CONTEXT AREA
    RET2 longword unsigned;                     /* LEVEL 2 SUBROUTINE RETURN
    CTX2 longword unsigned;                     /* LEVEL 2 CONTEXT AREA
    TQE longword unsigned;                      /* ADDRESS OF TIMER ENTRY
    CSPFL longword unsigned;                    /* Q OF FORK-INITIATED REQ'S FOR CSP
    CSPBL longword unsigned;                    /*  ...
    CSPIPID longword unsigned;                  /* PID OF CLUSTER SERVER (FOR SCH$WAKE)
    NEWTIME quadword unsigned;                  /* NEW VALUE OF TIME
    NEWTIME_REF quadword unsigned;              /* LOCAL REFERENCE FOR NEW TIME
    NEWQUORUM word unsigned;                    /* NEW VALUE FOR QUORUM
    ADJ_QUORUM word unsigned;                   /* QUORUM ADJUSTMENT REQUESTED VALUE
    FMERIT longword unsigned;                   /* FIGURE OF MERIT FOR OPTIMAL CLUSTER
    MEMSEQ word unsigned;                       /* MEMBERSHIP STATE SEQUENCE NUMBER
    QDVOTES word unsigned;                      /* VOTES HELD BY QUORUM DISK
    RANDOM longword unsigned;                   /* RANDOM NUMBER GENERATOR CONTEXT
    CLUDCB longword unsigned;                   /* ADDRESS OF QUORUM DISK CONTROL BLOCK
    QDNAME character dimension (16);            /* QUORUM DISK FULLDEVNAM
    FOREIGN_CLUSTER longword unsigned;          /* SHIFT REGISTER INDICATING FOREIGN CLUSTER SEEN
    FORK_BLOCK byte dimension (CLUBFKB$K_LENGTH); /* FORK BLOCK TO WAIT IN (CLUBFKB SUB-STRUCTURE)
    NODEMAP byte dimension (32);                /* BITMAP OF ALL POSSIBLE NODES
    CLUFCB byte dimension (CLUFCB$K_LENGTH);    /* CLUSTER FAILOVER CONTROL BLOCK
    HANG_FKB byte dimension (24);               /* FORK BLOCK TO USE TO BLOCK ACTIVITY AT IPL 4
    CLUBPWF byte dimension (CLUBPWF$K_LENGTH);  /* FORK BLOCK TO USE DURING POWER RECOVERY
    constant ''LENGTH'' equals . tag C;         /* LENGTH OF CLUB
    constant ''LENGTH'' equals . tag K;         /* LENGTH OF CLUB
end CLUBDEF;

end_module $CLUBDEF;
```

```
module $CLUDCBDEF;

/**
/* CLUDCB - Cluster Quorum Disk Control Block
/*-

aggregate CLUDCB structure prefix CLUDCB$;

    CLUDCBFL longword unsigned;          /* Forward Link (not used)
    CLUDCBBL longword unsigned;          /* Backward Link (not used)
    SIZE word unsigned;                  /* Size of CLUDCB (bytes)
    TYPE byte unsigned;                  /* Structure type
    SUBTYPE byte unsigned;               /* Structure subtype
    UCB longword unsigned;               /* Address of quorum disk UCB
    IRP longword unsigned;               /* Address of IRP
    TQE longword unsigned;               /* Address of timer queue entry
    ACT_COUNT longword unsigned;         /* Saved activity counter
    QFLBN longword unsigned;             /* Quorum file logical block number
    STATE structure word unsigned;       /* Quorum disk state bits
        QS_NOT_READY bitfield mask;      /* Not ready
        QS_READY bitfield mask;          /* Ready
        QS_ACTIVE bitfield mask;         /* Active
        QS_CLUSTER bitfield mask;        /* Active and this node is a cluster member
        QS_VOTE bitfield mask;           /* Potential vote
    end STATE;
    FLAGS structure word unsigned;       /* CLUDCB status bits
        QF_TIM bitfield mask;            /* Read or write timed out
        QF_RIP bitfield mask;            /* Read in progress
        QF_WIP bitfield mask;            /* Write in progress
        QF_ERROR bitfield mask;          /* Quorum disk error has been reported
        QF_CSPACK bitfield mask;         /* CSP request has been acknowledged
        QF_FIRST_ERR bitfield mask;      /* First error has already been seen
        QF_WRL_ERR bitfield mask;        /* Quorum disk is write-locked
    end flags;
    COUNTER byte unsigned;               /* Iteration counter
    BUFFER character length 512+4;       /* Quorum file buffer

    constant "LENGTH" equals . prefix CLUDCB$ tag K; /* Length of CLUDCB
    constant "LENGTH" equals . prefix CLUDCB$ tag C; /* Length of CLUDCB

    /* The quorum disk is specified with 4 sysgen parameters. DISK_QUORUM1
    /* to DISK_QUORUM4. Each parameter can specify 4 bytes.

    constant DISK_QUORUM equals 16 prefix CLUDCB$ tag S;

end CLUDCB;

end_module $CLUDCBDEF;
```

```
 module $CLUOPTDEF;

/++
/* CLUOPT - Cluster Optimal ReConfiguration Context Block
/*-

aggregate CLUOPT structure prefix CLUOPT$;
    PREV longword unsigned;             /* Link to previous CLUOPT block
    BEST longword fill;                 /* Link to best attained CLUOPT block
    SIZE word unsigned;                 /* Size of CLUOPT (bytes)
    TYPE byte unsigned;                 /* Structure type
    SUBTYPE byte unsigned;              /* Structure subtype
    CMERIT longword unsigned;           /* Figure of merit of nodes in CMAP
    ACMERIT longword unsigned;          /* Figure of merit of nodes in AMAP + CMAP
    CMAP byte dimension (32);           /* Map of nodes in proposed cluster
    AMAP byte dimension (32);           /* Map of nodes available for cluster
    RMAP byte dimension (32);           /* Map of nodes remaining for consideration
    constant 'LENGTH' equals . tag K;   /* Length of CLUOPT
    constant 'LENGTH' equals . tag C;   /* Length of CLUOPT
end CLUOPT;

end_module $CLUOPTDEF;
```

```
module $CONDEF;
/**
/*
/* Console function codes (defined in SRM).
/*
/*-


constant BOOTCPU equals 2  prefix CON tag $C;          /* Boot function code
constant CLRWARM equals 3  prefix CON tag $C;          /* Clear warm start flag
constant CLRCOLD equals 4  prefix CON tag $C;          /* Clear cold start flag

end_module $CONDEF;
```

```
module $CRBDEF;
/**
/* CRB - CHANNEL REQUEST BLOCK
/*
/* THERE IS ONE CHANNEL REQUEST BLOCK FOR EACH SET OF DEVICES WHOSE
/* ACCESS TO A SET OF CONTROLLERS MUST BE SYNCHRONIZED. EACH CHANNEL
/* CONTROL BLOCK ALLOWS UP TO FOUR CONTROLLERS TO WHICH THE INDIVIDUAL
/* DEVICES CAN BE ATTACHED.
/*-


aggregate CRBDEF structure prefix CRB$;
    WQFL longword unsigned;                     /*WAIT QUEUE FORWARD LINK
    WQBL longword unsigned;                     /*WAIT QUEUE BACKWARD LINK
    SIZE word unsigned;                         /*SIZE OF CRB IN BYTES
    TYPE byte unsigned;                         /*STRUCTURE TYPE FOR CRB
    TT_TYPE byte unsigned;                      /*controler type (DZ11, DZ32)
    REFC word unsigned;                         /*REFERENCE COUNT OF UCB'S
    MASK_OVERLAY union fill;
        "MASK" byte unsigned;                   /*CHANNEL ALLOCATION MASK
        MASK_BITS structure fill;
            BSY bitfield mask;                  /* CHANNEL IS BUSY (1=YES)
        end MASK_BITS;
    end MASK_OVERLAY;
    FILL_2 byte fill prefix CRBDEF tag $$;      /*SPARE UNUSED BYTE
    AUXSTRUC longword unsigned;                 /*Auxiliary structure addr (CDDB for class driver)
    TIMELINK_OVERLAY union fill;
        TIMELINK longword unsigned;             /*Thread of CRB's for periodic wakeup
        TT_MODEM longword unsigned;             /*modem control timer thread
    end TIMELINK_OVERLAY;
    DUETIME_OVERLAY union fill;
        DUETIME longword unsigned;              /*Due time for periodic wakeup
        DZ_MODEM longword unsigned;             /*DZ11 modem transition detection timer thread
    end DUETIME_OVERLAY;
    TOUTROUT_OVERLAY union fill;
        TOUTROUT longword unsigned;             /*Address of periodic wakeup routine
        TTY_TOUTROUT_FIELDS structure fill;
            DZ_RING byte unsigned;              /*last sampled ring for DZ11
            DZ_CARRIER byte unsigned;           /*last sampled carrier for DZ11
            DZ_DTR byte unsigned;               /*last output DTR for DZ11
            TT_TIMREFC byte unsigned;           /*lines with active modem timers
        end TTY_TOUTROUT_FIELDS;
    end TOUTROUT_OVERLAY;
    LINK longword unsigned;                     /*ADDRESS OF SECONDARY CRB
    INTD longword unsigned dimension 9;         /*INTERRUPT TRANSFER VECTOR
    constant "LENGTH" equals . prefix CRB$ tag K;   /*LENGTH OF STANDARD CRB
    constant "LENGTH" equals . prefix CRB$ tag C;   /*LENGTH OF STANDARD CRB
    INTD2 longword unsigned dimension 9;        /*SECOND INTERRUPT VECTOR
end CRBDEF;

end_module $CRBDEF;
```

```
module $VECDEF;

