```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR    RRR       MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

FILEID**RMSMAC


```
RRRRRRRR    MM       MM    SSSSSSSS  MM       MM    AAAAAA      CCCCCCCC
RRRRRRRR    MM       MM    SSSSSSSS  MM       MM    AAAAAA      CCCCCCCC
RR      RR  MMMM   MMMM  SS          MMMM   MMMM  AA      AA  CC
RR      RR  MMMM   MMMM  SS          MMMM   MMMM  AA      AA  CC
RR      RR  MM  MM  MM  SS           MM  MM  MM  AA      AA  CC
RR      RR  MM  MM  MM  SS           MM  MM  MM  AA      AA  CC
RRRRRRRR    MM       MM    SSSSSS    MM       MM  AA      AA  CC
RRRRRRRR    MM       MM    SSSSSS    MM       MM  AA      AA  CC
RR  RR      MM       MM          SS  MM       MM  AAAAAAAAAA  CC
RR  RR      MM       MM          SS  MM       MM  AAAAAAAAAA  CC
RR    RR    MM       MM          SS  MM       MM  AA      AA  CC              ....
RR    RR    MM       MM          SS  MM       MM  AA      AA  CC              ....
RR      RR  MM       MM  SSSSSSSS    MM       MM  AA      AA    CCCCCCCC      ....
RR      RR  MM       MM  SSSSSSSS    MM       MM  AA      AA    CCCCCCCC      ....

RRRRRRRR    EEEEEEEEEE    QQQQQQ
RRRRRRRR    EEEEEEEEEE    QQQQQQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RR  RR      EE          QQ  QQ  QQ
RR  RR      EE          QQ  QQ  QQ
RR    RR    EE          QQ    QQ
RR    RR    EE          QQ    QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
```

```
! RMSMAC.REQ  -  REQUIRE FILE FOR BLISS-32 INTERFACE TO RMS-32
!       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:     RMS-32 Interface from BLISS
!
! FUNCTIONAL DESCRIPTION:
!
!       This file is to be used as a REQUIRE FILE in any BLISS-32 program
!       using RMS-32.  Its MACROs permit the BLISS-32 program to allocate
!       and statically initialize the control blocks of interest to RMS-32,
!       dynamically initialize these control blocks, access the fields of
!       these control blocks, and invoke the functions supported by RMS-32.
!       Note that any changes made to any structure definitions must also
!       be reflected in the file  RMS32MAC.MAR  which contains the same
!       information for MACRO-32 programs.
!
! ENVIRONMENT:  Link (automatically) with STARLET.OLB
!
! AUTHOR:       Peter A. Belmont,       CREATION DATE:  27-May-77
!
! MODIFIED BY:
!
!       V03-017 DAS0005         David Solomon           06-Jun-1984
!               Remove extra comma in $XABTRM_INIT.
!
!       V03-016 DGB0037         Donald G. Blair         27-Mar-1984
!               Fix XAB$B_MTACC, which has never been initializable
!               from the $XABPRO macro.
```

```
        V03-015 DGB0035         Donald G. Blair         23-Mar-1984
                Add fields to protection xab: XAB$B_PROT_OPT,
                XAB$L_ACLBUF, XAB$W_ACLSIZ, XAB$W_ACLLEN,
                XAB$L_ACLCTX, XAB$L_ACLSTS

        V03-014 DAS0004         David Solomon           13-Feb-1984
                Fix typos in $XABJNL_INIT. Add support for FAB$V_LNM_MODE,
                FAB$V_CHAN_MODE, FAB$V_FILE_MODE, and XAB$B_PROT_MODE.

        V03-013 DAS0003         David Solomon           01-Aug-1983
                Fix missing comma in NOP changes in V03-012.

        V03-012 RAS0169         Ron Schaefer            12-Jul-1983
                Add NOP parameter and delete $XABACE macros.

        V03-011 LJA0066         Laurie J. Anderson      01-Mar-1983
                Add initialization of CXRBFZ (new field in XABCXR)
                length of the CXRBUF in bytes.
                Fix XABCXR_INIT which is missing a ).

        V03-010 JWH0190         Jeffrey W. Horn         21-Feb-1983
                Add $XABACE macros for the Access Control Entry XAB.

        V03-009 DAS0002         David Solomon           09-Feb-1983
                Add $XABTRM macros for the terminal XAB.

        V03-008 JWH0179         Jeffrey W. Horn         02-Feb-1983
                Fix bug in JWH0165.

