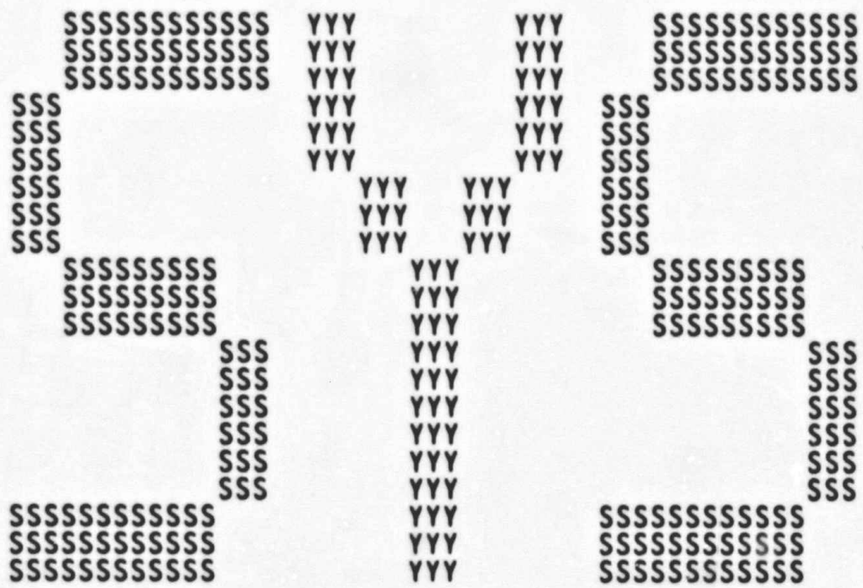
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