/*+
/* CRB INTERRUPT TRANSFER VECTOR STRUCTURE DEFINITIONS
/*-



aggregate VECDEF structure prefix VEC$;
    DISPATCH quadword unsigned;                          /*REGISTER SAVE AND DISPATCH INSTRUCTIONS
    IDB longword unsigned;                               /*ADDRESS OF ASSOCIATED IDB
    INITIAL longword unsigned;                           /*CONTROLLER INITIALIZATION ENTRY ADDRESS
    MAPREG_OVERLAY union fill;
        MAPREG word unsigned;                            /*STARTING MAP REGISTER ALLOCATED
        MAPREG_BITS structure fill;
            MAPREG bitfield length 15;                   /* MAP REGISTER NUMBER
            MAPLOCK bitfield mask;                       /* MAP REGISTER ALLOCATION PERMANENT
        end MAPREG_BITS;
    end MAPREG_OVERLAY;
    NUMREG byte unsigned;                                /*NUMBER OF MAP REGISTERS ALLOCATED
    DATAPATH_OVERLAY union fill;
        DATAPATH byte unsigned;                          /*BUFFERED DATAPATH ALLOCATED
        DATAPATH_BITS structure fill;
            DATAPATH bitfield length 5;                  /* DATAPATH NUMBER
            LWAE bitfield mask;                          /* LONGWORD ACCESS ENABLED
            FILL_1 bitfield fill prefix VECDEF tag $$;   /* SPARE BIT
            PATHLOCK bitfield mask;                      /* DATAPATH PERMANENT
        end DATAPATH_BITS;
    end DATAPATH_OVERLAY;
    ADP longword unsigned;                               /*ADDRESS OF ADP
    UNITINIT longword unsigned;                          /*ADDRESS OF UNIT INITIALIZE
    START longword unsigned;                             /*ADDRESS OF UNIT START
    UNITDISC longword unsigned;                          /*ADDRESS OF UNIT DISCONNECT
    constant "LENGTH" equals . prefix VEC$ tag K;        /*LENGTH OF STANDARD DISPATCHER
    constant "LENGTH" equals . prefix VEC$ tag C;        /*LENGTH OF STANDARD DISPATCHER
end VECDEF;

end_module $VECDEF;
```

```
module $CSBDEF;
/++
/* CSB - CLUSTER SYSTEM BLOCK.
/*
/*      THERE IS ONE CSB FOR EACH SYSTEM IN THE CLUSTER.
/*-


aggregate CSBDEF structure prefix CSB$;
    SYSQFL longword unsigned;                   /* SYSTEM QUEUE FORWARD LINK
    SYSQBL longword unsigned;                   /* SYSTEM QUEUE BACKWARD LINK
    SIZE word unsigned;                         /* SIZE OF CSB IN BYTES
    TYPE byte unsigned;                         /* STRUCTURE TYPE
    SUBTYPE byte unsigned;                      /* STRUCTURE SUBTYPE
    CDT longword unsigned;                      /* CDT ADDRESS
    PDT longword unsigned;                      /* PDT ADDRESS
    SENTQFL longword unsigned;                  /* SENT LIST HEAD LINK
    SENTQBL longword unsigned;                  /* SENT LIST TAIL LINK
    RESENDQFL longword unsigned;                /* RESEND LIST HEAD LINK
    RESENDQBL longword unsigned;                /* RESEND LIST TAIL LINK
    WARMCDRPQFL longword unsigned;              /* WARM CDRP QUEUE FORWARD LINK
    WARMCDRPQBL longword unsigned;              /* WARM CDRP QUEUE BACKWARD LINK
    SENDSEQNM word unsigned;                    /* NEXT SEQUENCE NUMBER TO SEND
    RCVDSEQNM word unsigned;                    /* LAST SEQUENCE NUMBER RECEIVED
    ACKRSEQNM word unsigned;                    /* LAST ACK RECEIVED SEQ. NUM.
    UNACKEDMSGS byte unsigned;                  /* NUMBER OF UNACKED MESSAGES
    REMACKLIM byte unsigned;                    /* REMOTE SIDE'S ACK LIMIT
    CURRCDRP longword unsigned;                 /* ADDRESS OF CDRP IN CRITICAL SECTION
    SWINCARN quadword unsigned;                 /* REMOTE SOFTWARE INCARN. NUM.
    ECOLVL byte unsigned;                       /* PROTOCOL ECO LEVEL
    VERNUM byte unsigned;                       /* PROTOCOL VERSION NUMBER
    WARMCDRPS byte unsigned;                    /* NUMBER OF CDRPS ON FREE QUEUE
    STATE byte unsigned;                        /* STATE OF CONNECTION
    constant (                                  /* STATE VALUES:
        OPEN,                                   /*  OPEN
        STATUS,                                 /*  SENDING/WAITING FOR STATUS
        RECONNECT,                              /*  ATTEMPTING TO RECONNECT
        NEW,                                    /*  BRAND NEW BLOCK
        CONNECT,                                /*  ATTEMPTING INITIAL CONNECTION
        ACCEPT,                                 /*  ACCEPTING INITIAL CONNECTION
        DISCONNECT,                             /*  DISCONNECT IN PROGRESS
        REACCEPT,                               /*  ACCEPTING RECONNECT REQUEST
        WAIT,                                   /*  TIME-OUT IN PROGRESS
        DEAD,                                   /*  NO CONNECTION POSSIBLE
        LOCAL                                   /*  LOCAL SYSTEM CSB
            ) equals 1 increment 1 tag K;
    TQE longword unsigned;                      /* ADDRESS OF TIMER QUEUE ENTRY
    TIMEOUT longword unsigned;                  /* TIME TO GIVE UP RECONNECTING
    CSID structure longword unsigned;           /* Cluster System ID
        CSID_IDX word unsigned;                 /* Slot index
        CSID_SEQ word unsigned;                 /* Sequence number
    end CSID;
    VOTES word unsigned;                        /* VOTES HELD BY NODE
    QUORUM word unsigned;                       /* QUORUM SET IN NODE
    LCKDIRWT word unsigned;                     /* LOCK MANAGER DISTRIBUTED DIRECTORY WEIGHT
    QDVOTES word unsigned;                      /* VOTES ASSIGNED TO QUORUM DISK
```

```
    PARTNERQFL longword unsigned;               /* LISTHEAD FWD PT  FOR BLOCK-XFER PARTNER BTXs
    PARTNERQBL longword unsigned;               /* LISTHEAD BACK P  FOR BLOCK-XFER PARTNER BTXs
    STATUS structure longword unsigned;         /* STATUS OF NODE I  CLUSTER
        /* THE FOLLOWING BITS ARE ALWAYS MEANINGFUL
        LONG_BREAK bitfield mask;               /*  LONG BREAK IN CONN  TION
        MEMBER bitfield mask;                   /*  NODE IS MEMBER OF LO AL CLUSTER
        REMOVED bitfield mask;                  /*  NODE REMOVED FROM CLU TER
        QF_SAME bitfield mask;                  /*  REMOTE QUORUM DISK MA HES LOCAL DISK
        FILL_2 bitfield length 4 fill;          /*  PAD TO BYTE BOUNDARY
        /*  THE FOLLOWING THREE BITS ARE SIGNIFICANT WHILE FORMING/JOINING
        /*  A CLUSTER
        CLUSTER bitfield mask;                  /*  REMOTE NODE IS CLUSTER MEMBER
        QF_ACTIVE bitfield mask;                /*  REMOTE NODE'S QUORUM FILE IS READABLE
                                                /*    AND MATCHES THIS NODE'S FILE
        SHUTDOWN bitfield mask;                 /*  REMOTE NODE READY FOR CLUSTER SHUTDOWN
        FILL_3 bitfield length 5 fill;          /*  PAD TO THIRD BYTE BOUNDARY
        /* THE FOLLOWING BITS ARE MEANINGFUL IF THIS NODE IS THE COORDINATOR
        LOCKED bitfield mask;                   /*  NODE LOCKED BY COORDINATOR
        SELECTED bitfield mask;                 /*  NODE SELECTED BY COORDINATOR
        FILL_5 bitfield length 6 fill;          /*  PAD TO BYTE BOUNDARY
        /* THE FOLLOWING BITS AR    LOCAL SIGNIFICANCE ONLY
        LOCAL bitfield mask;                    /*  MARK CSB FOR LOCAL SYSTEM
        STATUS_RCVD bitfield mask;              /*  STATUS RECEIVED FROM REMOTE SYSTEM
        SEND_STATUS bitfield mask;              /*  NEED TO SEND STATUS TO REMOTE SYSTEM
    end STATUS;
    CLUB longword unsigned;                     /* ADDRESS OF CLUB
    SB longword unsigned;                       /* ADDRESS OF SB FOR REMOTE SYSTEM
    REF_CNT byte unsigned;                      /* REFERENCE COUNT
    FILL_4 byte fill;                           /* PAD to word boundary
    NODES word unsigned;                        /* Number of nodes in remote cluster
    CNX_STS_R0 word unsigned;                   /* CONNECTION REQUEST R0 STATUS
    CNX_STS_R1 word unsigned;                   /* CONNECTION REQUEST R1 STATUS
    REFTIME quadword;                           /* CREATION/ADDITION/REMOVAL TIME
    CNCT byte dimension (16);                   /* CONNECT/ACCEPT DATA AREA
    NODEMAP byte dimension (32);                /* BITMAP OF NODE CONNECTIVITY
    constant 'LENGTH' equals . tag C;           /* LENGTH OF CSB
    constant 'LENGTH' equals . tag K;           /* LENGTH OF CSB
end CSBDEF;

end_module $CSBDEF;
```

```
module $CXBDEF;
/**
/* CXB - COMPLEX CHAINED BUFFER
/*
/* THESE OFFSETS ARE USED IN THE HEADER OF DISJOINT SEGMENTS
/* WHICH ARE TO BE PRESENTED TO THE USER AS A UNIT.
/*
/*-

aggregate   BDEF structure prefix CXB$;
    FL long   d unsigned;                       /*FORWARD QUEUE LINK
    BL longword unsigned;                       /*BACKWARD QUEUE LINK
    SIZE word unsigned;                         /*BLOCK SIZE
    TYPE byte unsigned;                         /*BLOCK TYPE
    CODE byte unsigned;                         /*BUFFER CODE
    "LE!GTH" word unsigned;                     /*LENGTH OF DATA
    OFFS T word unsigned;                       /*OFFSET TO START OF NSP MESSAGE
    LINK_OVERLAY union fill;
        LINK longword unsigned;                 /*LINK WORD FOR CHAINED DATA MESSAGE
        CHANNEL word unsigned;                  /* STORE CHANNEL NUMBER FOR AST
    end LINK_OVERLAY;
    IRP longword;                               /*IRP ADDRESS FOR TRANSMITS
    BOFF word unsigned;                         /*OFFSET TO DATALINK DATA
    BCNT word unsigned;                         /*SIZE OF DATALINK DATA
    END_ACTION longword unsigned;               /*POINTER TO I/O DONE ROUTINE
    SPARE1 longword unsigned;                   /*RESERVED
    SPARE0 longword unsigned;                   /*RESERVED
    constant "LENGTH" equals . prefix CXB$ tag K; /*LENGTH OF A STANDARD CXB
    constant "LENGTH" equals . prefix CXB$ tag C; /*LENGTH OF A STANDARD CXB