        V03-007 JWH0165         Jeffrey W. Horn         05-Jan-1983
                Fix Journal XAB macros to relect changes to XAB layout.

        V03-006 LJA0047         Laurie J. Anderson      28-Dec-1982
                Add comments/initialization for SEQ/REL NRP context

        V03-005 LJA0036         Laurie J. Anderson      27-Oct-1982
                Allow for initialization of the key buffer pointers in XABCXR
                Fix up initialization of context XAB's due to rearrangement.

        V03-004 CWH0001         CW Hobbs                27-Aug-1982
                Standard form service calls ($CLOSE, etc) are now
                defined in <VMSLIB.SRC>STARLET.SDL.  The declarations
                in this module have been commented away.

        V03-003 JWH0001         Jeffrey W. Horn         07-Jul-1982
                Add XABJNL, Journaling XAB.

        V03-002 LJA0010         Laurie Anderson         6-Jul-1982
                Fix problem with initialization of RAB$L_XAB

        V03-001 LJA0009         Laurie Anderson         14-Jun-1982
                Add RAB$L_XAB, to support new context XAB (XABCXR)
                Add two new XABs (of type context), one for FAB, other
                for RAB, XABCXF and XABCXR, respectively.

        V02-010 RAS0061         Ron Schaefer            15-Jan-1982
```

```
                Add FAB$W_GBC and fix the bad status-check macros.

        V02-009 KBT0003         Keith B Thompson        8-Jan-1982
                Remove COMP, change STRUCT to PROLOG and add MTACC

        V02-008 CDS0001         C Saether               4-Nov-1981
                Use the symbolic value "xab$c_none" to specify
                default value for the key xab "struct" and "compat" fields.

        V02-007 PSK0002         Paulina S Knibbe        26-Aug-1981
                Make more things symbolic

        V02-006 PSK0001         Paulina S Knibbe        21-Aug-1981
                Add support for new long key XAB fields.

        V02-005 RAS0069         Ron Schaefer            17-Jun-1981
                Correct the service macros (eg. $CLOSE) to have the
                correct linkage declaration and addressing mode.

        V02-004 KRM0015         Karl Malik              18-May-1981
                Modify $NAM and $NAM_INIT to include the additional
                40 bytes of the extended NAM block.

        V02-003 MCN0007         Maria del C. Nasr       12-May-1981
                Use old symbol for new length of backup date and time XAB.

        V02-002 RAS0042         Ron Schaefer            11-Feb-1981
                Add the missing SUC parameters to the following file
                processing macros:
                    $CLOSE, $CREATE, $DISPLAY, $ERASE, $EXTEND, $MODIFY, $OPEN

        V02-001 MCN0004         Maria del C. Nasr       17-Dec-1980
                Change definition of $XABDAT to include backup date and
                time.

REVISION HISTORY:

                Peter A. Belmont,       22-Jun-77
                - added $CLOSE, etc., to $RMS_CLOSE, etc.
                - revised RMS_OKSTATUS macro
                - changed "$K_" back to "$C_"
                - fixed ASYN in $RAB to recognize YES (ASYN=YES)

                Peter A. Belmont,       6-July-77
                - added the XXX_INIT and XXX_INI macros
                - which implement dynamic initialization of
                - control blocks.

                Christopher Cooper,     31-Oct-78
                - A lot has happened since the last update of this
                  history.
                - Added XABKEY and XABSUM initialization/declaration
                  macros.

                Alan Lehotsky           13-Dec-78
                - No support in $FAB and $FAB_INIT macros for DNM or
```

```
!                   FNM keywords. Cannot completely support the MACRO-32
!                   semantics, as we cannot force the file-name string
!                   into the $RMSNAM psect.
!
!                 Unknown RMS V1.5 person          Spring 1979
!                 - Support added for the ISAM segments, KEY XAB's and additional
!                   key fields in RAB's and FAB's.
!                 - Support DNM and FNM keywords by allowing the names to go into
!                   the $PLIT$ psect.
!
!                 Ron Schaefer              22-Oct-79
!                 -  Support for the POS and SIZ multiple fields within the
!                    $XABKEY and $XABKEY_INIT macros.
!                    Either POS or POSn forms are accepted.
!
!                 Ron Schaefer              30-Oct-79
!                 -  Support for AID, BKZ and DEQ fields in $XABALL and
!                    $XABALL_INIT.  Also remove spurious support for
!                    NOA, NOR and PVN fields in $XABSUM and $XABSUM_INIT,
!                    since these fields are readonly.
!
! DOCUMENTATION:
!
!       RMSMPP.RNO      Mini Project Plan, BLISS RMS Support Project
!       RMSSPC.RNO      Functional Specifications, BLISS RMS Support
!                       User Documentation, BLISS RMS-32 Support
!                       Maintenance Documentation, BLISS RMS-32 Support
!--