FILEID**SYSDEFQZ

```
  SSSSSSSS  YY      YY   SSSSSSSS  DDDDDDDD   EEEEEEEEEE  FFFFFFFFFF    QQQQQQ    ZZZZZZZZZZ
  SSSSSSSS  YY      YY   SSSSSSSS  DDDDDDDD   EEEEEEEEEE  FFFFFFFFFF    QQQQQQ    ZZZZZZZZZZ
SS          YY      YY  SS        DD      DD  EE          FF          QQ      QQ          ZZ
SS          YY      YY  SS        DD      DD  EE          FF          QQ      QQ          ZZ
SS            YY  YY    SS        DD      DD  EE          FF          QQ      QQ        ZZ
SS            YY  YY    SS        DD      DD  EE          FF          QQ      QQ        ZZ
  SSSSSS        YY        SSSSSS    DD      DD  EEEEEEEE    FFFFFFFF    QQ      QQ      ZZ
  SSSSSS        YY        SSSSSS    DD      DD  EEEEEEEE    FFFFFFFF    QQ      QQ      ZZ
        SS      YY              SS  DD      DD  EE          FF          QQ  QQ  QQ    ZZ
        SS      YY              SS  DD      DD  EE          FF          QQ  QQ  QQ    ZZ
        SS      YY              SS  DD      DD  EE          FF          QQ    QQ    ZZ          ....
        SS      YY              SS  DD      DD  EE          FF          QQ    QQ    ZZ          ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD   EEEEEEEEEE  FF            QQQQ  QQ  ZZZZZZZZZZ  ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD   EEEEEEEEEE  FF            QQQQ  QQ  ZZZZZZZZZZ  ....

  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
```

```
SY

mo
/*
/*
/*

ag

/*
/*
/*

en

ag

/*
/*
/*

en

ag

/*

en
```

```
{
{ Version:      'V04-000'
{
{****************************************************************************
{*                                                                          *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
{*  ALL RIGHTS RESERVED.                                                    *
{*                                                                          *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
{*  TRANSFERRED.                                                            *
{*                                                                          *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
{*  CORPORATION.                                                            *
{*                                                                          *
{*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
{*                                                                          *
{****************************************************************************

{++
{
{ FACILITY:  VAX/VMS System Macro Libraries
{
{ ABSTRACT:
{
{       This file contains the SDL source for all operating system control
{       blocks, from Q to Z.  That is, all control blocks from QAA to ZZZ.
{
{ ENVIRONMENT:
{
{       n/a
{
{--
{
{
{ AUTHOR:  The VMS Group        CREATION DATE:  1-Aug-1976
{
{ MODIFIED BY:
{
{       V03-125 ROW0410         Ralph O. Weber          6-AUG-1984
{               Define a UCB$L_STS bit indicating VAXcluster state transition
{               processing in progress, UCB$V_CLUTRAN.  This bit is used to
{               force mount verification to execute after (or as a part of) a
{               VAXcluster state transition.
{
{       V03-124 ACG0441         Andrew C. Goldstein,    2-Aug-1984  15:16
{               Add CLUSLOCK flag to VCB, DISMOUNT flag to UCB
{
{       V03-123 TCM0005         Trudy C. Matthews       23-Jul-1984  16:50
```

```
                Add RPB$B_CTRLLTR to $RPBDEF.

        V03-122 ACG0438         Andrew C. Goldstein,    23-Jul-1984  15:16
                Add VCA structures for cache interlocks

        V03-121 MMD0314         Meg Dumont,     19-Jul-1984  12:23
                Add $XGDEF to LIB.ML

        V03-120 ROW392          Ralph O. Weber          19-JUL-1984
                Add UCB$W_DEVSTS bit UCB$V_MSCP_WRTP which is the inclusive or
                of all the various MSCP write protection bits.

        V03-119 LY0501          Larry Yetto              6-JUL-1984 13:11
                Add subfields to UCB$L_MEDIA_ID

        V03-118 ROW0379         Ralph O. Weber          21-JUN-1984
                Add UCB$W_2P_MSCPUNIT to provide for future implementation of
                dual-path UDA support via MSCP unit number switching in the
                MSCP class driver.

        V03-117 ROW0357         Ralph O. Weber           1-MAY-1984
                Add UCB$V_MNTVERPND in UCB$L_STS.  This bit will be set along
                with UCB$V_MNTVERIP when it is necessary to stall a busy
                device due to loss of quorum.  Also add UCB$V_MSCP_PKACK in
                UCB$W_DEVSTS for MSCP devices.  This will be used by the disk
                class driver to indicate that a IO$_PACKACK operation is in
                progress.

        V03-116 CDS0012         Christian D. Saether    18-Apr-1984
                Add wcb$v_noacclock, remove wcb$v_acclkid.

        V03-115 GRR4115         Gregory R. Robert       12-Apr-1984
                Split $SMBDEF into symbiont services and print symbiont
                sections.  Move latter into $PSMDEF.

        V03-114 LJK0265         Lawrence J. Kenah       9-Apr-1984
                Add SHL_SIZE cell and OLD_SHL_SIZE constant to $SHLDEF
                to allow image activator and ANALYZE /IMAGE to handle
                shareable image list elements of different sizes.

        V03-113 LMP0221         L. Mark Pilant,         7-Apr-1984  13:13
                Add a pointer to the ORB in the UCB.  Also remove the
                definitions, in the UCB, of the owner UIC, volume
                protection, and the ACL queue segment list head.

        V03-112 ROW0337         Ralph O. Weber           7-APR-1984
                Remove UCB$L_CANLINK.  This field is no longer used.

        V03-111 MHB0118         Mark Bramhall           26-Mar-1984
                Add UAF$V_DISRECONNECT.

        V03-110 MIR0370         Michael I. Rosenblum    20-Mar-1984
                Add definitions for TTY$V_ST_TABRIGHT and CTSLOW
                Add UCB cell to store the cursor positioning from the
                last read.
```

```
V03-109 SSA0020         Stan Amway              20-Mar-1984
        Moved UCB$W_QLEN from disk/tape extension to main UCB.
        Changed UCB$W_DCCB in disk/tape extension to UCB$L_DCCB.

V03-108 ROW0325         Ralph O. Weber          19-MAR-1984
        Remove UCB$L_MVIOQFL and UCB$L_MVIOQBL; these fields are no
        longer used.  Add UCB$Q_MSCP_RESV, eight bytes of space
        reserved for use by new MSCP features implemented during the
        Version 4 life-time.  Add VCB$L_SHAD_RESV to the standard
        disk VCB and VCB$Q_SHDM_RESV to the shadow set member VCB,
        both fields provide reserved space for implementation of
        volume shadowing during the Version 4 life-time.

V03-107 JWT0163         Jim Teague              10-Mar-1984
        Enlarge name fields in shareable image list ($SHLDEF)

V03-106 ACG0400         Andrew C. Goldstein,    10-Mar-1984  2:01
        Add quota cache structures for cluster-wide quota cacheing

V03-105 WHM0001         Bill Matthews           02-Mar-1984
        Added field SLV$A_SYSVECS( address of vectors in SYS.EXE)
        to $SLVDEF.

V03-104 CDS0011         Christian D. Saether     8-Mar-1984
        Add WRITE_TURN flag to WCB.

V03-103 PRD0072         Paul R. DeStefano       27-Feb-1984
        Added SB$L_CSB (link to newest Cluster System Block)
        to $SBDEF.

V03-102 ROW0316         Ralph O. Weber          27-FEB-1984
        Change UCB$V_TU_OVRSEQCHK to UCB$V_TU_OVRSQCHK in
        UCB$W_DEVSTS.  Add a fourth type of VCB for shadow set
        members.  Add fields to the disk VCB for shadow set and
        enhanced mount verification handling.

V03-101 SSA0011         Stan Amway              27-Feb-1984
        Changed UCB$L_DCCB in disk/tape UCB extension to UCB$W_DCCB.
        Moved UCB$W_QLEN to disk/tape UCB extension.
        Deleted UCB$L_WRTCNT.

V03-100 MMD0244         Meg Dumont,     24-Feb-1984  14:59
        Add support for VCB$V_FIL_ACCESS

V03-099 SSA0010         Stan Amway              14-Feb-1984
        Added UCB$V_DATACACHE as a disk-specific bit in UCB$W_DEVSTS.
        Added UCB$L_DCCB in disk/tape UCB extension.

        Added UCB$L_WRTCNT & UCB$W_QLEN to support MONITOR disk class.
        (Done on behalf of Tom Cafarella.)

V03-098 ROW0303         Ralph O. Weber          10-FEB-1984
        Add definition of UCB$M_AST_ARMED defined as the sign bit of
        UCB$W_DIRSEQ.

V03-097 ROW0300         Ralph O. Weber           9-FEB-1984
```

```
                Add UCB$V_MSCP_FLOVR, a bit which a MSCP driver toggles
                (changes the state of) whenever a device sucessfully moves
                from one controller to another.

        V03-096 ROW0297         Ralph O. Weber          7-FEB-1984
                Add UCB$V_TU_SEQNOP a device dependent status bit which the
                tape class driver sets when the wait count is bumped due to a
                sequential nop operation being in progress.

        V03-095 PCG0001         Peter George            06-Feb-1984
                Add UAF$V_DISREPORT.

        V03-094 LMP0188         L. Mark Pilant,         4-Feb-1984  11:19
                Add a classification block to the VCB.

        V03-093 TMK0004         Todd M. Katz            04-Feb-1984
                Add a NI device extension to $UCBDEF.

        V03-092 KPL0100         Peter Lieberwirth       2-Feb-1984
                Lengthen RPB BOOTNDT field to a word.  Leave byte definition
                as well.  Only newly-defined FLAGS byte need move as a result,
                it shifts left one byte.

        V03-091 ROW0282         Ralph O. Weber          14-JAN-1984
                Add UCB$V_SUPMVMSG bit to UCB$L_STS.  When set, this bit
                supresses success status messages from mount verification.
                Add UCB$L_WAIT_CDDB to the MSCP UCB extension.  This field
                points to a CDDB which is waiting for mount verification to
                complete before beginning single CDRP processing.

        V03-090 ACG0385         Andrew C. Goldstein,    9-Jan-1984  17:07
                Replace $UAFDEF (authorization file) with new V4 format

        V03-089 LMP0177         L. Mark Pilant,         7-Dec-1983  10:22
                Add an ACL queue listhead to the UCB.

        V03-088 CDS0010         Christian D. Saether    6-Dec-1983
                Add VCB$V_NOSHARE flag.

        V03-087 SRB0106         Steve Beckhardt         6-Dec-1983
                Made several changes to $RSBDEF.

        V03-086 ROW0256         Ralph O. Weber          16-NOV-1983
                Move the general disk UCB$W_DEVSTS bit definitions from being
                a UCB$L_DPC subfield to being a UCB$W_DEVSTS subfield.  Also
                redo the MSCP and TU device status bit definitions to
                accomodate these recently discovered disk device status bits.

        V03-085 ROW0255         Ralph O. Weber          14-NOV-1983
                Restore UCB$W_EC1 and UCB$W_EC2 symbols which were lost during
                the preparation of ROW0253.

        V03-084 TMK0003         Todd M. Katz            12-Nov-1983
                Lets try again and this time fix both TMK0002 and ROW0253.
                My previous attempt doubly defined the symbol UCB$K_2P_LENGTH.
                Replace one of the definitions with local symbol #2P_LENGTH.
```

```
        V03-083 TMK0002         Todd M. Katz            12-Nov-1983
                Fix ROW0253 and system build by defining UCB$K_2P_LENGTH.

        V03-082 ROW0253         Ralph O. Weber          11-NOV-1983
                Make changes to the UCB to accomidate more class driver fields
                becoming publicly available.  Make DEVSTS definitions for
                class driver bits.  Add 2P synonyms for all dual-path fields.
                Restructure disk and tape specific sections of the device
                dependent UCB to add class driver fields and to overlay some
                unused local-disk fields with these class driver fields.

        V03-081 TMK0001         Todd M. Katz            26-Oct-1983
                Add UAF$L_JTQUOTA within $UAFDEF.

        V03-080 CDS0009         Christian D. Saether    14-Oct-1983
                Remove VCB$L_ALLOCLKID.  Add various VCB and RVT
                fields to support volume activity blocking for rebuild.

        V03-079 CDS0008         Christian D. Saether    10-Oct-1983
                Add VCB$L_ALLOCLKID.

        V03-078 JSV0411         Joost Verhofstad        27-SEP-1983
                Add UCB$W_JNL_PROT

        V03-077 GRR0005         Gregory R. Robert       26-Sep-1983
                Added $SMBDEF, public symbiont definitions.

        V03-076 CWH3076         CW Hobbs                10-Sep-1983
                Add a RPB$B_FLAGS field to $RPBDEF, and define a bit
                RPB$V_NOSYSDISK, which means that the boot volume is no
                longer present (e.g. for S/A BACKUP from console).

        V03-075 ROW0220         Ralph O. Weber          7-SEP-1983
                Switch bit positions of the Phase 1 in progress and Phase 2 in
                progress flags in the Replacement and Caching Table structure
                definition, $RCTDEF.  This causes the definition to conform
                with the DSDF specification, from which it is derived.

                Also define UCB$V_LCL_VALID in UCB$L_STS.  This bit being set
                indicates that one of the reasons a disk is volume valid is a
                PACKACK operation performed on the local node.

        V03-074 CDS0007         Christian D. Saether    24-Aug-1983
                Add RVT$T_VLSLCKNAM to the RVT.

        V03-073 ROW0211         Ralph O. Weber          16-AUG-1983
                Add two LENGTH symbols to $UCBDEF.  UCB$K_LCL_DISK_LENGTH is
                the length of a device-independent UCB for a local disk.
                UCB$K_LCL_TAPE_LENGTH is the length of a device-independent
                UCB for a local magnetic tape.  (Someday there will be
                different LENGTH symbols for remote disks and tapes.)

                Also make UCB$W_BCR have a longword form for the convenience
                of DBDRIVER.
```

```
{       V03-072 ROW0204         Ralph O. Weber          5-AUG-1983
{               Move UCB$L_CPID from its current overlay with UCB$L_DUETIM to
{               a new overlay with UCB$L_LOCKID.
{
{       V03-071 KTA3072         Kerbey T. Altmann       02-Aug-1983
{               Add UCB$B_ONLCNT - count of number of ONLINEs to
{               a disk or tape.
{               Change location of RVT$L_STRUCLKID added by CDS0006.
{
{       V03-070 CDS0006         Christian D. Saether    2-Aug-1983
{               Remove RVX structure.
{               Add RVT$L_STRUCLKID.
{
{       V03-069 NPK3029         N. kronenberg           29-JUL-1983
{               Redefine the SCS$C_ST status codes to conform to
{               the latest SCA format (error type*8+severity.)
{
{       V03-068 LY0401          Larry Yetto             29-JUL-1983 12:15:59
{               Add UCB$L_JNL_BTXSEQNO
{
{       V03-067 JLV0283         Jake VanNoy             28-JUL-1983
{               Correct fill problem in UCBDEF.
{
{       V03-066 JSV0367         Joost Verhofstad        28-JUL-1983
{               Change journal name length to 18
{
{       V03-065 MMD0190         Meg Dumont,     28-Jul-1983  9:45
{               Changed bit in VCBDEF from AUTO to NOAUTO to make mag tape
{               AVL/AVR consistent between DCL and MOUNT system service
{
{       V03-064 LY0398          Larry Yetto                     28-JUL-1983
{               Add UCB$L_JNL_WCBFL and UCB$L_JNL_WCBBL. Remove
{               VCB$L_JNL_WCBFL and VCB$L_JNL_WCBBL.
{
{       V03-063 CDS0005         Christian D. Saether            28-Jun-1983
{               Add VCB$L_VOLLKID, VCA$L_EXTCLKID, and VCA$L_FIDCLKID.
{
{       V03-062 MIR0052         Michael I. Rosenblum            27-Jun-1983
{               Add missing deffinitions for UCB$L_RTT_BANDEXCL and BANDEXMSK
{
{       V03-061 MIR0051         Michael I. Rosenblum            24-Jun-1983
{               Add $TTYUCBDEF to this module.  Also note this Definition
{               Must follow $UCBDEF as it gets a local symbol defined
{               in $UCBDEF.
{
{       V03-060 CDS0004         Christian D. Saether    23-Jun-1983
{               Add VCB$T_VOLCKNAM to end of vcb instead of in middle.
{
{       V03-059 RLRDPATH1       Robert L. Rappaport     23-Jun-1983
{               Add UCB$L_DP_ALTUCB to Dual Path section of UCB.
{
{       V03-058 CDS0003         Christian D. Saether    23-Jun-1983
{               Add VCB$T_VOLCKNAM field.
{
{       V03-057 WMC0055         Wayne Cardoza           21-Jun-1983
{               Increase number of vector pages in SGNDEF
```

```
{
{       V03-056 PRB0198         Paul R. Beck            13-JUN-1983 13:28
{               Add RUCBDEF and RUHDEF (formerly defined in [RUF.SRC]RUF.SDL)
{
{       V03-055 LY0381          Larry Yetto             13-JUN-1983 07:57:45
{               Add WCB$L_JNL_RC
{
{       V03-054 WMC0054         Wayne Cardoza           29-May-1983
{               New protection and spare fields in SLVDEF.
{
{       V03-053 MLJ0113         Martin L. Jack          27-May-1983
{               Add UAF$B_QUEPRI.
{
{       V03-052 RLRDPATH        Robert L. Rappaport     25-May-1983
{               Create a new UCB extension for Dual Ported Devices that
{               resides after the errorlogging extension and before
{               the DISK_UCB_EXTENSION.  Place in it UCB$L_DP_DDB and
{               UCB$L_DP_LINK.
{
{       V03-051 LY0375          Larry Yetto             24-MAY-1983 15:44:29
{               Add cluster write Q list head to UCB journal extention.
{               Also added status bits DIRENTER and CHKVAL in $RSBDEF.
{
{       V03-050 KTA3052         Kerbey T. Altmann       24-May-1983
{               Updated SCSDEF for split up of layers.
{
{       V03-049 LY0366          Larry Yetto             18-MAY-1983 17:08:36
{               Add UCB$V_JNL_UNMAST, UCB$L_JNL_RMBLK, UCB$L_JNL_ACBM,
{               and UCB$L_JNL_LSEQNO
{
{       V03-048 JLV0253         Jake VanNoy             18-MAY-1983
{               Add symbols to $TASTDEF.
{
{       V03-047 MMD0151         Meg Dumont,     26-Apr-1983  9:06
{               Add VCB$B_LBLCNT to the MTAACP portion of VCBDEF
{
{       V03-046 LY0356          Larry Yetto             21-APR-1983 08:16:48
{               Add UCB$L_JNL_FAILQFL and UCB$L_JNL_FAILQBL
{
{       V03-045 MSH0004         Maryann Hinden          14-Apr-1983
{               Add SPNB$W_REFC.
{
{       V03-044 JWH0208         Jeffrey W. Horn         12-Apr-1983
{               Add SLV$B_PROT and SLV$T_FACILTY to $SLVDEF.
{
{       V03-043 TCM0004         Trudy C. Matthews       12-Apr-1983
{               Add UCB$L_LOCKID to $UCBDEF.
{
{       V03-042 JWT0104         Jim Teague              29-Mar-1983
{               Add UAF$T_CLITABLES field to $UAFDEF.
{
{       V03-041 JSV0201         Joost Verhofstad        28-MAR-1983
{               Add monitor counts in UCB for journals
{
{       V03-040 ROW0171         Ralph O. Weber          25-MAR-1983
{               Extend UCB$L_DEVCHAR to a quadword, UCB$Q_DEVCHAR.
```

```
;               Also create symbol for second portion of the quadword,
;               UCB$L_DEVCHAR2.
;
;       V03-039 SRB0072         Steve Beckhardt         25-Mar-1983
;               Added some fields to $RSBDEF.
;
;       V03-038 MSH0003         Maryann Hinden          25-Mar-1983
;               Delete SPNB$L_HEADER definition, add SPNB$C_HDRSIZ.
;
;       V03-037 STJ3075         Steven Jeffreys,25-Mar-1983
;               - Add VCB$V_ERASE and VCB$V_NOHIGHWATER.
;
;       V03-036 MMD0108         Meg Dumont,     11-Mar-1983  12:32
;               Add defs to VL1DEF for handling of VOL1 owner id field
;               and added VL2DEF to handle the VOL2 label.
;
;       V03-035 MMD0106         Meg Dumont,     10-Mar-1983  9:55
;               Add fields in VCB for AVR, AVL and the VOL2, HDR4
;               additions to MTAACP
;
;       V03-034 SRB0069         Steve Beckhardt         9-Mar-1983
;               Removed SYSNAM status bit from $RSBDEF.
;
;       V03-033 LY0316          Larry Yetto             07-mar-1983
;               Add WCB$V_JDB field to WCB$B_JNL_STAT
;
;       V03-032 WMC0001         Wayne Cardoza           06-Mar-1983
;               Add UAF$B_MAXDETACH
;
;       V03-031 JSV0161         Joost Verhofstad        28-FEB-1983
;               Add VCB$L_JNLIOCNT
;
;       V03-030 RLRMXBCNT       Robert L. Rappaport     24-Feb-1982
;               Add UCB$L_MAXBCNT.
;
;       V03-029 MSH0002         Maryann Hinden          24-Feb-1983
;               Add $SPNBDEF.
;
;       V03-028 ROW0139         Ralph O. Weber          18-FEB-1983
;               Move JNL_CLS and JNL_SLV to UCB$W_DEVSTS.  Overlay UCB$L_PDT
;               with UCB$L_JNL_MCSID.  Remove UCB$W_JNL_CHAR, UCB$L_JNL_CDT,
;               UCB$L_SCSFL, and UCB$L_SCSBL journal UCB extension.
;
;       V03-027 RSH0005         R. Scott Hanna          10-Feb-1983
;               Added $RDIDEF which defines the rights database
;               identifier block offsets.
;
;       V03-026 TCM0003         Trudy C. Matthews       9-Feb-1983
;               Added new VMB input flags to $RPBDEF: AUTOTEST and CRDTEST.
;
;       V03-025 SRB0065         Steve Beckhardt         21-Jan-1983
;               Added new resource CLUSTRAN to $RSNDEF.
;
;       V03-024 DWT0067         David W. Thiel          20-Jan-1983
;               Add $SPPBDEF.  Add fields to $SBDEF.
```

```
{       V03-023 CDS0002         Christian D. Saether    27-Dec-1982
{               Move WCB$L_ACCLKID to avoid ambush by inexplicit assumptions.
{
{       V03-022 STJ3045         Steven T. Jeffreys      16-Dec-1982
{               Add $SLVDEF macro definition.
{
{       V03-021 SRB0057         Steve Beckhardt         16-Dec-1981
{               Added some new fields and reordered others in $RSBDEF
{               for distributed lock manager support.
{
{       V03-020 CDS0001         Christian D. Saether    9-Dec-1982
{               Add WCB$L_ACCLKID.
{
{       V03-019 ACG0303         Andrew C. Goldstein,    9-Dec-1982  15:13
{               Add FILL attribute to extraneous names
{
{       V03-018 NPK3010         N. Kronenberg   15-Nov-1982
{               Modify $SBDEF to add node name, hardware type and version,
{               and DDB pointer.  Add $SBODEF and $SYSAPDEF.
{
{       V03-017 MMD0001         Meg Dumont,     11-Nov-1982  14:42
{               Add bit inMODE field of VCBDEF to allow users to
{               enable EOT handling in the MTAACP.
{
{         V03-016 TCM0002               Trudy C. Matthews       31-Oct-1982
{                 Add new field to Restart Parameter Block: RPB$L_BADPGS.
{
{         V03-015 KTA0017               Kerbey T. Altmann       21-Oct-1982
{                 Add new fields to SCS system block.
{
{         V03-014 CWH0014               CW Hobbs                20-Oct-1982
{                 Add some warnings to $WSLDEF about strange constants
{
{         V03-013 JSV0081               Joost Verhofstad        8-Oct-1982
{                 Add WCB$L_JNL_PUIC
{
{         V03-012 SRB0054               Steve Beckhardt         6-Oct_1982
{                 Added new resource for SCS waits in RSNDEF.
{
{         V03-011 JSV0068               Joost Verhofstad        22-Sep-1982
{                 Add some journaling specific WCB fields and change the fixed
{                 constants produced by conversion routines from MDL, into
{                 computed constants for the WCB and VCB.
{
{         V03-010 ROW0122               Ralph O. Weber          12-SEP-1982
{                 Work over the UCB definition.  Change machined SDL into
{                 human senseable SDL.  Extend UCB$W_STS to a longword,
{                 defining UCB$L_STS.  Add a spare word for alignment after
{                 UCB$W_DEVSTS.  Move UCB$L_DEVDEPND2 next to UCB$L_DEVDEPEND
{                 and create UCB$Q_DEVDEPEND.
{
{       V03-009 JSV0063               Joost Verhofstad        09-Sep-1982
{               Change names of symbols that existed in V3.0 and
{                are used for journaling and have changed. This is
{                to allow builds of journaling facilities on both
```

```
{                the latest system and V3.0 (for early field test)
{
{        V03-008 MSH0001                 Maryann Hinden          09-Sep-1982
{                Add UAS definitions.
{
{        V03-006 JSV0031                 Joost Verhofstad        27-Jul-1982
{                Add some UCB and WCB fields for journaling
{
{        V03-005 JSV0022                 Joost Verhofstad        16-Jul-1982
{                Include errorlog UCB fields in journai UCB
{
{        V03-005 JSV0019                 Joost Verhofstad        12-Jul-1982
{                Add UCB$V_KNOWN_JNL
{
{        V03-004 LY0027                  Larry Yetto             29-Jun-1982
{                Define UCB$L_JNL_NDL to be the same as UCB$L_JNL_ASID
{
{        V03-003         JSV008          Joost Verhofstad        10-Jun-1982
{                Add UCB, VCB and WCB fields for journals
{
{        V03-002         KTA0100         Kerbey T. Altmann       07-Jun-1982
{                Add field MEDIA_ID to UCB.
{
{        V03-001         KDM0078         Kathleen D. Morse       15-Mar-1982
{                Add RPB$V_FINDMEM flag, for 11/782 installations.
{**
```

```
module $RBMDEF;
/*+
/* RBM        - realtime bit map of SPTs available for real time processes
/*-


aggregate RBMDEF structure prefix RBM$;
    STARTVPN longword unsigned;                 /* Starting VPN of bit map.
    FREECOUNT longword unsigned;                /* Number of free SPTs.
    SIZE word unsigned;                         /* Size of control block.
    TYPE byte unsigned;                         /* Type of control block.
    FILL_1 byte fill prefix RBMDEF tag $$;      /* Spare byte.
    constant "LENGTH" equals . prefix RBM$ tag K;   /* Length of block so far.
    constant "LENGTH" equals . prefix RBM$ tag C;   /* Length of block so far.
    BITMAP longword unsigned;                   /* Start of bit map.

end RBMDEF;

end_module $RBMDEF;
```

```
module $RDIDEF;
/*++
/* Rights Database Identifier Block definitions. This structure contains the
/* RMS Internal File Identifiers (IFI's) and Internal Stream Identifiers
/* (ISI's) for the rights database. This structure is allocated from the
/* process allocation region pool.
/*--

aggregate RDIDEF structure prefix RDI$;

    #ISI_MAX = 10;
    constant ISI_MAX equals #ISI_MAX;   /* Maximum number of concurrent record streams

    SIZE longword unsigned;             /* Size of allocated block
    IFI_READ longword unsigned;         /* Internal File Identifier for read operations
    IFI_WRITE longword unsigned;        /* Internal File Identifier for write operations
    ISI_VEC longword unsigned dimension 0:#ISI_MAX; /* Internal Stream Identifier vector

end RDIDEF;

end_module $RDIDEF;
```

```
module $RDPDEF;
/*
/* REMOTE DEVICE PROTOCOL DEFINITIONS
/*


aggregate RDPDEF structure prefix RDP$;
    OPCODE word unsigned;                                   /*OPERATION CODE
    MOD word unsigned;                                      /*OPERATION CODE MODIFIERS
    REFID longword unsigned;                                /*REFERENCE ID
    UNIT_OVERLAY union fill;
        UNIT word unsigned;                                 /*DEVICE UNIT NUMBER
        constant HEADERLEN equals . prefix RDP$ tag K;      /*HEADER LENGTH
        constant HEADERLEN equals . prefix RDP$ tag C;      /*HEADER LENGTH
        SIZE word unsigned;                                 /*SIZE OF MESSAGE (ACP/DRIVER USE ONLY)
    end UNIT_OVERLAY;
    PARAM1 longword unsigned;                               /*PARAMETER 1
    PARAM2 longword unsigned;                               /*PARAMETER 2
    PARAM3 longword unsigned;                               /*PARAMETER 3
    PARAM4 longword unsigned;                               /*PARAMETER 4
    PARAM5 longword unsigned;                               /*PARAMETER 5
    PARAM6 longword unsigned;                               /*PARAMETER 6
/*
/* RESPONSE FROM REMOTE PACKET DEFINITIONS
/*
                                                            /*RESPONSE PACKET OPCODES
    constant(
          ATTN                                              /* ATTENTION
        , "END"                                             /* I/O REQUEST COMPLETE
        , LOG                                               /* ERROR LOG
        ) equals -1 increment -1  prefix RDP tag $C;
end RDPDEF;

aggregate RDPDEF1 structure prefix RDP$;
    FILL_3 byte dimension 10 fill prefix RDPDEF tag $$;
    STATUS quadword unsigned;                               /*END PACKET I/O STATUS
/*
/* TERMINAL SPECIFIC PARAMETER DEFINITIONS
/*
/*  READ/WRITE REQUEST
end RDPDEF1;

aggregate RDPDEF2 structure prefix RDP$;
    FILL_4 byte dimension 10 fill prefix RDPDEF tag $$;
    TT_BCNT longword unsigned;                              /*BYTE COUNT
    TT_CARCON_OVERLAY union fill;
        TT_CARCON longword unsigned;                        /*WRITE CARRIAGE CONTROL
        TT_TIMOUT longword unsigned;                        /*READ TIMEOUT
    end TT_CARCON_OVERLAY;
    TT_WDATA_OVERLAY union fill;
        TT_WDATA character;                                 /*WRITE DATA
        TT_TERM character;                                  /*BYTE OF SIZE + TERMINATOR MASK
                                                            /*WORD OF SIZE + PROMPT STRING
/* SET MODE/CHARACTERISTICS REQUEST
    end TT_WDATA_OVERLAY;
end RDPDEF2;
```

```
aggregate RDPDEF3 structure prefix RDP$;
    FILL_5 byte dimension 10 fill prefix RDPDEF tag $$;
    TT_CHAR_OVERLAY union fill;
        TT_CHAR quadword unsigned;                          /*CHARACTERISTICS
        TT_ASTPRM longword unsigned;                        /*AST PARAMETER