    DLL_OVERLAY union fill;
        DLL character length 32;                /*SCRATCH AREA FOR DATALINK LAYER
        STATION quadword unsigned;              /*REMOTE STATION ADDRESS
    end DLL_OVERLAY;
    constant DLL  equals 32  prefix CXB tag $C; /*SIZE OF CXB$T_DLL
                                                /*** this field must be quadword
                                                /*** aligned
    constant HEADER equals . prefix CXB$ tag K; /*CXB SIZE UP TO THIS POINT
    constant HEADER equals . prefix CXB$ tag C; /*CXB SIZE UP TO THIS POINT

    constant TRAILER equals 4  prefix CXB tag $C; /*SPACE AFTER CXB DATA FOR CRC CODE
    FILL_1 longword fill prefix CXBDEF tag $$;    /*THIS REPRESENTS THE SPACE TAKEN FOR
                                                  /*THE CRC TRAILER
    constant OVERHEAD equals . prefix CXB$ tag K; /*CXB$C_HEADER + CXB$C_TRAILER
    constant OVERHEAD equals . prefix CXB$ tag C; /*CXB$C_HEADER + CXB$C_TRAILER
end CXBDEF;

end_module $CXBDEF;
```

```
module $DDBDEF;
/*+
/* DDB - DEVICE DATA BLOCK
/*
/* THERE IS ONE DEVICE DATA BLOCK FOR EACH CONTROLLER IN A SYSTEM.
/*+


aggregate DDBDEF structure prefix DDB$;
    LINK longword unsigned;                         /*ADDRESS OF NEXT DDB IN LIST (0=LAST)
    UCB longword unsigned;                          /*ADDRESS OF FIRST UCB FOR THIS DDB
    SIZE word unsigned;                             /*SIZE OF DDB IN BYTES
    TYPE byte unsigned;                             /*TYPE OF DATA STRUCTURE FOR DDB
    FILL_1 byte fill prefix DDBDEF tag $$;          /*SPARE UNUSED BYTE
    DDT longword unsigned;                          /*ADDRESS OF THE DRIVER DISPATCH TABLE
    ACPD_OVERLAY union fill;
        ACPD longword unsigned;                     /*NAME OF DEFAULT ACP FOR DEVICE UNITS
        ACPD_FIELDS structure fill;
            FILL_4 byte dimension 3 fill prefix DDBDEF tag $$;
            ACPCLASS byte unsigned;                 /*CLASS CODE OF DEFAULT ACP
                                                    /*ACP CLASS CODE FOR DISKS
            constant(
                  PACK                              /*LARGE DISK PACKS
                , CART                              /*DISK CARTRIDGES
                , SLOW                              /*SLOW (CHEAP) DISKS (E.G., FLOPPY)
                , TAPE                              /*BLOCK STRUCTURED TAPE (E.G., TU58)
                ) equals 1 increment 1  prefix DDB tag $K;
        end ACPD_FIELDS;
    end ACPD_OVERLAY;

    NAME_OVERLAY union fill;                        /* GENERIC PATHNAME
        NAME character length 16;                   /*   OF THE DEVICE
                                                    /* AS AN
        NAME_ASCIC structure fill;                  /*   ASCIC STRUCTURE
            NAME_LEN byte unsigned;                 /*      CHARACTER COUNT
            NAME_STR character length 15;           /*      CHARACTER STRING
        end NAME_ASCIC;
    end NAME_OVERLAY;

    DRVNAM_OVERLAY union fill;                      /* DEVICE DRIVER NAME
        DRVNAME character length 16;                /*
                                                    /* AS AN
        DRVNAM_ASCIC structure fill;                /*   ASCIC STRUCTURE
            DRVNAM_LEN byte unsigned;               /*      CHARACTER COUNT
            DRVNAM_STR character length 15;         /*      CHARACTER STRING
        end DRVNAM_ASCIC;
    end DRVNAM_OVERLAY;

    SB longword unsigned;                           /*ADDR OF SYSTEMBLOCK
    CONLINK longword unsigned;                      /*NEXT DDB IN CONNECTION SUB-CHAIN
    ALLOCLS longword unsigned;                      /*DEVICE ALLOCATION CLASS
    "2P_UCB" structure longword unsigned;           /* ADDRESS OF FIRST UCB ON SECONDARY PATH
        DP_UCB longword unsigned;                   /* OLD STYLE SYNONYM FOR ABOVE
    end "2P_UCB";
    constant "LENGTH" equals . prefix DDB$ tag K;   /*LENGTH OF STANDARD DDB
```

```
    constant "LENGTH" equals . prefix DDB$ tag C;       /*LENGTH OF STANDARD DDB
end DDBDEF;

end_module $DDBDEF;
```

```
module $DDTDEF;
/*+
/* DDT - DRIVER DISPATCH TABLE
/*
/* EACH DEVICE DRIVER HAS A DRIVER DISPATCH TABLE.
/*-



aggregate DDTDEF structure prefix DDT$;
    START longword unsigned;                    /*ADDRESS OF DRIVER START I/O ROUTINE
    UNSOLINT longword unsigned;                 /*ADDRESS OF UNSOLICITED INTERRUPT ROUTINE
    FDT longword unsigned;                      /*ADDRESS OF FUNCTION DECISION TABLE
    CANCEL longword unsigned;                   /*ADDRESS OF CANCEL I/O ENTRY POINT
    REGDUMP longword unsigned;                  /*ADDRESS OF DEVICE REGISTER DUMP ROUTINE
    DIAGBUF word unsigned;                      /*SIZE OF DIAGNOSTIC BUFFER IN BYTES
    ERRORBUF word unsigned;                     /*SIZE OF ERROR LOG BUFFER IN BYTES
    UNITINIT longword unsigned;                 /*UNIT INITIALIZATION ENTRY POINT
    ALTSTART longword unsigned;                 /*ALTERNATE START I/O ENTRY POINT
    MNTVER longword unsigned;                   /*ADDRESS OF MOUNT VERIFICATION ROUTINE
    CLONEDUCB longword unsigned;                /*ADDRESS OF CLONED UCB ENTRY POINT
    FDTSIZE word unsigned;                      /*SIZE OF FDT IN BYTES
    filler word fill;                           { filler to gain longword alignment
    MNTV_SSSC longword unsigned;                /*ADDRESS OF SHADOW SET STATE CHANGE MV ENTRY
    MNTV_FOR longword unsigned;                 /*ADDRESS OF FOREIGN DEVICE MV ENTRY
    MNTV_SQD longword unsigned;                 /*ADDRESS OF SEQUENTIAL DEVICE MV ENTRY
       stant "LENGTH" equals . prefix DDT$ tag K;   /*LENGTH OF DDT
    constant "LENGTH" equals . prefix DDT$ tag C;   /*LENGTH OF DDT

end DDTDEF;

end_module $DDTDEF;
```

```
module $DJIDEF;
/*
/* Item codes for interface from job controller to LOGINOUT.
/*
aggregate ITEM_HEADER structure prefix DJI$;
    ITEM_SIZE           word unsigned;          /* Item size
    ITEM_CODE           word unsigned;          /* Item code
        constant (                              {  Define BATCH_OUTPUT items
            CPU_MAXIMUM                         /* (longword)   CPU maximum (10 ms units)
        ,   FILE_IDENTIFICATION                 /* (28 bytes)   DVI, FID, DID of command procedure
        ,   FLAGS                               /* (longword)   flags
        ,   JOB_NAME                            /* (string)     job name
        ,   LOG_QUEUE                           /* (string)     log file queue
        ,   LOG_SPECIFICATION                   /* (string)     log file specification
        ,   PARAMETER_1                         /* (string)     value of P1
        ,   PARAMETER_2                         /* (string)     value of P2
        ,   PARAMETER_3                         /* (string)     value of P3
        ,   PARAMETER_4                         /* (string)     value of P4
        ,   PARAMETER_5                         /* (string)     value of P5
        ,   PARAMETER_6                         /* (string)     value of P6
        ,   PARAMETER_7                         /* (string)     value of P7
        ,   PARAMETER_8                         /* (string)     value of P8
        ,   RESTART                             /* (string)     value of BATCH$RESTART
        ,   USERNAME                            /* (string)     username
        ,   WSDEFAULT                           /* (longword)   working set default
        ,   WSEXTENT                            /* (longword)   working set extent
        ,   WSQUOTA                             /* (longword)   working set quota
        ) equals 1 increment 1 prefix DJI$;
        constant (                              {  Define BATCH_INPUT items
            INPUT_FLAGS                         /* (longword)   flags
        ,   CONDITION_VECTOR                    /* (1 to 3 longwords) error conditions
        ,   FILE_SPECIFICATION                  /* (string)     filespec of failed logfile
        ) equals 32769 increment 1 prefix DJI$;
end;
/*
/* Structure of FLAGS item.
/*
aggregate FLAGS structure fill prefix DJI$;
    FLAGS structure longword unsigned;
        DELETE_FILE             bitfield mask;  /* delete command procedure
        LOG_DELETE              bitfield mask;  /* delete log file
        LOG_NULL                bitfield mask;  /* log specification is NLA0:
        LOG_SPOOL               bitfield mask;  /* spool log file
        NOTIFY                  bitfield mask;  /* spool log file with /NOTIFY
        RESTARTING              bitfield mask;  /* job is restarting
        TERMINATE               bitfield mask;  /* job should terminate
        USE_CPU_MAXIMUM         bitfield mask;  /* use specified CPU_MAXIMUM
        USE_WSDEFAULT           bitfield mask;  /* use specified WSDEFAULT
        USE_WSEXTENT            bitfield mask;  /* use specified WSEXTENT
        USE_WSQUOTA             bitfield mask;  /* use specified WSQUOTA
    end;
end;
/*
/* Structure of INPUT_FLAGS item.
/*
aggregate INPUT_FLAGS structure fill prefix DJI$;
```