!++
! $RMS_OFFSET
!
!       User macro.  Takes name of an RMS-32 field as
!       parameter and evaluates to the BYTE offset
!       of that field from the base of the control block.
!
!       Use:
!               OFFSET = $RMS_OFFSET(FAB$L_ABC);
!--

MACRO

        $RMS_OFFSET(NAME) = $RMS_OFFSET_1(%REMOVE(NAME)) %,
        $RMS_OFFSET_1 (X) = X %;

!++
! STATIC INITIALIZATION MACRO UTILITIES
!
!       Internal macros.  Used by the initialization
```

```
!       macros below to assign initial values to
!       the fields of the RMS-32 control blocks.
!
!--

MACRO

!++
! $RMS_BITFLD (and its support macros ...)
!
!       Internal macro.  Allows the initialization
!       of a field with the OR of one or more (named) bits.
!
!       $RMS_BITFLD(ALLOC,PREFIX,VALUE) builds an
!       initial value of size ALLOC ( BYTE, WORD,
!       or LONG ), setting the bits whose names
!       are of the form PREFIX//VAL where
!       VAL is either VALUE or one of the elements
!       of VALUE when VALUE is a tuple (i.e., <A,B,C>).
!
!--

        $RMS_BITS(A,B)[]=
            %NAME(A,B)   $RMS_OR(%REMAINING)   $RMS_BITS(A,%REMAINING) %,

        $RMS_OR[]=
            OR %,

        $RMS_BITFLD(ALLOC,PREFIX,VALUE)=
            %IF %NULL(VALUE)
            %THEN ALLOC(0)
            %ELSE ALLOC($RMS_BITS(PREFIX,%REMOVE(VALUE)))
            %FI %,

!++
! $RMS_BITFLD_INI
!
!       Internal macro.  Permits the dynamic initialization
!       of a field with the OR of one or more named bits.
!
!       $RMS_BITFLD_INI(BLK,NAME,PREFIX,VALUE) stores
!       into BLK[NAME] the value obtained by OR'ing
!       together the bits whose names are PREFIX//VAL
!       where VAL is either VALUE or is one of the
!       elements of VALUE when VALUE is a tuple.
!
!--

    $RMS_BITFLD_INI(BLK,NAME,PREFIX,VALUE)=
        %IF %NULL(VALUE) %THEN %EXITMACRO %FI
        BLOCK[BLK,%REMOVE(NAME);0,BYTE] =
        $RMS_BITS(PREFIX,%REMOVE(VALUE)) %,
```

```
!++
! $RMS_CODFLD
!
!       Internal macro.  Allows the initialization
!       of a field with a named value.
!
!       $RMS_CODFLD(ALLOC,PREFIX,VALUE)
!       allocates a value of size ALLOC (BYTE, WORD, or LONG)
!       and initializes it with the value whose name
!       is PREFIX//VALUE.
!--

        $RMS_CODFLD(ALLOC,PREFIX,VALUE)=
            ALLOC(%NAME(PREFIX,%REMOVE(VALUE))) %,



!++
! $RMS_CODFLD_INI
!
!       Internal macro.  Allows the dynamic initialization
!       of a field with a named value.
!
!       $RMS_CODFLD_INI(BLK,NAME,PREFIX,VALUE)
!       assigns to BLK[NAME] the value whose name is
!       PREFIX//VALUE
!--

        $RMS_CODFLD_INI(BLK,NAME,PREFIX,VALUE)=
        %IF %NULL(VALUE) %THEN %EXITMACRO %FI
            BLOCK[BLK,%REMOVE(NAME):0,BYTE] =
                (%NAME(PREFIX,%REMOVE(VALUE))) %,


!++
! $RMS_VALFLD
!
!       Internal macro.  Permits the initialization
!       of a field with a general value.
!
!       $RMS_VALFLD(ALLOC,VALUE) allocates
!       a field of size ALLOC (BYTE, WORD, or LONG)
!       and initializes it with the (general) value, VALUE.
!--