    end TT_CHAR_OVERLAY;
    TT_SPEED longword unsigned;                             /*LINE SPEED
    TT_FILL longword unsigned;                              /*FILL SPECIFIER
    TT_PARITY longword unsigned;                            /*PARITY FLAGS
    TT_CHAR2 longword unsigned;                             /* Remaining longword of characters
/*  READ REQUEST END PACKET
end RDPDEF3;

aggregate RDPDEF4 structure prefix RDP$;
    FILL_6 byte dimension 10 fill prefix RDPDEF tag $$;
    FILL_1 quadword fill prefix RDPDEF tag $$;              /*I/O STATUS
    TT_RDATA character;                                     /*WORD OF SIZE + READ DATA
/*  SENSE MODE/CHARACTERISTICS END PACKET
end RDPDEF4;

aggregate RDPDEF5 structure prefix RDP$;
    FILL_7 byte dimension 10 fill prefix RDPDEF tag $$;
    FILL_2 quadword fill prefix RDPDEF tag $$;              /*I/O STATUS
    TT_SCHAR quadword unsigned;                             /*SENSED CHARACTERISTICS
    TT_SCHAR2 longword unsigned;                            /* Additional longword of characters

/* Broadcast message attention packet
end RDPDEF5;

aggregate RDPDEF6 structure prefix RDP$;
    FILL_8 byte dimension 10 fill prefix RDPDEF tag $$;
    TT_BRDTOTSIZE word unsigned;                            /* Total size of data
    TT_BRDMSG word unsigned;                                /* Message code
    TT_BRDUNIT word unsigned;                               /* Unit number
    TT_BRDNAME character length 16;                         /* Device name as counted string
    constant TT_BRDNAME equals 16  prefix RDP tag $C;       /* Size of name field
    TT_BRDTXTSIZE_OVERLAY union fill;
        TT_BRDTXTSIZE word unsigned;                        /* Count for message text
        TT_BRDTXTSIZE_FIELDS structure fill;
            FILL_9 byte dimension 2 fill prefix RDPDEF tag $$;
            TT_BRDTEXT character length 0 tag T;            /* Message text start

/* Out of band attention packet

        end TT_BRDTXTSIZE_FIELDS;
    end TT_BRDTXTSIZE_OVERLAY;
end RDPDEF6;

aggregate RDPDEF7 structure prefix RDP$;
    FILL_10 byte dimension 10 fill prefix RDPDEF tag $$;
    TT_OOTBAND byte unsigned;                               /* Out of band character

/* ATTENTION PACKET MODIFIERS
    constant(
          TT_UNSOL                                          /*UNSOLICITED DATA
```

```
        , TT_HANGUP                                     /*MODEM HANGUP
        , TT_CTRLC                                      /*CONTROL/C
        , TT_CTRLY                                      /*CONTROL/Y
        , TT_STARTRCV                                   /* Start a receive to the net
        , TT_BRDCST                                     /* Broadcast message for mailbox
        , TT_OUTBAND                                    /* Out of band AST
        ) equals 0 increment 1 prefix RDP tag $C;

end RDPDEF7;

end_module $RDPDEF;
```

```
module $RBFDEF;
/*
/*      Remote buffer as stored in dynamic memory
/*
/*      This structure must be identical to the abcve structure except
/*      for the header, which is the header for a buffered io buffer.
/*


/*
/*      Buffered io buffer header
/*


aggregate RBFDEF structure prefix RBF$;
    MSGDAT longword unsigned;                           /* Address of message data
    USRBFR longword unsigned;                           /* User buffer address
    SIZE word unsigned;                                 /* Size of structure
    TYPE byte unsigned;                                 /* Type of structure, DYN$C_BUFIO
    SPARE byte unsigned;                                /* Alignment
    DATSIZE word unsigned;                              /* Data size

/*
/*      End of header
/*

    OPCODE word unsigned;                               /*OPERATION CODE
    MOD word unsigned;                                  /*OPERATION CODE MODIFIERS
    REFID longword unsigned;                            /*REFERENCE ID
    UNIT word unsigned;                                 /*DEVICE UNIT NUMBER
/*      S       SIZE,0,W                    /*SIZE OF MESSAGE (ACP/DRIVER USE ONLY)
    constant HEADERLEN equals . prefix RBF$ tag K;      /*HEADER LENGTH
    constant HEADERLEN equals . prefix RBF$ tag C;      /*HEADER LENGTH
    PARAM1 longword unsigned;                           /*PARAMETER 1
    PARAM2 longword unsigned;                           /*PARAMETER 2
    PARAM3 longword unsigned;                           /*PARAMETER 3
    PARAM4 longword unsigned;                           /*PARAMETER 4
    PARAM5 longword unsigned;                           /*PARAMETER 5
    PARAM6 longword unsigned;                           /*PARAMETER 6
/*
/* RESPONSE FROM REMOTE PACKET DEFINITIONS
/*
                                                        /*RESPONSE PACKET OPCODES
    constant(
          ATTN                                          /* ATTENTION
        , "END"                                         /* I/O REQUEST COMPLETE
        , LOG                                           /* ERROR LOG
        ) equals -1 increment -1  prefix RBF tag $C;
end RBFDEF;

aggregate RBFDEF1 structure prefix RBF$;
    FILL 3 byte dimension 24 fill prefix RBFDEF tag $$;
    STATUS quadword unsigned;                           /*END PACKET I/O STATUS
/*
/* TERMINAL SPECIFIC PARAMETER DEFINITIONS
/*
```

```
/*  READ/WRITE REQUEST
end RBFDEF1;

aggregate RBFDEF2 structure prefix RBF$;
    FILL_4 byte dimension 24 fill prefix RBFDEF tag $$;
    TT_BCNT longword unsigned;                          /*BYTE COUNT
    TT_CARCON_OVERLAY union fill;
        TT_CARCON longword unsigned;                    /*WRITE CARRIAGE CONTROL
        TT_TIMOUT longword unsigned;                    /*READ TIMEOUT
    end TT_CARCON_OVERLAY;
    TT_WDATA_OVERLAY union fill;
        TT_WDATA character;                             /*WRITE DATA
        TT_TERM character;                              /*BYTE OF SIZE + TERMINATOR MASK
                                                        /*WORD OF SIZE + PROMPT STRING
/* SET MODE/CHARACTERISTICS REQUEST
    end TT_WDATA_OVERLAY;
end RBFDEF2;

aggregate RBFDEF3 structure prefix RBF$;
    FILL_5 byte dimension 24 fill prefix RBFDEF tag $$;
    TT_CHAR_OVERLAY union fill;
        TT_CHAR quadword unsigned;                      /*CHARACTERISTICS
        TT_ASTPRM longword unsigned;                    /*AST PARAMETER
    end TT_CHAR_OVERLAY;
    TT_SPEED longword unsigned;                         /*LINE SPEED
    TT_FILL longword unsigned;                          /*FILL SPECIFIER
    TT_PARITY longword unsigned;                        /*PARITY FLAGS
    TT_CHAR2 longword unsigned;                         /* Another longword of characters
/*  READ REQUEST END PACKET
end RBFDEF3;

aggregate RBFDEF4 structure prefix RBF$;
    FILL_6 byte dimension 24 fill prefix RBFDEF tag $$;
    FILL_1 quadword fill prefix RBFDEF tag $$;          /*I/O STATUS
    TT_RDATA character;                                 /*WORD OF SIZE + READ DATA
/*  SENSE MODE/CHARACTERISTICS END PACKET
end RBFDEF4;

aggregate RBFDEF5 structure prefix RBF$;
    FILL_7 byte dimension 24 fill prefix RBFDEF tag $$;
    FILL_2 quadword fill prefix RBFDEF tag $$;          /*I/O STATUS
    TT_SCHAR quadword unsigned;                         /*SENSED CHARACTERISTICS
    TT_SCHAR2 longword unsigned;                        /* Another longword of characters

/* Broadcast message attention packet
end RBFDEF5;

aggregate RBFDEF6 structure prefix RBF$;
    FILL_8 byte dimension 24 fill prefix RBFDEF tag $$;
    TT_BRDTOTSIZE word unsigned;                        /* Total size of data
    TT_BRDMSG word unsigned;                            /* Message code
    TT_BRDUNIT word unsigned;                           /* Unit number
    TT_BRDNAME character length 16;                     /* Device name as counted string
    constant TT_BRDNAME equals 16 prefix RBF tag $C;    /* Size of name field
    TT_BRDTXTSIZE_OVERLAY union fill;
        TT_BRDTXTSIZE word unsigned;                    /* Count for message text
```

```
        TT_BRDTXTSIZE_FIELDS structure fill;
            FILL_9 byte dimension 2 fill prefix RBFDEF tag $$;
            TT_BRDTEXT character length 0 tag T;        /* Message text start

/* Out of band attention packet

        end TT_BRDTXTSIZE_FIELDS;
    end TT_BRDTXTSIZE_OVERLAY;
end RBFDEF6;

aggregate RBFDEF7 structure prefix RBF$;
    FILL_10 byte dimension 24 fill prefix RBFDEF tag $$;
    TT_OUTBAND byte unsigned;                           /* Out of band character

/* ATTENTION PACKET MODIFIERS
    constant(
          TT_UNSOL                                      /*UNSOLICITED DATA
        , TT_HANGUP                                     /*MODEM HANGUP
        , TT_CTRLC                                      /*CONTROL/C
        , TT_CTRLY                                      /*CONTROL/Y
        , TT_STARTRCV                                   /* Start a receive to the net
        , TT_BRDCST                                     /* Broadcast message for mailbox
        , TT_OUTBAND                                    /* Out of band AST
        ) equals 0 increment 1 prefix RBF tag $C;

end RBFDEF7;

end_module $RBFDEF;
```

```
module $RCTDEF;
/*+
/* RCT - Replacement and Caching Table sector !0 layout.
/*       The RCT is a structure residing on disks controlled by MSCP
/*       speaking disk controllers.  The RCT is maintained by the intelligent
/*       controllers and the disk class driver.  The disk class driver mainly
/*       gets involved in RCT manipulations during host initiated bad
/*       block replacement.


aggregate RCTDEF structure prefix RCT$;
    VOLSER quadword unsigned;                                   /* Volume serial number
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                                    /* Flags word
        FLAGS_BITS structure fill;
            WB bitfield mask;                                   /* Write back caching in use
            FILL_1 bitfield length 6 fill prefix RCTDEF tag $$;
            FE bitfield mask;                                   /* Forced Error flag for block being replaced
            FILL_2 bitfield length 5 fill prefix RCTDEF tag $$;
            BR bitfield mask;                                   /* Replacement caused by Bad RBN
            RP2 bitfield mask;                                  /* Replacement in Progress phase 2
            RP1 bitfield mask;                                  /* Replacement in Progress phase 1
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    FILL_3 word fill prefix RCTDEF tag $$;                      /* Reserved word
    LBN longword unsigned;                                      /* LBN curently being replaced.
    RBN longword unsigned;                                      /* RBN allocated to replace LBN
    BAD_RBN longword unsigned;                                  /* If BR flag, RBN of bad replacement block
    WB_CTRL quadword unsigned;                                  /* Serial ! of last controller doing Write back
    WB_INCAR longword unsigned;                                 /* Write back incarnation !
    INCARTIME_OVERLAY union fill;
        INCARTIME quadword unsigned;                            /* Date-time of last update of incarnation no.
/*
/* Structure of a Replacement Block Descriptor
/*
        INCARTIME_BITS0 structure fill;
            LBN bitfield mask length 28;                        /* Space for LBN replaced by this RBN
            CODE bitfield mask length 4;                        /* Describes how this descriptor being used
        end INCARTIME_BITS0;

        INCARTIME_BITS1 structure fill;
            FILL_4 bitfield length 28 fill prefix RCTDEF tag $$;/* LBN
            NONPRIME bitfield mask;                             /* Set implies allocated, but not prime RBN
            ALLOCATED bitfield mask;                            /* This RBN allocated
            UNUSABLE bitfield mask;                             /* This RBN unusable
            NULL bitfield mask;                                 /* This marks a NULL entry
        end INCARTIME_BITS1;
                                                                /* Values of CODE
        constant EMPTY  equals 0  prefix RCT tag $K;            /* Unallocated (empty) replacement block
        constant ALOCPRIME      equals 2  prefix RCT tag $K;/* Allocated replace blk - primary RBN
        constant ALOCNONP       equals 3  prefix RCT tag $K;/* Allocated replace blk - non-primary RBN
        constant UNUSABLE       equals 4  prefix RCT tag $K;/* Unusable replacement block
        constant ALTUNUSE       equals 5  prefix RCT tag $K;/* Alternate unusable replacement block
        constant NULL   equals 8  prefix RCT tag $K;            /* Null entry - no corresponding RBN sector
    end INCARTIME_OVERLAY;
```

```
end RCTDEF;

end_module $RCTDEF;

module $RDTDEF;
/*+
/* RDT - SCS RESPONSE DESCRIPTOR TABLE
/*
/* ONE RESPONSE DESCRIPTOR (RD) IS ALLOCATED FOR EACH SCS MESSAGE
/* SENT FOR WHICH THE SENDER EXPECTS A MATCHING RESPONSE.
/*-


aggregate RDTDEF structure prefix RDT$ origin FILL_2;
    WAITFL longword unsigned;                          /*RD WAIT QUEUE FWD LINK
    WAITBL longword unsigned;                          /*RD WAIT QUEUE BACK LINK
    SIZE word unsigned;                                /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                /*SCS STURCTURE TYPE
    SUBTYP byte unsigned;                              /*SCS STRUCT SUBTYPE FOR RDT
    FREERD longword unsigned;                          /*ADDR OF 1ST FREE RD
    MAXRDIDX longword unsigned;                        /*MAXIMUM ! OF DESCRIPTORS
    FILL_1 longword fill prefix RDTDEF tag $$;         /*RESERVED FOR FUTURE USE
    constant "LENGTH"   equals 24  prefix RDT tag $C;  /*LENGTH OF NEG PORTION OF STRUCTURE
/*
    FILL_2 byte fill prefix RDTDEF tag $$;
end RDTDEF;

end_module $RDTDEF;
```

```
module $RDDEF;
/*+
/* RD - SCS RESPONSE DESCRIPTOR FORMAT
/*-


aggregate RDDEF structure prefix RD$;
    CDRP_OVERLAY union fill;
        CDRP longword unsigned;                 /*ADDR OF ASSOC CDRP OR
                                                /* OR OTHER CONTEXT BLOCK
        LINK longword unsigned;                 /* OR LINK TO NEXT FREE RD
    end CDRP_OVERLAY;
    STATE_OVERLAY union fill;
        STATE word unsigned;                    /*RD STATE FLAGS
        STATE_BITS structure fill;
            BUSY bitfield;                      /* ALLOCATED IF SET
            PERM bitfield;                      /* PERMANENTLY ALLOCATED RD IF SET
        end STATE_BITS;
    end STATE_OVERLAY;
    SEQNUM word unsigned;                       /*SEQUENCE NUMBER OF RD
    constant "LENGTH" equals . prefix RD$ tag K;    /*LENGTH OF RD
    constant "LENGTH" equals . prefix RD$ tag C;    /*LENGTH OF RD

end RDDEF;

end_module $RDDEF;
```

```
module $RPBDEF;
/*+
/* RESTART PARAMETER BLOCK DEFINITIONS
/*-

aggregate RPBDEF structure prefix RPB$;
    BASE longword unsigned;                          /*PHYSICAL BASE ADDRESS OF 64K BLOCK
    RESTART longword unsigned;                       /*POINTER TO RESTART ROUTINE (PHYSICAL)
    CHKSUM longword unsigned;                        /*CHECKSUM OF BYTES 0-7F OF RESTART ROUTINE
    RSTRTFLG longword unsigned;                      /*RESTART IN PROGRESS FLAG
    HALTPC longword unsigned;                        /*PC AT RESTART/HALT
    HALTPSL longword unsigned;                       /*PSL AT RESTART/HALT
    HALTCODE longword unsigned;                      /*CODE DESCRIBING RESTART REASON
    BOOTR0_OVERLAY union fill;
        BOOTR0 longword unsigned;                    /*SAVED BOOT PARAMETER R0
        BOOTR0_FIELDS structure fill;
            RODEVTYP byte unsigned;                  /* DEVICE TYPE SUBFIELD
            FILL_1 byte fill prefix RPBDEF tag $$;   /* RESERVED
            ROUBVEC word unsigned;                   /* UNIBUS INT VECTOR SUBFIELD
        end BOOTR0_FIELDS;
    end BOOTR0_OVERLAY;
    BOOTR1_OVERLAY union fill;
        BOOTR1 longword unsigned;                    /*SAVED BOOT PARAMETER R1
        BOOTR1_BITS structure fill;
            NEXUS bitfield length 4;                 /*NEXUS OF SYSTEM DEVICE ADAPTER
            ABUS bitfield length 2;                  /*ABUS ADAPTER NUMBER OF SBIA
        end BOOTR1_BITS;
    end BOOTR1_OVERLAY;
    BOOTR2 longword unsigned;                        /*SAVED BOTT PARAMETER R2
    BOOTR3 longword unsigned;                        /*SAVED BOOT PARAMETER R3
    BOOTR4 longword unsigned;                        /*SAVED BOOT PARAMETER R4
    BOOTR5_OVERLAY union fill;
        BOOTR5 longword unsigned;                    /*SAVED BOOT PARAMETER R5
        BOOTR5_BITS structure fill;
            CONV bitfield;                           /* CONVERSATIONAL BOOTSTRAP
            DEBUG bitfield;                          /* KEEP DEBUGGER CODE
            INIBPT bitfield;                         /* INITIAL BREAKPOINT
            BBLOCK bitfield;                         /* TRANSFER TO BOOTBLOCK
            DIAG bitfield;                           /* BOOT DIAGNOSTIC FILE
            BOOBPT bitfield;                         /* BOOTSTRAP BREAKPOINT
            HEADER bitfield;                         /* USE START ADDRESS FROM IMAGE HEADER
            NOTEST bitfield;                         /* FLAG TO INHIBIT MEMORY TESTING
            SOLICT bitfield;                         /* SOLICIT BOOT FILE NAME
            HALT bitfield;                           /* HALT BEFORE TRANSFER
            NOPFND bitfield;                         /* INHIBIT PFN DELETION
            MPM bitfield mask;                       /* MULTI-PROCESSOR BOOT, USE MA780 ONLY
            USEMPM bitfield mask;                    /* USE MA780 AS IF IT WERE LOCAL MEMORY
            MEMTEST bitfield mask;                   /* USE STRICTER TEST TO VALIDATE MEMORY
            FINDMEM bitfield mask;                   /* FIND SUFFICIENT MEMORY TO BOOT (>512K)
            AUTOTEST bitfield mask;                  /* USED BY DIAGNOSTIC SUPERVISOR
            CRDTEST bitfield mask;                   /* REMOVE PAGES WITH CRD ERRORS
            FILL_2 bitfield length 11 fill prefix RPBDEF tag $$;
            TOPSYS bitfield mask length 4;           /*SYSTEM DIRECTORY NUMBER
        end BOOTR5_BITS;
    end BOOTR5_OVERLAY;
```

```
    IOVEC longword unsigned;                    /*ADDRESS OF BOOTSTRAP QIO VECTOR
    IOVECSZ longword unsigned;                  /*SIZE OF BOOT QIO ROUTINE
    FILLBN longword unsigned;                   /*LOGICAL BLOCK NUMBER OF BOOT FILE
    FILSIZ longword unsigned;                   /*SIZE OF BOOT FILE
    PFNMAP quadword unsigned;                   /*DESCRIPTOR FOR PFN BITMAP
    PFNCNT longword unsigned;                   /*COUNT OF PHYSICAL PAGES
    SVASPT longword unsigned;                   /*SYSTEM VIRTUAL ADDRESS OF SPT
    CSRPHY longword unsigned;                   /*UBA DEVICE CSR ADDRESS (PHYSICAL)
    CSRVIR longword unsigned;                   /*UBA DEVICE CSR ADDRESS (VIRTUAL)
    ADPPHY longword unsigned;                   /*ADAPTER CONFIGURATION REGISTER (PHYSICAL)
    ADPVIR longword unsigned;                   /*ADAPTER CONFIGURATION REGISTER (VIRTUAL)
    UNIT word unsigned;                         /*UNIT NUMBER
    DEVTYP byte unsigned;                       /*DEVICE TYPE CODE
    SLAVE byte unsigned;                        /*SLAVE UNIT NUMBER
    FILE character length 40;                   /*BOOT FILE NAME (ASCIC)
    CONFREG byte unsigned dimension 16;         /*ARRAY OF ADAPTER TYPES
    HDRPGCNT byte unsigned;                     /*COUNT OF HEADER PAGES
    BOOTNDT_OVERLAY union fill;
        BOOTNDT word unsigned;                  /*16-BIT BOOT ADAPTER NEXUS DEVICE TYPE
        BOOTNDT byte unsigned;                  /* 8-BIT BOOT ADAPTER NEXUS DEVICE TYPE
    end BOOTNDT_OVERLAY;
    FLAGS structure byte unsigned;              /*MISCELLANEOUS FLAG BITS
        NOSYSDISK bitfield mask;                /* BOOT DISK IS NOT PRESENT
    end FLAGS;
    ISP longword unsigned;                      /*PWR FAIL INTERRUPT STACK POINTER
    PCBB longword unsigned;                     /*PROCESS CONTROL BLOCK BASE
    SBR longword unsigned;                      /*SYSTEM BASE REGISTER
    SCBB longword unsigned;                     /*SYSTEM CONTROL BLOCK BASE
    SISR longword unsigned;                     /*SOFTWARE INTERRUPT SUMMARY REGISTER
    SLR longword unsigned;                      /*SYSTEM LENGTH REGISTER
    MEMDSC_OVERLAY union fill;
        MEMDSC longword unsigned dimension 16;  /*MEMORY DESCRIPT. - PAGCNT, TR, BASE PFN
        MEMDSC_BITS structure fill;
            PAGCNT bitfield length 24;          /* COUNT OF PAGES FOR THIS MEMORY
            TR bitfield length 8;               /* TR NUMBER FOR THIS MEMORY
            BASEPFN bitfield length 32;         /* BASE PFN FOR THIS MEMORY
        end MEMDSC_BITS;
        constant MEMDSCSIZ      equals 8  prefix RPB tag $C;/*NUMBER OF BYTES IN ONE MEM DESCRIPTOR
        constant NMEMDSC        equals 8  prefix RPB tag $C;/*NUMBER OF MEMORY DESCRIPTORS IN RPB
    end MEMDSC_OVERLAY;
    BUGCHK longword unsigned;                   /*BUGCHECK LOOP ADDRESSS FOR MP SECONDARY
    WAIT byte unsigned dimension 4;             /*BUGCHECK LOOP CODE FOR MP SECONDARY
    BADPGS longword unsigned;                   /*NUMBER OF BAD PAGES FOUND IN MEM SCAN
    CTRLLTR byte unsigned;                      /*CONTROLLER LETTER DESIGNATOR
    constant "LENGTH" equals . prefix RPB$ tag K;   /*LENGTH OF RPB
    constant "LENGTH" equals . prefix RPB$ tag C;   /*LENGTH OF RPB
end RPBDEF;

end_module $RPBDEF;
```

```
module $RSBDEF;
/++
/* RSB - RESOURCE BLOCK
/*
/* RESOURCE BLOCKS REPRESENT RESOURCES FOR WHICH THERE ARE LOCKS OUTSTANDING.
/* EACH RESOURCE BLOCK MAY HAVE ONE OR MORE LOCK BLOCKS (LKB) QUEUED TO IT.
/*-


aggregate RSBDEF structure prefix RSB$;
    HSHCHN longword unsigned;                   /*HASH CHAIN
    HSHCHNBK longword unsigned;                 /*HASH CHAIN BACK POINTER
    SIZE word unsigned;                         /*SIZE OF RSB
    TYPE byte unsigned;                         /*STRUCTURE TYPE
    DEPTH byte unsigned;                        /*DEPTH IN TREE
    GGMODE byte unsigned;                       /*GROUP GRANT MODE
    CGMODE byte unsigned;                       /*CONVERSION GRANT MODE
    STATUS_OVERLAY union fill;
        STATUS word unsigned;                   /*STATUS
        STATUS_BITS structure fill;
            DIRENTRY bitfield mask;             /* ENTERED IN DIR. DURING FAILOVER
            VALINVLD bitfield mask;             /* VALUE BLOCK INVALID
        end STATUS_BITS;
    end STATUS_OVERLAY;
    GRQFL longword unsigned;                    /*GRANTED QUEUE FORWARD LINK
    GRQBL longword unsigned;                    /*GRANTED QUEUE BACKWARD LINK
    CVTQFL longword unsigned;                   /*CONVERSION QUEUE FORWARD LINK
    CVTQBL longword unsigned;                   /*CONVERSION QUEUE BACKWARD LINK
    WTQFL longword unsigned;                    /*WAIT QUEUE FORWARD LINK
    WTQBL longword unsigned;                    /*WAIT QUEUE BACKWARD LINK
    VALBLK quadword unsigned;                   /*VALUE BLOCK
    FILL_1 quadword fill prefix RSBDEF tag $$;  /*MORE VALUE BLOCK
    CSID longword unsigned;                     /*SYSTEM ID OF MASTER SYS.
    VALSEQNUM longword unsigned;                /*VALUE BLOCK SEQ. NUMBER
    REFCNT word unsigned;                       /*SUB RSB REFERENCE COUNT
    BLKASTCNT word unsigned;                    /*BLOCKING AST COUNT
    HASHVAL word unsigned;                      /*HASH VALUE
    RQSEQNM word unsigned;                      /*REQUEST SEQUENCE NUMBER
    PARENT longword unsigned;                   /*ADDRESS OF PARENT RSB
    GROUP word unsigned;                        /*GROUP NUMBER
    RMOD byte unsigned;                         /*ACCESS MODE OF RESOURCE
    RSNLEN byte unsigned;                       /*RESOURCE NAME LENGTH
    constant "LENGTH" equals . prefix RSB$ tag K;       /*LENGTH OF FIXED PART OF RSB
    constant "LENGTH" equals . prefix RSB$ tag C;       /*LENGTH OF FIXED PART OF RSB
    RESNAM character length 0 tag T;                    /*START OF RESOURCE NAME
    constant MAXLEN equals 31  prefix RSB tag $K;       /*MAXIMUM LENGTH OF RESOURCE NAME
end RSBDEF;

end_module $RSBDEF;
```

```
module $RSNDEF;
/**
/* RESOURCE NAME DEFINITIONS
/*-

                                                        /*0 ORIGIN IN INCREMENTS OF 1

constant(
        ASTWAIT                                         /*WAIT FOR AST EVENT, CHANNEL INTERLOCK
      , MAILBOX                                         /*MAILBOX SPACE
      , NPDYNMEM                                        /*NON-PAGED DYNAMIC MEMORY
      , PGFILE                                          /*PAGING FILE SPACE
      , PGDYNMEM                                        /*PAGED DYNAMIC MEMORY
      , BRKTHRU                                         /*TERMINAL BROADCAST
      , IACLOCK                                         /*IMAGE ACTIVATION INTERLOCK
      , JQUOTA                                          /*JOB POOLED QUOTA
      , LOCKID                                          /*LOCKIDS
      , SWPFILE                                         /*SWAPPING FILE SPACE
      , MPLEMPTY                                        /*MODIFIED PAGE LIST EMPTY
      , MPWBUSY                                         /*MODIFIED PAGE WRITER BUSY
      , SCS                                             /*SYSTEM COMMUNICATION
      , CLUSTRAN                                        /*CLUSTER STATE TRANSITION
      , MAX                                             /*MAXIMUM RESOURCE NUMBER
      ) equals 1 increment 1  prefix RSN tag $;

end_module $RSNDEF;
```

```
module $RUCBDEF;
/**
/* Internal control block for use by Recovery Unit services.
/*
/* This structure is created when a process first issues a
/* RUF service and is pointed to by CTL$L_RUF.
/*-

aggregate RUCBDEF structure fill prefix RUCB$;
    RUID_OVERLAY union fill;
        RUID character length 16;                       /* 128-bit id
        constant ID_LEN equals . prefix RUCB$ tag K;    /* length of the ID
        constant ID_LEN equals . prefix RUCB$ tag C;    /* length of the ID
        RUID_FIELDS structure fill;
            ID_TIME quadword unsigned;                  /* system time in 100ns. units
            ID_CSID longword unsigned;                  /* cluster ID
            ID_EPID longword unsigned;                  /* cluster PID of initiating process
        end RUID_FIELDS;
    end RUID_OVERLAY;
    RU_CTRL_OVERLAY union fill;
        RU_CTRL longword unsigned;                      /* control longword
        RU_CTRL_FIELDS structure fill;
            STATE byte unsigned;                        /* RU current state
            constant ACTIVE     equals 1  prefix RUCB tag $C;/* active
            constant PH1_INIT   equals 2  prefix RUCB tag $C;/* phase 1 commit started
            constant PH1_FIN    equals 3  prefix RUCB tag $C;/* phase 1 commit completed
            constant PH2_INIT   equals 4  prefix RUCB tag $C;/* phase 2 commit started
            constant PH2_FIN    equals 5  prefix RUCB tag $C;/* phase 2 commit completed
            constant CANCEL     equals 6  prefix RUCB tag $C;/* cancel in progress
            constant RESET      equals 7  prefix RUCB tag $C;/* reset to markpoint in progress
            CTRL_OVERLAY union fill;
                CTRL byte unsigned;                     /* control flags
                CTRL_BITS structure fill;
                    INIT bitfield mask;                 /* RUCB has been initialized
                    ACTIVE bitfield mask;               /* Recovery Unit is currently active
                    INHANDLER bitfield mask;            /* processing handlers
                    RUSYNC bitfield mask;               /* handler has specified synchronous RUs
                end CTRL_BITS;
            end CTRL_OVERLAY;
            SRVCODE word unsigned;                      /* current operation change mode code
        end RU_CTRL_FIELDS;
    end RU_CTRL_OVERLAY;
    HACTION longword unsigned;                          /* handler action code
    HREASON longword unsigned;                          /* handler invocation reason code (see $RUHRSNDEF)
    MARKPT longword unsigned;                           /* most recent markpoint value
    RUH_EXEC longword unsigned;                         /* exec mode handler listhead
    RUH_SUPV longword unsigned;                         /* supervisor mode handler listhead
    RUH_USER longword unsigned;                         /* user mode handler listhead
    SV_PCPSL quadword unsigned;                         /* saved user's PC and PSL from original CHMK
    SV_RSB longword unsigned;                           /* saved return address from RUF$CALL_HANDLERS
    FREESP longword unsigned;                           /* free handler cell listhead
    AILIST longword unsigned;                           /* ptr to list of AI jnls touched in RU
    BILIST longword unsigned;                           /* ptr to list of BI jnls touched in RU
    IMPURE longword unsigned;                           /* ptr to impure storage area for ENDRU
    AMODE byte unsigned;                                /* mode of caller of RUF$START
    SV_AMODE byte unsigned;                             /* saved access mode (R4) from RUF$CALL_HANDLERS
```

```
    FILL_1 byte dimension 2 fill prefix RUCBDEF tag $$; /* spare
    constant "LENGTH" equals . prefix RUCB$ tag K;      /* length of RUCB
    constant "LENGTH" equals . prefix RUCB$ tag C;      /* length of RUCB
end RUCBDEF;

end_module $RUCBDEF;
```

```
module $RUHDEF;
/**
/*       Recovery Unit Handler storage cell definitions
/*-

aggregate RUHDEF structure fill prefix RUH$;
    LINK longword unsigned;                         /* Next cell address ptr
    ADDR longword unsigned;                         /* Routine addr
    PARAM longword unsigned;                        /* Routine parameter
    constant SIZE equals . prefix RUH$ tag K;       /* Size of cell