```
    INPUT_FLAGS structure longword unsigned;
        NO_FILE                 bitfield mask;  /* do not return a file
    end;
end;

end_module $DJIDEF;
```

```
module $DPTDEF;
/*+
/* DPT - DRIVER PROLOGUE TABLE
/*
/* EACH DEVICE DRIVER HAS A DRIVER PROLOGUE TABLE.
/*-



aggregate DPTDEF structure prefix DPT$;
    FLINK longword unsigned;                        /*FORWARD LINK TO NEXT DPT
    BLINK longword unsigned;                        /*BACKWARD LINK TO PREVIOUS DPT
    SIZE word unsigned;                             /*SIZE OF DRIVER
    TYPE byte unsigned;                             /*STRUCTURE TYPE
    REFC byte unsigned;                             /*COUNT OF DDB'S THAT REFERENCE DRIVER
    ADPTYPE byte unsigned;                          /*ADAPTER TYPE CODE
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                        /*DRIVER LOADER FLAGS
        FLAGS_BITS structure fill;
            SUBCNTRL bitfield mask;                 /*DEVICE IS A SUB-CONTROLLER
            SVP bitfield mask;                      /*DEVICE REQUIRES A SYSTEM PAGE
            NOUNLOAD bitfield mask;                 /*DRIVER IS NOT TO BE UNLOADED
            SCS bitfield mask;                      /*SCS CODE MUST BE LOADER WITH DRIVER
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    UCBSIZE word unsigned;                          /*SIZE OF UCB
    INITTAB word unsigned;                          /*OFFSET TO INIT TABLE
    REINITTAB word unsigned;                        /*OFFSET TO RE-INIT TABLE
    UNLOAD word unsigned;                           /*OFFSET TO UNLOAD ACTION ROUTINE
    MAXUNITS word unsigned;                         /*MAXIMUM UNITS THAT CAN BE CONNECTED
    VERSION word unsigned;                          /*DRIVER PROLOGUE VERSION NUMBER
    constant VERSION    equals 4  prefix DPT tag $C;/*CURRENT VERSION NUMBER
    DEFUNITS word unsigned;                         /*DEFAULT NUMBER OF UNITS
    DELIVER word unsigned;                          /*OFFSET TO DRIVER UNIT DELIVERY ROUTINE
    VECTOR word unsigned;                           /*OFFSET TO VECTOR TABLE (IN TTDRIVER)
    NAME character length 12;                       /*DRIVER NAME (COUNTED STRING)
    LINKTIME quadword unsigned;                     /*LINK DATE AND TIME FROM IMAGE HEADER
    ECOLEVEL longword unsigned;                     /*ECO LEVEL FROM IMAGE HEADER
    constant "LENGTH" equals . prefix DPT$ tag K;   /*LENGTH OF PROLOGUE TABLE
    constant "LENGTH" equals . prefix DPT$ tag C;   /*LENGTH OF PROLOGUE TABLE

end DPTDEF;

end_module $DPTDEF;
```

```
module $DYNDEF;

/++
/* DATA STRUCTURE TYPE DEFINITIONS
/*
/* EACH DATA STRUCTURE THAT IS ALLOCATED FROM THE DYNAMIC MEMORY
/* POOL SHOULD HAVE A VALID TYPE IN ITS 11TH BYTE.
/*-


                                                        /*BASE AND OFFSET OF 1

constant(
        ADP                                             /*UNIBUS ADAPTER CONTROL BLOCK
      , ACB                                             /*AST CONTROL BLOCK
      , AQB                                             /*ACP QUEUE BLOCK
      , CEB                                             /*COMMON EVENT BLOCK
      , CRB                                             /*CHANNEL REQUEST BLOCK
      , DDB                                             /*DEVICE DESCRIPTOR BLOCK
      , FCB                                             /*FILE CONTROL BLOCK
      , FRK                                             /*FORK BLOCK
      , IDB                                             /*INTERRUPT DISPATCH BLOCK
      , IRP                                             /*I/O REQUEST PACKET
      , LOG                                             /*LOGICAL NAME BLOCK
      , PCB                                             /*PROCESS CONTROL BLOCK
      , PQB                                             /*PROCESS QUOTA BLOCK
      , RVT                                             /*RELATIVE VOLUME TABLE
      , TQE                                             /*TIMER QUEUE ENTRY
      , UCB                                             /*UNIT CONTROL BLOCK
      , VCB                                             /*VOLUME CONTROL BLOCK
      , WCB                                             /*WINDOW CONTROL BLOCK
      , BUFIO                                           /*BUFFERED I/O BLOCK
      , TYPAHD                                          /*TERMINAL TYPEAHEAD BUFFER
      , GSD                                             /*GLOBAL SECTION DESCRIPTOR BLOCK
      , MVL                                             /*MAGNETIC TAPE VOLUME LIST
      , NET                                             /*NETWORK MESSAGE BLOCK
      , KFE                                             /*KNOWN FILE ENTRY
      , MTL                                             /*MOUNTED VOLUME LIST ENTRY
      , BRDCST                                          /*BROADCAST MESSAGE BLOCK
      , CXB                                             /*COMPLEX CHAINED BUFFER
      , NDB                                             /* NETWORK NODE DESCRIPTOR BLOCK
      , SSB                                             /* LOGICAL LINK SUBCHANNEL STATUS BLOCK
      , DPT                                             /* DRIVER PROLOGUE TABLE
      , JPB                                             /* JOB PARAMETER BLOCK
      , PBH                                             /* PERFORMANCE BUFFER HEADER
      , PDB                                             /* PERFORMANCE DATA BLOCK
      , PIB                                             /* PERFORMANCE INFORMATION BLOCK
      , PFL                                             /* PAGE FILE CONTROL BLOCK
      , JNLWCB                                          /* JOURNAL WINDOW CONTROL BLOCK
      , PTR                                             /* POINTER CONTROL BLOCK
      , KFRH                                            /* KNOWN FILE IMAGE HEADER
      , DCCB                                            /* Data Cache Control Block
      , EXTGSD                                          /*EXTENDED GLOBAL SECTION DESCRIPTOR
      , SHMGSD                                          /*SHARED MEMORY GLOBAL SECTION DESCRIPTOR
      , SHB                                             /*SHARED MEMORY CONTROL BLOCK
      , MBX                                             /*MAILBOX CONTROL BLOCK
```

```
    , IRPE                                  /*I/O REQUEST PACKET EXTENSION
    , SLAVCEB                               /*SLAVE COMMON EVENT BLOCK
    , SHMCEB                                /*SHARED MEMORY MASTER COMMON EVENT BLOCK
    , JIB                                   /*JOB INFORMATION BLOCK
    , TWP                                   /* Terminal driver write packet
    , RBM                                   /* Realtime SPT bit map
    , VCA                                   /* Disk volume cache block
    , CDB                                   /*X25 LES CHANNEL DATA BLOCK
    , LPD                                   /*X25 LES PROCESS DESCRIPTOR
    , LKB                                   /*LOCK BLOCK
    , RSB                                   /*RESOURCE BLOCK
    , LKID                                  /*LOCK ID TABLE
    , RSHT                                  /*RESOURCE HASH TABLE
    , CDRP                                  /*CLASS DRIVER REQUEST PACKET
    , ERP                                   /* ERRORLOG PACKET
    , CIDG                                  /*DATAGRAM BUFFER FOR CI PORT
    , CIMSG                                 /*MESSAGE BUFFER FOR CI PORT
    , XWB                                   /*DECNET LOGICAL LINK CONTEXT BLOCK
                                            /*    (REPLACES "NDB" BLOCK)
    , WQE                                   /*DECNET WORK QUEUE BLOCK
                                            /*    (REPLACES "NET" BLOCK)
    , ACL                                   /*ACCESS CONTROL LIST QUEUE ENTRY
    , LNM                                   /*LOGICAL NAME BLOCK
    , UNUSED_2                              /*UNUSED
    , RIGHTSLIST                            /*RIGHTS LIST
    , KFD                                   /* Known File Device Directory block
    , KFPB                                  /* Known File list Pointer Block
    , CIA                                   /* Compound Intrusion Analysis block
    , PMB                                   /* Page Fault Monitor Control Block
    , PFB                                   /* Page Fault Monitor Buffer
    , CHIP                                  /* Internal CHKPRO block
    , ORB                                   /* Objects Rights Block
    ) equals 1 increment 1  prefix DYN tag $C;
/*
/* THE FOLLOWING CODES ARE SUBTYPABLE, THAT IS EACH CODE REFERS TO A GENERIC
/* FUNCTION AND WITHIN THAT FUNCTION THERE MAY BE MANY DIFFERENT SUB-TYPES
/* OF BLOCKS.  THIS SCHEME WAS ADOPTED TO PRESERVE TYPES.  THE SUB-TYPE IS
/* IN THE 12TH BYTE.
/*
constant SUBTYPE        equals 96  prefix DYN tag $C;   /* START OF SUBTYPABLES
constant SCS    equals 96  prefix DYN tag $C;           /* SYSTEM COMMUNICATION SERVICES
constant(
      SCS_CDL                                           /* CONNECT DISPATCH LIST
    , SCS_CDT                                           /* CONNECT DISPATCH TABLE
    , SCS_DIR                                           /* DIRECTORY BLOCK
    , SCS_PB                                            /* PATH BLOCK
    , SCS_PDT                                           /* PORT DESCRIPTOR TABLE
    , SCS_RDT                                           /* REQUEST DESCRIPTOR TABLE
    , SCS_SB                                            /* SYSTEM BLOCK
    , SCS_SPPB                                          /* SCA POLLER PROCESS BLOCK
    , SCS_SPNB                                          /* SCA POLLER NAME BLOCK
    , SCS_UQB                                           /* MSCP SERVER UNIT BLOCK
    , SCS_HQB                                           /* MSCP SERVER HOST BLOCK
    ) equals 1 increment 1  prefix DYN tag $C;
constant CI     equals 97  prefix DYN tag $C;           /* CI PORT SPECIFIC
constant(
```

```
      CI_BDT                                            /* BUFFER DESCRIPTOR TABLE
    , CI_FQDT                                           /* FREE QUE DESCRIPTOR TABLE
    ) equals 1 increment 1  prefix DYN tag $C;
constant LOADCODE       equals 98  prefix DYN tag $C;   /* LOADABLE CODE
constant(
      NON_PAGED                                         /* NON PAGED CODE
    , PAGED                                             /* PAGED CODE
    , LC_MP                                             /* MULTIPROCESSOR CODE
    , LC_SCS                                            /* SCS CODE
    , LC_CLS                                            /* CLUSTER CODE
    , LC_CHREML                                         /* CHAR/DECIMAL INS EMUL
    , LC_FPEMUL                                         /* FLOAT PNT EMULATOR
    , LC_MSCP                                           /* MSCP SERVER
    , LC_SYSL                                           /* SYSLOA
    ) equals 1 increment 1  prefix DYN tag $C;
constant INIT   equals 99  prefix DYN tag $C;           /* STRUCTURES SET UP BY INIT
constant(
      PCBVEC                                            /* PROCESS CONTROL BLOCK VECTOR
    , PHVEC                                             /* PROCESS HEADER VECTOR
    , SWPMAP                                            /* SWAPPER MAP
    , MPWMAP                                            /* MODIFIED PAGE WRITER MAP
    , PRCMAP                                            /* PROCESS BITMAP
    , BOOTCB                                            /* BOOT CONTROL BLOCK
    , CONF                                              /* CONFIGURATION ARRAYS
    , CST                                               /* CLUSTER SYSTEM TABLE
    ) equals 1 increment 1  prefix DYN tag $C;
constant CLASSDRV       equals 100  prefix DYN tag $C;  /* CLASS DRIVER MAJOR STRUCTURE TYPE CODE
constant(
      CD_CDDB                                           /* CLASS DRIVER DATA BLOCK
    , CD_BBRPG                                          /* BAD BLOCK REPLACEMENT PAGE
    , CD_SHDW_WRK                                       /* SHADOW SET WORK BUFFER
    ) equals 1 increment 1  prefix DYN tag $C;
constant CLU            equals 101  prefix DYN tag $C;  /* CLUSTER MAJOR STRUCTURE TYPE CODE
constant(
      CLU_CSB                                           /* CONNECTION STATUS BLOCK
    , CLU_CLUVEC                                        /* CLUSTER SYSTEM VECTOR
    , CLU_CLUB                                          /* CLUSTER BLOCK
    , CLU_BTX                                           /* CLUSTER BLOCK TRANSFER EXTENSION
    , CLU_CLUDCB                                        /* CLUSTER DISK QUORUM CONTROL BLOCK
    , CLU_CLUOPT                                        /* CLUSTER OPTIMAL RECONFIGURATION CONTEXT BLOCK
    , CLU_LCKDIR                                        /* LOCK MANAGER DISTRIBUTED DIRECTORY VECTOR
    ) equals 1 increment 1 prefix DYN tag $C;
constant PGD            equals 102  prefix DYN tag $C;  /* PAGED DYNAMIC MEMORY
constant (
      PGD_F11BC                                         /* F11BXQP BUFFER CACHE.
    ) equals 1 increment 1  prefix DYN tag $C;
constant JNL            equals 103  prefix DYN tag $C;  /* JOURNALING STRUCTURE
constant(                                               /* JOURNALING SUBTYPES
      JNL_ABL                                           /* AI-BI LIST
    , JNL_ADL                                           /* ALLOCATED DEVICE LIST
    , JNL_BCB                                           /* BUFFER CONTROL BLOCK
    , JNL_ACBM                                          /* JOURNAL ACCESS BIT MAP
    , JNL_BUF                                           /* JOURNAL BUFFER
    , JNL_DB                                            /* JOURNAL DATA BLOCK
    , JNL_SFT                                           /* SPOOL FILE TABLE
    , JNL_NDL                                           /* NAME TABLE DEVICE LIST
```