        $RMS_VALFLD(ALLOC,VALUE)=
            ALLOC(VALUE) %,
```

```
!++
! $RMS_VALFLD_INI
!
!       Internal macro.  Permits the dynamic initialization
!       of a field with a general value.
!
!       $RMS_VALFLD_INI(BLK,NAME,VALUE) stores the value VALUE
!       the field B[K[NAME].
!--

        $RMS_VALFLD_INI(BLK,NAME,VALUE)=
        %IF %NULL(VALUE) %THEN %EXITMACRO %FI
        %IF %IDENTICAL(VALUE,0) %THEN %EXITMACRO %FI
            BLOCK[BLK,%REMOVE(NAME):0,BYTE] = VALUE %,


!++
! $RMS_8FLD
!
!       Internal macro.  Permits the initialization of 8 contiguous fields
!       named by the technique XXX$A_FLDn, where n ranges from 0 to 7.
!
!       $RMS_8FLD(ALLOC,VALUE0,VALUE1,VALUE2,VALUE3,VALUE4,VALUE5,VALUE6,VALUE7)
!       allocates a 8 continguous fields of size ALLOC (BYTE, WORD or LONG)
!       and initializes them with the values VALUE0 thru VALUE7.  A value of 0
!       is supplied for any values that are null.
!--

        $RMS_8FLD(ALLOC,VAL0,VAL1,VAL2,VAL3,VAL4,VAL5,VAL6,VAL7)=
        ALLOC(
            %IF %NULL(VAL0) %THEN (0) %ELSE (VAL0) %FI,
            %IF %NULL(VAL1) %THEN (0) %ELSE (VAL1) %FI,
            %IF %NULL(VAL2) %THEN (0) %ELSE (VAL2) %FI,
            %IF %NULL(VAL3) %THEN (0) %ELSE (VAL3) %FI,
            %IF %NULL(VAL4) %THEN (0) %ELSE (VAL4) %FI,
            %IF %NULL(VAL5) %THEN (0) %ELSE (VAL5) %FI,
            %IF %NULL(VAL6) %THEN (0) %ELSE (VAL6) %FI,
            %IF %NULL(VAL7) %THEN (0) %ELSE (VAL7) %FI
        )%,

!++
! $RMS_8FLD_INI
!
!       Internal macro.  Permits the dynamic initialization of 8 contiguous
!       fields named by the technique XAB$f_FLDn, where n ranges from 0 to 7.
!
!       $RMS_8FLD-INI(BLK,NAME)[VALUE_LIST]
```

```
!       initializes up to 8 fields with the values in VALUE_LIST.
!
!--

        $RMS_8FLD_INI(BLK,NAME) [VAL]=
            %IF %COUNT GEQ 8
            %THEN
                %ERROR('Excess POS or SIZ values.')
                %EXITMACRO
            %FI
            $RMS_VALFLD_INI(BLK, (%NAME(NAME,%COUNT)), VAL)  %,


!++
! $RMS_VALPRO (and support macros ...)
!
!       Internal macro.  Permits the initialization
!       of a protection-word following the RSX-11M
!       and STARLET conventions.
!
!--
        $RMS_VALPR2(RWEDCHAR)[]=
            AND NOT
            %IF       %IDENTICAL(RWEDCHAR,'R') %THEN XAB$M_NOREAD
            %ELSE %IF %IDENTICAL(RWEDCHAR,'W') %THEN XAB$M_NOWRITE
            %ELSE %IF %IDENTICAL(RWEDCHAR,'E') %THEN XAB$M_NOEXE
            %ELSE %IF %IDENTICAL(RWEDCHAR,'D') %THEN XAB$M_NODEL
            %ELSE 0 %ERROR('Illegal value ''', RWEDCHAR, ''' of parameter PRO')
            %FI %FI %FI %FI
            $RMS_VALPR2(%REMAINING) %,

        $RMS_VALPR1(RWEDGROUP)=
            %B'1111'  $RMS_VALPR2(%EXPLODE(RWEDGROUP)) %,

        $RMS_VALPRO(SYSTEM,OWNER,GROUP,WORLD)=
            %IF %LENGTH GTR 4
            %THEN
                WORD(0)
                %ERROR('Illegal value of parameter PRO')
            %ELSE
                WORD(
                ($RMS_VALPR1(SYSTEM)) OR ($RMS_VALPR1(OWNER))^4 OR
                ($RMS_VALPR1(GROUP))^8 OR ($RMS_VALPR1(WORLD))^12)
            %FI %,


!++
! $RMS_VALPRO_INI
!
!       Internal macro.  Permits the dynamic initialization
!       of a protection word following the RSX-11M and STARLET
!       convention.
!
!--
```