    constant SIZE equals . prefix RUH$ tag C;       /* Size of cell
end RUHDEF;

end_module $RUHDEF;
```

```
module $RVTDEF;
/**
/* RVT - RELATIVE VOLUME TABLE
/*
/* A RELATIVE VOLUME MAPPING TABLE IS REQUIRED FOR EVERY MULTIVOLUME
/* STRUCTURE THAT IS MOUNTED IN A SYSTEM.
/*-



aggregate RVTDEF structure prefix RVT$;
    STRUCLKID longword unsigned;                        /* LOCK ID OF VOLUME SET LOCK.
    REFC word unsigned;                                 /* REFERENCE COUNT
    ACTIVITY word unsigned;                             /* ACTIVITY COUNT/FLAG
    SIZE word unsigned;                                 /* SIZE OF RVT IN BYTES
    TYPE byte unsigned;                                 /* STRUCTURE TYPE OF RVT
    NVOLS byte unsigned;                                /* NUMBER OF VOLUMES IN SET
    STRUCNAME character length 12;                      /* STRUCTURE (VOLUME SET) NAME
    VLSLCKNAM character length 12;                      /* Volume set lock name.
    BLOCKID longword unsigned;                          /* Blocking lock id.
    ACB byte unsigned dimension 28;                     /* ACB for blocking ast.
    constant "LENGTH" equals . prefix RVT$ tag K;       /* LENGTH OF STANDARD RVT
    constant "LENGTH" equals . prefix RVT$ tag C;       /* LENGTH OF STANDARD RVT
    UCBLST longword unsigned;                           /* ADDRESSES OF THE RESPECTIVE UCB'S
    constant MINSIZE    equals 18  prefix RVT tag $C;   /* MINIMUM NUMBER OF ENTRIES TO ALLOCATE

end RVTDEF;

end_module $RVTDEF;
```

```
module $SBDEF;
/*+
/* SB - SCS SYSTEM BLOCK
/*
/* THE SB HAS INFORMATION ABOUT KNOWN SYSTEMS IN A CPU CLUSTER.
/*-


aggregate SBDEF structure prefix SB$;
    FLINK longword unsigned;                    /*FWD LINK TO NEXT SB
    BLINK longword unsigned;                    /*BACK LINK TO PREVIOUS SB
    SIZE word unsigned;                         /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                         /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                       /*SCS STRUCT SUBTYPE FOR SB
    PBFL longword unsigned;                     /*LINK TO NEXT PATH BLOCK
    PBBL longword unsigned;                     /*LINK TO PREVIOUS PATH BLOCK
    PBCONNX longword unsigned;                  /*ADDR OF NEXT PB TO USE FOR
                                                /* A CONNECTION
    SYSTEMID byte unsigned dimension 6;         /*SYSTEM ID
    FILL_1 word fill prefix SBDEF tag $$;       /*RESERVED WORD
    MAXDG word unsigned;                        /*MAXIMUM DATAGRAM SIZE
    MAXMSG word unsigned;                       /*MAXIMUM MESSAGE SIZE
    SWTYPE character length 4;                  /*SOFTWARE TYPE, 1-4 CHAR
    SWVERS character length 4;                  /*SOFTWARE VERSION, 1-4 CHAR
    SWINCARN quadword unsigned;                 /*SOFTWARE INCARNATION #
    HWTYPE character length 4;                  /*HW TYPE; 1-4 CHAR, BLANK FILL
    HWVERS byte unsigned dimension 12;          /*HW VERSION #
    NODENAME character length 16;               /*SCS NODENAME, COUNTED ASCII STRING
    DDB longword unsigned;                      /*DDB LIST HEAD
    TIMEOUT word;                               /*SCA PROCESS POLLER, WAITING TIME REMAINING
    ENBMSK byte unsigned dimension 2;           /*SCA PROCESS POLLER, PROCESS ENABLE MASK
    CSB longword unsigned;                      /*LINK TO NEWEST CLUSTER SYSTEM BLOCK
    constant ''LENGTH'' equals . prefix SB$ tag K;  /*LENGTH OF SB
    constant ''LENGTH'' equals . prefix SB$ tag C;  /*LENGTH OF SB

end SBDEF;

end_module $SBDEF;
```

```
module $SBODEF;
/++
/* SBO - SCS CONFIG_SYS OUTPUT ARRAY FORMAT
/*
/* THE OUTPUT ARRAY RETURNED FROM CALL TO SCS$CONFIG_SYS.  DATA IS MOSTLY COPIED FROM
/* THE SYSTEM BLOCK (SB) BEING LOOKED UP.
/*-



aggregate SBODEF structure prefix SBO$;
    SYSTEMID byte unsigned dimension 6;                 /*SYSTEM ID
    FILL_1 word fill prefix SBODEF tag $$;              /*RESERVED WORD
    MAXDG word unsigned;                                /*MAXIMUM DG SIZE
    MAXMSG word unsigned;                               /*MAXIMUM MSG SIZE
    SWTYPE character length 4;                          /*SW TYPE, 1-4 CHAR, BLNK FILL
    SWVERS character length 4;                          /*SW VERSION, 1-4 CHAR, BLNK FILL
    SWINCARN quadword unsigned;                         /*SW INCARNATION #
    HWTYPE character length 4;                          /*HW TYPE, 1-4 CHAR BLNK FILL
    HWVERS byte unsigned dimension 12;                  /*HW VERSION, 1-4 CHAR BLNK FILL
    NODENAME character length 16;                       /*NODE NAME, COUNTED ASCII STRING
    constant VC1 equals . prefix SBO$ tag C;            /*START OF 12 BYTE SPECIFIER OF
    constant VC1 equals . prefix SBO$ tag K;            /* 1ST VC (PATH BLK) TO SYSTEM
    RSTATION1 byte unsigned dimension 6;                /*REMOTE STATION OF 1ST VC
    FILL_1 word fill prefix SBODEF tag $$;              /*RESERVED WORD
    LPORT1 character length 4;                          /*LOCAL PORT NAME OF 1ST VC
    NXT_SYSID byte unsigned dimension 6;                /*ID OF NEXT SYSTEM IN CONFIGURATION
    FILL_1 word fill prefix SBODEF tag $$;              /*RESERVED WORD
    constant "LENGTH" equals . prefix SBO$ tag K;       /*LENGTH OF SBO ARRAY
    constant "LENGTH" equals . prefix SBO$ tag C;       /*LENGTH OF SBO ARRAY

end SBODEF;
end_module $SBODEF;
```

```
module $SCSDEF;
/*+
/* SCS MESSAGE DEFINITIONS
/*
/* THIS STRUCTURE DEFINES OFFSETS AND FIELDS WITHIN THE SCS PORTION OF
/* A CLUSTER MESSAGE.  OFFSETS ARE DEFINED RELATIVE TO THE START OF THE
/* APPLICATION DATA OR SCS CONTROL MESSAGE DATA.  THE FULL MESSAGE FORMAT
/* CONSISTS OF A PORT DRIVER LAYER HEADER (SEE STRUCTURE PPD) FOLLOWED
/* BY THE SCS HEADER LAYER FOLLOWED BY THE APPLICATION DATA OR SCS CONTROL
/* MESSAGE DATA.
/*-


aggregate SCSDEF structure prefix SCS$ origin MIN_CR;
    PPD byte unsigned dimension 16;                 /*16 BYTES OF PPD HEADER
    "LENGTH" word unsigned;                         /*MESSAGE LENGTH (INCLUDES ALL
                                                    /* BYTES FROM SCS$W_LENGTH ON,
                                                    /* NOT INCLUDING SCS$W_LENGTH)
                                                    /* (FIELD SHARED BY PPD)
                                                    /*DEFINE LENGTHS OF SCS CONTROL MSGS:
    constant OVHD       equals 14  prefix SCS tag $C;   /* SCS LAYER OVERHEAD
    constant CON_REQL   equals 66  prefix SCS tag $C;   /* CONNECT_REQ LENGTH
    constant CON_RSPL   equals 18  prefix SCS tag $C;   /* CONNECT_RSP LENGTH
    constant ACCP_REQL  equals 66  prefix SCS tag $C;   /* ACCEPT_REQ LENGTH
    constant ACCP_RSPL  equals 18  prefix SCS tag $C;   /* ACCEPT_RSP LENGTH
    constant REJ_REQL   equals 18  prefix SCS tag $C;   /* REJECT_REQ LENGTH
    constant REJ_RSPL   equals 14  prefix SCS tag $C;   /* REJECT_RSP LENGTH
    constant DISC_REQL  equals 18  prefix SCS tag $C;   /* DISCONNECT_REQ LENGTH
    constant DISC_RSPL  equals 14  prefix SCS tag $C;   /* DISCONNECT_RSP LENGTH
    constant CR_REQL    equals 18  prefix SCS tag $C;   /* CREDIT_REQ LENGTH
    constant CR_RSPL    equals 14  prefix SCS tag $C;   /* CREDIT_RSP LENGHT
    FILL_1 word fill prefix SCSDEF tag $$;          /*WORD RESERVED FOR PPD LAYER
    MTYPE word unsigned;                            /*SCS MESSAGE TYPE
                                                    /*SCS MESSAGE TYPE CODES:
                                                    /* 0 ORIGIN, INCREMENTS OF 1
    constant(
          CON_REQ                                   /* CONNECT_REQ
        , CON_RSP                                   /* CONNECT_RSP
        , ACCP_REQ                                  /* ACCEPT_REQ
        , ACCP_RSP                                  /* ACCEPT_RSP
        , REJ_REQ                                   /* REJECT_REQ
        , REJ_RSP                                   /* REJECT_RSP
        , DISC_REQ                                  /* DISCONNECT_REQ
        , DISC_RSP                                  /* DISCONNECT_RSP
        , CR_REQ                                    /* CREDIT_REQ
        , CR_RSP                                    /* CREDIT_RSP
        , APPL_MSG                                  /* APPLICATION MESSAGE
        , APPL_DG                                   /* APPLICATION DATAGRAM
        ) equals 0 increment 1  prefix SCS tag $C;
    CREDIT word unsigned;                           /*CREDIT BEING EXTENDED
    DST_CONID longword unsigned;                    /*DESTINATION (RECVING) CONNX ID
    SRC_CONID longword unsigned;                    /*SOURCE (SENDING) CONNX ID
    constant APPL_BASE equals . prefix SCS$ tag K;  /*BASE OF APPLICTION MESSAGE DATA
    constant APPL_BASE equals . prefix SCS$ tag C;  /*BASE OF APPLICTION MESSAGE DATA
    MIN_CR word unsigned;                           /*MINIMUM SEND CREDIT
```



```
/*
/*
/*

aggr

end;

/*
/*
/*

aggr

end;

/*
/*
/*

aggr

end;
```

```
    STATUS word unsigned;                                   /*STATUS/REASON
                                                            /*DEFINE STATUS/REASON CODES:
    constant STNORMAL equals 1 prefix SCS$ tag K;           /* NORMAL, SUCCESS
    constant STNORMAL equals 1 prefix SCS$ tag C;           /*
    constant STNOMAT  equals 10 prefix SCS$ tag K;          /* NO MATCHING LISTENER
    constant STNOMAT  equals 10 prefix SCS$ tag C;          /*
    constant STNORS   equals 18 prefix SCS$ tag K;          /* NO RESOURCES
    constant STNORS   equals 18 prefix SCS$ tag C;          /*
    constant STDISC   equals 25 prefix SCS$ tag K;          /* DISCONNECTED
    constant STDISC   equals 25 prefix SCS$ tag C;          /*
    constant STINSFCR equals 33 prefix SCS$ tag K;          /* INSUFF CREDIT
    constant STINSFCR equals 33 prefix SCS$ tag C;          /*

    constant CON_BASE equals . prefix SCS$ tag K;           /*BASE OF CONNECT/ACCEPT INFO TO
    constant CON_BASE equals . prefix SCS$ tag C;           /*BASE OF CONNECT/ACCEPT INFO TO
                                                            /* GIVE TO SYSAP'S
                                                            /*FORMAT OF CONNECT/ACCEPT REQ MSGS:
    DST_PROC character length 16;                           /* DESTINATION PROCESS NAME
    SRC_PROC character length 16;                           /* SOURCE PROCESS NAME
    CON_DAT byte unsigned dimension 16;                     /* CONNECT DATA

end SCSDEF;

/*
/* DEFINITION OF THE REQUEST/SEND DATA OFFSETS
/*

aggregate SCSDEF1 structure prefix SCS$ origin SND_BOFF;
    LCONID longword unsigned;                               /* LOCAL CONNECTION ID
    RSPID longword unsigned;                                /* LOCAL RESPONSE ID
    XCT_LEN longword unsigned;                              /* TRANSACTION LENGTH
    SND_NAME longword unsigned;                             /* SEND BUFFER NAME
    SND_BOFF longword unsigned;                             /*  AND OFFSET
    REC_NAME longword unsigned;                             /* RECEIVE BUFFER NAME
    REC_BOFF longword unsigned;                             /*  AND OFFSET
end SCSDEF1;

end_module $SCSDEF;
```

```
module $SCSCMGDEF;
/*+
/* SCSCMG - SCS CONNECTION MANAGEMENT MESSAGE FORMAT
/*
/* THIS PORTION OF A CONNECT/ACCEPT MESSAGE IS SEEN BY A
/* SYSTEM APPLICATION.
/*-



aggregate SCSCMGDEF structure prefix SCSCMG$;
    RECNAM character length 16;                 /*RECEIVE PROCESS NAME
    SNDNAM character length 16;                 /*SENDER PROCESS NAME
    SNDDAT byte unsigned dimension 16;          /*SENDER CONNECT DATA

end SCSCMGDEF;

end_module $SCSCMGDEF;
```

```
module $SDIRDEF;
/++
/* SDIR - SCS DIRECTORY ENTRY
/*
/* THIS DATA STRUCTURE IS ALLOCATED FOR EACH LOCAL PROCESS THAT WANTS
/* TO BE KNOWN TO SCS.
/*-



aggregate SDIRDEF structure prefix SDIR$;
    FLINK longword unsigned;                    /*FWD LINK
    BLINK longword unsigned;                    /*BCK LINK
    SIZE word unsigned;                         /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                         /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                       /*SCS STRUCTURE SUBTYPE FOR SDIR
    PROCNAM byte unsigned dimension 16;         /*ASCII STRING FOR PROCESS NAME
    PROCINF byte unsigned dimension 16;         /*ASCII STRING FOR PROCESS INFO
    CONID longword unsigned;                    /*CONNECTION ID
    constant "LENGTH" equals . prefix SDIR$ tag K;
    constant "LENGTH" equals . prefix SDIR$ tag C;

end SDIRDEF;

end_module $SDIRDEF;
```

```
module $SGNDEF;

/*+
/* SYSGEN PARAMETER DEFINITIONS
/*-


                                                        /*
constant BALSETCNT      equals 24  prefix SGN tag $C;   /* NUMBER OF PROCESSES IN BALANCE SET
constant DFWSCNT        equals 100  prefix SGN tag $C;  /* DEFAULT WORKING SET COUNT
constant DFWSQUOTA      equals 120  prefix SGN tag $C;  /* DEFAULT WORKING SET QUOTA
constant GBLSECCNT      equals 40  prefix SGN tag $C;   /* GLOBAL SECTION COUNT
constant MAXGPGCNT      equals 2*1024  prefix SGN tag $C;/* GLOBAL PAGE COUNT (GPT SIZE)
constant MAXPAGCNT      equals 128*32*4  prefix SGN tag $C;/* PHYSICAL MEMORY SIZE IN PAGES
constant MAXPGFL        equals 4096  prefix SGN tag $C; /* DEFAULT MAXIMUM PAGING FILE
constant MAXPSTCNT      equals 5  prefix SGN tag $C;    /* MAX NUMBER OF PST ENTRIES
constant MAXVPGCNT      equals 8*8*128  prefix SGN tag $C;/* MAX PROCESS VIRTUAL SIZE (PAGES)
constant MAXWSCNT       equals 1024  prefix SGN tag $C; /* MAX WORKING SET SIZE (PAGES)
constant MINWSCNT       equals 10  prefix SGN tag $C;   /* MIN WORKING SET SIZE (PAGES)
constant NPAGEDYN       equals 52*512  prefix SGN tag $C;/* NON-PAGED DYNAMIC POOL SIZE
constant NPROCS equals 64  prefix SGN tag $C;           /* MAX NUMBER OF PROCESSES
constant PAGEDYN        equals 2*16*512  prefix SGN tag $C;/* PAGED DYNAMIC POOL SIZE IN BYTES
constant PHYPAGCNT      equals 32*128  prefix SGN tag $C;/* ACTUAL PHYSICAL PAGE COUNT
constant SYSDWSCNT      equals 40  prefix SGN tag $C;   /* DEFAULT SYSTEM WORKING SET COUNT
constant SYSVECPGS      equals 5  prefix SGN tag $C;    /* NO. OF PAGES OF SYSTEM SERVICE VECTORS
constant SYSWSCNT       equals 96  prefix SGN tag $C;   /* SYSTEM WORKING SET COUNT

end_module $SGNDEF;
```

```
module $SHBDEF;
/++
/* SHARED MEMORY CONTROL BLOCK DEFINITIONS
/*-
/*
/* The UETP for the MA780 depends on some of the following definitions.  Please
/* let someone in that group know if the definitions change substantially.
/*



aggregate SHBDEF structure prefix SHB$;
    LINK longword unsigned;                         /*LINK TO NEXT SHB
    DATAPAGE longword unsigned;                     /*VIRTUAL ADDRESS OF DATAPAGE
    SIZE word unsigned;                             /*SIZE OF SHB IN BYTES
    TYPE byte unsigned;                             /*STRUCTURE TYPE FOR SHB
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                        /*FLAGS
        FLAGS_BITS structure fill;
            CONNECT bitfield mask;                  /* MEMORY IS CONNECTED, USEABLE
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    REFCNT longword unsigned;                       /*COUNT OF REFERENCES TO MEMORY
    BASGSPFN longword unsigned;                     /*BASE PFN FOR GLOBAL SECTION PAGES
    NEXUS byte unsigned;                            /*NEXUS OF PORT
    PORT byte unsigned;                             /*PORT NUMBER
    FILL_1 word fill prefix SHBDEF tag $$;          /* UNUSED
    POOLEND longword unsigned;                      /*ADDRESS PAST LAST BYTE OF POOL
    ADP longword unsigned;                          /*ADAPTER CONTROL BLOCK ADDRESS
    constant "LENGTH" equals . prefix SHB$ tag K;   /*LENGTH OF CONTROL BLOCK
    constant "LENGTH" equals . prefix SHB$ tag C;   /*LENGTH OF CONTROL BLOCK
end SHBDEF;
end_module $SHBDEF;
```

```
module $SHDDEF;
/*+
/* SHARED MEMORY DATAPAGE DEFINITIONS
/*-
/*
/* The UETP for the MA780 depends on some of the following definitions.  Please
/* let someone in that group know if the definitions change substantially.
/*

constant MAXPORTS       equals 16  prefix SHD tag $C;   /*MAXIMUM NUMBER PORTS HANDLED BY THIS
                                                        /*DATA STRUCTURE
/**** START OF CONSTANT FIELDS:

aggregate SHDDEF structure prefix SHD$;
    MBXPTR longword unsigned;                           /*RELATIVE POINTER TO MAILBOX TABLE
    GSDPTR longword unsigned;                           /*RELATIVE POINTER TO GSD TABLE
    CEFPTR longword unsigned;                           /*RELATIVE POINTER TO CEF TABLE
    GSBITMAP longword unsigned;                         /*RELATIVE POINTER TO BITMAP
    GSPAGCNT longword unsigned;                         /*CNT OF PAGES ALLOTTED FOR GBL SECTIONS
    GSPFN longword unsigned;                            /*RELATIVE PFN OF 1ST GBL SECTION PAGE
    GSDMAX word unsigned;                               /*MAX GSD'S (SIZE OF TABLE)
    MBXMAX word unsigned;                               /*MAX MAILBOXES (SIZE OF TABLE)
    CEFMAX word unsigned;                               /*MAX CEF CLUSTERS (SIZE OF TABLE)
    FILL_1 word fill prefix SHDDEF tag $$;              /*UNUSED
    NAME character length 16;                           /*NAME OF MEMORY (COUNTED STRING)
    constant NAMLENGTH  equals 16  prefix SHD tag $C;   /*MAXIMUM LENGTH OF NAME OF MEMORY
    INITTIME quadword unsigned;                         /*INITIALIZATION TIME
/**** END OF CONSTANT FIELDS.
    CRC longword unsigned;                              /*CRC OF CONSTANT FIELDS
    GSDQUOTA word unsigned dimension 16;                /*COUNT OF GSD'S CREATED (ONE/PORT)
    MBXQUOTA word unsigned dimension 16;                /*COUNT OF MAILBOXES CREATED (ONE/PORT)
    CEFQUOTA word unsigned dimension 16;                /*COUNT OF CLUSTERS CREATED (ONE/PORT)
    PORTS byte unsigned;                                /*NUMBER OF PORTS
    INITLCK byte unsigned;                              /*OWNER OF INIT LOCK
    BITMAPLCK byte unsigned;                            /*OWNER OF GS BITMAP LOCK
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                            /*FLAGS FOR LOCKING DATA STRUCTURES
        FLAGS_BITS structure fill;
            INITLCK bitfield mask;                      /*COMMON DATA PAGE BEING INITIALIZED
            BITMAPLCK bitfield mask;                    /*BITMAP BEING MODIFIED
            GSDLCK bitfield mask;                       /*GLOBAL SECTION DSC TABLE BEING SEARCHED
            MBXLCK bitfield mask;                       /*MAILBOX TABLE BEING SEARCHED
            CEFLCK bitfield mask;                       /*COMMON EVENT FLAG TABLE BEING SEARCHED
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    GSDLOCK byte unsigned;                              /*OWNER OF GSD TABLE LOCK
    MBXLOCK byte unsigned;                              /*OWNER OF MBX TABLE LOCK
    CEFLOCK byte unsigned;                              /*OWNER OF CEF TABLE LOCK
    FILL_2 byte fill prefix SHDDEF tag $$;              /*UNUSED
    PRQWAIT word unsigned;                              /*PORTS WAITING FOR INTER-PROCESSOR REQUEST BLOCKS
                                                        /* (ONE BIT/PORT)
    POLL word unsigned;                                 /*PORTS ACTIVELY USING THE MEMORY
                                                        /* (ONE BIT/PORT)
    RESWAIT word unsigned dimension 16;                 /*PORTS WAITING FOR A RESOURCE
                                                        /* (ONE BIT/PORT, ONE MASK/RESOURCE)
```

```
    RESAVAIL word unsigned dimension 16;                /*PORTS NEEDING TO REPORT RESOURCE AVAILABLE
                                                        /* (ONE BIT/PORT, ONE MASK/RESOURCE)
    RESSUM word unsigned;                               /*PORTS WITH RESOURCES TO REPORT
                                                        /* (ONE BIT/PORT)
    FILL_3 word fill prefix SHDDEF tag $$;              /*UNUSED
    FILL_4 longword fill prefix SHDDEF tag $$;          /*UNUSED
/**** NOTE: THE FOLLOWING FIELDS MUST BE QUADWORD ALIGNED:
    PRQ quadword unsigned;                              /*FREE INTER-PROCESSOR REQUEST BLOCK LISTHEAD
    POOL quadword unsigned;                             /*FREE POOL BLOCK LISTHEAD
    PRQWRK quadword unsigned dimension 16;              /*INTER-PROCESSOR REQUEST WORK QUEUE LISTHEADS
                                                        /* (ONE LISTHEAD/PORT)
    constant ''LENGTH'' equals . prefix SHD$ tag K;     /*LENGTH OF DATAPAGE
    constant ''LENGTH'' equals . prefix SHD$ tag C;     /*LENGTH OF DATAPAGE
end SHDDEF;

end_module $SHDDEF;
```

```
module $SHLDEF;
/*+
/* SHL - SHAREABLE IMAGE LIST
/*
/* THIS LIST IS CREATED IN THE IMAGE FIXUP SECTION BY THE LINKER AND
/* USED BY THE IMAGE ACTIVATOR FOR DOING SHAREABLE IMAGE FIXUPS.
/*-

aggregate SHLDEF structure prefix SHL$;
    BASEVA longword unsigned;                       /* Base address of this shareable image
    SHLPTR longword unsigned;                       /* Pointer from SHL in shareable image
                                                    /*  to associated SHL in executable image
    "IDENT" longword unsigned;                      /* GSMATCH
    PERMCTX longword unsigned;                      /* Permanent sharable image context
    SHL_SIZE byte unsigned;                         /* Size of SHL elements
    FILL_1 byte fill prefix SHLDEF tag $$;          /* Spare for extensions
    FILL_2 word fill prefix SHLDEF tag $$;          /* Spare for extensions
    FILL_3 longword fill prefix SHLDEF tag $$;      /* Spare for extensions
    constant OLD_SHL_SIZE equals 56 prefix SHL tag $C;  /* Size of "old" SHL
    constant MAXNAMLNG  equals 39  prefix SHL tag $C;   /* Maximum length of image name
    IMGNAM_OVERLAY union fill;
        IMGNAM character length 40;                 /* Shareable image name (ASCIC string)
        constant "LENGTH" equals . prefix SHL$ tag K;   /* Length of shareable image list element
        constant "LENGTH" equals . prefix SHL$ tag C;   /* Length of shareable image list element
        NAMLNG byte unsigned;                       /* Synonym for name count
    end IMGNAM_OVERLAY;
end SHLDEF;

end_module $SHLDEF;
```

```
module $SLVDEF;
/*
/*
/* Define symbolic offsets for System Loadable Vectors.  These symbols
/* are used by the various pieces of the loadable EXEC, notably SCSVEC,
/* to create a list of vectors in system space and a corresponding image
/* that will be loaded into pool and connected to the system vectors.
/*
/*
aggregate SLV structure prefix SLV$;
    CODESIZE    longword unsigned;      /* Loadable image size (in bytes)
    INITRTN     longword unsigned;      /* Offset to init. routine
    SIZE        word unsigned;          /* Same as SLV$L_CODESIZE
    TYPE        byte unsigned;          /* Structure type (DYN$C_LOADCODE)
    SUBTYP      byte unsigned;          /* Sturcture Subtype
    PROT_R      byte unsigned;          /* writeable protection for image
    PROT_W      byte unsigned;          /* read-only protection for image
    SPARE       word unsigned;          /* spare field for future use
    SYSVECS     address;                /* address of vectors in SYS.EXE
    FACILITY    character length 16;    /* facility name (.ASCIC)
    LIST        character length 640;   /* Start of vector list (MAXVEC*5)
constant "LENGTH" equals .;             /* SLV$K_LENGTH
end SLV;
/*
/*
/* Define vector type codes.  The codes LODUMMY and HIDUMMY are
/* used as placeholders, to make the definition of the upper and
/* lower bound vector type symbols automatic.  New vector type codes
/* should be added at the end of the list, but before HIDUMMY.
/*
/*
constant (LODUMMY,                      /*
          LDATA,                        /* Longword pointer to data
          AJUMP,                        /* Aligned jump
          UJUMP,                        /* Unaligned jump
          SDATA,                        /* Specified data
          SJUMP,                        /* Specified jump
          HIDUMMY                       /*
         )
        equals 0 increment 1 prefix SLV tag $K;
constant MINTYPE equals SLV$K_LODUMMY+1 prefix SLV tag $K; /* Lower bound of vector type codes
constant MAXTYPE equals SLV$K_HIDUMMY-1 prefix SLV tag $K; /* Upper bount of vector type codes
constant MAXVEC equals 128 prefix SLV tag $K;              /* Max. # of vectors in list.

end_module $SLVDEF;
```

```
module $SMBDEF;                         /* Symbiont interface definitions

/*+
/* Symbolic definitions for the symbiont to job controller interface.
/*
/*      Public definitions of message types, item codes, and
/*      other constants utilied by the symbiont to job controller
/*      interface facility.
/*
/*-

/*
/*      Structure level
/*

constant SMBMSG$K_STRUCTURE_LEVEL   equals 1;   /* Current structure level
constant SMBMSG$K_STRUCTURE_LEVEL_1 equals 1;   /* Structure level 1


/*
/*      Request header
/*

aggregate REQUEST_HEADER structure prefix SMBMSG$;
    REQUEST_CODE        word unsigned;          /* Request code

        constant (                              /*  Define request codes
            PAUSE_TASK,                         /*  - STOP /QUEUE
            RESET_STREAM,                       /*  - STOP /QUEUE /RESET
            RESUME_TASK,                        /*  - START /QUEUE (when paused)
            START_STREAM,                       /*  - START /QUEUE (when stopped)
            START_TASK,                         /*  - task available
            STOP_STREAM,                        /*  - STOP /QUEUE /NEXT
            STOP_TASK,                          /*  - STOP /QUEUE /ABORT or /REQUEUE
            TASK_COMPLETE,                      /*  - stream is idle
            TASK_STATUS,                        /*  - asynchronous status update
            MAX_REQUEST_CODE                    /* MUST BE LAST
        ) equals 1 increment 1;

    STRUCTURE_LEVEL     byte unsigned;          /* Message structure level
    STREAM_INDEX        byte unsigned;          /* Stream index
end;


/*
/*      Item header
/*

aggregate ITEM_HEADER structure prefix SMBMSG$;
    ITEM_SIZE           word unsigned;          /* Item size
    ITEM_CODE           word unsigned;          /* Item code

        constant (                              /*  Define item codes
            ACCOUNTING_DATA,                    /*  - accounting information
            ACCOUNT_NAME,                       /*  - account name
```

```
        AFTER_TIME,                     /*  - /AFTER value
        ALIGNMENT_PAGES,                /*  - /ALIGN count
        BOTTOM_MARGIN,                  /*  - trailing blank lines
        CHARACTERISTICS,                /*  - /CHARACTERISTICS value
        CHECKPOINT_DATA,                /*  - checkpoint information
        CONDITION_VECTOR,               /*  - task error messages
        DEVICE_NAME,                    /*  - /ON value
        DEVICE_STATUS,                  /*  - device status
        ENTRY_NUMBER,                   /*  - job entry number
        EXECUTOR_QUEUE,                 /*  - this output queue
        FILE_COPIES,                    /*  - /COPIES value
        FILE_COUNT,                     /*  - current file copy number
        FILE_SETUP_MODULES,             /*  - file setup module list
        FIRST_PAGE,                     /*  - first page to print
        FORM_LENGTH,                    /*  - lines per page
        FORM_NAME,                      /*  - name of physical form
        FORM_SETUP_MODULES,             /*  - form setup module list
        FORM_WIDTH,                     /*  - columns per line
        FILE_IDENTIFICATION,            /*  - device, fid, and did
        FILE_SPECIFICATION,             /*  - file name
        JOB_COPIES,                     /*  - /JOB_COUNT value
        JOB_COUNT,                      /*  - current job copy number
        JOB_NAME,                       /*  - /NAME value
        JOB_RESET_MODULES,              /*  - job reset module list
        LAST_PAGE,                      /*  - last page to print
        LEFT_MARGIN,                    /*  - leading blank columns
        LIBRARY_SPECIFICATION,          /*  - library name
        MAXIMUM_STREAMS,                /*  - maximum supported symbiont
        MESSAGE_VECTOR,                 /*  - error messages to print
        NOTE,                           /*  - /NOTE value
        PAGE_SETUP_MODULES,             /*  - page setup module list
        PARAMETER_1,                    /*  - user parameter 1
        PARAMETER_2,                    /*  - user parameter 2
        PARAMETER_3,                    /*  - user parameter 3
        PARAMETER_4,                    /*  - user parameter 4
        PARAMETER_5,                    /*  - user parameter 5
        PARAMETER_6,                    /*  - user parameter 6
        PARAMETER_7,                    /*  - user parameter 7
        PARAMETER_8,                    /*  - user parameter 8
        PRINT_CONTROL,                  /*  - printing control
        PRIORITY,                       /*  - queue priority
        QUEUE,                          /*  - generic queue name
        REFUSE_REASON,                  /*  - reason task refused
        RELATIVE_PAGE,                  /*  - /BACKWARD, /FORWARD values
        REQUEST_CONTROL,                /*  - request control
        REQUEST_RESPONSE,               /*  - request code being responded to
        RIGHT_MARGIN,                   /*  - trailing blank columns
        SEARCH_STRING,                  /*  - /SEARCH value
        SEPARATION_CONTROL,             /*  - separation control
        STOP_CONDITION,                 /*  - reason for print abort
        TIME_QUEUED,                    /*  - time queued
        TOP_MARGIN,                     /*  - leading blank lines
        UIC,                            /*  - UIC of submittor
        USER_NAME,                      /*  - username
                                        /*
        MAX_ITEM_CODE                   /* MUST BE LAST
```