```
    , JNL_JMT                                       /* JOURNAL MERGE TABLE
    , JNL_RM                                        /* JOURNAL REMASTER BLOCK
    , JNL_RRP                                       /* RECOVERY REQUEST BLOCK
    , JNL_RCPC                                      /* RCP CONTROL BLOCK
    , JNL_RUL                                       /* RECOVERY UNIT LIST
    , JNL_VCL                                       /* VCB LIST
    , JNL_VLE                                       /* VCB ELEMENT
    , JNL_CWQ                                       /* CLUSTER WRITE Q ENTRY
    , JNL_RC                                        /* READ CONTEXT
    , JNL_MSG                                       /* JOURNAL MESSAGE
    , JNL_BXSTS                                     /* BLOCK XFER STATUS BLOCK
    , JNL_MSGDATA                                   /* CI MESSAGE DATA
    , JNL_DIOREAD                                   /* DIO read data
    ) equals 1 increment 1  prefix DYN tag $C;

/*
/* SPECIAL DYNAMIC MEMORY TYPES.  THESE ARE HANDLED SPECIALLY BY
/* EXE$DALONONPAGED.
/*
constant SPECIAL        equals 128  prefix DYN tag $C;  /* START OF SPECIAL TYPES
                                                        /* BASE OF 128 AND OFFSET OF 1

constant(
      SHRBUFIO                                          /* SHARED MEMORY BUFFERED I/O
    ) equals 128 increment 1  prefix DYN tag $C;

end_module $DYNDEF;
```

```
module $EMBHDDEF;

/*
/* ERROR MESSAGE BUFFER HEADER
/*
/*                  ******        CAUTION  ******
/*
/*        ALL OF THE EMBxxDEF STRUCTURES ASSUME THAT THE HEADER IS
/*        EXACTLY ONE LONGWORD IN LENGTH.  IF THIS FIELD CHANGES,
/*        IF EFFECTS ALL OF THE OTHER STRUCTURES.
/*        ALL MESSAGES HAVE TYPE, TIME, SYSTEM ID, AND ERROR SEQUENCE IN THE
/*        SAME RELATIVE LOCATIONS.
/*



aggregate EMBHDDEF structure prefix EMB$ origin HD_SID;
    SIZE word unsigned;                                 /*SIZE OF ERROR MESSAGE IN BYTES
    BUFIND byte unsigned;                               /*ALLOCATION BUFFER INDICATOR (0 OR 1)
    VALID byte unsigned;                                /*ERROR MESSAGE VALID INDICATOR
    constant "LENGTH"  equals 4  prefix EMB tag $K;     /*LENGTH OF FIXED PART OF MESSAGE HEADER
    HD_SID longword unsigned;                           /*SYSTEM ID
    HD_ENTRY_OVERLAY union fill.
        HD_ENTRY word unsigned;                         /*ERROR MESSAGE ENTRY TYPE
        HD_ENTRY_FIELDS structure fill;
            DEVTYP byte unsigned;                       /*DEVICE TYPE
            DEVCLS byte unsigned;                       /*DEVICE CLASS
        end HD_ENTRY_FIELDS;
    end HD_ENTRY_OVERLAY;
    HD_TIME quadword unsigned;                          /*TIME OF MESSAGE ENTRY
    HD_ERRSEQ word unsigned;                            /*ERROR SEQUENCE FOR MESSAGE
    constant HD_LENGTH equals . prefix EMB$ tag K;      /*LENGTH OF PART COMMON TO ALL MESSAGES
    constant HD_LENGTH equals . prefix EMB$ tag C;      /*LENGTH OF PART COMMON TO ALL MESSAGES
end EMBHDDEF;

end_module $EMBHDDEF;

module $EMBBCDEF;

/*
/* BUGCHECK ERROR MESSAGE BUFFER FORMAT (SYSTEM AND USER)
/*



aggregate EMBBCDEF structure prefix EMB$;
    BC_SID longword unsigned;                           /*SYSTEM ID
    BC_ENTRY word unsigned;                             /*ENTRY TYPE
    BC_TIME quadword unsigned;                          /*TIME IN 64 BITS
    BC_ERRSEQ word unsigned;                            /*ERROR SEQUENCE NUMBER
    BC_KSP longword unsigned;                           /*KERNEL STACK POINTER
    BC_ESP longword unsigned;                           /*EXECUTIVE STACK POINTER
    BC_SSP longword unsigned;                           /*SUPERVISOR STACK POINTER
    BC_USP longword unsigned;                           /*USER STACK POINTER
    BC_ISP longword unsigned;                           /*INTERRUPT STACK POINTER
    BC_R0 longword unsigned;                            /*REGISTER R0
```

```
    BC_R1 longword unsigned;                        /*REGISTER R1
    BC_R2 longword unsigned;                        /*REGISTER R2
    BC_R3 longword unsigned;                        /*REGISTER R3
    BC_R4 longword unsigned;                        /*REGISTER R4
    BC_R5 longword unsigned;                        /*REGISTER R5
    BC_R6 longword unsigned;                        /*REGISTER R6
    BC_R7 longword unsigned;                        /*REGISTER R7
    BC_R8 longword unsigned;                        /*REGISTER R8
    BC_R9 longword unsigned;                        /*REGISTER R9
    BC_R10 longword unsigned;                       /*REGISTER R10
    BC_R11 longword unsigned;                       /*REGISTER R11
    BC_AP longword unsigned;                        /*ARGUMENT POINTER
    BC_FP longword unsigned;                        /*FRAME POINTER
    BC_SP longword unsigned;                        /*CURRENT STACK POINTER
    BC_PC longword unsigned;                        /*PROGRAM COUNTER
    BC_PSL longword unsigned;                       /*PROCESSOR STATUS
    BC_CODE longword unsigned;                      /*BUGCHECK CODE
    BC_PID longword unsigned;                       /*CURRENT PROCESS ID
    BC_LNAME character length 16;                   /*CURRENT PROCESS NAME
    constant BC_LENGTH equals . prefix EMB$ tag K;  /*SIZE OF FIXED PART OF BUGCHECK MESSAGE
    constant BC_LENGTH equals . prefix EMB$ tag C;  /*SIZE OF FIXED PART OF BUGCHECK MESSAGE
end EMBBCDEF;

end_module $EMBBCDEF;

module $EMBCRDEF;

/*
/* CRASH-RESTART ERROR MESSAGE BUFFER FORMAT
/*


aggregate EMBCRDEF structure prefix EMB$;
    CR_SID longword unsigned;                       /*SYSTEM ID
    CR_ENTRY word unsigned;                         /*ENTRY TYPE
    CR_TIME quadword unsigned;                      /*TIME IN 64 BITS
    CR_ERRSEQ word unsigned;                        /*ERROR SEQUENCE NUMBER
    CR_KSP longword unsigned;                       /*KERNEL STACK POINTER
    CR_ESP longword unsigned;                       /*EXECUTIVE STACK POINTER
    CR_SSP longword unsigned;                       /*SUPERVISOR STACK POINTER
    CR_USP longword unsigned;                       /*USER STACK POINTER
    CR_ISP longword unsigned;                       /*INTERRUPT STACK POINTER
    CR_R0 longword unsigned;                        /*REGISTER R0
    CR_R1 longword unsigned;                        /*REGISTER R1
    CR_R2 longword unsigned;                        /*REGISTER R2
    CR_R3 longword unsigned;                        /*REGISTER R3
    CR_R4 longword unsigned;                        /*REGISTER R4
    CR_R5 longword unsigned;                        /*REGISTER R5
    CR_R6 longword unsigned;                        /*REGISTER R6
    CR_R7 longword unsigned;                        /*REGISTER R7
    CR_R8 longword unsigned;                        /*REGISTER R8
    CR_R9 longword unsigned;                        /*REGISTER R9
    CR_R10 longword unsigned;                       /*REGISTER R10
    CR_R11 longword unsigned;                       /*REGISTER R11
    CR_AP longword unsigned;                        /*ARGUMENT POINTER
```

```
    CR_FP longword unsigned;                        /*FRAME POINTER
    CR_SP longword unsigned;                        /*CURRENT STACK POINTER
    CR_PC longword unsigned;                        /*PROGRAM COUNTER
    CR_PSL longword unsigned;                       /*PROCESSOR STATUS
    CR_P0BR longword unsigned;                      /*PROGRAM REGION BASE REGISTER
    CR_P0LR longword unsigned;                      /*PROGRAM REGION LIMIT REGISTER
    CR_P1BR longword unsigned;                      /*CONTROL REGION BASE REGISTER
    CR_P1LR longword unsigned;                      /*CONTROL REGION LIMIT REGISTER
    CR_SBR longword unsigned;                       /*SYSTEM BASE REGISTER
    CR_SLR longword unsigned;                       /*SYSTEM LIMIT REGISTER
    CR_PCBB longword unsigned;                      /*PROCESS CONTROL BLOCK BASE REGISTER
    CR_SCBB longword unsigned;                      /*SYSTEM CONTROL BLOCK BASE REGISTER
    CR_ASTLVL longword unsigned;                    /*AST DELIVERY LEVEL REGISTER
    CR_SISR longword unsigned;                      /*SOFTWARE INTERRUPT SUMMARY REGISTER
    CR_ICCS longword unsigned;                      /*INTERVAL TIMER CONTROL STATUS REGISTER
    CR_CPUREG longword unsigned;                    /*START OF CPU-SPECIFIC IPR'S
end EMBCRDEF;

aggregate EMBCRDEF1 structure prefix EMB$;
    FILL_1 byte dimension 148 fill prefix EMBCRDEF tag $$;
    CR_ICR longword unsigned;                       /*INTERVAL COUNT REGISTER
    CR_TODR longword unsigned;                      /*TIME OF DAY REGISTER
    CR_ACCS longword unsigned;                      /*ACCELERATOR CONTROL REGISTER
    CR_SBIFS longword unsigned;                     /* SBI FAULT STATUS
    CR_SBISC longword unsigned;                     /* SBI COMPARATOR REGISTER
    CR_SBIMT longword unsigned;                     /* SBI MAINT REGISTER
    CR_SBIER longword unsigned;                     /* SBI ERROR REGISTER
    CR_SBITA longword unsigned;                     /* SBI TIMEOUT ADDR REGISTER
    CR_SBIS longword unsigned dimension 16;         /* SBI SILO
end EMBCRDEF1;

aggregate EMBCRDEF2 structure prefix EMB$;
    FILL_2 byte dimension 148 fill prefix EMBCRDEF tag $$;
    FILL_4 byte dimension 12 fill prefix EMBCRDEF tag $$;/*Allow room for ICR,TODR,ACCS
    CR_TBDR longword unsigned;                      /* TB DISABLE REGISTER
    CR_CADR longword unsigned;                      /* CACHE DISABLE REGISTER
    CR_MCESR longword unsigned;                     /* MACHINE CHECK ERROR SUMMARY
    CR_CAER longword unsigned;                      /* CACHE ERROR REGISTER
    CR_CMIERR longword unsigned;                    /* CMI ERROR SUMMARY REGISTER
                                                    /* 16 UNUSED LONGWDS IN EMB
end EMBCRDEF2;

aggregate EMBCRDEF3 structure prefix EMB$;
    FILL_3 byte dimension 244 fill prefix EMBCRDEF tag $$;
    CR_CODE longword unsigned;                      /*BUGCHECK/CRASH CODE
    CR_PID longword unsigned;                       /*CURRENT PROCESS ID
    CR_LNAME character length 16;                   /*CURRENT PROCESS NAME
    constant CR_LENGTH equals . prefix EMB$ tag K;  /*SIZE OF FIXED PART OF BUGCHECK MESSAGE
    constant CR_LENGTH equals . prefix EMB$ tag C;  /*SIZE OF FIXED PART OF BUGCHECK MESSAGE
end EMBCRDEF3;

end_module $EMBCRDEF;

module $EMBDVDEF;

/*
```

```
/* DEVICE ERROR MESSAGE BUFFER FORMAT (ERROR AND TIMEOUT)
/*


aggregate EMBDVDEF structure prefix EMB$;
    DV_SID longword unsigned;                       /*SYSTEM ID
    DV_ENTRY word unsigned;                         /*ENTRY TYPE (1=ERROR, 96=TIMEOUT)
    DV_TIME quadword unsigned;                      /*TIME OF ERROR
    DV_ERRSEQ word unsigned;                        /*ERROR SEQUENCE NUMBER
    DV_ERTCNT byte unsigned;                        /*REMAINING NUMBER OF ERROR RETRIES
    DV_ERTMAX byte unsigned;                        /*MAXIMUM NUMBER OF ERROR RETRIES
    DV_IOSB quadword unsigned;                      /*FINAL I/O STATUS
    DV_STS word unsigned;                           /*FINAL DEVICE STATUS
    DV_CLASS byte unsigned;                         /*DEVICE CLASS
    DV_TYPE byte unsigned;                          /*DEVICE TYPE
    DV_RQPID longword unsigned;                     /*REQUESTER PROCESS ID
    DV_BOFF word unsigned;                          /*BYTE OFFSET IN PAGE
    DV_BCNT word unsigned;                          /*TRANSFER BYTE COUNT
    DV_MEDIA longword unsigned;                     /*STARTING MEDIA ADDRESS
    DV_UNIT word unsigned;                          /*PHYSICAL UNIT NUMBER
    DV_ERRCNT word unsigned;                        /*UNIT ERROR COUNT
    DV_OPCNT longword unsigned;                     /*UNIT OPERATION COUNT
    DV_OWNUIC longword unsigned;                    /*VOLUME OWNER UIC
    DV_CHAR longword unsigned;                      /*DEVICE CHARACTERISTICS
    DV_SLAVE byte unsigned;                         /*SLAVE CONTROLLER NUMBER
    FILL_1 byte fill prefix EMBDVDEF tag $$;        /*SPARE UNUSED BYTES
    DV_FUNC word unsigned;                          /*I/O FUNCTION VALUE
    DV_NAME character length 16;                    /*DEVICE NAME
    DV_REGSAV longword unsigned;                    /*START OF REGISTER SAVE AREA
end EMBDVDEF;

end_module $EMBDVDEF;

module $EMBTSDEF;

/*
/* TIME STAMP MSG FORMAT
/*


aggregate EMBTSDEF structure prefix EMB$;
    TS_SID longword unsigned;                       /*SYSTEM ID
    TS_ENTRY word unsigned;                         /*ENTRY TYPE
    TS_TIME quadword unsigned;                      /*TIME IN 64 BITS
    TS_ERRSEQ word unsigned;                        /*ERROR SEQ !
    constant TS_LENGTH equals . prefix EMB$ tag K;  /*LENGTH OF TIME STAMP MSG
    constant TS_LENGTH equals . prefix EMB$ tag C;  /*LENGTH OF TIME STAMP MSG
end EMBTSDEF;

end_module $EMBTSDEF;

module $EMBSSDEF;

/*
```

```
/* SYSTEM SERVICE MESSAGE
/*
/*        NOTE:        SYSTEM SERVICE MESSAGE COVERS:
/*
/*                1) THE MESSAGES FROM THE SERVICE
/*                2) OPERATOR MESSAGES
/*                3) NETWORK MESSAGES
/*
/*        ONLY THE TYPE FIELD IS DIFERENT
/*



aggregate EMBSSDEF structure prefix EMB$;
    SS_SID longword unsigned;                          /*SYSTEM ID
    SS_ENTRY word unsigned;                            /*ENTRY TYPE
    SS_TIME quadword unsigned;                         /*TIME IN 64 BITS
    SS_ERRSEQ word unsigned;                           /*ERROR SEQUENCE NUMBER
    SS_MSGSZ word unsigned;                            /*MESSAGE TEXT SIZE IN BYTES
    constant SS_LENGTH equals . prefix EMB$ tag K;     /*LENGTH OF CONSTANT PART
    constant SS_LENGTH equals . prefix EMB$ tag C;     /*LENGTH OF CONSTANT PART
    SS_MSGTXT byte unsigned;                           /*FIRST BYTE OF MESSAGE TEXT
end EMBSSDEF;

end_module $EMBSSDEF;

module $EMBVMDEF;

/*
/* VOLUME MOUNT/DISMOUNT MESSAGE TYPE
/*



aggregate EMBVMDEF structure prefix EMB$;
    VM_SID longword unsigned;                          /*SYSTEM ID
    VM_ENTRY word unsigned;                            /*ENTRY TYPE = EMB$K_VM OR EMB$K_VD
    VM_TIME quadword unsigned;                         /*TIME IN 64 BIT FORMAT
    VM_ERRSEQ word unsigned;                           /*ERROR SEQUENCE NUMBER
    VM_OWNUIC longword unsigned;                       /*OWNER UIC OF THE VOLUME
    VM_ERRCNT longword unsigned;                       /*UNIT ERROR COUNT FROM UCB
    VM_OPRCNT longword unsigned;                       /*UNIT OPERATION COUNT FROM UCB
    VM_UNIT word unsigned;                             /*DEVICE UNIT NUMBER
    VM_NAMLNG byte unsigned;                           /*LENGTH OF DEVICE GENERIC NAME
    VM_NAMTXT character length 15;                     /*DEVICE GENERIC NAME
    VM_VOLNUM word unsigned;                           /*VOLUME NUMBER WITHIN SET
    VM_NUMSET word unsigned;                           /*NUMBER OF VOLUMES WITHIN SET
    VM_LABEL character length 12;                      /*VOLUME LABEL
    constant VM_LENGTH equals . prefix EMB$ tag K;     /*LENGTH OF BUFFER
    constant VM_LENGTH equals . prefix EMB$ tag C;     /*LENGTH OF BUFFER
end EMBVMDEF;

end_module $EMBVMDEF;

module $EMBSUDEF;
/*
```

```
/* SYSTEM STARTUP MESSAGE
/*


aggregate EMBSUDEF structure prefix EMB$;
    SU_SID longword unsigned;                          /*SYSTEM ID
    SU_ENTRY word unsigned;                            /*ENTRY TYPE (IE: BOOT OR POWER RECOVERY)
    SU_TIME quadword unsigned;                         /*CONTENTS OF SYSTEM TIME QUADWORD
    SU_ERRSEQ word unsigned;                           /*ERROR SEQUENCE NUMBER
    SU_DAYTIM longword unsigned;                       /*CONTENTS OF TIME OF DAY CLOCK
    constant SU_LENGTH equals . prefix EMB$ tag K;     /*LENGTH OF MESSAGE
    constant SU_LENGTH equals . prefix EMB$ tag C;     /*LENGTH OF MESSAGE
end EMBSUDEF;

end_module $EMBSUDEF;

module $EMBMCDEF;