SYS

end

end

```
        ) equals 1 increment 1;
end;


/*
/*      ACCOUNTING_DATA item
/*

aggregate ACCOUNTING_DATA structure prefix SMBMSG$;
    PAGES_PRINTED       longword unsigned;          /* Pages printed
    qio_puts            longword unsigned;          /* Lines printed
    rms_gets            longword unsigned;          /* File reads
    CPU_TIME            longword unsigned;          /* Processor time
#s_accounting_data = .;
end;


/*
/*      CHECKPOINT_DATA item
/*

aggregate CHECKPOINT_DATA structure prefix SMBMSG$;
    FILLER              byte unsigned;              /* Reserved
    CHECKPOINT_LEVEL    byte unsigned;              /* Checkpoint structure level
    OFFSET              word unsigned;              /* Offset into record
    CARCON              longword unsigned;          /* Carriage control
    PAGE                longword unsigned;          /* Page number
    RECORD_NUMBER       longword unsigned;          /* Record number
    USER_KEY            quadword;                   /* User positioning key
#s_checkpoint_data = .;
end;


/*
/*      DEVICE_STATUS item
/*

aggregate DEVICE_STATUS structure prefix SMBMSG$;
    DEVICE_FLAGS        structure longword unsigned;    /* Device flags
        LOWERCASE           bitfield mask;              /*  - supports lowercase
        PAUSE_TASK          bitfield mask;              /*  - symbiont initiated pause
        REMOTE              bitfield mask;              /*  - device is remote
        SERVER              bitfield mask;              /*  - server symbiont
        STALLED             bitfield mask;              /*  - task stalled
        STOP_STREAM         bitfield mask;              /*  - symbiont requesting stop stream
        TERMINAL            bitfield mask;              /*  - device is a terminal
        UNAVAILABLE         bitfield mask;              /*  - device unavailable
        FILLER              bitfield length 32-^ fill;  /*  - force longword
        end;
end;
```

```
/*
/*      PRINT_CONTROL item
/*

aggregate PRINT_CONTROL structure prefix SMBMSG$;
    PRINT_FLAGS         structure longword unsigned;    /* Print flags
        DOUBLE_SPACE        bitfield mask;              /*  - double space
        PAGE_HEADER         bitfield mask;              /*  - print page headers
        PAGINATE            bitfield mask;              /*  - insert <FF>'s
        PASSALL             bitfield mask;              /*  - binary print file
        SEQUENCED           bitfield mask;              /*  - print sequence numbers
        SHEET_FEED          bitfield mask;              /*  - pause at every TOF
        TRUNCATE            bitfield mask;              /*  - truncate on overflow
        WRAP                bitfield mask;              /*  - wrap on overflow
        FILLER              bitfield length 32-^ fill;  /*  - force longword
    end;
end;


/*
/*      REQUEST_CONTROL item
/*

aggregate REQUEST       structure prefix SMBMSG$;
    REQUEST_FLAGS       structure longword unsigned;    /* Print flags
        ALIGNMENT_MASK      bitfield mask;              /*  - print A's and 9's
        PAUSE_COMPLETE      bitfield mask;              /*  - pause when request complete
        RESTARTING          bitfield mask;              /*  - job is restarting
        TOP_OF_FILE         bitfield mask;              /*  - rewind before resume
        FILLER              bitfield length 32-^ fill;  /*  - force longword
    end;
end;


/*
/*      SEPARATION_CONTROL item
/*

aggregate SEPARATION_CONTROL    structure prefix SMBMSG$;
    SEPARATION_FLAGS            structure longword unsigned;    /* Print flags
        FILE_BURST          bitfield mask;              /*  - print file burst page
        FILE_FLAG           bitfield mask;              /*  - print file flag page
        FILE_TRAILER        bitfield mask;              /*  - print file trailer page
        FILE_TRAILER_ABORT  bitfield mask;              /*  - print file trailer page
        JOB_FLAG            bitfield mask;              /*  - print job flag page
        JOB_BURST           bitfield mask;              /*  - print job burst page
        JOB_RESET           bitfield mask;              /*  - execute job reset sequence
        JOB_RESET_ABORT     bitfield mask;              /*  - execute job reset sequence
        JOB_TRAILER         bitfield mask;              /*  - print job trailer page
        JOB_TRAILER_ABORT   bitfield mask;              /*  - print job trailer page
        FILLER              bitfield length 32-^ fill;  /*  - force longword
    end;
end;
```

```
end_module $SMBDEF;
```

```
module $SPNBDEF;
/++
/* SPNB - SCA POLLER NAME BLOCK
/*
/* THIS DATA STRUCTURE CONTAINS A LIST OF PROCESS NAMES WHICH WILL
/* BE SEARCHED FOR ON THE GIVEN REMOTE NODE.
/*-



aggregate SPNBDEF structure prefix SPNB$;
    FLINK longword unsigned;                          /*FWD LINK
    BLINK longword unsigned;                          /*BCK LINK
    SIZE word unsigned;                               /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                               /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                             /*SCS STRUCTURE SUBTYPE FOR SPNB
    SB longword unsigned;                             /*SYSTEM BLOCK OF REMOTE NOTE
    ROUTINE longword unsigned;                        /*ADDRESS OF ROUTINE TO BE CALLED WHEN PROCESS FOUND
    INDEX byte unsigned;                              /*INDEX INTO PROCESS LIST OF NEXT PROCESS TO SEARCH FOR
    REFC word unsigned;                               /*NUMBER OF REFERENCES TO SPNB
    FREE byte unsigned dimension 1;                   /*FREE BYTE
    constant "HDRSIZ" equals . prefix SPNB$ tag C;    /*SIZE OF HEADER
    NAMLST byte unsigned;                             /*START OF VARIABLE LENGTH LIST OF ADDRESSES OF PROCESS NAMES
                                                      /*LIST IS ZERO TERMINATED
end SPNBDEF;

end_module $SPNBDEF;
```

```
module $SPPBDEF;
/++
/* SPPB - SCA POLLER PROCESS BLOCK
/*
/* THIS DATA STRUCTURE DESCRIBES A PROCESS NAME KNOWN
/* TO THE SCA DIRECTORY POLLER.
/*-



aggregate SPPBDEF structure prefix SPPB$;
    FLINK longword unsigned;                          /*FWD LINK
    BLINK longword unsigned;                          /*BCK LINK
    SIZE word unsigned;                               /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                               /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                             /*SCS STRUCTURE SUBTYPE FOR SPPB
    PROCNAM byte unsigned dimension 16;               /*ASCII STRING FOR PROCESS NAME
    RTN longword unsigned;                            /*ADDRESS OF NOTIFICATION ROUTINE
    CTX longword unsigned;                            /*CONTEXT FOR NOTIFICATION ROUTINE
    BIT word unsigned;                                /*BIT ASSIGNED TO THIS PROCESS NAME
    UNUSED_1 word unsigned fill;                      /*WORD RESERVED FOR EXPANSION
    constant "LENGTH" equals . prefix SPPB$ tag K;
    constant "LENGTH" equals . prefix SPPB$ tag C;

end SPPBDEF;

end_module $SPPBDEF;
```

```
module $STATEDEF;

/*+
/* SCHEDULING STATES
/*-


                                                        /* DEFINITIONS START AT 1

constant(
       COLPG                                            /*COLLIDED PAGE WAIT
     , MWAIT                                            /*MUTEX AND MISCELLANEOUS RESOURCE WAIT
     , CEF                                              /*COMMON EVENT FLAG WAIT STATE
     , PFW                                              /*PAGE FAULT WAIT
     , LEF                                              /*LOCAL EVENT FLAG WAIT
     , LEFO                                             /*LOCAL EVENT FLAG WAIT OUT OF BALANCE SET
     , HIB                                              /*HIBERNATE WAIT
     , HIBO                                             /*HIBERNATE WAIT OUT OF BALANCE SET
     , SUSP                                             /*SUSPENDED
     , SUSPO                                            /*SUSPENDED OUT OF THE BALANCE SET
     , FPG                                              /*FREEPAGE WAIT
     , COM                                              /*COMPUTE, IN BALANCE SET STATE
     , COMO                                             /*COMPUTE, OUT OF BALANCE SET STATE
     , CUR                                              /*CURRENT PROCESS STATE
     ) equals 1 increment 1  prefix SCH tag $C;

end_module $STATEDEF;
```

```
module $SYSAPDEF;

/**
/* SYSAP - FLAGS USED IN THE SYSAP-SCS INTERFACE
/*-



                                                    /*OPTIONS FOR DISPOSING OF
                                                    /* SENT DATAGRAM:
                                                    /* 0 ORIGIN, INCR OF 1:
constant (
      DISPQ                                         /* DISPOSE ON DG FREE QUEUE
    , DISPRET                                       /* DISPOSE BY RETURN TO SYSAP
    , DISPPO                                        /* DISPOSE BY RETURN TO POOL
    ) equals 0 increment 1 prefix SYSAP tag $C;

                                                    /*FLAGS SPECIFYING TYPE OF DG
                                                    /* REC'D FROM REMOTE SYSAP:
                                                    /* 0 ORIGIN, INCR OF 1:
constant (
      DGREC                                         /* DG REC'D FROM REMOTE
    , DGSNT                                         /* DG SENT
    ) equals 0 increment 1 prefix SYSAP tag $C;

end_module $SYSAPDEF;
```

```
module $TASTDEF;
/*
/* TERMINAL AST PACKET. THIS STRUCTURE IS USED BY TERMINAL SERVICES TO
/* DELIVER OUT OF BAND CHARACTER ASTS.
/*

aggregate TASTDEF structure prefix TAST$;
    FILL_1 longword dimension 7 fill prefix TASTDEF tag $$;/*RESERVE ACB REGION
    FLINK longword unsigned;                            /*FORWARD LINK
    AST longword unsigned;                              /*SAVED AST ADDRESS
    ASTPRM longword unsigned;                           /*SAVED AST PARAMETER
    PID longword unsigned;                              /*SAVED PID
    RMOD byte unsigned;                                 /*SAVED RMOD
    CTRL_OVERLAY union fill;
        CTRL byte unsigned;                             /*CONTROL FIELD
        CTRL_BITS structure fill;
            MASK_DSBL bitfield mask;                    /*DISABLE MASK PROCESSING
            INCLUDE bitfield mask;                      /*INCLUDE CHARACTER
            ONE_SHOT bitfield mask;                     /*ONE SHOT AST
            BUSY bitfield mask;                         /*BLOCK BUSY
            LOST bitfield mask;                         /*AST LOST
            ABORT bitfield mask;                        /*ABORT I/O
        end CTRL_BITS;
    end CTRL_OVERLAY;
    CHAN word unsigned;                                 /*CHANNEL
    "MASK" longword unsigned;                           /*OUT OF BAND MASK
    constant "LENGTH" equals . prefix TAST$ tag K;
    constant "LENGTH" equals . prefix TAST$ tag C;

    STATUS_BITS structure;
        FILL bitfield length 14;                        /* First byte and spares
        ABO bitfield mask;                              /* ABORT flag
        INC bitfield mask;                              /* INCLUDE flag
    end STATUS_BITS;

end TASTDEF;

end_module $TASTDEF;
```

```
module $TQEDEF;
/*+
/* TQE - TIME QUEUE ENTRY
/*
/* TIME QUEUE ENTRIES ARE UTILIZED TO SET TIMERS, WAKE UP PROCESSES, AND
/* FOR INTERNAL SYSTEM SUBROUTINES.
/*-


aggregate TQEDEF structure prefix TQE$;
    TQFL longword unsigned;                                 /*TIME QUEUE FORWARD LINK
    TQBL longword unsigned;                                 /*TIME QUEUE BACKWARD LINK
    SIZE word unsigned;                                     /*SIZE OF TQE IN BYTES
    TYPE byte unsigned;                                     /*STRUCTURE TYPE FOR TQE
    RQTYPE_OVERLAY union fill;
        RQTYPE byte unsigned;                               /*TIME QUEUE ENTRY TYPE
        RQTYPE_BITS structure fill;
            FILL_1 bitfield length 2 fill prefix TQEDEF tag $$;/* STARTING OFFSET
            REPEAT bitfield mask;                           /* REPEAT REQUEST (1=YES)
            ABSOLUTE bitfield mask;                         /* Absolute expiration time specified
        end RQTYPE_BITS;
    end RQTYPE_OVERLAY;
    PID_OVERLAY union fill;
        PID longword unsigned;                              /*TIMER OR WAKE REQUEST PROCESS ID
        FPC longword unsigned;                              /*TIMER SUBROUTINE ADDRESS
    end PID_OVERLAY;
    AST_OVERLAY union fill;
        AST longword unsigned;                              /*ADDRESS OF AST ROUTINE
        FR3 longword unsigned;                              /*TIMER SUBROUTINE SAVED R3
    end AST_OVERLAY;
    ASTPRM_OVERLAY union fill;
        ASTPRM longword unsigned;                           /*AST PARAMETER
        FR4 longword unsigned;                              /*TIMER SUBROUTINE SAVED R4
    end ASTPRM_OVERLAY;
    TIME quadword unsigned;                                 /*ABSOLUTE EXPIRATION TIME
    DELTA quadword unsigned;                                /*DELTA REPEAT TIME
    RMOD byte unsigned;                                     /*ACCESS MODE OF REQUEST
    EFN byte unsigned;                                      /*EVENT FLAG NUMBER AND EVENT GROUP
    FILL_2 word fill prefix TQEDEF tag $$;                  /*SPARE WORD
    RQPID longword unsigned;                                /*REQUESTER PROCESS ID
    constant "LENGTH" equals . prefix TQE$ tag K;           /*LENGTH OF STANDARD TQE
    constant "LENGTH" equals . prefix TQE$ tag C;           /*LENGTH OF STANDARD TQE

/*
/* TIME QUEUE ENTRY REQUEST TYPE DEFINITIONS
/*

    constant TMSNGL     equals 0  prefix TQE tag $C;    /*TIMER ENTRY SINGLE SHOT REQUEST
    constant SSREPT     equals 1+TQE$M_REPEAT  prefix TQE tag $C;/*SYSTEM SUBROUTINE REPEAT REQUEST
    constant SSSNGL     equals 1  prefix TQE tag $C;    /*SYSTEM SUBROUTINE SINGLE SHOT REQUEST
    constant WKREPT     equals 2+TQE$M_REPEAT  prefix TQE tag $C;/*WAKE ENTRY REPEAT REQUEST
    constant WKSNGL     equals 2  prefix TQE tag $C;    /*WAKE ENTRY SINGLE SHOT REQUEST

end TQEDEF;
```

```
end_module $TQEDEF;
```

```
module $UAFDEF;
/+++
/* User authorization file format
/* Note: With the exception of the username and account name,
/* all strings are blank padded counted strings. Username and
/* account name are uncounted, blank padded.
/*--


aggregate UAFDEF structure prefix UAF$;
    RTYPE byte unsigned;                    /* UAF record type
    constant (
        USER_ID                             /* main user ID record
        ) equals 1 increment 1 tag C;
    VERSION byte unsigned;                  /* UAF format version
    constant (
        VERSION1                            /* this version
        ) equals 1 increment 1 tag C;
    USRDATOFF word unsigned;                /* offset of counted string of user data

    USERNAME structure character length 32; /* username
        FILL_0 character length 31 fill;
        USERNAME_TAG character;             /* tag to differentiate records
    end USERNAME;
    UIC structure longword unsigned;        /* user ID code
        MEM word unsigned;                  /* member subfield
        GRP word unsigned;                  /* group subfield
    end UIC;
    SUB_ID longword unsigned;               /* user sub-identifier
    PARENT_ID quadword unsigned;            /* identifier of owner of this account
    constant KEYED_PART equals . tag C;     /* ISAM keys come this far
    ACCOUNT character length 32;            /* account name
    OWNER character length 32;              /* owner's name
    DEFDEV character length 32;             /* default device
    DEFDIR character length 64;             /* default directory
    LGICMD character length 64;             /* login command file
    DEFCLI character length 32;             /* default command interpreter
    CLITABLES character length 32;          /* user CLI tables

    PWD structure quadword unsigned;        /* hashed password
        PWD longword unsigned;              /* 32 bit subfield
    end PWD;
    PWD2 quadword unsigned;                 /* second password
    LOGFAILS word unsigned;                 /* count of login failures
    SALT word unsigned;                     /* random password salt
    ENCRYPT byte unsigned;                  /* encryption algorithm
    constant(                               /* encryption codes
          AD_II                             /* AUTODIN-II 32 bit crc code
        , PURDY                             /* Purdy polynomial over salted input
        , PURDY_V                           /* Purdy polynomial + variable length username
        ) equals 0 increment 1 tag C;
    ENCRYPT2 byte unsigned;                 /* encryption algorithm for 2nd pwd
    PWD_LENGTH byte unsigned;               /* minimum password length
    FILL_1 byte dimension 1 fill tag $$;
    EXPIRATION quadword unsigned;           /* expiration date for account
    PWD_LIFETIME quadword unsigned;         /* password lifetime
```

```
    PWD_DATE quadword unsigned;                 /* date of password change
    PWD2_DATE quadword unsigned;                /* date of 2nd password change
    LASTLOGIN_I quadword unsigned;              /* date of last interactive login
    LASTLOGIN_N quadword unsigned;              /* date of last non-interactive login
    PRIV quadword unsigned;                     /* process privilege vector
    DEF_PRIV quadword unsigned;                 /* default process privileges
    MIN_CLASS structure;                        /* minimum security class
        FILL_2 byte dimension 20 fill;
    end MIN_CLASS;
    MAX_CLASS structure;                        /* maximum security class
        FILL_3 byte dimension 20 fill;
    end MAX_CLASS;
    FLAGS structure longword unsigned;          /* user flags longword
        DISCTLY bitfield;                       /* no user control-y
        DEFCLI bitfield;                        /* only allow user default CLI
        LOCKPWD bitfield;                       /* disable SET PASSWORD command
        CAPTIVE bitfield;                       /* captive account (no overrides)
        DISACNT bitfield;                       /* no interactive login
        DISWELCOM bitfield;                     /* skip welcome message
        DISMAIL bitfield;                       /* skip new mail message
        NOMAIL bitfield;                        /* disable mail delivery
        GENPWD bitfield;                        /* passwords must be generated
        PWD_EXPIRED bitfield;                   /* password has expired
        PWD2_EXPIRED bitfield;                  /* 2nd password has expired
        AUDIT bitfield;                         /* audit all actions
        DISREPORT bitfield;                     /* skip last login messages
        DISRECONNECT bitfield;                  /* inhibit reconnections
    end FLAGS;
    NETWORK_ACCESS_P byte unsigned dimension 3; /* hourly network access, primary
    NETWORK_ACCESS_S byte unsigned dimension 3; /* hourly network access, secondary
    BATCH_ACCESS_P byte unsigned dimension 3;   /* hourly batch access, primary
    BATCH_ACCESS_S byte unsigned dimension 3;   /* hourly batch access, secondary
    LOCAL_ACCESS_P byte unsigned dimension 3;   /* hourly local access, primary
    LOCAL_ACCESS_S byte unsigned dimension 3;   /* hourly local access, secondary
    DIALUP_ACCESS_P byte unsigned dimension 3;  /* hourly dialup access, primary
    DIALUP_ACCESS_S byte unsigned dimension 3;  /* hourly dialup access, secondary
    REMOTE_ACCESS_P byte unsigned dimension 3;  /* hourly remote access, primary
    REMOTE_ACCESS_S byte unsigned dimension 3;  /* hourly remote access, secondary
    FILL_4 byte dimension 12 fill tag $$;       /* space for 2 more access types
    PRIMEDAYS structure byte unsigned;          /* bits representing primary days
        MONDAY bitfield;                        /* bit clear means this is a primary day
        TUESDAY bitfield;                       /* bit set means this is an off day
        WEDNESDAY bitfield;
        THURSDAY bitfield;
        FRIDAY bitfield;
        SATURDAY bitfield;
        SUNDAY bitfield;
    end PRIMEDAYS;
    FILL_5 byte dimension 1 fill tag $$;

    PRI byte unsigned;                          /* base process priority
    QUEPRI byte unsigned;                       /* maximum job queuing priority
    MAXJOBS word unsigned;                      /* maximum jobs for UIC allowed
                                                /* 0 means no limit
    MAXACCTJOBS word unsigned;                  /* maximum jobs for account allowed
                                                /* 0 means no limit
```

```
    MAXDETACH word unsigned;                    /* maximum detached processes for UIC
                                                /* 0 means no limit
    PRCCNT word unsigned;                       /* subprocess creation limit
    BIOLM word unsigned;                        /* buffered I/O limit
    DIOLM word unsigned;                        /* direct I/O limit
    TQCNT word unsigned;                        /* timer queue entry limit
    ASTLM word unsigned;                        /* AST queue limit
    ENQLM word unsigned;                        /* enqueue limit
    FILLM word unsigned;                        /* open file limit
    SHRFILLM word unsigned;                     /* shared file limit
    WSQUOTA longword unsigned;                  /* working set size quota
    DFWSCNT longword unsigned;                  /* default working set size
    WSEXTENT longword unsigned;                 /* working set size limit
    PGFLQUOTA longword unsigned;                /* page file quota
    CPUTIM longword unsigned;                   /* CPU time quota
    BYTLM longword unsigned;                    /* buffered I/O byte count limit
    PBYTLM longword unsigned;                   /* paged buffer I/O byte count limit
    JTQUOTA longword unsigned;                  /* job-wide logical name table creation quota
    PROXY_LIM word unsigned;                    /* number of proxies user can grant
    PROXIES word unsigned;                      /* number of proxies granted
    ACCOUNT_LIM word unsigned;                  /* number of sub-accounts allowed
    ACCOUNTS word unsigned;                     /* number of sub-accounts in use
    FILL_99 byte dimension 64 fill tag $$;
    constant FIXED equals . prefix UAF$ tag K;  /* length of fixed portion
    constant FIXED equals . prefix UAF$ tag C;  /* length of fixed portion
    FILL_100 byte dimension 768 fill tag $$;    /* user-extensible area
    constant "LENGTH" equals . prefix UAF$ tag K;
    constant "LENGTH" equals . prefix UAF$ tag C;
end UAFDEF;

end_module $UAFDEF;
```

```
module $UASDEF;
/*+
/* UNIBUS ADDRESS SPACE REGISTER DEFINITIONS FOR DW750
/*      (SECOND UNIBUS ADAPTER ON 11/750)
/*-


aggregate UASDEF structure prefix UAS$;
        FILL_1 byte dimension 5216 fill prefix UASDEF tag $$;

        IP structure;                                                   /* INTER-PROCESSOR EXERCISER COMMUNICATOR
                FILL_2 word dimension 2 fill prefix UASDEF tag $$; /* ADDRESS AND DATA REGISTERS NOT CURRENTLY USED

                CR1_OVERLAY union fill;
                        IP_CR1 word unsigned;                   /* THE THIRD IPEC REGISTER, CR1
                        CR1_BITS structure fill;
                                FILL_3 bitfield length 12 fill prefix UASDEF tag $$; /* SKIP BITS OF NO INTEREST
                                IP_CR1_PIE bitfield mask;       /* POWERFAIL INTERRUPT ENABLE
                                IP_CR1_PDN bitfield mask;       /* POWER DOWN STATUS BIT
                        end CR1_BITS;
                end CR1_OVERLAY;

        end IP;

end UASDEF;

end_module $UASDEF;
```

```
module $UBADEF;
/*+
/* UNIBUS ADAPTER REGISTER OFFSET DEFINITIONS
/*-



aggregate UBADEF structure prefix UBA$;
    CSR_OVERLAY union fill;
        CSR longword unsigned;                          /*CONFIGURATION STATUS REGISTER
        CSR_BITS structure fill;
            CSR_ADCOD bitfield length 8;                /* ADAPTER CODE FIELD
            FILL_1 bitfield length 8 fill prefix UBADEF tag $$;/* RESERVED BITS
            CSR_UBIC bitfield mask;                     /* UNIBUS INITIALIZATION COMPLETE
            CSR_UBPDN bitfield mask;                    /* UNIBUS POWER DOWN
            CSR_UBIIP bitfield mask;                    /* UNIBUS INITIALIZATION IN PROGRESS
            FILL_2 bitfield length 2 fill prefix UBADEF tag $$;/* RESERVED BITS
            CSR_OT bitfield mask;                       /* OVER TEMPERATURE
            CSR_PU bitfield mask;                       /* ADAPTER POWER UP
            CSR_PD bitfield mask;                       /* ADAPTER POWER DOWN
            FILL_3 bitfield length 2 fill prefix UBADEF tag $$;/* RESERVED BITS
            CSR_XMFLT bitfield mask;                    /* TRANSMITTER FAULT
            CSR_MT bitfield mask;                       /* MULTIPLE TRANSMITTERS
            CSR_IS bitfield mask;                       /* INTERLOCK SEQUENCE FAULT
            CSR_URD bitfield mask;                      /* UNEXPECTED READ DATA
            CSR_WS bitfield mask;                       /* WRITE SEQUENCE DATA
            CSR_PE bitfield mask;                       /* SBI PARITY ERROR
        end CSR_BITS;
    end CSR_OVERLAY;
    CR_OVERLAY union fill;
        CR longword unsigned;                           /*CONTROL REGISTER
        CR_BITS structure fill;
            CR_INIT bitfield mask;                      /* ADAPTER INITIALIZATION
            CR_UBPF bitfield mask;                      /* UNIBUS POWER FAIL
            CR_CNFIE bitfield mask;                     /* CONFIGURATION INTERRUPT ENABLE
            CR_SUEFIE bitfield mask;                    /* SBI TO UNIBUS ERROR FIELD INTERRUPT ENABLE
            CR_USEFIE bitfield mask;                    /* UNIBUS TO SBI ERROR FIELD INTERRUPT ENABLE
            CR_BRIE bitfield mask;                      /* BUS REQUEST INTERRUPT ENABLE
            CR_IFSIE bitfield mask;                     /* INTERRUPT FIELD SWITCH INTERRUPT ENABLE
            CR_ARLVL bitfield mask length 2;            /* ADAPTER REQUEST LEVEL
            FILL_4 bitfield length 17 fill prefix UBADEF tag $$;/* RESERVED BITS
            CR_MRDSB bitfield length 5;                 /* MAP REGISTER DISABLE
        end CR_BITS;
    end CR_OVERLAY;
    SR_OVERLAY union fill;
        SR longword unsigned;                           /*STATUS REGISTER
        SR_BITS structure fill;
            SR_SSYNC bitfield mask;                     /* UNIBUS SLAVE SYNC TIMEOUT
            SR_UBSTO bitfield mask;                     /* UNIBUS SELECT TIMEOUT
            SR_LER bitfield mask;                       /* LOST ERROR
            SR_MRPE bitfield mask;                      /* MAP REGISTER PARITY ERROR
            SR_IVMR bitfield mask;                      /* INVALID MAP REGISTER
            SR_DPPE bitfield mask;                      /* DATAPATH PARITY ERROR
            SR_CXTMO bitfield mask;                     /* COMMAND TRANSMISSION TIMEOUT
            SR_CXTER bitfield mask;                     /* COMMAND TRANSMISSION ERROR
            SR_CRD bitfield mask;                       /* CORRECTED READ DATA
```

```
            SR_RDS bitfield mask;                       /* READ DATA SUBSTITUTE
            SR_RDTO bitfield mask;                      /* READ DATA TIMEOUT
            SR_BRID bitfield mask;                      /* BUS REQUEST INTERRUPT DONE
            FILL_5 bitfield length 12 fill prefix UBADEF tag $$;/* RESERVED BITS
            SR_BRRVF bitfield length 4;                 /* BUS REQUEST RECEIVE VECTOR FULL
            SR_BRSVF bitfield mask;                     /* BUS REQUEST SEND VECTOR FULL
            SR_RIE bitfield mask;                       /* REQUEST INTERRUPT ENABLED
            SR_UBIFS bitfield mask;                     /* UNIBUS INTERRUPT FIELD SWITCH
        end SR_BITS;
    end SR_OVERLAY;
    DCR longword unsigned;                              /*DIAGNOSTIC CONTROL REGISTER
    FMER_OVERLAY union fill;
        FMER longword unsigned;                         /*FAILED MAP ENTRY REGISTER
        FMER_BITS structure fill;
            FMER_MRN bitfield length 9;                 /* FAILED MAP REGISTER NUMBER
        end FMER_BITS;
    end FMER_OVERLAY;
    FUBAR_OVERLAY union fill;
        FUBAR longword unsigned;                        /*FAILED UNIBUS ADDRESS REGISTER
        FUBAR_BITS structure fill;
            FUBAR_ADR bitfield length 18;               /* FAILED SBI TO UNIBUS ADDRESS
        end FUBAR_BITS;
    end FUBAR_OVERLAY;
    FILL_6 longword dimension 2 fill prefix UBADEF tag $$;/*RESERVED REGISTERS
    BRSVR longword unsigned dimension 4;                /*BUS REQUEST SEND VECTOR REGISTERS
    BRRVR_OVERLAY union fill;
        BRRVR longword unsigned dimension 4;            /*BUS REQUEST RECEIVE VECTOR REGISTER
        BRRVR_BITS structure fill;
            BRRVR_IVA bitfield length 16;               /* INTERRUPT VECTOR ADDRESS
            FILL_7 bitfield length 15 fill prefix UBADEF tag $$;/* RESERVED BITS
            BRRVR_AIR bitfield mask;                    /* ADAPTER INTERRUPT REQUEST PENDING
        end BRRVR_BITS;
    end BRRVR_OVERLAY;
    DPR_OVERLAY union fill;
        DPR longword unsigned dimension 16;             /*DATAPATH REGISTERS
        DPR_BITS structure fill;
            DPR_ADDR bitfield length 16;                /* BUFFERED UNIBUS ADDRESS
            DPR_STATE bitfield length 8;                /* BUFFER STATE FLAGS
            FILL_8 bitfield length 5 fill prefix UBADEF tag $$;/* RESERVED BITS
            DPR_DPF bitfield mask;                      /* DATAPATH FUNCTION
            DPR_XMTER bitfield mask;                    /* BUFFER TRANSFER ERROR
            DPR_BNE bitfield mask;                      /* BUFFER NOT EMPTY
        end DPR_BITS;
    end DPR_OVERLAY;
    FILL_9 byte dimension 1920 fill prefix UBADEF tag $$;/* VALUE IS 2048-128
    MAP_OVERLAY union fill;
        MAP longword unsigned dimension 496;            /*MAP REGISTERS
        MAP_BITS structure fill;
            MAP_ADDR bitfield length 21;                /* SBI PAGE ADDRESS
            MAP_DPD bitfield length 4;                  /* DATAPATH DESIGNATOR
            MAP_BO bitfield mask;                       /* BYTE OFFSET
            FILL_10 bitfield length 5 fill prefix UBADEF tag $$;/* RESERVED BITS
            MAP_VALID bitfield mask;                    /* MAP REGISTER VALID
        end MAP_BITS;
        constant MAXDP  equals 15  prefix UBA tag $C;   /*MAXIMUM DATAPATH !
```

```
    end MAP_OVERLAY;
end UBADEF;

end_module $UBADEF;
```

```
SYS

MOD
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
agg

/*
/*
/*

/*
/*
/*
/*

/*

/*
```

```
module $UBIDEF;
/*+
/* UNIBUS INTERCONNECT (VAX 11/750 & 11/730) REGISTER OFFSETS AND FIELDS
/*-


aggregate UBIDEF union prefix UBI$;
    DPR longword unsigned dimension 4;                  /*DATAPATH REGISTERS
                                                        /* (DPR 0 NOT IMPLEMENTED)
        DPR_BITS structure fill;
            DPR_PUR bitfield mask;                      /* DATAPATH PURGE
            FILL_1 bitfield length 28 fill prefix UBIDEF tag $$;/* RESERVED BITS
            DPR_UCE bitfield mask;                      /* UNCORRECTABLE ERROR
            DPR_NXM bitfield mask;                      /* NON-EXISTENT MEMORY
            DPR_ERROR bitfield mask;                    /* ERROR (UCE!NXM)
        end DPR_BITS;
end UBIDEF;

aggregate UBIDEF1 structure prefix UBI$;
    FILL_6 byte dimension 16 fill prefix UBIDEF tag $$;
    DSR_OVERLAY union fill;
        DSR longword unsigned dimension 4;              /*DIAGNOSTIC STATUS REGISTERS
                                                        /* (DSR 0 NOT IMPLEMENTED)
        DSR_BITS structure fill;
            FILL_2 bitfield length 27 fill prefix UBIDEF tag $$;/* RESERVED BITS
            DSR_CD bitfield mask;                       /* ALL 4 BYTES IN BDP FULL
            DSR_BF bitfield length 4;                   /* BYTE 0,1,2,3 IN BDP HAS VALID DATA
        end DSR_BITS;
    end DSR_OVERLAY;
end UBIDEF1;

aggregate UBIDEF2 structure prefix UBI$;
    FILL_7 byte dimension 16 fill prefix UBIDEF tag $$;
    SR_OVERLAY union fill;
        SR longword unsigned;                           /*UB STATUS REGISTER:
        SR_BITS structure fill;
            FILL_3 bitfield length 14 fill prefix UBIDEF tag $$;/* RESERVED BITS
            SR_UWE bitfield mask;                       /* UNCORRECTED WRITE ERROR
            SR_MRPE bitfield mask;                      /* MAP REGISTER PARITY ERROR
            SR_NXM bitfield mask;                       /* NONEXISTENT MEMORY REF
            FILL_4 bitfield length 14 fill prefix UBIDEF tag $$;/* RESERVED BITS
            SR_UCE bitfield mask;                       /* UNCORRECTED READ ERROR
        end SR_BITS;
                                                        /*END OF CPU_SPECIFIC REGISTERS
    end SR_OVERLAY;
end UBIDEF2;

aggregate UBIDEF3 structure prefix UBI$;
    FILL_5 byte dimension 2048 fill prefix UBIDEF tag $$;/*RESERVE ^X800 BYTES
    MAP longword unsigned dimension 496;                /*MAP REGISTERS, SAME FORMAT AS UBA
    constant MAXDP      equals 3  prefix UBI tag $C;    /*MAXIMUM DATAPATH !
    constant PURCNT     equals 10  prefix UBI tag $C;   /*MAX ! OF TESTS OF PURGE DONE
end UBIDEF3;

end_module $UBIDEF;
```