/*
/* MACHINE CHECK LOG BUFFER FORMAT
/*



aggregate EMBMCDEF structure prefix EMB$;
    MC_SID longword unsigned;                          /*SYSTEM ID
    MC_ENTRY word unsigned;                            /*ENTRY TYPE
    MC_TIME quadword unsigned;                         /*TIME IN 64 BITS
    MC_ERRSEQ word unsigned;                           /*ERROR SEQUENCE NUMBER
    MC_SUMCOD byte unsigned;                           /*SUMMARY CODE
    MC_TOPF byte unsigned;                             /*TIME OUT PENDING FLAG
    MC_OPCODE byte unsigned;                           /*OPCODE OF INSTRUCTION CAUSING CHECK
    MC_CACHEF byte unsigned;                           /*CACHE DISABLE FLAG, 1=GROUP 0, 2=G 1
    MC_CES longword unsigned;                          /*CPU ERROR STATUS
    MC_UPC longword unsigned;                          /*MICRO-PC AT FAULT TIME
    MC_VA longword unsigned;                           /*VIRTUAL ADDRESS AT FAULT TIME
    MC_D longword unsigned;                            /*CPU D REGISTER AT FAULT TIME
    MC_TBER0 longword unsigned;                        /*TRANSLATION BUFFER STATUS REG 0
    MC_TBER1 longword unsigned;                        /*TRANSLATION BUFFER STATUS REG 1
    MC_TIMOAD longword unsigned;                       /*PHYSICAL ADDRESS CAUSING SBI TIMEOUT
    MC_PARITY longword unsigned;                       /*CACHE STATUS REGISTER
    MC_SBIERR longword unsigned;                       /*SBI ERROR REGISTER
    MC_PC longword unsigned;                           /*PC OF INSTRUCTION CAUSING CHECK
    MC_PSL longword unsigned;                          /*PSL OF MACHINE AT FAULT TIME
    constant MC_LENGTH equals . prefix EMB$ tag K;     /*LENGTH OF MACHINE CHECK FRAME
    constant MC_LENGTH equals . prefix EMB$ tag C;     /*LENGTH OF MACHINE CHECK FRAME
end EMBMCDEF;

end_module $EMBMCDEF;

module $EMBSEDEF;

/*
/* SOFT ECC DETECTED ERRORS AND SBI ALERT BUFFER FORMAT
/*
```

```
aggregate EMBSEDEF structure prefix EMB$;
    SE_SID longword unsigned;                           /*SYSTEM ID
    SE_ENTRY word unsigned;                             /*ENTRY TYPE
    SE_TIME quadword unsigned;                          /*TIME IN 64 BITS
    SE_ERRSEQ word unsigned;                            /*ERROR SEQUENCE NUMBER
    SE_NUMCON longword unsigned;                        /*NUMBER OF MEMORY CONTROLLERS
    constant SE_LENGTH equals . prefix EMB$ tag K;      /*LENGTH OF FIXED PART OF MSG
    constant SE_LENGTH equals . prefix EMB$ tag C;      /*LENGTH OF FIXED PART OF MSG
    SE_TR longword unsigned;                            /*ADAPTOR TR NUMBER
    SE_A longword unsigned;                             /*MEMORY REGISTER A
    SE_B longword unsigned;                             /*MEMORY REGISTER B
    SE_C longword unsigned;                             /*MEMORY REGISTER C
    SE_PC longword unsigned;                            /*PC OF INSTRUCTION AT FAULT TIME
    SE_PSL longword unsigned;                           /*PSL OF MACHINE AT FAULT TIME
end EMBSEDEF;

end_module $EMBSEDEF;

module $EMBSBDEF;

/*
/* SBI FAULT BUFFER FORMAT AND ASYNCHRONOUS WRITE ERROR FORMAT
/*


aggregate EMBSBDEF structure prefix EMB$;
    SB_SID longword unsigned;                           /*SYSTEM ID
    SB_ENTRY word unsigned;                             /*ENTRY TYPE
    SB_TIME quadword unsigned;                          /*TIME IN 64 BITS
    SB_ERRSEQ word unsigned;                            /*ERROR SEQUENCE NUMBER
    SB_FAULT longword unsigned;                         /*SBI FAULT/STATUS REGISTER
    SB_SILCMP longword unsigned;                        /*SBI SILO COMPARATOR
    SB_MAINT longword unsigned;                         /*SBI MAINTENANCE
    SB_ERROR longword unsigned;                         /*SBI ERROR REG
    SB_TIMOUT longword unsigned;                        /*SBI TIMEOUT REG
    SB_SILO longword unsigned dimension 16;             /*SBI SILO REG
    SB_SBIRGS longword unsigned dimension 16;           /*REGISTER A'S ON BUS (OR 0)
    SB_PC longword unsigned;                            /*PC OF INSTRUCTION AT FAULT TIME
    SB_PSL longword unsigned;                           /*PSL OF MACHINE AT FAULT TIME
    constant SB_LENGTH equals . prefix EMB$ tag K;      /*LENGTH OF SBI ERROR BUFFER
    constant SB_LENGTH equals . prefix EMB$ tag C;      /*LENGTH OF SBI ERROR BUFFER
end EMBSBDEF;

end_module $EMBSBDEF;

module $EMBUIDEF;

/*
/* UNDEFINED ADAPTER INTERRUPT BUFFER FORMAT
/*
```

```
aggregate EMBUIDEF structure prefix EMB$;
    UI_SID longword unsigned;                           /*SYSTEM ID
    UI_ENTRY word unsigned;                             /*ENTRY TYPE
    UI_TIME quadword unsigned;                          /*TIME IN 64 BITS
    UI_ERRSEQ word unsigned;                            /*ERROR SEQUENCE NUMBER
    UI_TR longword unsigned;                            /*ADAPTER TR NUMBER
    UI_CSR longword unsigned;                           /*ADAPTER CONGIGURATION STATUS REGISTER
    constant UI_LENGTH equals . prefix EMB$ tag K;      /*LENGTH OF MESSAGE
    constant UI_LENGTH equals . prefix EMB$ tag C;      /*LENGTH OF MESSAGE
end EMBUIDEF;

end_module $EMBUIDEF;

module $EMBUEDEF;


/*
/* ERROR BUFFER FORMAT FOR UNIBUS ERROR SUMMARY REGISTER
/*        ***** USED ONLY BY 11/730 ****
/*


aggregate EMBUEDEF structure prefix EMB$;
    UE_SID longword unsigned;                           /*SYSTEM ID
    UE_ENTRY word unsigned;                             /*ENTRY TYPE
    UE_TIME quadword unsigned;                          /*TIME IN 64 BITS
    UE_ERRSEQ word unsigned;                            /*ERROR SEQUENCE NUMBER
    UE_UBERR longword unsigned;                         /*UNIBUS ERROR REGISTER
    constant UE_LENGTH equals . prefix EMB$ tag K;      /*LENGTH OF MESSAGE
    constant UE_LENGTH equals . prefix EMB$ tag C;      /*LENGTH OF MESSAGE
end EMBUEDEF;

end_module $EMBUEDEF;

module $EMBSPDEF;


/*
/* ERROR BUFFER FORMAT FOR SAVING SOFTWARE PARAMETERS FOR CLASS DRIVER THAT
/*        CORRESPOND TO A LOGGED MESSAGE (SEE EMBLMDEF BELOW) ORIGINATING
/*        IN AN INTELLIGENT MASS STORAGE CONTROLLER.
/*


aggregate EMBSPDEF structure prefix EMB$;
    SP_SID longword unsigned;                           /* System ID
    SP_ENTRY word unsigned;                             /* Entry type (of this errorlog buffer)
    SP_TIME quadword unsigned;                          /* Time this entry created
    SP_ERRSEQ word unsigned;                            /* Error Sequence Number
    SP_CLASS byte unsigned;                             /* Device Class
    SP_TYPE byte unsigned;                              /* Device Type
    SP_BOFF word unsigned;                              /* Byte OFFset of data transfer
    SP_BCNT longword unsigned;                          /* Byte Count of data transfer
    SP_MEDIA longword unsigned;                         /* Media address (LBN) of data transfer
    SP_RQPID longword unsigned;                         /* Requesting PID
```

```
    SP_IOSB quadword unsigned;                          /* Final I/O status
    SP_FUNC word unsigned;                              /* I/O function code
    SP_UNIT word unsigned;                              /* Unit number of drive
    SP_OPCNT longword unsigned;                         /* Cummulative operation count this unit
    SP_ERRCNT word unsigned;                            /* Cummulative error count for this unit
    SP_UCBSTS word unsigned;                            /* Copy of UCB$W_STS field
    SP_OWNUIC longword unsigned;                        /* Unit's owner's UIC
    SP_CHAR longword unsigned;                          /* Device Characteristics
    SP_CMDREF longword unsigned;                        /* Command Reference number (RSPID)
    SP_DEVNAM character length 16;                      /* Device name
    constant SP_LENGTH equals . prefix EMB$ tag K;
    constant SP_LENGTH equals . prefix EMB$ tag C;

end EMBSPDEF;

end_module $EMBSPDEF;

module $EMBLMDEF;

/*
/* LOGGED MESSAGE (DEVICE DEPENDENT CONTENTS).  DRIVER LOGS MESSAGE
/*        WHICH MAY COME DIRECT FROM INTELLIGENT MASS STORAGE CONTROLLER.
/*


aggregate EMBLMDEF structure prefix EMB$;
    LM_SID longword unsigned;                           /* System ID
    LM_ENTRY word unsigned;                             /* Entry type (i.e. Logged Message)
    LM_TIME quadword unsigned;                          /* Time this entry created
    LM_ERRSEQ word unsigned;                            /* Error sequence number
    LM_CLASS byte unsigned;                             /* Device Class
    LM_TYPE byte unsigned;                              /* Device Type
    LM_UNIT word unsigned;                              /* Device unit number
    LM_DEVNAM character length 16;                      /* Device name
    LM_MSGTYP word unsigned;                            /* Type of logged message
    constant LM_LENGTH equals . prefix EMB$ tag K;
    constant LM_LENGTH equals . prefix EMB$ tag C;

end EMBLMDEF;

end_modu e $EMBLMDEF;

module $EMBLTDEF;

/*
/* LOGGED MESSAGE MESSAGE TYPES
/*


constant DM     equals 01  prefix EMB tag $C;       /* Disk MSCP message
constant DM     equals 01  prefix EMB tag $K;       /* Disk MSCP message
constant TM     equals 02  prefix EMB tag $C;       /* Tape MSCP message
constant TM     equals 02  prefix EMB tag $K;       /* Tape MSCP message
constant PM     equals 03  prefix EMB tag $C;       /* Port (CI) message
constant PM     equals 03  prefix EMB tag $K;       /* Port (CI) message
```

```
constant UM      equals 04  prefix EMB tag $C;          /* Port (UDA) message
constant UM      equals 04  prefix EMB tag $K;          /* Port (UDA) message
constant AVATN   equals 05  prefix EMB tag $C;          /* Available Attention Message
constant AVATN   equals 05  prefix EMB tag $K;          /* Available Attention Message
constant DUPUN   equals 06  prefix EMB tag $C;          /* Duplicate Unit ! Attention Message
constant DUPUN   equals 06  prefix EMB tag $K;          /* Duplicate Unit ! Attention Message
constant IVCMD   equals 07  prefix EMB tag $C;          /* Invalid Command Log message.
constant IVCMD   equals 07  prefix EMB tag $K;          /* Invalid Command Log message.
constant ACPTH   equals 08  prefix EMB tag $C;          /* Access Path Attention Message
constant ACPTH   equals 08  prefix EMB tag $K;          /* Access Path Attention Message
constant INVSTS  equals 09  prefix EMB tag $C;          /* Invalid Status in End Message
constant INVSTS  equals 09  prefix EMB tag $K;          /* Invalid Status in End Message
constant INVATT  equals 10  prefix EMB tag $C;          /* Invalid Attention Message
constant INVATT  equals 10  prefix EMB tag $K;          /* Invalid Attention Message
constant NOUNIT_DG      equals 11  prefix EMB tag $C;   /* No unit in Datagram
constant NOUNIT_DG      equals 11  prefix EMB tag $K;   /* No unit in Datagram

end_module $EMBLTDEF;

module $EMBETDEF;
/*
/* ERROR MESSAGE ENTRY TYPE DEFINITIONS
/*

constant DE      equals 01  prefix EMB tag $C;          /*DEVICE ERROR
constant DE      equals 01  prefix EMB tag $K;          /*DEVICE ERROR
constant MC      equals 02  prefix EMB tag $C;          /*MACHINE CHECK
constant MC      equals 02  prefix EMB tag $K;          /*MACHINE CHECK
constant BE      equals 04  prefix EMB tag $C;          /*BUS ERROR
constant BE      equals 04  prefix EMB tag $K;          /*BUS ERROR
constant SA      equals 05  prefix EMB tag $C;          /*SBI ALERT
constant SA      equals 05  prefix EMB tag $K;          /*SBI ALERT
constant SE      equals 06  prefix EMB tag $C;          /*SOFT ECC ERROR
constant SE      equals 06  prefix EMB tag $K;          /*SOFT ECC ERROR
constant AW      equals 07  prefix EMB tag $C;          /*ASYNCHRONOUS WRITE ERROR
constant AW      equals 07  prefix EMB tag $K;          /*ASYNCHRONOUS WRITE ERROR
constant HE      equals 08  prefix EMB tag $C;          /*HARD ECC ERROR
constant HE      equals 08  prefix EMB tag $K;          /*HARD ECC ERROR
constant UBA     equals 09  prefix EMB tag $C;          /* 11/780 Unibus Adapter error
constant UBA     equals 09  prefix EMB tag $K;          /* 11/780 Unibus Adapter error
constant SI      equals 10  prefix EMB tag $C;          /* 11/750 Fault through SBI vector
constant SI      equals 10  prefix EMB tag $K;          /* 11/750 Fault through SBI vector
constant UE      equals 11  prefix EMB tag $C;          /* 11/730 Unibus Error
constant UE      equals 11  prefix EMB tag $K;          /* 11/730 Unibus Error
constant MBA     equals 12  prefix EMB tag $C;          /* 11/780 Massbus Adapter Error
constant MBA     equals 12  prefix EMB tag $K;          /* 11/780 Massbus Adapter Error
constant SBIA    equals 13  prefix EMB tag $C;          /* 11/790 SBIA error
constant SBIA    equals 13  prefix EMB tag $K;          /* 11/790 SBIA error
constant CRD     equals 14  prefix EMB tag $C;          /* 11/790 CRD log
constant CRD     equals 14  prefix EMB tag $K;          /* 11/790 CRD log
constant EMM     equals 15  prefix EMB tag $C;          /* 11/790 Environmental MOnitor
constant EMM     equals 15  prefix EMB tag $K;          /* 11/790 Environmental MOnitor
constant HLT     equals 16  prefix EMB tag $C;          /* 11/790 Processor Error Halt
constant HLT     equals 16  prefix EMB tag $K;          /* 11/790 Processor Error Halt
constant CRBT    equals 17  prefix EMB tag $C;          /* 11/790 Console Reboot
```

```
constant CRBT    equals 17  prefix EMB tag $K;          /* 11/790 Console Reboot
constant CS      equals 32  prefix EMB tag $C;          /*COLD START (IE: SYSTEM BOOT)
constant CS      equals 32  prefix EMB tag $K;          /*COLD START (IE: SYSTEM BOOT)
constant NF      equals 35  prefix EMB tag $K;          /*NEW FILE CREATED
constant NF      equals 35  prefix EMB tag $C;          /*NEW FILE CREATED
constant WS      equals 36  prefix EMB tag $C;          /*WARM START (IE: SYSTEM POWER RECOVERY)
constant WS      equals 36  prefix EMB tag $K;          /*WARM START (IE: SYSTEM POWER RECOVERY)
constant CR      equals 37  prefix EMB tag $C;          /*CRASH RE-START
constant CR      equals 37  prefix EMB tag $K;          /*CRASH RE-START
constant TS      equals 38  prefix EMB tag $C;          /*TIME STAMP ENTRY
constant TS      equals 38  prefix EMB tag $K;          /*TIME STAMP ENTRY
constant SS      equals 39  prefix EMB tag $C;          /*SYSTEM SERVICE MESSAGE
constant SS      equals 39  prefix EMB tag $K;          /*SYSTEM SERVICE MESSAGE
constant SBC     equals 40  prefix EMB tag $C;          /*SYSTEM BUGCHECK
constant SBC     equals 40  prefix EMB tag $K;          /*SYSTEM BUGCHECK
constant OM      equals 41  prefix EMB tag $C;          /*OPERATOR MESSAGE
constant OM      equals 41  prefix EMB tag $K;          /*OPERATOR MESSAGE
constant NM      equals 42  prefix EMB tag $C;          /*NETWORK MESSAGE
constant NM      equals 42  prefix EMB tag $K;          /*NETWORK MESSAGE
constant VM      equals 64  prefix EMB tag $C;          /*VOLUME MOUNT
constant VM      equals 64  prefix EMB tag $K;          /*VOLUME MOUNT
constant VD      equals 65  prefix EMB tag $C;          /*VOLUME DISMOUNT
constant VD      equals 65  prefix EMB tag $K;          /*VOLUME DISMOUNT
constant DT      equals 96  prefix EMB tag $C;          /*DEVICE TIMEOUT
constant DT      equals 96  prefix EMB tag $K;          /*DEVICE TIMEOUT
constant UI      equals 97  prefix EMB tag $C;          /*UNDEFINED INTERRUPT
constant UI      equals 97  prefix EMB tag $K;          /*UNDEFINED INTERRUPT
constant DA      equals 98  prefix EMB tag $C;          /* Asynchronous Device Attention
constant DA      equals 98  prefix EMB tag $K;          /* Asynchronous Device Attention
constant SP      equals 99  prefix EMB tag $C;          /* Software Parameters
constant SP      equals 99  prefix EMB tag $K;          /* Software Parameters
constant LM      equals 100  prefix EMB tag $C;         /* Logged Message
constant LM      equals 100  prefix EMB tag $K;         /* Logged Message
constant LOGMSCP equals 101  prefix EMB tag $C;         /* Logged MSCP Message
constant LOGMSCP equals 101  prefix EMB tag $K;         /* Logged MSCP Message
constant UBC     equals 112  prefix EMB tag $C;         /*USER BUGCHECK
constant UBC     equals 112  prefix EMB tag $K;         /*USER BUGCHECK

end_module $EMBETDEF;
```

```
module $EO1DEF;
/**
/* EOF1 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE FIRST LABEL IN FILE TRAILER LABEL SET.  IT IS EQUIVALENT TO
/* HDR1 EXCEPT FOR THE FOLLOWING FIELDS.
/*-


aggregate EO1DEF structure prefix EO1$;
    EO1LID longword unsigned;                       /*LABEL IDENTIFIER AND NUMBER 'EOF1'
    FILL_1 character length 50 fill prefix EO1DEF tag $$;/*SAME AS HDR1
    BLOCKCNT character length 6;                    /*BLOCK COUNT
end EO1DEF;

end_module $EO1DEF;
```

```
module $EO2DEF;

/**
/* EOF2 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE SECOND LABEL IN THE FILE TRAILER LABEL SET.  IT IS EQUIVALENT
/* TO HDR2 EXCEPT FOR THE FOLLOWING FIELDS.
/*-


aggregate EO2DEF union prefix EO2$;
    EO2LID longword unsigned;                    /*LABEL IDENTIFIER AND NUMBER 'EOF2'
end EO2DEF;

end_module $EO2DEF;
```

```
module $EO3DEF;

/**
/* EOF3 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE THIRD LABEL IN THE FILE TRAILER LABEL SET.  IT IS EQUIVALENT
/* TO HDR3 EXCEPT FOR THE FOLLOWING FIELDS.
/*-


aggregate EO3DEF union prefix EO3$;
    EO3LID longword unsigned;                       /*LABEL IDENTIFIER AND NUMBER 'EOF3'
end EO3DEF;

end_module $EO3DEF;
```

```
module $EO4DEF;

/**
/* EOF4 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE FOURTH LABEL IN THE FILE TRAILER LABEL SET.  IT IS EQUIVALENT
/* TO HDR4 EXCEPT FOR THE FOLLOWING FIELDS.
/*-


aggregate EO4DEF union prefix EO4$;
    EO4LID longword unsigned;                       /*LABEL IDENTIFIER AND NUMBER 'EOF4'
end EO4DEF;

end_module $EO4DEF;
```

```
module $ERLDEF;
/*
/* ERROR LOG ALLOCATION BUFFER HEADER
/*



aggregate ERLDEF structure prefix ERL$;
    BUSY byte unsigned;                                 /*NUMBER OF BUSY MESSAGES IN BUFFER
    MSGCNT byte unsigned;                               /*NUMBER OF COMPLETED MESSAGES IN BUFFER
    BUFIND byte unsigned;                               /*BUFFER INDICATOR OF RESPECTIVE BUFFER
    FLAGS byte unsigned;                                /*BUFFER CONTROL FLAGS
    NEXT longword unsigned;                             /*ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER
    END_OVERLAY union fill;
        "END" longword unsigned;                        /*ADDRESS OF END OF BUFFER + 1
        constant 'LENGTH' equals . prefix ERL$ tag K;   /*LENGTH OF ALLOCATION BUFFER HEADER
        constant 'LENGTH' equals . prefix ERL$ tag C;   /*LENGTH OF ALLOCATION BUFFER HEADER

        END_BITS structure fill;
            LOCK bitfield mask;                         /*BUFFER ALLOCATION INTERLOCK
            TIMER bitfield mask;                        /*TIMER ACTIVE
        end END_BITS;

    end END_OVERLAY;
end ERLDEF;

end_module $ERLDEF;
```

SYSDEFAE
SDL

IMGACTCTX
SDL

SYSDEFFL
SDL