SYS

/*

/*

/*

/*-

/*

/**

/*-

/*

/*

/*

/**

/*-

/*

/**

```
module $UBMDDEF;
/++
/* UBMD - UNIBUS Map Descriptor used to record UNIBUS map registers
/*        and datapaths allocated.
/*-



aggregate UBMDDEF structure prefix UBMD$;
    MAPREG word unsigned;                       /* Starting map register
    NUMREG byte unsigned;                       /* Number of registers in extent
    DATAPATH byte unsigned;                     /* Associated Buffered datapath

end UBMDDEF;

end_module $UBMDDEF;
```

```
{+
{ UCB - UNIT CONTROL BLOCK
{
{ THERE IS ONE UCB FOR EACH DEVICE UNIT IN A SYSTEM.
{-

module $UCBDEF;

aggregate UCBDEF structure prefix UCB$;
    FQFL_OVERLAY union fill;
        FQFL longword unsigned;                 /*FORK QUEUE FORWARD LINK
        UNIT_SEED word unsigned;                /* UNIT NUMBER SEED
        MB_SEED word unsigned;                  /* MB -- UNIT NUMBER SEED
        RQFL longword unsigned;                 /* NET -- RCV QUEUE FORWARD LINK
    end FQFL_OVERLAY;
    FQBL_OVERLAY union fill;
        FQBL longword unsigned;                 /*FORK QUEUE BACKWARD LINK
        RQBL longword unsigned;                 /* NET -- RCV QUEUE BACKWARD LINK
    end FQBL_OVERLAY;
    SIZE word unsigned;                         /*SIZE OF UCB IN BYTES
    TYPE byte unsigned;                         /*STRUCTURE TYPE FOR UCB
    FIPL byte unsigned;                         /*FORK INTERRUPT PRIORITY LEVEL
    FPC_OVERLAY union fill;
        FPC longword unsigned;                  /*FORK PC
        ASTQFL longword unsigned;               /* MB -- AST QUEUE LISTHEAD FORWARD LINK
        PARTNER character;                      /* NET -- PARTNER'S NODENAME
    end FPC_OVERLAY;
    FR3_OVERLAY union fill;
        FR3 longword unsigned;                  /*FORK R3
        ASTQBL longword unsigned;               /* MB -- AST QUEUE LISTHEAD BACKWARD LINK
    end FR3_OVERLAY;
    FR4_OVERLAY union fill;
        FR4 longword unsigned;                  /*FORK R4
        MB_FR4_FIELDS structure fill;
            MSGMAX word unsigned;               /* MB -- MAXIMUM MESSAGES ALLOWED
            MSGCNT word unsigned;               /* MB -- CURRENT NUMBER OF MESSAGES
        end MB_FR4_FIELDS;
        FIRST longword unsigned;                /* NET -- ADDR OF 1ST SEG OF CHAINED MSG
    end FR4_OVERLAY;
    BUFQUO_OVERLAY union fill;
        BUFQUO word unsigned;                   /* BUFFERED I/O QUOTA CHARGED FOR THIS UCB
        DSTADDR word unsigned;                  /* NET -- REMOTE CONNECT NO.
    end BUFQUO_OVERLAY;
    SRCADDR word unsigned;                      /* NET -- LOCAL CONNECT NO.
    ORB longword unsigned;                      /* OBJECT'S RIGHTS BLOCK ADDRESS
    LOCKID_OVERLAY union fill;
        LOCKID longword unsigned;               /*DEVICE LOCK ID
        CPID longword unsigned;                 /*PID CHARGED FOR BUFQUO BY UCBCREDEL
    end LOCKID_OVERLAY;
    CRB longword unsigned;                      /*ADDRESS OF PRIMARY CHANNEL REQUEST BLOCK
    DDB longword unsigned;                      /*BACKPOINTER TO DEVICE DATA BLOCK
    PID longword unsigned;                      /*PROCESS ID OF OWNER PROCESS
    LINK longword unsigned;                     /*ADDRESS OF NEXT UCB FOR RESPECTIVE DDB
    VCB longword unsigned;                      /*ADDRESS OF VOLUME CONTROL BLOCK
    DEVCHAR union fill;                         /*DEVICE CHARACTERISTIC BITS
        DEVCHAR quadword unsigned;              /* Device characteristic bits quadword
```

```
        DEVCHAR_Q_BLOCK structure fill;
            DEVCHAR longword unsigned;              /*  Original device characteristic bits
            DEVCHAR2 longword unsigned;             /*  Extended device characteristic bits
        end DEVCHAR_Q_BLOCK;
    end DEVCHAR;
    DEVCLASS byte unsigned;                         /*DEVICE CLASS
    DEVTYPE byte unsigned;                          /*DEVICE TYPE
    DEVBUFSIZ word unsigned;                        /*DEVICE DEFAULT BUFFER SIZE
    DEVDEPEND_Q_OVERLAY union fill;                 /*DEVICE DEPENDENT DATA
        DEVDEPEND quadword unsigned;                /*Device dependent quadword
        DEVDEPEND_Q_BLOCK structure;
            DEVDEPEND_OVERLAY union fill;
                DEVDEPEND longword unsigned;        /*  First device dependent longword
                DISK_DEVDEPEND structure;           /*    Disk fields
                    SECTORS byte unsigned;          /*      Sectors per track
                    TRACKS byte unsigned;           /*      Track per cylinder
                    CYLINDERS word unsigned;        /*      Cylinders per disk
                end DISK_DEVDEPEND;
                TERM_DEVDEPEND structure;           /*    Terminal fields
                    TERM_DEVDEPEND_FILL byte dimension 3
                        fill prefix UCBDEF tag $$;
                    VERTSZ byte unsigned;           /*      Vertical page size (lines per page)
                end TERM_DEVDEPEND;
                NET_DEVDEPEND structure;            /*    Network fields
                    LOCSRV byte unsigned;           /*      Local link services
                    REMSRV byte unsigned;           /*      Remote link services
                    BYTESTOGO word unsigned;        /*      No. of bytes left in rcv bfr
                end NET_DEVDEPEND;
                JNL_SEQNO longword unsigned;        /*    Journal -- Running sequence number
            end DEVDEPEND_OVERLAY;
            DEVDEPND2_OVERLAY union fill;
                DEVDEPND2 longword unsigned;        /*  Second device dependent long word
                TT_DEVDP1 longword unsigned;        /*    Terminal -- Device dependent long word
            end DEVDEPND2_OVERLAY;
        end DEVDEPEND_Q_BLOCK;
    end DEVDEPEND_Q_OVERLAY;
    IOQFL longword unsigned;                        /*I/O QUEUE LISTHEAD FORWARD LINK
    IOQBL longword unsigned;                        /*I/O QUEUE LISTHEAD BACKWARD LINK
    UNIT word unsigned;                             /*PHYSICAL DEVICE UNIT NUMBER
    CHARGE_OVERLAY union fill;
        CHARGE word unsigned;                       /*MAILBOX BYTE COUNT QUOTA CHARGE
        RWAITCNT word unsigned;                     /* CLASS DRIVERS -- THREADS WAITING RESOURCES
        CTRLR_ALLOC_FIELDS structure fill;
            CM1 byte unsigned;                      /* LEVEL 1 CONTROLLER ALLOCATION MASK
            CM2 byte unsigned;                      /* LEVEL 2 CONTROLLER ALLOCATION MASK
        end CTRLR_ALLOC_FIELDS;
    end CHARGE_OVERLAY;
    IRP longword unsigned;                          /*CURRENT I/O REQUEST PACKET ADDRESS
    REFC word unsigned;                             /*REFERENCE COUNT OF PROCESSES
    DIPL_OVERLAY union fill;
        DIPL byte unsigned;                         /*DEVICE INTERRUPT PRIORITY LEVEL
        STATE byte unsigned;                        /* NET -- LINK STATE FOR NETWORK TRANSITIONS
    end DIPL_OVERLAY;
    AMOD byte unsigned;                             /*ALLOCATION ACCESS MODE
    AMB longword unsigned;                          /*ASSOCIATED UNIT CONTROL BLOCK POINTER
    STS_OVERLAY union fill;
```

```
        STS longword unsigned;                  /*DEVICE UNIT STATUS
        STS word unsigned;
        STS_BITS structure fill;
            TIM bitfield mask;                  /* TIME OUT ENABLED (1=YES)
            INT bitfield mask;                  /* INTERRUPT EXPECTED (1=YES)
            ERLOGIP bitfield mask;              /* ERROR LOG IN PROGRESS ON UNIT (1=YES)
            CANCEL bitfield mask;               /* CANCEL I/O ON UNIT (1=YES)
            ONLINE bitfield mask;               /* UNIT ONLINE (1=YES)
            POWER bitfield mask;                /* POWER FAILED WHILE UNIT BUSY (1=YES)
            TIMOUT bitfield mask;               /* UNIT TIMED OUT (1=YES)
            INTTYPE bitfield mask;              /* RECEIVER INTERRUPT IF SET
            BSY bitfield mask;                  /* UNIT IS BUSY (1=YES)
            MOUNTING bitfield mask;             /* DEVICE IS BEING MOUNTED
            DEADMO bitfield mask;               /* DEALLOCATE AT DISMOUNT
            VALID bitfield mask;                /* VOLUME IS SOFTWARE VALID
            UNLOAD bitfield mask;               /* UNLOAD VOLUME AT DISMOUNT
            TEMPLATE bitfield mask;             /* SET IF THIS IS TEMPLATE UCB
                                                {  FROM WHICH OTHER UCB'S FOR
                                                {  THIS DEVICE TYPE ARE MADE
            MNTVERIP bitfield mask;             /* MOUNT VERIFICATION IN PROGRESS
            WRONGVOL bitfield mask;             /* WRONG VOLUME DETECTED DURING MOUNT VERIFICATION
            DELETEUCB bitfield mask;            /* DELETE THIS UCB WHEN REFC REACHES ZERO
            LCL_VALID bitfield mask;            /* VOLUME IS VALID ON THE LOCAL NODE
            SUPMVMSG bitfield mask;             /* IF SET, SUPPRESS SUCCESS TYPE MOUNT VER. MSGS.
                                                {  CLEARED BY ANY ERROR IN MOUNT VERIFICATION
            MNTVERPND bitfield mask;            /* MOUNT VERIFICATION IS PENDING ON BUSY DEVICE.
            DISMOUNT bitfield mask;             /* DISMOUNT IN PROGRESS
            CLUTRAN bitfield mask;              /* VAXcluster STATE TRANSITION IN PROGRESS
        end STS_BITS;
    end STS_OVERLAY;
    DEVSTS_OVERLAY union fill;
        DEVSTS word unsigned;                   /*DEVICE DEPENDENT STATUS
        DEVSTS_GENERAL_BITS structure fill;     /* Generally used bits
            JOB bitfield mask;                  /*   Job Controller notified
            devsts_gen_fill bitfield length 5 fill;
            TEMPL_BSY bitfield mask;            /*   Template UCB is busy
        end DEVSTS_GENERAL_BITS;
        DEVSTS_MAILBX_BITS structure fill;      /* Mailbox status bits
            PRMMBX bitfield mask;               /*   Permanent mailbox
            DELMBX bitfield mask;               /*   Mailbox marked for delete
            devsts_mb_fill bitfield length 1 fill;
            SHMMBX bitfield mask;               /*   Shared memory mailbox
        end DEVSTS_MAILBX_BITS;
        DEVSTS_TERM_BITS structure fill;        /* Terminal status bits
            devsts_tt_fill bitfield length 1 fill;
            TT_TIMO bitfield mask;              /*   Terminal read timeout in progress
            TT_NOTIF bitfield mask;             /*   Terminal user notified of unsolicted data
            TT_HANGUP bitfield mask;            /*   Process hang up
            TT_DEVSTS_FILL bitfield length 15-^; /*   fill to the end the word
            TT_NOLOGINS bitfield mask;          /* NOLOGINS ALLOWED
        end DEVSTS_TERM_BITS;
        DEVSTS_NET_BITS structure fill;         /* Network status bits
            devsts_net_fill1 bitfield length 2 fill;
            NT_BFROVF bitfield mask;            /*   Too many bytes rcvd
            devsts_net_fill2 bitfield fill;
            NT_NAME bitfield mask;              /*   Link has declared a connect name
```

/*

```
        NT_BREAK bitfield mask;                 /*   Link is being broken
    end DEVSTS_NET_BITS;
    DEVSTS_DISKS structure fill;                /* Disk (all disks) status bits
                                                {  (using the low order byte, for
                                                {  compatibility with class driver
                                                {  usage of the high order word)
        ECC bitfield mask;                      /*   ECC correction was made
        DIAGBUF bitfield mask;                  /*   Diagnostic buffer specified
        NOCNVRT bitfield mask;                  /*   No LBN to media address conversion
        DX_WRITE bitfield mask;                 /*   Console floppy write operation
        DATACACHE bitfield mask;                /*   Data blocks being cached
    end DEVSTS_DISKS;
    DEVSTS_MSCP_CLASS_BITS structure fill;      /* MSCP class driver bits
        byte_fill bitfield length 8 fill;       {  (using the high order byte only)
        MSCP_MNTVERIP bitfield mask;            /*   Mount verification in progress
        MSCP_INITING bitfield mask;             /*   UCB is being initialized
        MSCP_WAITBMP bitfield mask;             /*   RWAITCNT has been bumped
        MSCP_FLOVR bitfield mask;               /*   Bit toggled everytime a failover succeeds.
        MSCP_PKACK bitfield mask;               /*   Set when a IO$_PACKACK is in progress.
        MSCP_WRTP bitfield mask;                /*   Unit MSCP write protected in some way.
    end DEVSTS_MSCP_CLASS_BITS;
    DEVSTS_TAPE_CLASS_BITS structure fill;      /* Tape class driver bits
                                                {  (using the low order byte only)
        TU_OVRSQCHK bitfield mask;              /*   Override sequence checking
        TU_TRACEACT bitfield mask;              /*   IRP trace table active
        TU_SEQNOP bitfield mask;                /*   Sequential NOP tape operation in progress
    end DEVSTS_TU_CLASS_BITS;
    DEVSTS_JOURNAL_BITS structure fill;         /* Journal status bits
        devsts_jnl_fill bitfield length 4 fill;
        PERM_JNL bitfield mask;                 /*   Permanent journal device
        KNOWN_JNL bitfield mask;                /*   Known journal
        JNL_CLS bitfield mask;                  /*   Cluster journal
        JNL_SLV bitfield mask;                  /*   Slave journal UCB
        CDELE_PND bitfield mask;                /*   Cluster delete operation pending
        JNL_UNMAST bitfield mask;               /*   Device unmastered
    end DEVSTS_JOURNAL_BITS;
end DEVSTS_OVERLAY;
QLEN word unsigned;                             /* Device queue length
DUETIM longword unsigned;                       /*DUE TIME FOR I/O COMPLETION
OPCNT longword unsigned;                        /*COUNT OF OPERATIONS COMPLETED
SVPN_OVERLAY union fill;
    SVPN longword unsigned;                     /*SYSTEM VIRTUAL PAGE/MAP REGISTER NUMBER
    LOGADR longword unsigned;                   /* MB -- LOGICAL NAME BLOCK ADDRESS
end SVPN_OVERLAY;
SVAPTE longword unsigned;                       /*SYSTEM VIRTUAL ADDRESS OF PTE
BOFF word unsigned;                             /*BYTE OFFSET IN FIRST PAGE
BCNT word unsigned;                             /*BYTE COUNT OF TRANSFER
ERTCNT byte unsigned;                           /*ERROR LOG DEVICE CURRENT ERROR RETRY COUNT
ERTMAX byte unsigned;                           /*ERROR LOG DEVICE MAXIMUM ERROR RETRY COUNT
ERRCNT word unsigned;                           /*DEVICE ERROR COUNT
PDT_OVERLAY union fill;
    PDT longword unsigned;                      /*ADDR OF PORT DESCRIPTOR TABLE
    JNL_MCSID longword unsigned;                /*MASTER NODE'S CSID (JOURNALING)
end PDT_OVERLAY;
DDT longword unsigned;                          /*ADDR OF DDT (OPTIONAL BUT PREFERRED)
MEDIA_ID_OVERLAY union fill ;
```

```
        MEDIA_ID longword unsigned;                     /*BIT ENCODED MEDIA IDENTIFICATION
        MEDIA_ID_SUBFIELDS structure fill;
            MEDIA_ID_NN bitfield length 7;              /* MEDIA NAME NUMBER
            MEDIA_ID_N2 bitfield length 5;              /* MEDIA NAME CHAR 2
            MEDIA_ID_N1 bitfield length 5;              /* MEDIA NAME CHAR 1
            MEDIA_ID_N0 bitfield length 5;              /* MEDIA NAME CHAR 0
            MEDIA_ID_T1 bitfield length 5;              /* MEDIA TYPE CHAR 1
            MEDIA_ID_T0 bitfield length 5;              /* MEDIA TYPE CHAR 0
        end MEDIA_ID_SUBFIELDS;
    end MEDIA_ID_OVERLAY;
    constant "LENGTH" equals . prefix UCB$ tag K;       /*LENGTH OF STANDARD UCB
    constant "LENGTH" equals . prefix UCB$ tag C;       /*LENGTH OF STANDARD UCB
    #UCB_LENGTH = .;              /* for $TTYUCBDEF
end UCBDEF;

/*
/* DEVICE DEPENDENT UCB EXTENSIONS
/*
/* MAILBOX
/*

aggregate UCBDEF3 structure prefix UCB$;
    FILL_7 byte dimension UCB$K_LENGTH fill prefix UCBDEF tag $$;
    MB_WAST longword unsigned;                          /*WRITE ATTENTION AST BLOCK ADDR
    MB_RAST longword unsigned;                          /*READ ATTENTION AST BLOCK ADDR
    MB_MBX longword unsigned;                           /*MAILBOX CONTROL BLOCK ADDR
    MB_SHB longword unsigned;                           /*SHARED MEM. CONTROL BLOCK ADDR
    MB_WIOQFL longword unsigned;                        /*WRITE I/O QUEUE FORWARD LINK
    MB_WIOQBL longword unsigned;                        /*WRITE I/O QUEUE BACKWARD LINK
    MB_PORT longword unsigned;                          /*SHARED MEM. PORT NUMBER
    constant MB_LENGTH equals . prefix UCB$ tag K;      /*SIZE OF MAILBOX UCB
    constant MB_LENGTH equals . prefix UCB$ tag C;      /*SIZE OF MAILBOX UCB
end UCBDEF3;

/*
/* ERROR LOG DEVICES (ALL)
/*

aggregate UCBDEF4 structure prefix UCB$;
    FILL_8 byte dimension UCB$K_LENGTH fill prefix UCBDEF tag $$;
    SLAVE byte unsigned;                                /*SLAVE CONTROLLER NUMBER
    SPR byte unsigned;                                  /*SPARE UNUSED BYTE
    FEX byte unsigned;                                  /*FUNCTION DISPATCH TABLE INDEX
    CEX byte unsigned;                                  /*CASE TABLE FUNCTION EXECUTION INDEX
    EMB longword unsigned;                              /*ADDRESS OF ERROR MESSAGE BUFFER
    FILL_1 word fill prefix UCBDEF tag $$;              /*SPARE UNUSED WORD
    FUNC word unsigned;                                 /*I/O FUNCTION MODIFIERS
    DPC longword unsigned;                              /*SAVED DRIVER SUBROUTINE RETURN ADDRESS
    constant ERL_LENGTH equals .;                       /*SIZE OF ERROR LOG UCB
    constant ERL_LENGTH equals . tag C;                 /*SIZE OF ERROR LOG UCB
end UCBDEF4;

/*
/* DUAL PORTED DEVICES (ALL DISKS AND MOST TAPES)
/*
```

```
aggregate DUALPATH_EXTENSION structure prefix UCB$;
    fill_dualpath byte dimension UCB$K_ERL_LENGTH fill;
    DUAL_PATH union fill;                                   {  Setup two versions of the dual-path names
        OLD_DUAL_PATH structure fill;                       {  This is the old way of naming things
            DP_DDB longword unsigned;                       /* Pointer to alternate DDB
            DP_LINK longword unsigned;                      /* Address of next UCB for this DDB
            DP_ALTUCB longword unsigned;                    /* Addr of alternate UCB for this unit
        end OLD_DUAL_PATH;
        PREFERED_DUAL_PATH structure fill;                  {  This is the prefered way of naming things
            "2P_DDB" longword unsigned;                     /* Pointer to alternate DDB
            "2P_LINK" longword unsigned;                    /* Address of next UCB for this DDB
            "2P_ALTUCB" longword unsigned;                  /* Addr of alternate UCB for this unit
        end PREFERED_DUAL_PATH;
    end DUAL_PATH;
    constant DP_LENGTH equals .;                            /* Size of dual path UCB
    constant DP_LENGTH equals . tag C;                      /* size of dual path UCB
    constant "2P_LENGTH" equals .;                          /* Size of dual path UCB
    constant "2P_LENGTH" equals . tag C;                    /* size of dual path UCB
    #2P_LENGTH  = .;
end DUALPATH_EXTENSION;

/*
/* ALL DISKS AND TAPES
/*

aggregate DISKTAPE_UCB_EXTENSION structure prefix UCB$;
    fill_disktape byte dimension #2P_LENGTH fill;
    DIRSEQ structure word unsigned;                         /* Directory sequence number
        filler bitfield length 15 fill;                     {   skip value portion of sequence number
        AST_ARMED bitfield mask;                            /*  Blocking AST armed flag
    end DIRSEQ;
    ONLCNT byte unsigned;                                   /* Online count
    reserved byte fill;                                     {  Reserved byte
    DISKTAPE_OVERLAY union fill;
        MAXBLOCK longword unsigned;                         /* Random access device highest block
        RECORD longword unsigned;                           /* Current tape position or frame counter
    end DISKTAPE_OVERLAY;
    constant LCL_TAPE_LENGTH equals .;                      /* Size of local tape UCB
    constant LCL_TAPE_LENGTH equals . tag C;                /* Size of local tape UCB
    MAXBCNT  longword unsigned;                             /* Maximum transfer BCNT
    DCCB longword unsigned;                                 /* Pointer to data cache control block
    #BEGIN_LOCAL_DISKS = .;
    #BEGIN_MSCP = .;
end DISKTAPE_UCB_EXTENSION;

aggregate LCL_DISK_UCB_EXTENSION structure prefix UCB$;
    fill_lcl_disk byte dimension #BEGIN_LOCAL_DISKS fill;
    MEDIA structure longword unsigned;                      /* Media address (longword)
        DA word unsigned;                                   /* Saved desired sector/track address register
        DC word unsigned;                                   /* Saved desired cylinder address register
    end MEDIA;
    BCR structure longword unsigned;                        /* Byte count register
        BCR word unsigned;
    end BCR;
    EC1 word unsigned;                                      /* ECC position register
    EC2 word unsigned;                                      /* ECC pattern register
```

```
    OFFSET word unsigned;                       /* Current offset register contents
    OFFNDX byte unsigned;                       /* Current offset table index
    OFFRTC byte unsigned;                       /* Current offset retry count
    constant LCL_DISK_LENGTH equals .;          /* Size of local disk UCB
    constant LCL_DISK_LENGTH equals . tag C;    /* Size of local disk UCB

/*
/* FLOPPY DEPENDENT BIT DEFINITIONS
/*
    DX_BUF longword unsigned;                   /*ADDRESS OF SECTOR BUFFER
    DX_BFPNT longword unsigned;                 /*CURRENT SECTOR BUFFER POINTER
    DX_RXDB longword unsigned;                  /*SAVED RECEIVER DATA BUFFER
    DX_BCR word unsigned;                       /*CURRENT FLOPPY BYTE COUNT
    DX_SCTCNT byte unsigned;                    /*CURRENT SECTOR BYTE COUNT
    FILL_2 byte fill prefix UCBDEF tag $$;      /*SPARE UNUSED BYTE

end LCL_DISK_UCB_EXTENSION;
/*
/* MSCP DISKS AND TAPES UCB EXTENSION
/*

aggregate MSCP_UCB_EXTENSION structure prefix UCB$;
    mscp_fill byte dimension #BEGIN_MSCP fill;
    CDDB longword unsigned;                     /* Pointer to active CDDB
    "2P_CDDB" longword unsigned;                /* Pointer to alternate CDDB
    CDDB_LINK longword unsigned;                /* Pointer to next UCB in CDDB chain
    CDT longword unsigned;                      /* Pointer to active CDT
    UNIT_ID quadword unsigned;                  /* Unique MSCP unit identifier
    MSCPUNIT word unsigned;                     /* Primary path MSCP unit number
    "2P_MSCPUNIT" word unsigned;                /* Secondary path MSCP unit number
    MSCPDEVPARAM longword unsigned;             /* MSCP device-dependent parameters
    WAIT_CDDB longword unsigned;                /* Address of the CDDB waiting for mnt. ver.
                                                {  to complete on this UCB
    UNIT_FLAGS word unsigned;                   /* MSCP unit flags
    reserved word unsigned fill;                {  reserved word, for alignment
    MSCP_RESV quadword unsigned;                /* Reserved for MSCP enhancements
    constant MSCP_DISK_LENGTH equals .;         /* Size of MSCP disk UCB
    constant MSCP_TAPE_LENGTH equals .;         /* Size of MSCP tape UCB

end MSCP_UCB_EXTENSION;

/*
/* NETWORK LOGICAL LINK (NETWORK MAILBOX) EXTENSION
/*

aggregate UCBDEF7 structure prefix UCB$;
    FILL_11 byte dimension UCB$K_LENGTH fill prefix UCBDEF tag $$;
    NT_DATSSB longword unsigned;                /* ADDRESS OF DATA SUBCHANNEL STATUS BLOCK
    NT_INTSSB longword unsigned;                /* ADDRESS OF INT/LS SSB
    NT_CHAN word unsigned;                      /* DDCMP CHANNEL NO.
    FILL_3_OVERLAY union fill;
        FILL_3 word fill prefix UCBDEF tag $$;  /* DUMMY FIELD
        FILL_3_BITS structure fill;
            LTYPE bitfield length 2;            /* LINK TYPE BITS
            SEGFLO bitfield;                    /* SEGMENT REQUEST COUNTS
            MSGFLO bitfield;                    /* MESSAGE REQUEST COUNTS
```

```
        MSGACK bitfield;                                /* MESSAGE ACK/NAK
        BACKP bitfield mask;                            /* BACKPRESSURE (1=> NO FLOW)
        LNKPRI bitfield length 2;                       /* LINK PRIORITY (IGNORED)
    end FILL_3_BITS;
                                                        /* NETWORK CONSTANTS
    constant LOGLNK equals 1  prefix UCB tag $C;        /* CONNECT IS FOR LOGICAL LINK (NOT SINGLE MSG)
end FILL_3_OVERLAY;
end UCBDEF7;

/*
/* JOURNAL DEVICE EXTENSION
/*

aggregate UCBDEF8 structure prefix UCB$;
    FILL_12 byte dimension UCB$K_ERL_LENGTH fill prefix UCBDEF tag $$;
    JNL_BCB_OVERLAY union fill;
        JNL_BCB longword unsigned;                      /* ADDRESS OF BUFFER CONTROL BLOCK
        JNL_ADL longword unsigned;                      /* ALLOCATED JOURNAL DEVICE LIST
    end JNL_BCB_OVERLAY;
    JNL_RUL longword unsigned;                          /* ADDRESS OF RUL (RU JOURNALS ONLY)
    JNL_WQFL longword unsigned;                         /* WAIT QUEUE FORWARD LINK
    JNL_WQBL longword unsigned;                         /* WAIT QUEUE BACKWARD LINK
    JNL_FQFL longword unsigned;                         /* FORCE QUEUE FORWARD LINK
    JNL_FQBL longword unsigned;                         /* FORCE QUEUE BACKWARD LINK
    JNL_NAM character;                                  /* JOURNAL NAME LENGTH
    JNL_NAM byte unsigned dimension 18;                 /* JOURNAL NAME
    FILL_4 byte fill prefix UCBDEF tag $$;              /* SPARE
    JNL_QUOT longword unsigned;                         /* QUOTA FOR RU JOURNALS
    JNL_ID word unsigned;                               /* JOURNAL ID (TAPES ONLY)
    JNL_MUNIT word unsigned;                            /* MASTER'S DEVICE UNIT NUMBER
    JNL_MASK longword unsigned;                         /* JOURNAL MASK
    JNL_ASID_OVERLAY union fill;
        JNL_ASID longword unsigned;                     /* ASSIGN ID
        JNL_NDL longword unsigned;                      /* POINTER TO NAME TABLE DEVICE LIST
    end JNL_ASID_OVERLAY;
    JNL_REFC longword unsigned;                         /* LOCAL REFERENCE COUNT
    JNL_TREFC longword unsigned;                        /* TOTAL REFERENCE COUNT
    JNL_MXENT word unsigned;                            /* MAXIMUM ENTRY SIZE
    JNL_PROT word unsigned;                             /* PROTECTION MASK
    JNL_WRCNT longword unsigned;                        /* WRITE COUNT
    JNL_BWCNT longword unsigned;                        /* BUFFER WRITE COUNT
    JNL_EXCNT longword unsigned;                        /* JOURNAL EXTEND COUNT
    JNL_FAILQFL longword unsigned;                      /* FAIL OVER WAIT Q FORWARD LINK
    JNL_FAILQBL longword unsigned;                      /* FAIL OVER WAIT Q BACKWARD LINK
    JNLSEQ_OVLY union fill;
        JNL_LSEQNO      longword unsigned;              /* LOCAL SEQUENCE NUMBER
        JNL_BTXSEQNO    longword unsigned;              /* BLOCK TRANSFER SEQ NUM (PROTO UCB ONLY)
    end JNLSEQ_OVLY;
    JNL_ACBM    longword unsigned;                      /* ADDRESS OF ACCESS BIT MAP
    JNL_RMBLK   longword unsigned;                      /* ADDRESS OF REMASTER BLOCK
    JNL_CWQFL   longword unsigned;                      /* Cluster write Q forward link
    JNL_CWQBL   longword unsigned;                      /* Cluster write Q backward link
    JNL_WCBFL   longword unsigned;                      /* WCB LISTHEAD FORWARD LINK
    JNL_WCBBL   longword unsigned;                      /* WCB LISTHEAD BACKWARD LINK
    FILL_5 longword fill prefix UCBDEF tag $$;          /* SPARE
    constant JNL_LENGTH equals . prefix UCB$ tag C;     /* JOURNAL UCB LENGTH
```

```
end UCBDEF8;

/*
/* NI DEVICE EXTENSION
/*

aggregate UCBDEF9 structure prefix UCB$;
    FILL_13 byte dimension UCB$K_LENGTH fill prefix UCBDEF tag $$;
    NI_HWAPTR longword unsigned;                    /*ADDRESS OF NI DEVICE HARDWARE ADDRESS
    NI_MLTPTR longword unsigned;                    /*ADDRESS OF PROTOCOL MULTICAST TABLE
    constant NI_LENGTH equals . prefix UCB$ tag K;  /*SIZE OF NI DEVICE UCB
    constant NI_LENGTH equals . prefix UCB$ tag C;  /*SIZE OF NI DEVICE UCB
end UCBDEF9;

end_module $UCBDEF;
```

```
MODULE $TTYUCBDEF;
/*
/*  $TTYUCBDEF follows here only because there is no way to get the
/*  UCB$K_LENGTH symbol into another module. TTYUCBDEF was formerly
/*   included in TTYDEF.MAR.
/*
/* TERMINAL DRIVER DEFINITIONS
/*
/* These definitions define the device dependent extensions of the UCB.
/* Certain portions of the ucb are assumed to be contiguous and must not
/* be split. These areas are documented in the following definitions.
/*

aggregate TTYUCBDEF structure prefix UCB$;

        TT_UCBFILL byte dimension #UCB_LENGTH fill prefix UCBDEF tag $$;
/*
/* Logical terminal UCB extension
/*

        TL_CTRLY        longword unsigned;      /* CONTROL Y AST BLOCK LIST HEAD
        TL_CTRLC        longword unsigned;      /* CONTROL C AST BLOCK LIST HEAD
        TL_OUTBAND      longword unsigned;      /* OUT OF BAND CHARACTER MASK
        TL_BANDQUE      longword unsigned;      /* OUT OF BAND AST QUEUE
        TL_PHYUCB       longword unsigned;      /* THE PHYSICAL UCB ADDRESS
        TL_CTLPID       longword unsigned;      /* CONTROLING PID (USED WITH SPAWN)
        TL_BRKTHRU      quadword unsigned;      /* FACILITY BROADCAST BITMASK

        constant TL_LENGTH equals . tag C;
        constant TL_LENGTH equals . tag K;
/*
/* Terminal class driver dependant region
/* Split here between local and remote terminal UCB's
/*
        TTYRTTUCB union;        /* local/remote union (overlay)

        TTYUCB structure;       /* this structure defines remainder of local ucb

/* READ TIMEOUT CONTROL

        TT_RDUE         longword unsigned;      /* ABSTIME WHEN READ TIMEOUT DUE
        TT_RTIMOU       longword unsigned;      /* ADDRESS OF READ TIMEOUT ROUTINE

/* TERMINAL DRIVER STATE TABLE

        TT_STATE_OVERLAY union fill;
        TT_STATE        quadword unsigned;      /* CURRENT UNIT STATE VECTOR
        TT_STATE_Q_BLOCK structure fill;
          TT_STATET_OVERLAY union fill;
            TT_STATE1   longword unsigned;
            TT_STATE1_FIELDS structure fill prefix TTV$;
                ST_POWER        bitfield mask;  /*
                ST_CTRLS        bitfield mask;  /*
                ST_FILL         bitfield mask;  /*
                ST_CURSOR       bitfield mask;  /*
                ST_SENDLF       bitfield mask;  /*
```

```
            ST_BACKSPACE    bitfield mask;  /*
            ST_MULTI        bitfield mask;  /*
            ST_WRITE        bitfield mask;
            ST_EOL          bitfield mask;  /*
            ST_EDITREAD     bitfield mask;  /*
            ST_RDVERIFY     bitfield mask;  /*
            ST_RECALL       bitfield mask;  /*
            ST_READ         bitfield mask;  /*
          end TT_STATE1_FIELDS;
        end TT_STATE1_OVERLAY;

        TT_STATE2_OVERLAY union fill;
          TT_STATE2 longword unsigned;
          TT_STATE2_FIELDS structure fill prefix TTY$;
            ST_CTRLO        bitfield mask;  /*
            ST_DEL          bitfield mask;  /*
            ST_PASALL       bitfield mask;  /*
            ST_NOECHO       bitfield mask;  /*
            ST_WRTALL       bitfield mask;  /*
            ST_PROMPT       bitfield mask;  /*
            ST_NOFLTR       bitfield mask;  /*
            ST_ESC          bitfield mask;  /*
            ST_BADESC       bitfield mask;  /*
            ST_NL           bitfield mask;  /*
            ST_REFRSH       bitfield mask;  /*
            ST_ESCAPE       bitfield mask;  /*
            ST_TYPFUL       bitfield mask;  /*
            ST_SKIPLF       bitfield mask;  /*
            ST_ESC_O        bitfield mask;  /*
            ST_WRAP         bitfield mask;  /*
            ST_OVRFLO       bitfield mask;  /*
            ST_AUTOP        bitfield mask;  /*
            ST_CTRLR        bitfield mask;  /*
            ST_SKIPCRLF     bitfield mask;  /*
            ST_EDITING      bitfield mask;  /*
            ST_TABEXPAND    bitfield mask;  /*
            ST_QUOTING      bitfield mask;  /*
            ST_OVERSTRIKE   bitfield mask;  /*
            ST_TERMNORM     bitfield mask;  /*
            ST_ECHAES       bitfield mask;  /*
            ST_PRE          bitfield mask;  /*
            ST_NINTMULTI    bitfield mask;  /*
            ST_RECONNECT    bitfield mask;  /*
            ST_CTSLOW       bitfield mask;  /*
            ST_TABRIGHT     bitfield mask;  /*
          end TT_STATE2_FIELDS;
        end TT_STATE2_OVERLAY;
      end TT_STATE_Q_BLOCK;
      end TT_STATE_OVERLAY;

      TT_LOGUCB       longword unsigned;      /* ADDRESS OF THE LOGICAL UCB

/* DEFAULT CHARACTERISTICS

      TT_DECHAR       longword unsigned;      /* DEFAULT DEVICE CHARACTERISTICS
      TT_DECHA1       longword unsigned;      /* DEFAULT DEVICE CHAR EXTENSIONS
```

```
/* WRITE QUEUE POINTERS

        TT_WFLINK       longword unsigned;      /* Write queue forward link.
        TT_WBLINK       longword unsigned;      /* Write queue backward link.
        TT_WRTBUF       longword unsigned;      /* Current write buffer block.
/* ADDRESS AND LENGTH OF MULTI-ECHO STRING

        TT_MULTI        longword unsigned;      /* CURRENT MULTIECHO BUFFER ADDRESS
        TT_MULTILEN     word unsigned;          /* LENGTH OF STRING TO OUTPUT
        TT_SMLTLEN      word unsigned;          /* SAVED MULTI LENGTH
        TT_SMLT         longword unsigned;      /* AND THE SAVED ADDRESS
/* Typeahead buffer address

        TT_TYPAHD       longword unsigned;       /* TYPEAHEAD BUFFER ADDRESS

/*-- ************************************************************************

/* DEFAULT SPEED, FILL ,PARITY (MUST BE CONTIGUOUS)

/*++ ************************************************************************

        TT_DESPEE       word unsigned;          /* DEFAULT SPEED
        TT_DECRF        byte unsigned;          /* DEFAULT CR FILL
        TT_DELFF        byte unsigned;          /* DEFAULT LF FILL

        TT_DEPARI       byte unsigned;          /* DEFAULT PARITY/CHAR SIZE
        TT_DEFSPE_SPARE1 byte unsigned;
        TT_DEFSPE_SPARE2 word unsigned;
/*-- ************************************************************************
/*
/* DEFAULT TERMINAL TYPE AND SIZE (MUST BE CONTIGUOUS)
/*
/*++  ************************************************************************

        TT_DETYPE       byte unsigned;          /* DEFAULT TERMINAL TYPE
        TT_DESIZE       word unsigned;          /* DEFAULT LINE SIZE
        TT_SPARE1       byte unsigned;          /* SPARE BYTE MUST FOLLOW
/*-- ************************************************************************

/* SPEED, FILL, PARITY (MUST BE CONTIGUOUS)

/*++ ************************************************************************

        TT_SPEED_OVERLAY union fill;
            TT_SPEED    word unsigned;          /* SPEED CODES (SPLIT SPEED)
            TT_SPEED_FIELDS structure fill;
                TT_TSPEED byte unsigned;        /* TRANSMIT SPEED
                TT_RSPEED byte unsigned;        /* RECEIVE SPEED
            end TT_SPEED_FIELDS;
        end TT_SPEED_OVERLAY;

        TT_CRFILL       byte unsigned;          /* NUMBER FILLS TO OUTPUT ON CR
        TT_LFFILL       byte unsigned;          /* NUMBER FILLS TO OUTPUT ON LF

        PARITY_OVERLAY union fill;
```

```
            TT_PARITY   byte unsigned;          /* PARITY AND CHARACTER SIZE DEFINITIONS
            TT_PARITY_BITS structure fill;
                TT_XXPARITY     bitfield mask;  /* UNUSED ??
                TT_DISPARERR    bitfield mask;  /* SPECIFY DISREGARD PARITY ERRORS
                TT_USERFRAME    bitfield mask;  /* SPECIFY USER FRAME SETUP
                TT_LEN bitfield mask length 2;  /* CHARACTER LENGTH
                TT_STOP         bitfield mask;  /* STOP BITS
                TT_PARTY        bitfield mask;  /* PARITY ENABLED
                TT_ODD          bitfield mask;  /* ODD PARITY
            end TT_PARITY_BITS;
        end PARITY_OVERLAY;
        TT_PAR_SPARE1   byte unsigned;
        TT_PAR_SPARE2   word unsigned;
/*-- ***************************************************************************

/* CURRENT CURSOR AND LINE POSITION FOR FORMATTED OPERATIONS

        TT_CURSOR       word unsigned;          /* CURRENT CURSOR POSITION
        TT_LINE         byte unsigned;          /* CURRENT LINE ON PAGE
        TT_LASTC        byte unsigned;          /* LAST FORMATTED OUTPUT CHARACTER

/* Number of back spaces to output for non-ansi terminals

        TT_BSPLEN       word unsigned;          /* NUMBER OF BACKSPACES

/* FILL HANDLING

        TT_FILL         byte unsigned;          /* CURRENT FILL COUNT

/* ESCAPE SYNTAX RULE STATE.

        TT_ESC          byte unsigned;          /* CURRENT  READ ESCAPE SYNTAX STATE
        TT_ESC_O        byte unsigned;          /* OUPUT ESCAPE STATE

/* Count of characters in interrupt string

        TT_INTCNT       byte unsigned;

/* Bit used for modem control

        TT_UNITBIT      word unsigned;          /* BIT USED TO ENABLE AND DISABLE MODEM CONTROL.

/* PORT SPECIFIC OUTPUT CONTROL

        TT_HOLD_OVERLAY union fill;
            TT_HOLD word unsigned;      /* UNIT HOLDING TANK AND PORT DISPATCH
            TT_HOLD_BITS structure fill prefix TTY$;
                TANK_CHAR       byte unsigned;  /* CHARACTER
                TANK_PREMPT     bitfield mask;  /* SEND PREMPT CHARACTER
                TANK_STOP       bitfield mask;  /* STOP OUTPUT
                TANK_HOLD       bitfield mask;  /* CHAR IN TANK
                TANK_BURST      bitfield mask;  /* BURST ACTIVE
                TANK_DMA        bitfield mask;  /* DMA ACTIVE **** SHOULD MOVE BEFORE BURST ****
            end TT_HOLD_BITS;
        end TT_HOLD_OVERLAY;
```

```
        TT_PREMPT       byte unsigned;          /* THE BYTE USED TO PREMPT INPUT
        TT_OUTYPE       byte unsigned;          /* TYPE OF OUTPUT THAT THIS CALL

/* CLASS & PORT VECTOR POINTERS

        TT_GETNXT       longword unsigned;      /* ADDRESS OF CLASS INPUT ROUTINE
        TT_PUTNXT       longword unsigned;      /* ADDRESS OF CLASS OUTPUT ROUTINE
        TT_CLASS        longword unsigned;      /* ADDRESS OF CLASS VECTOR
        TT_PORT         longword unsigned;      /* ADDRESS OF PORT VECTOR

        TT_OUTADR       longword unsigned;      /* ADDRESS OF OUTPUT  CURRENT STREAM
        TT_OUTLEN       word unsigned;          /* LENGTH OF OUTPUT STREAM

        TT_PRTCTL_OVERLAY union fill;
            TT_PRTCTL   word unsigned;          /* THE PORT DRIVER CONTROL WORD
            TT_PRTCTL_BITS structure fill prefix TTY$;
                PC_NOTIME bitfield mask;        /* IF SET NO TIMEOUT WILL BE CALCULATED
                PC_DMAENA bitfield mask;        /* DMA CURRENTLY ENABLED
                PC_DMAAVL bitfield mask;        /* DMA SUPPORTED ON THIS PORT
                PC_PRMMAP bitfield mask;        /* UNIT CAN HAVE PERMANENT MAP REGISTERS
                PC_MAPAVL bitfield mask;        /* MAP REGISTER CURRENTLY ALLOCATED
                PC_XOFAVL bitfield mask;        /* AUTO XOFF SUPPORTED ON THIS PORT
                PC_XOFENA bitfield mask;        /* AUTO XOFF CURRENTLY ENABLED
                PC_NOCRLF bitfield mask;        /* don't do free linefeed after creturn
            end TT_PRTCTL_BITS;
        end TT_PRTCTL_OVERLAY;

/* MODEM CONTROL DEFINITIONS

        TT_DS_RCV       byte unsigned;          /* CURRENT RECEIVE MODEM
        TT_DS_TX        byte unsigned;          /* CURRENT TRANSMIT MODEM
        TT_DS_ST        word unsigned;          /* CURRENT MODEM STATE
        TT_DS_TIM       word unsigned;          /* CURRENT MODEM TIMEOUT

        TT_MAINT_OVERLAY union fill;
            TT_MAINT    byte unsigned;          /* MAINTENANCE PARAMETERS
            TT_MAINT_BITS structure fill;
                TT_MAINT_FILL bitfield length 7;
                TT_DSBL bitfield mask;          /* LINE DISABLED
            end TT_MAINT_BITS;
        end TT_MAINT_OVERLAY;

        TT_OLDCPZORG    byte unsigned;          /* spare byte make this longword alligned

        constant TT_CLSLEN equals . tag C;
        constant TT_CLSLEN equals . tag K;

/**********************************************************************
/*
/* Terminal Port driver dependant extension region

        TP_MAP  longword unsigned;              /* UNIBUS MAP REGISTERS
        TP_STAT_OVERLAY union fill;
            TP_STAT     byte unsigned;          /* DMA PORT SPECIFIC STATUS
            TP_STAT_BITS structure fill prefix TTY$; /* BITS DEFINED IN THE DMA STATUS WORD
                TP_ABORT bitfield mask;         /* DMA ABORT REQUESTED ON THIS LINE
```

```
            TP_ALLOC bitfield mask;         /* ALLOC MAP FORK IN PROGRESS
            TP_DLLOC bitfield mask;         /* DEALLOCATE MAP FORK IN PROGRESS
        end TP_STAT_BITS;
    end TP_STAT_OVERLAY;

    TP_SPARE1       byte unsigned;
    TP_SPARE2       word unsigned;

    constant TP_LENGTH equals . tag C;
    constant TP_LENGTH equals . tag K;
    constant TT_LENGTH equals . tag C;
    constant TT_LENGTH equals . tag K;

    TT_STATE_SX structure prefix TTY$;
            SX_POWER        bitfield;       /*
            SX_CTRLS        bitfield;       /*
            SX_FILL         bitfield;       /*
            SX_CURSOR       bitfield;       /*
            SX_SENDLF       bitfield;       /*
            SX_BACKSPACE    bitfield;       /* OUTPUT BACKSPACES FOR SEVERAL LOOPS
            SX_MULTI        bitfield;       /*
            SX_WRITE        bitfield;       /* Write state
            SX_EOL          bitfield;       /*
            SX_EDITREAD     bitfield;       /*
            SX_RDVERIFY     bitfield;       /*
            SX_RECALL       bitfield;       /*
            SX_READ         bitfield;       /*
            SX_FILLBITS     bitfield length 32-^;   /* END OF FIRST LONGWORD

            SX_CTRLO        bitfield;       /*
            SX_DEL          bitfield;       /*
            SX_PASALL       bitfield;       /*
            SX_NOECHO       bitfield;       /*
            SX_WRTALL       bitfield;       /*
            SX_PROMPT       bitfield;       /*
            SX_NOFLTR       bitfield;       /*
            SX_ESC          bitfield;       /*
            SX_BADESC       bitfield;       /*
            SX_NL           bitfield;       /* New line must directly precede
            SX_REFRSH       bitfield;       /* refresh, or all breaks.
            SX_ESCAPE       bitfield;       /*
            SX_TYPFUL       bitfield;       /*
            SX_SKIPLF       bitfield;       /*
            SX_ESC_O        bitfield;       /*
            SX_WRAP         bitfield;       /*
            SX_OVRFLO       bitfield;       /*
            SX_AUTOP        bitfield;       /*
            SX_CTRLR        bitfield;       /*
            SX_SKIPCRLF     bitfield;       /*
            SX_EDITING      bitfield;       /*
            SX_TABEXPAND    bitfield;       /*
            SX_QUOTING      bitfield;       /*
            SX_OVERSTRIKE   bitfield;       /*
            SX_TERMNORM     bitfield;       /*
            SX_ECHAES       bitfield;       /*
            SX_PRE          bitfield;       /*
```

```
            SX_NINTMULTI    bitfield;       /*
            SX_RECONNECT    bitfield;       /*
            SX_CTSLOW       bitfield;       /*
            SX_TABRIGHT     bitfield;       /*
        end TT_STATE_SX;
    end TTYUCB;

/* remote terminal extension

    RTTUCB structure;

        RTT_NETUCB          longword unsigned;      /* NET DEVICE UCB
        RTT_NETWIND         longword unsigned;      /* NET DEVICE WCB
        RTT_IRPFL           longword unsigned;      /* IRP QUEUE
        RTT_IRPBL           longword unsigned;      /* IRP QUEUE
        RTT_NETIRP          longword unsigned;      /* READ NET IIRP
        RTT_BANDINCL        longword unsigned;      /* OUT OF BAND INCLUDES
        RTT_BANDINMSK       longword unsigned;      /* OUT OF BAND INCLUDE MASK
        RTT_BANDEXCL        longword unsigned;      /* out of band exclude mask
        RTT_BANDEXMSK       longword unsigned;      /* out of band exclude

        RTT_PROVRS          byte unsigned;          /* PROTOCOL VERSION
        RTT_PROECO          byte unsigned;          /* PROTOCOL ECO
        RTT_LINK            word unsigned;          /* LINK NUMBER (for LOGINOUT)

        RTT_OBJ             byte unsigned;          /* OBJECT NUMBER CONNECTED
        RTT_SYSTYPE         word unsigned;          /* SYSTEM TYPE (VMS=7)
        RTT_FILLBYTE        byte unsigned;          /* fill - use when convenient

/*      CTERM driver only

        CT_FLAGS_OVERLAY union fill;
            CT_FLAGS        word unsigned;          /* MISC FLAGS
            CT_FLAGS_BITS structure fill prefix "FLG$";
                WIIRP_BSY   bitfield mask;          /* WIIRP BUSY
                CTRLO       bitfield mask;          /* CTRLO IN PROGRESS
                CANCTRLO    bitfield mask;          /* CANCEL CTRLO ON WRITE
                INWRTFDT    bitfield mask;          /* IN WRITE FDT
                QUOTA       bitfield mask;          /* QUOTA CHARGED
                VAXTOVAX    bitfield mask;          /* VAX TO VAX
                BUFFER      bitfield mask;          /* DO BUFFERED WRITES
            end CT_FLAGS_BITS;
        end CT_FLAGS_OVERLAY;

        CT_QCTPCNT          word unsigned;          /* QUEUED CTP COUNT
        CT_WIIRP            longword unsigned;      /* WRITE IIRP
        CT_TQE              longword unsigned;      /* TQE ADDRESS
        CT_NETQFL           longword unsigned;      /* WAITING FOR WRITE
        CT_NETQBL           longword unsigned;      /*   TO NET QUEUE
        CT_STALLQFL         longword unsigned;      /* IRPs BEING HELD
        CT_STALLQBL         longword unsigned;      /*   QUEUE
        CT_WRTCTP           longword unsigned;      /* BUFFERED WRITE CTP
        CT_WRTCUR           longword unsigned;      /* CURRENT FILL POINTER
        CT_WRTSIZ           word unsigned;          /* REMAINING SIZE
        CT_WRTCNT           word unsigned;          /* COUNT SINCE LAST TQE
        CT_MAXMSG           word unsigned;          /* MAX WRITE TO NET SIZE
```

```
        CT_MAXREAD          word unsigned;          /* MAX READ IN SERVER
        CT_LEGALMSG         longword unsigned;      /* LEGAL MESSAGE MASK
        CT_VERSION          byte unsigned;          /* CTERM VERSION
        CT_ECO              byte unsigned;          /* CTERM ECO
        CT_FILLWORD         word unsigned;          /* fill

        CT_DEBUG_FILL       CHARACTER LENGTH 4*10;  /* 10 LONGWORD FOR DEBUG

    constant RTT_LENGTH equals . tag C;             /* Length must be same for both RTTDRIVER
    constant RTT_LENGTH equals . tag K;             /* and CTDRIVER.

    end RTTUCB;

end TTYRTTUCB;      /* end union

end TTYUCBDEF;

end_module $TTYUCBDEF;
```

```
module $VADEF;

/*+
/* VIRTUAL ADDRESS VIELDS
/*-


aggregate VADEF  union prefix VA$;
    VADEF_BITS0 structure fill;
        "BYTE" bitfield mask length 9;                  /*BYTE VIELD
        VPN bitfield mask length 21;                    /*VIRTUAL PAGE NUMBER
        P1 bitfield mask;                               /*P1 SPACE
        SYSTEM bitfield mask;                           /*SYSTEM SPACE
    end VADEF_BITS0;
    VADEF_BITS1 structure fill;
        FILL_1 bitfield length 9 fill prefix VADEF tag $$;
        VPG bitfield mask length 23;                    /*VIRTUAL PAGE INCLUDING P1 & S
    end VADEF_BITS1;
end VADEF;

end_module $VADEF;
```

```
module $VCADEF;
/*+
/*
/* VCA - Volume Cache Block. This block contains the specialized caches for
/* a disk volume; to wit, the file ID cache, the extent cache, and the quota
/* file cache. The file ID cache and extent cache are together in one block;
/* the quota cache is located separately in another block. Both are pointed to
/* by the VCB.
/*
/*-



aggregate VCADEF structure prefix VCA$;
    FIDCACHE longword unsigned;                 /* pointer to file ID cache
    EXTCACHE longword unsigned;                 /* pointer to extent cache
    SIZE word unsigned;                         /* block size
    TYPE byte unsigned;                         /* block type code
    FLAGS structure byte;                       /* cache flags
        FIDC_VALID bitfield mask;               /* FID cache valid
        EXTC_VALID bitfield mask;               /* Extent cache valid
        FIDC_FLUSH bitfield mask;               /* FID cache to be flushed
        EXTC_FLUSH bitfield mask;               /* Extent cache to be flushed
    end FLAGS;
    constant "LENGTH" equals . tag K;           /* length of block header
    constant "LENGTH" equals . tag C;           /* length of block header
/*
/* The file ID cache consists of the cache header, followed by a longword
/* vector of file numbers, densely packed.
/*
end VCADEF;

aggregate VCADEF1 structure prefix VCA$;
    FIDSIZE word unsigned;                      /* number of entries allocated
    FIDCOUNT word unsigned;                     /* number of entries present
    FIDCLKID longword unsigned;                 /* FID cache lock id.
    FIDCACB byte unsigned dimension 28;         /* FID cache blocking ACB
    FIDLIST longword unsigned;                  /* first entry in list
/*
/* The extent cache consists of the cache header, followed by a quadword
/* vector of extents, densely packed. Each quadword contains block count
/* and starting LBN.
/*
end VCADEF1;

aggregate VCADEF2 structure prefix VCA$;
    EXTSIZE word unsigned;                      /* number of entries allocated
    EXTCOUNT word unsigned;                     /* number of entries present
    EXTTOTAL longword unsigned;                 /* total number of blocks contained in cache
    EXTLIMIT word unsigned;                     /* limit of volume to be cached, in percent/10
    FILL_2 word fill tag $$;                    /* spare
    EXTCLKID longword unsigned;                 /* EXT cache lock id.
    EXTCACB byte unsigned dimension 28;         /* Extent cache blocking ACB.
    EXTLIST quadword unsigned;                  /* first entry in list
end VCADEF2;
```

```
aggregate VCADEF3 structure prefix VCA$;
    EXTBLOCKS longword unsigned;                    /* number of blocks
    EXTLBN longword unsigned;                       /* starting LBN
/*
/* The quota cache consists of the cache header, followed by the cache
/* entries. Each cache entry is a block as defined below.
/*
end VCADEF3;

aggregate VCADEF4 structure prefix VCA$;
    QUOSIZE word unsigned;                          /* number of entries allocated
    QUOLRU word unsigned;                           /* current LRU counter
    QUOCLKID longword unsigned;                     /* whole cache lock ID
    FILL_3 byte dimension 3 fill tag $$;            /* 2nd longword & block size & type
    QUOCFLAGS structure byte;                       /* cache flags
        CACHEVALID bitfield mask;                   /* cache is valid
        CACHEFLUSH bitfield mask;                   /* cache is to be flushed
    end QUOCACHEFLAGS;
    QUOACB byte unsigned dimension 28;              /* ACB to deliver blocking AST
    QUOFLUSHACB byte unsigned dimension 28;         /* ACB to deliver cache flush AST
    QUOLIST longword unsigned;                      /* start of entries

end VCADEF4;

aggregate VCADEF5 structure prefix VCA$;
    QUOLOCK structure;                              /* lock status block
        QUOSTATUS_OVERLAY union fill;
            QUOSTATUS word unsigned;                /* $ENQ status
            QUOINDEX word unsigned;                 /* index in cache of this entry
        end QUOSTATUS_OVERLAY;
        QUOLRUX word unsigned;                      /* LRU index for entry
        QUOLKID longword unsigned;                  /* lock ID of cache entry
        QUORECNUM byte unsigned dimension 3 tag L; /* record number
        QUOFLAGS structure byte unsigned;           /* flags byte
            QUOVALID bitfield mask;                 /* valid entry is present
            QUODIRTY bitfield mask;                 /* dirty flag
        end QUOFLAGS;
        USAGE longword unsigned;                    /* current usage
        PERMQUOTA longword unsigned;                /* permanent quota
        OVERDRAFT longword unsigned;                /* overdraft limit
    end QUOLOCK;
    QUOUIC longword unsigned;                       /* UIC
    constant QUOLENGTH equals . tag K;              /* length of quota cache entry
    constant QUOLENGTH equals . tag C;              /* length of quota cache entry
end VCADEF5;

end_module $VCADEF;
```

```
{+
{ VCB - VOLUME CONTROL BLOCK
{
{ THERE IS ONE VOLUME CONTROL BLOCK FOR EACH MOUNTED DEVICE UNIT IN A
{ SYSTEM. IT CONTAINS INFORMATION NECESSARY TO CONTROL ACCESS TO AND
{ VERIFY CERTAIN VOLUME PARAMETERS IN THE CASE A DEVICE UNIT SHOULD
{ ERRONEOUSLY GO OFFLINE.
{-

module $VCBDEF;

aggregate VCBDEF_COMMON structure prefix VCB$;          {  COMMON VCB DEFINITIONS
    FORWARD_LINK union fill;
        FCBFL longword unsigned;                        /* FCB listhead forward link
        BLOCKFL longword unsigned;                      /*  or - Blocked request listhead forward link
        MEMQFL longword unsigned;                       /*  or - Shadow set members queue forward link
    end FORWARD_LINK;
    BACKWARD_LINK union fill;
        FCBBL longword unsigned;                        /* FCB listhead backward link
        BLOCKBL longword unsigned;                      /*  or - Blocked request listhead backward link
        MEMQBL longword unsigned;                       /*  or - Shadow set members queue backward link
    end BACKWARD_LINK;
    SIZE word unsigned;                                 /* Size of VCB in bytes
    TYPE byte unsigned;                                 /* structure type of VCB

    constant MRKLEN equals .;                           /* Mark length
    constant MRKLEN equals . tag C;                     /* Mark length
    #VCBMARK2 = .;                                      /* Second mark point

    VOLSTS union fill;
        STATUS byte unsigned;                           /* Volume status:
        DISK_BITS structure fill;                       {   for disks:
            WRITE_IF bitfield mask;                     /*   Index file is write accessed
            WRITE_SM bitfield mask;                     /*   Storage map is write accessed
            HOMBLKBAD bitfield mask;                    /*   Primary home block is bad
            IDXHDRBAD bitfield mask;                    /*   Primary index file header is bad
            NOALLOC bitfield mask;                      /*   Allocation/deallocation inhibited (bad bitmaps)
            EXTFID bitfield mask;                       /*   Volume has 24 bit file numbers
            GROUP bitfield mask;                        /*   Volume is mounted /group
            SYSTEM bitfield mask;                       /*   Volume is mounted /system
        end DISK_BITS;
        TAPE_BITS structure fill;                       {   for tapes:
            PARTFILE bitfield mask;                     /*   Partial file exists on tape
            LOGICEOVS bitfield mask;                    /*   Positioned at logical end of volume set
            WAIMOUVOL bitfield mask;                    /*   Wait for volume mount
            WAIREWIND bitfield mask;                    /*   Wait for rewind completion
            WAIUSRLBL bitfield mask;                    /*   Wait for user label
            CANCELIO bitfield mask;                     /*   Cancel I/O
            MUSTCLOSE bitfield mask;                    /*   Must close file
            NOWRITE bitfield mask;                      /*   Don't write trailers
        end TAPE_BITS;
        SHADOW_BITS structure fill;                     {   for shadow set members
            SHADMAST bitfield mask;                     /*   This VCB is for shadow set master
            NEWSSMEMB bitfield mask;                    /*   New shadow set member
            FAILED bitfield mask;                       /*   Member failed out of shadow set
        end SHADOW_BITS;
```

```
    end VOLSTS;
    TRANS word unsigned;                    /* VOLUME TRANSACTION COUNT
    RVN word unsigned;                      /* RELATIVE VOLUME NUMBER
    AQB longword unsigned;                  /* ADDRESS OF AQB
    VOLNAME character length 12;            /* VOLUME LABEL BLANK FILLED
    RVT longword unsigned;                  /* ADDRESS OF UCB OR RELATIVE VOLUME TABLE
    #VCBMARK3 = . ;                         /* THIRD MARK POINT
    constant COMLEN equals . prefix VCB$ tag K;     /* LENGTH OF COMMON AREA
end VCBDEF_COMMON;

aggregate VCBDEF_DISKS structure prefix VCB$;
    filldisks byte dimension #VCBMARK3 fill;
    constant COMLEN equals . prefix VCB$ tag C;     /* LENGTH OF COMMON AREA
    HOMELBN longword unsigned;              /* LBN OF VOLUME HOME BLOCK
    HOME2LBN longword unsigned;             /* LBN OF ALTERNATE VOLUME HOME BLOCK
    IXHDR2LBN longword unsigned;            /* LBN OF ALTERNATE INDEX FILE HEADER
    IBMAPLBN longword unsigned;             /* LBN OF INDEX FILE BITMAP
    SBMAPLBN longword unsigned;             /* LBN OF STORAGE BITMAP
    IBMAPSIZE byte unsigned;                /* SIZE OF INDEX FILE BITMAP
    SBMAPSIZE byte unsigned;                /* SIZE OF STORAGE BITMAP
    IBMAPVBN byte unsigned;                 /* CURRENT VBN IN INDEX FILE BIT MAP
    SBMAPVBN byte unsigned;                 /* CURRENT VBN IN STORAGE MAP
    CLUSTER word unsigned;                  /* VOLUME CLUSTER SIZE
    EXTEND word unsigned;                   /* VOLUME DEFAULT FILE EXTENSION LENGTH
    FREE longword unsigned;                 /* NUMBER OF FREE BLOCKS ON VOLUME
    MAXFILES longword unsigned;             /* MAXIMUM NUMBER OF FILES ALLOWED ON VOLUME
    WINDOW byte unsigned;                   /* VOLUME DEFAULT WINDOW SIZE
    LRU_LIM byte;                           /* VOLUME DIRECTORY LRU SIZE LIMIT
    FILEPROT word unsigned;                 /* VOLUME DEFAULT FILE PROTECTION
    MCOUNT word unsigned;                   /* MOUNT COUNT
    EOFDELTA byte unsigned;                 /* INDEX FILE EOF UPDATE COUNT
    RESFILES byte unsigned;                 /* NUMBER OF RESERVED FILES ON VOLUME
    RECORDSZ word unsigned;                 /* NUMBER OF BYTES IN A RECORD
    BLOCKFACT byte unsigned;                /* VOLUME BLOCKING FACTOR
    STATUS2_OVERLAY union fill;             /* SECOND STATUS BYTE
        STATUS2 byte unsigned;
        STATUS2_BITS structure fill;
            WRITETHRU bitfield;             /* VOLUME IS TO BE WRITE-THROUGH CACHED
            NOCACHE bitfield;               /* ALL CACHEING IS DISABLED ON VOLUME
            MOUNTVER bitfield;              /* VOLUME CAN UNDERGO MOUNT VERIFICATION
            ERASE bitfield;                 /* ERASE DATA WHEN BLOCKS REMOVED FROM FILE
            NOHIGHWATER bitfield;           /* TURN OFF HIGH-WATER MARKING (D = ON)
            NOSHARE bitfield;               /* non-shared mount
            CLUSLOCK bitfield;              /* CLUSTER WIDE LOCKING NECESSARY
        end STATUS2_BITS;
    end STATUS2_OVERLAY;
    QUOTAFCB longword unsigned;             /* ADDRESS OF FCB OF DISK QUOTA FILE
    CACHE longword unsigned;                /* ADDRESS OF VOLUME CACHE BLOCK
    QUOCACHE longword unsigned;             /* ADDRESS OF VOLUME QUOTA CACHE
    QUOSIZE word unsigned;                  /* LENGTH OF QUOTA CACHE TO ALLOCATE
    PENDERR word unsigned;                  /* COUNT OF PENDING WRITE ERRORS
    SERIALNUM longword unsigned;            /* VOLUME SERIAL NUMBER (DISKS ONLY)
    JNLIOCNT longword unsigned;             /* JOURNALING IO COUNT
    RETAINMIN quadword unsigned;            /* MINIMUM FILE RETENTION PERIOD
    RETAINMAX quadword unsigned;            /* MAXIMUM FILE RETENTION PERIOD
    VOLLKID longword unsigned;              /* VOLUME LOCK ID
```

```
    VOLCKNAM character length 12;                   /* NAME FOR VOLUME LOCKS
    BLOCKID longword unsigned;                      /* VOLUME BLOCKING LOCK.
    MOUNTTIME quadword unsigned;                    /* VOLUME MOUNT TIME
    MEMHDFL longword unsigned;                      /* SHADOW SET MEMBERS QUEUE HEADER FL
    MEMHDBL longword unsigned;                      /* SHADOW SET MEMBERS QUEUE HEADER BL
    ACTIVITY word unsigned;                         /* ACTIVITY COUNT/FLAG
    fill_1 byte fill;
    SHAD_STS byte unsigned;                         /* STATUS BYTE RELATIVE TO MEMHDFL
    SHAD_RESV longword unsigned;                    /* RESERVED FOR SHADOW SET PROCESSING
    ACB byte unsigned dimension 28;                 /* ACB FOR BLOCKING AST.
    MIN_CLASS structure;                            /* MINIMUM CLASSIFICATION
        FILL_2 byte dimension 20 fill;
    end MIN_CLASS;
    MAX_CLASS structure;                            /* MAXIMUM CLASSIFICATION
        FILL_3 byte dimension 20 fill;
    end MAX_CLASS;
    constant "LENGTH" equals . prefix VCB$ tag K;   /* LENGTH OF STANDARD VCB
    constant "LENGTH" equals . prefix VCB$ tag C;   /* LENGTH OF STANDARD VCB

end VCBDEF_DISKS;

/*
/* SHADOW SET MEMBER VOLUME CONTROL BLOCK FIELDS
/*

aggregate VCBDEF_SHADOW structure prefix VCB$;
    fillshadow byte dimension #VCBMARK3 fill;
    MEM_UCB longword unsigned;                      /* Shadow set member UCB address
    MAST_UCB longword unsigned;                     /* Shadow set master UCB address
    MAST_VCB longword unsigned;                     /* Shadow set master VCB address
    WORKQFL longword unsigned;                      /* Work queue forward link
    WORKQBL longword unsigned;                      /* Work queue backward link
    MSCP_STS longword unsigned;                     /* MSCP status information
    SHDW_RESV quadword unsigned;                    /* Reserved for future enhancements
    constant SHAD_LEN equals .;                     /* Shadow set member VCB length
end VCBDEF_SHADOW;

/*
/* MTAACP VOLUME CONTROL BLOCK FIELDS
/*

aggregate VCBDEF2 structure prefix VCB$;
    FILL_3 byte dimension #VCBMARK3 fill prefix VCBDEF tag $$;
    CUR_FID_OVERLAY union fill;
        CUR_FID longword unsigned;                  /* CURRENT FILE IDENTIFICATION
        CUR_FID_FIELDS structure fill;
            CUR_NUM word unsigned;                  /* CURRENT FILE SECTION NUMBER
            CUR_SEQ word unsigned;                  /* CURRENT FILE SEQUENCE NUMBER
        end CUR_FID_FIELDS;
    end CUR_FID_OVERLAY;
    START_FID_OVERLAY union fill;
        START_FID longword unsigned;                /* FILE IDENTIFICATION AT START OF SEARCH
        START_FID_FIELDS structure fill;
            START_NUM word unsigned;                /* FILE SECTION NUMBER AT START OF SEARCH
            START_SEQ word unsigned;                /* FILE SEQUENCE NUMBER AT START OF SEARCH
        end START_FID_FIELDS;
```

```
    end START_FID_OVERLAY;
    MODE_OVERLAY union fill;
        MODE word unsigned;                             /* MODE OF OPERATION
        MODE_BITS structure fill;
            OVREXP bitfield;                            /* OVERRIDE EXPIRATION
            OVRACC bitfield;                            /* OVERRIDE ACCESS
            OVRLBL bitfield;                            /* OVERRIDE LABELS
            OVRSETID bitfield;                          /* OVERRIDE SET IDENTIFIER
            INTCHG bitfield;                            /* INTERCHANGE TAPE
            EBCDIC bitfield;                            /* EBCDIC CODE SET
            NOVOL2 bitfield;                            /* DO NOT WRITE A VOL2 LABEL
            NOHDR3 bitfield;                            /* DO NOT WRITE HDR3 LABELS
            STARFILE bitfield;                          /* CURRENT FILE IS A STARLET PRODUCED FILE
            ENUSEREOT bitfield;                         /* SET WHEN USER HANDLING OF EOT IS ENABLED
            BLANK bitfield;                             /* SET FOR AVL WHEN NO READ SHOULD HAPPEN FIRST
            INIT bitfield;                              /* SET FOR AVL WHEN NEXT VOL MOUNTED SHOULD BE INITED
            NOAUTO bitfield;                            /* MTAACP NOT RUNNING IN AVL AND AVR MODE
            OVRVOLO bitfield;                           /* OVERRIDE THEVOL1 OWNER IDENT FIELD
            FIL_ACCESS bitfield;                        /* SET IF ACCESS ROUTINE ALLOWS CHECK OF VMS PROTECTION ON FILE
        end MODE_BITS;

    end MODE_OVERLAY;
    TM byte unsigned;                                   /* NUMBER OF TM'S INTO FILE
    CUR_RVN byte unsigned;                              /* CURRENT RELATIVE VOLUME
    ST_RECORD longword unsigned;                        /* NUMBER OF RECORDS UP TO AND INCLUDING LAST TAPE MARK
    MVL longword unsigned;                              /* ADDRESS OF MAGNETIC TAPE VOLUME LIST
    WCB longword unsigned;                              /* ADDRESS OF WINDOW FOR THIS VOLUME
    VPFL longword unsigned;                             /* VIRTUAL PAGE LIST HEAD
    VPBL longword unsigned;                             /* VIRTUAL PAGE LIST TAIL
    USRLBLAST longword unsigned;                        /* ADDRESS OF USER LABEL AST CONTROL BLOCK
    LBLCNT byte unsigned;                               /* Count of HDRn labels read on file open

/* NOTE THAT FCP AND MTAACP SHARE VCB$W_MCOUNT(DISPLACEMENT 76)

end VCBDEF2;

aggregate VCBDEF3 structure prefix VCB$;
    FILL_4 byte dimension #VCBMARK2 fill prefix VCBDEF tag $$;
    QNAMECNT byte unsigned;                             /* BYTE COUNT OF QUEUE NAME
    QNAME character length 20;                          /* ASCII NAME OF QUEUE FOR THIS DEVICE
/*
/* JOURNAL ACP VOLUME CONTROL BLOCK FIELDS
/*
end VCBDEF3;

aggregate VCBDEF4 structure prefix VCB$;
    FILL_5 byte dimension #VCBMARK3 fill prefix VCBDEF tag $$;
    JNL_CHAR_OVERLAY union fill;
        JNL_CHAR longword unsigned;                     /* journal media characteristics
        JNL_CHAR_BITS structure fill;
            JNL_DISK bitfield mask;                     /* journal is on disk
            JNL_TAPE bitfield mask;                     /* journal is on tape
            JNL_TMPFI bitfield mask;                    /* temporary file
        end JNL_CHAR_BITS;
    end JNL_CHAR_OVERLAY;
    JNL_JFTA longword unsigned;                         /* JOURNAL FILE TABLE ADDRESS (IN ACP)
```

```
    JNL_IRPS longword unsigned dimension 2;             /* PREALLOCATED FREE IRP QUEUE HEADER
    JNL_JMT longword unsigned;                          /* ADDRESS OF JMT (JOURNAL MERGE TABLE)
    JNL_UCB longword unsigned;                          /* UCB ADDRESS
    JNL_JMTFL longword unsigned;                        /* JMT FORWARD LINK
    JNL_JMTBL longword unsigned;                        /* JMT BACKWARD LINK
    JNL_MODE byte unsigned;                             /* ACCESS MODE OF CREATOR
    JNL_COP word unsigned;                              /* NUMBER OF JOURNAL FILE COPIES
    FILL_2 byte fill prefix VCBDEF tag $$;              /* SPARE
    JNL_MASK longword unsigned;                         /* MASK
    constant JNL_LENGTH equals . prefix VCB$ tag K;     /* LENGTH OF JOURNAL VCB
    constant JNL_LENGTH equals . prefix VCB$ tag C;     /* LENGTH OF JOURNAL VCB
end VCBDEF4;

end_module $VCBDEF;
```

```
module $VL1DEF;
/*+
/* VOL1 ANSI MAGNETIC TAPE LABEL
/* THIS IS THE FIRST BLOCK ON EVERY ANSI LABELED MAGNETIC TAPE.
/* IT IDENTIFIES THE VOLUME AND ITS PROTECTION.
/*-


aggregate VL1DEF structure prefix VL1$;
    VL1LID longword unsigned;                           /*LABEL IDENTIFIER AND NUMBER 'VOL1'
    VOLLBL character length 6;                          /*VOLUME LABEL
    VOLACCESS byte unsigned;                            /*VOLUME ACCESS
    FILL_1 character length 13 fill prefix VL1DEF tag $$;/*SPACES
    SYSCODE character length 13;                        /* SYSTEM CODE
    OWNER_UNION union fill;
        OWNER_IDENT character length 14;                /* VOL1 OWNER ID FIELD
        OLD_VOLOWNER structure fill;
           VOLOWNER character length 13;                /*VOLUME OWNER IDENTIFICATION
           DECSTDVER byte unsigned;                             /*DEC STANDARD VERSION
        end OLD_VOLOWNER;
    end OWNER_UNION;
    FILL_2 character length 28 fill prefix VL1DEF tag $$;/*SPACES
    LBLSTDVER byte unsigned;                            /*LABEL STANDARD VERSION '3'
end VL1DEF;

end_module $VL1DEF;
```

```
module $VL2DEF;
/*+
/* VOL2 ANSI MAGNETIC TAPE LABEL
/* THIS IS BLOCK IS WRITTEN TO TAPES WHEN A VMS PROTECTION IS SPECIFIED
/*-


aggregate VL2DEF structure prefix VL2$;
    VL2LID longword unsigned;                   /*LABEL IDENTIFIER AND NUMBER 'VOL2'
    VOLOWNER character length 15;               /*VOLUME OWNER IDENTIFICATION
end VL2DEF;

end_module $VL2DEF;
```

```
module $WCBDEF;

/++
/* WCB - WINDOW CONTROL BLOCK
/*
/* THERE IS A WINDOW CONTROL BLOCK FOR EACH FILE ACCESSED BY A PROCESS.
/* IT CONTAINS MAPPING INFORMATION SUCH THAT A LARGE PERCENTAGE OF VIRTUAL
/* FILE I/O CAN BE MAPPED FROM VIRTUAL TO LOGICAL BLOCK NUMBERS WITHOUT
/* HAVING TO READ THE RESPECTIVE FILE HEADER.
/*-

aggregate WCBDEF structure prefix WCB$;
    WLFL longword unsigned;                          /* WINDOW LIST FORWARD LINK
    WLBL longword unsigned;                          /* WINDOW LIST BACKWARD LINK
    SIZE word unsigned;                              /* SIZE OF WINDOW BLOCK IN BYTES
    TYPE byte unsigned;                              /* STRUCTURE TYPE OF WCB
    ACCESS_OVERLAY union fill;
        ACCESS byte unsigned;                        /* ACCESS CONTROL BYTE
        ACCESS_BITS structure fill;
            READ bitfield mask;                      /* READ ACCESS ALLOWED (1=YES)
            WRITE bitfield mask;                     /* WRITE ACCESS ALLOWED (1=YES)
            NOTFCP bitfield mask;                    /* FILE NOT ACCESSED BY FCP IF SET
            SHRWCB bitfield mask;                    /* SHARED WINDOW
            OVERDRAWN bitfield mask;                 /* FILE ACCESSOR HAS OVERDRAWN HIS QUOTA
            COMPLETE bitfield mask;                  /* SET WINDOW MAPS ENTIRE FILE
            CATHEDRAL bitfield mask;                 /* LARGE, COMPLEX WINDOW (SIC) TO MAP
                                                     /* FILE COMPLETELY
            EXPIRE bitfield mask;                    /* FILE EXPIRATION DATE MAY NEED TO BE SET
        end ACCESS_BITS;
    end ACCESS_OVERLAY;
    PID_OVERLAY union fill;
        PID longword unsigned;                       /* PROCESS ID OF ACCESSOR PROCESS
        PID_FIELDS structure fill;
            FILL_5 byte dimension 2 fill prefix WCBDEF tag $$;
            REFCNT word unsigned;                    /* REFERENCE COUNT FOR SHARED WINDOW
        end PID_FIELDS;
    end PID_OVERLAY;
    ORGUCB longword unsigned;                        /* ADDRESS OF ORIGINAL UCB FROM CCB
    ACON_OVERLAY union fill;
        ACON word unsigned;                          /* ACCESS CONTROL INFORMATION
                                                     /* NOTE - THESE BITS TRACK THE BITS
                                                     /* IN FIB$L_ACCTL
        ACON_BITS0 structure fill;
            NOWRITE bitfield;                        /* NO OTHER WRITERS
            DLOCK bitfield;                          /* ENABLE DEACCESS LOCK
            FILL_1 bitfield length 2 fill prefix WCBDEF tag $$;/* UNUSED
            SPOOL bitfield;                          /* SPOOL FILE ON CLOSE
            WRITECK bitfield;                        /* ENABLE WRITE CHECK
            SEQONLY bitfield;                        /* SEQUENTIAL ONLY ACCESS
            FILL_2 bitfield fill prefix WCBDEF tag $$;  /* SPARE
            WRITEAC bitfield;                        /* WRITE ACCESS
            READCK bitfield;                         /* ENABLE READ CHECK
            NOREAD bitfield;                         /* NO OTHER READERS
            NOTRUNC bitfield;                        /* NO TRUNCATES
        end ACON_BITS0;
```

```
        ACON_BITS1 structure fill;
            FILL_3 bitfield length 2 fill prefix WCBDEF tag $$;
                7* THE FOLLOWING FIELD OVERLAYS THE FIRST
                /* UNUSED FLAG IN WCB$W_ACON ABOVE.
            NOACCLOCK bitfield;                          /* NO ACCESS LOCK CHECKING
            FILL_4 bitfield length 8 fill prefix WCBDEF tag $$;
            READINIT bitfield;                           /* A READINIT WAS DONE OVER THIS CHANNEL
            WRITE_TURN bitfield;                         /* FORCE WINDOW TURN ON WRITES
        end ACON_BITS1;
    end ACON_OVERLAY;
    #WCBMARK2 = . ;                                      /* SIZE DEVICE INDEPENDENT PART OF WCB
    NMAP word unsigned;                                  /* NUMBER OF MAPPING POINTERS
    FCB longword unsigned;                               /* ADDRESS OF FCB
    RVT longword unsigned;                               /* ADDRESS OF RELATIVE VOLUME TABLE
    LINK longword unsigned;                              /* LINK TO NEXT WINDOW SEGMENT
    READS longword unsigned;                             /* COUNT OF READS PERFORMED
    WRITES longword unsigned;                            /* COUNT OF WRITES PERFORMED
    STVBN longword unsigned;                             /* STARTING VBN MAPPED BY WINDOW
    constant MAP equals . prefix WCB$ tag K;             /* MAP POINTERS START HERE
    constant MAP equals . prefix WCB$ tag C;             /* MAP POINTERS START HERE
    constant "LENGTH" equals . prefix WCB$ tag K;        /* LENGTH OF STANDARD WCB SANS POINTERS
    constant "LENGTH" equals . prefix WCB$ tag C;        /* LENGTH OF STANDARD WCB SANS POINTERS
                                                         /* NOTE THAT VIRTUAL MAPPING
                                                         /* NEEDS P1_COUNT IMMEDIATELY
                                                         /* FOLLOWING STVBN
    P1_COUNT word unsigned;                              /* COUNT FIELD OF FIRST POINTER
    P1_LBN longword unsigned;                            /* LBN FIELD OF SECOND POINTER
    P2_COUNT word unsigned;                              /* COUNT FIELD OF SECOND POINTER
    P2_LBN longword unsigned;                            /* LBN FIELD OF FIRST POINTER
                                                         /* FORMAT OF RETRIEVAL POINTER
end WCBDEF;

aggregate WCBDEF1 structure prefix WCB$;
    COUNT word unsigned;                                 /* COUNT FIELD
    LBN longword unsigned;                               /* LBN FIELD
end WCBDEF1;

aggregate WCBDEF2 structure prefix WCB$ origin FILL_6;
    PREVCOUNT word unsigned;                             /* PREVIOUS RETRIEVAL POINTER
    PREVLBN longword unsigned;
                                                         /* RETRIEVAL POINTER FORMAT
    FILL_6 byte fill prefix WCBDEF tag $$;
end WCBDEF2;

aggregate WCBDEF3 structure prefix WCB$;
    FILL_7 byte dimension #WCBMARK2 fill prefix WCBDEF tag $$;
    JNL_REFC word unsigned;                              /* REFERENCE COUNT AT $ASSJNL TIME
    JNL_FCOD word unsigned;                              /* FACILITY CODE OWNER JOURNAL CHANNEL
    JNL_STAT_OVERLAY union fill ;
        JNL_STAT byte unsigned;                          /* STATUS
        JNL_STAT_BITS structure fill ;
            JDB bitfield mask ;                          /* JDB ALLREADY WRITTEN OVER THIS CHANNEL
        end JNL_STAT_BITS ;
    end JNL_STAT_OVERLAY ;
    JNL_ACMOD byte unsigned;                             /* ACCESS MODE
    JNL_PROT word unsigned;                              /* PROTECTION MASK
```

```
    JNL_AID longword unsigned;                          /* ASSIGN SEQUENCE NUMBER
    JNL_SEQ longword unsigned;                          /* SEQUENCE NUMBER LAST ENTRY WRITTEN
    JNL_PRCNAM character length 16;                     /* PROCESS NAME OF CHANNEL OWNER
    JNL_UIC longword unsigned;                          /* UIC USED FOR ENTRIES WRITTEN OVER CHANNEL
    JNL_PUIC longword unsigned;                         /* UIC OF PROCESS
    JNL_TIME quadword unsigned;                         /* TIME AT WHICH CHANNEL WAS ASSIGNED
    JNL_WLFL longword unsigned;                         /* FORWARD LINK WCB QUEUE
    JNL_WLBL longword unsigned;                         /* BACKWARD LINK WCB QUEUE
    JNL_RC longword unsigned;                           /* READ CONTEXT BLOCK
    constant JNL_LEN equals . prefix WCB$ tag K;        /* LENGTH WCB FOR JOURNAL CHANNELS
    constant JNL_LEN equals . prefix WCB$ tag C;        /* LENGTH WCB FOR JOURNAL CHANNELS
end WCBDEF3;

end_module $WCBDEF;
```

```
module $WSLDEF;
/*+
/* WORKING SET LIST DEFINITIONS
/*-



aggregate WSLDEF  union prefix WSL$;
    WSLDEF_BITS structure fill;
        VALID bitfield mask;                                /*WSL ENTRY VALID
        PAGTYP bitfield mask length 3;                      /*PAGE TYPE (SEE PFNDEF FOR VALUES)
        PFNLOCK bitfield mask;                              /*PAGE FRAME LOCK
                                                            /*THE PRECEDING 5 BITS MUST BE IN ORDER
        WSLOCK bitfield mask;                               /*WORKING SET LOCK
        GOODPAGE bitfield mask;                             /*THIS PAGE SHOULD REMAIN IN WS ONE MORE PASS
        FILL_1 bitfield fill prefix WSLDEF tag $$;          /*SPARE BIT
        MODIFY bitfield mask;                               /*SAVED MODIFY BIT
    end WSLDEF_BITS;
                                                            /*THE FOLLOWING 5 BITS MUST BE IN ORDER

    constant "LENGTH"   equals 4  prefix WSL tag $C;        /*SIZE OF WS LIST ENTRY
/*
/* PAGE TYPE VIELD DEFINITIONS
/*
/*      N.B.:   These constants have been adjusted by left-shifting the constant by the offset to the field WSL$V_PAGTYP.
/*              To use these when explicitly extracting the field, the adjustment must be removed.  For example:
/*
/*                  IF .wsle [wsl$v_pagtyp] EQL (wsl$c_system^-1)                ! Or (wsl$c_system/2)
/*
    constant PROCESS    equals %X00 prefix WSL tag $C;  /*PROCESS PAGE
    constant SYSTEM     equals %X02 prefix WSL tag $C;  /*SYSTEM PAGE
    constant "GLOBAL"   equals %X04 prefix WSL tag $C;  /*GLOBAL PAGE (READ ONLY)
    constant GBLWRT     equals %X06 prefix WSL tag $C;  /*GLOBAL WRITABLE PAGE
    constant PPGTBL     equals %X08 prefix WSL tag $C;  /*PROCESS PAGE TABLE
    constant GPGTBL     equals %X0A prefix WSL tag $C;  /*GLOBAL PAGE TABLE
end WSLDEF;

end_module $WSLDEF;
```

```
module $WQHDEF;
/*+
/* WAIT QUEUE HEADER DEFINITIONS
/*-



aggregate WQHDEF structure prefix WQH$;
    WQFL longword unsigned;                          /*HEAD OR FORWARD LINK
    WQBL longword unsigned;                          /*TAIL OR BACKWARD LINK
    WQCNT word unsigned;                             /*WAIT QUEUE COUNT
    WQSTATE word unsigned;                           /*STATE NUMBER FOR WAIT
    constant "LENGTH" equals . prefix WQH$ tag K;    /*LENGTH OF WAIT QUEUE HEADER
    constant "LENGTH" equals . prefix WQH$ tag C;    /*LENGTH OF WAIT QUEUE HEADER

end WQHDEF;

end_module $WQHDEF;
```

```
{+
{ XG - Definitions for the fields within the XGDRIVER.
{
{-

module $XGDEF;
constant PRIXMT equals 0  prefix XG tag $C;              /* Primary xmt use vector slot 0
constant SECXMT equals 1  prefix XG tag $C;              /* Secondary xmt use vector slot 1
constant PRIRCV equals 2  prefix XG tag $C;              /* Primary rcv use vector slot 2
constant SECRCV equals 3  prefix XG tag $C;              /* Secondary rcv use vector slot 3
constant RCV_CSR        equals 0  prefix XG tag $C;      /* Receive CSR
constant XMT_CSR        equals 2  prefix XG tag $C;      /* Transmit CSR
constant MISC_REG       equals 4  prefix XG tag $C;      /* Set misc bits
constant IND_ADDR       equals 6  prefix XG tag $C;      /* Use to access the ind reg (IR)
constant(
      PROTOCOL                                           /* 0th IR def's the protocol char
    , RCV_ERR                                            /* 1st IR def's rcv errors
    , XMT_ERR                                            /* 2nd IR def's xmt errors
    , SYNC                                               /* 3rd IR def's sync charateristics
    , MODEM                                              /* 4th IR def's modem state change
    , STN_ADDR                                           /* 5th IR use to set station address
    , PRI_RCV                                            /* 6th and 7th IR used to define
    , PRI_RCV1                                           /* primary rcv buffer and address
    , SEC_RCV                                            /* 8th and 9th IR used to define
    , SEC_RCV1                                           /* secondary rcv buffer and address
    , PRI_XMT                                            /* 10th and 11th IR used to define
    , PRI_XMT1                                           /* primary xmt buffer and address
    , SEC_XMT                                            /* 12th and 13th IR used to define
    , SEC_XMT1                                           /* secondary xmt buffer and address
    , TERM_CHAR                                          /* 14th used to describe term char
    , FREE                                               /* 15th unused register
    ) equals 0 increment 1  prefix XG tag $C;

/* Bit def's for RCV and XMT CSR


aggregate XGDEF  union fill prefix XG$;
    XGDEF_BITS0 structure fill;
        ENABLE bitfield mask;                            /* Enable the receiver
        FILL_1 bitfield fill prefix XGDEF tag $$;        /* reserved
        PRM_SEC bitfield mask;                           /* 0 = prim 1 = sec buffer and addr
        TERM_IDL bitfield mask;                          /* Term char for RCV's Idle for XMT's
        DATA_SET_IE bitfield mask;                       /* Enable intrpts for data set change
        INT_ENABLE bitfield mask;                        /* Enable intrpts for rcv and xmt's
        ACT_DSC bitfield mask;                           /* Active (rcv's) Data set change (xmt')
        DONE_S bitfield mask;                            /* Sec buffer proceesing is finished
        ILP_XCS bitfield mask;                           /* Interal loopback (rcv) XMT clock src
        LOOP_TYPE bitfield mask length 2;                /* Loopb type for devices like CPI which support many
        FILL_2 bitfield fill prefix XGDEF tag $$;        /* reserved
        RESIDUAL bitfield mask;                          /* Bit protocols only
        PRI_SEC_STN bitfield mask;                       /* 0 = control 1 = tributary station
        ERROR bitfield mask;                             /* Error on rcv or xmt
        DONE_P bitfield mask;                            /* Primary buffer processing complete
```

```
    end XGDEF_BITS0;

/* Misc reg definitions

    XGDEF_BITS1 structure fill;
        IND_REG bitfield length 4;                          /* Ind reg address to access
        FILL_3 bitfield length 3 fill prefix XGDEF tag $$;/* reserved
        MASTER_RESET bitfield mask;                         /* Master reset bit
        FILL_4 bitfield length 2 fill prefix XGDEF tag $$;/* reserved
        USER_RCV_FLAG bitfield mask;                        /* User receive flag
        FILL_5 bitfield fill prefix XGDEF tag $$;
        CTS_FLAG bitfield mask;                             /* Clear to send flag
        CARRIER_FLAG bitfield mask;                         /* Carrier detect flag
        RING_FLAG bitfield mask;                            /* Ring indicator flag
        DSR_FLAG bitfield mask;                             /* Data set ready flag
    end XGDEF_BITS1;

/* Protocol parameter definitions Indirect register 0

    XGDEF_BITS2 structure fill;
        ERR_CNTRL bitfield length 3;                        /* Error control def CRC_CCITT 1's
        PROTOCOL bitfield length 3;                         /* Protocol type def DDCMP
        STRIP_SYNC bitfield mask;                           /* Set to strip excess sync characters
        FILL_6 bitfield fill prefix XGDEF tag $$;           /* reserved
        RCV_BPC bitfield length 3;                          /* RCV bits/char default is 8
        FILL_7 bitfield length 2 fill prefix XGDEF tag $$;/* reserved
        XMT_RCV bitfield length 3;                          /* XMT bits/char default is 8
    end XGDEF_BITS2;

/* Receive errors definitions Indirect register 1

    XGDEF_BITS3 structure fill;
        FILL_8 bitfield fill prefix XGDEF tag $$;           /* reserved
        LATENCY_RCV bitfield mask;                          /* RCV latency error
        NXM_RCV bitfield mask;                              /* Non-existant memory error
        BCC_ERR bitfield mask;                              /* Block check error
        VRC_ERR bitfield mask;                              /* Byte prot only char parity error
        ABORT bitfield mask;                                /* Bit prot only
        BUFOVR bitfield mask;                               /* When char COUNT and msg len aren't eq
        FILL_9 bitfield fill prefix XGDEF tag $$;           /* Reserved
        RES_BIT_CNT bitfield mask length 3;                 /* Residual bit count
        FILL_10 bitfield length 5 fill prefix XGDEF tag $$;/* Reserved
    end XGDEF_BITS3;

/* Transmit error definitions Indirect register 2

    XGDEF_BITS4 structure fill;
        MSG_LEN bitfield mask;                              /* Char count indicates a buff too small
        NXM_XMT bitfield mask;                              /* Non existant memory
        LATENCY_XMT bitfield mask;                          /* XMT latency error
        FILL_11 bitfield length 5 fill prefix XGDEF tag $$;/* Reserved
        XMT_BRG bitfield length 4;                          /* Baud rate
        FILL_12 bitfield length 4 fill prefix XGDEF tag $$;/* Reserved
    end XGDEF_BITS4;
```

```
/* Sync information definitions Indirect register 3

    XGDEF_BITS5 structure fill;
        NMB_OF_SYNC bitfield length 5;                  /* Number of syncs to send bettweenmsgs
        FILL_13 bitfield length 3 fill prefix XGDEF tag $$;/* Reserved
        SYNC bitfield length 8;                         /* Contains the sync char
    end XGDEF_BITS5;


/* Data set change register Indirect register 4

    XGDEF_BITS6 structure fill;
        FILL_14 bitfield length 4 fill prefix XGDEF tag $$;/* Reserved
        CTS bitfield mask;                              /* Clear to send
        CARRIER bitfield mask;                          /* Carrier detect
        RING_IND bitfield mask;                         /* Ring indicator
        DSR bitfield mask;                              /* Data set ready
        USER_XMT bitfield mask;                         /* User transmit
        DTR bitfield mask;                              /* Data terminal ready
        DATA_SGNL bitfield mask;                        /* Data signal rate
        FILL_15 bitfield fill prefix XGDEF tag $$;      /* reserved
        RTS bitfield mask;                              /* Request to send
        FILL_16 bitfield length 3 fill prefix XGDEF tag $$;/* Reserved
    end XGDEF_BITS6;

/* Internal clock def's TX.CSR<8>

    constant INTCLK_OFF equals 0  prefix XG tag $C;     /* No internal clock
    constant INTCLK_ON  equals 1  prefix XG tag $C;     /* Set internal clock

/* Error control definitions IR0<0:3>

    constant(
          ERR_CRC1                                      /* CRC-CCITT preset to 1's
        , ERR_CRC0                                      /* CRC-CCITT preset to 0's
        , ERR_LVE                                       /* LRC/VRC even
        , ERR_CRC16                                     /* CRC-16 preset to 0's
        , ERR_LRC0                                      /* LRC odd
        , ERR_LRCE                                      /* LRC even
        , ERR_LVO                                       /* LRC/VRC odd
        , NOCON                                         /* No error control
        ) equals 0 increment 1  prefix XG tag $C;


/* Protocol definitions  IR0<3:3>

    constant PRO_DDCMP  equals 0  prefix XG tag $C;     /* DDCMP
    constant PRO_SDLC   equals 1  prefix XG tag $C;     /* SDLC
    constant PRO_HDLC   equals 2  prefix XG tag $C;     /* HDLC
    constant BISYNC     equals 3  prefix XG tag $C;     /* BISYNC
    constant GENBYTE    equals 7  prefix XG tag $C;     /* General byte

/* Bits per char definitions. RCV: IR0<8:10>  XMT:IR0<13:15>

    constant(
          BPC_8
```

```
        , BPC_1
        , BPC_2
        , BPC_3
        , BPC_4
        , BPC_5
        , BPC_6
        , BPC_7
        ) equals 0 increment 1  prefix XG tag $C;

/* Baud rate generator definitions IR2<8:11>

    constant(
          BRG_800
        , BRG_1200
        , BRG_1760
        , BRG_2152
        , BRG_2400
        , BRG_4800
        , BRG_9600
        , BRG_19200
        ) equals 0 increment 1  prefix XG tag $C;

/* Sync character definitions IR3<8:15>

    constant SYNC_DDCMP equals 150  prefix XG tag $C;   /* Set sync character to HEX 96
    constant SYNC_HDLC  equals 0  prefix XG tag $C;     /* Set no sync character
    constant SYNC_BISYNC        equals 50  prefix XG tag $C;/* Set sync character to HEX 32

/* Struct of parameter buffer

end XGDEF;

    aggregate XGDEF1 structure fill prefix XG$;
    ERR_CNTRL byte unsigned;                        /* Set the type of error control to use
    PROTOCOL byte unsigned;                         /* Set protocol type
    TX_BPC byte unsigned;                           /* Set XMT bits per char
    RX_BPC byte unsigned;                           /* Set RCV bits per char
    BAUD byte unsigned;                             /* Set line speed
    NUM_SYNC byte unsigned;                         /* Set number of sync to send
    SYNC_REG byte unsigned;                         /* Set sync char to send
    ICLK byte unsigned;                             /* Set the internal clock
    BPC byte unsigned;                              /* RCV/XMT bits per char
    MNTLOOP_OVERLAY union fill;
        MNTLOOP byte unsigned;                      /* Maint loopb type
/*
/* Bit def for interface with the frame routine
/*
/* XG$V_BUFFER_CHAR          clear        Buffer char in the next position
/*                          set         Use XG$V_BUFFER_IN_PREV_POS
/* XG$V_BUFFER_IN_PREV_POS
/*                          clear        ignore the char
/*                          set         Buffer in previous position
/* XG$V_COMPLETE_READ          set         complete framed buffer to user
```

```
/*
        MNTLOOP_BITS structure fill;
            BUFFER_CHAR bitfield mask;
            BUFFER_IN_PREV_POS bitfield mask;
            COMPLETE_READ bitfield mask;
            FILL_17 bitfield length 28 fill prefix XGDEF tag $$;/* reserved
            NEW_FRAME bitfield mask;                        /* set if new rcv message
        end MNTLOOP_BITS;
    end MNTLOOP_OVERLAY;
    end XGDEF1;

end_module $XGDEF;
```

SYSDEFMP
SDL

SYSDEFQZ
SDL

ALLOCPFN
LIS

ACCOUNT
LIS

CMODSSDSP
MAR

SYSMAR
MAR

SYSPARAM
MAR

ASTDEL
LIS