```
        $RMS_VALPRO_INI(BLK,SYSTEM,OWNER,GROUP,WORLD)=
            %IF %LENGTH GTR 5 %THEN
                %ERROR('Illegal value of parameter PRO')
                %EXITMACRO %FI
            %IF %LENGTH EQL 1 %THEN %EXITMACRO %FI
            BLOCK[BLK,XAB$W_PRO;0,BYTE] =
                ($RMS_VALPR1(SYSTEM) OR ($RMS_VALPR1(OWNER))^4 OR
                ($RMS_VALPR1(GROUP))^8 OR ($RMS_VALPR1(WORLD))^12)
                %,


!++
! $RMS_VALUIC
!
!       Internal macro.  Permits the initialization
!       of the two-word item, MEMBER/GROUP
!--

        $RMS_VALUIC(GROUP,MEMBER)=
            %IF %LENGTH NEQ 2
            %THEN
                REP 2 OF WORD(0)
                %ERROR('Illegal value of parameter UIC')
            %ELSE
                WORD(MEMBER,GROUP)
            %FI %,


!++
! $RMS_VALUIC_INI
!
!       Internal macro.  Permits the dynamic initialization
!       of the two-word item, MEMBER/GROUP
!--

        $RMS_VALUIC_INI(BLK,GROUP,MEMBER)=
            %IF %LENGTH NEQ 3
            %THEN %ERROR('Illegal value of parameter UIC')
                %EXITMACRO %FI
            %IF %IDENTICAL(GROUP,0) AND %IDENTICAL(MEMBER,0)
            %THEN %EXITMACRO %FI
            BLOCK[BLK,XAB$L_UIC;0,BYTE]
                = ((MEMBER) OR (GROUP)^16) %,


!++
! $RMS_VALRFI
!
!       Internal macro.  Permits the initialization
!       of the three-word RFI field with
```

```
!       FILENO, SEQNO, RVN
!
!--

        $RMS_VALRFI(FILENO,SEQNO,RVN)=
            %IF %LENGTH NEQ 3
            %THEN
                REP 3 OF WORD(0)
                %ERROR('Illegal value of parameter RFI')
            %ELSE
                WORD(FILENO,SEQNO,RVN)
            %FI %,


!++
! $RMS_VALRFI_INI
!
!       Internal macro.  Permits the dynamic initialization
!       of the three-word RFI field with FILENO, SEQNO, and RVN.
!
!--

        $RMS_VALRFI_INI(BLK,FILENO,SEQNO,RVN)=
            %IF %LENGTH NEQ 4
             %THEN %ERROR('Illegal value of parameter RFI')
                %EXITMACRO %FI
            %IF %IDENTICAL(FILENO,0) AND
                %IDENTICAL(SEQNO ,0) AND
                %IDENTICAL(RVN   ,0) %THEN %EXITMACRO %FI
            BLOCK[BLK,XAB$W_RFI0;0,BYTE]= FILENO;
            BLOCK[BLK,XAB$W_RFI2;0,BYTE]= SEQNO;
            BLOCK[BLK,XAB$W_RFI4;0,BYTE]= RVN
        %;
```

```
!++
! $FAB_DECL
!
!       Used to declare a FAB control block where
!       initialization is not required.
!--

MACRO
        $FAB_DECL =
        BLOCK[FAB$C_BLN,BYTE]   %;

!++
! $FAB
!
!       Used to allocate and statically initialize
!       a FAB control block.
!--

KEYWORDMACRO

        $FAB(
                FAC=GET,        SHR,            FNA=0,          FNS=0,
                DNA=0,          DNS=0,          RTV=0,          ORG=SEQ,
                RAT,            FOP,            XAB=0,          MRS=0,
                JNL=0,          MRN=0,          ALQ=0,          DEQ=0,
                BLS=0,          NAM=0,          RFM=VAR,        FSZ=0,
                FNM,            DNM,            BKS=0,          CTX=0,
                GBC=0,          LNM_MODE=0,     CHAN_MODE=0,    FILE_MODE=0
                )=

        $FAB_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       FAB$C_BID),     ! BID
                $RMS_VALFLD(BYTE,                       FAB$C_BLN),     ! BLN
                $RMS_VALFLD(WORD,                       0),             ! IFI
                $RMS_BITFLD(LONG,       FAB$M_,         FOP),           ! FOP
                $RMS_VALFLD(LONG,                       0),             ! STS
                $RMS_VALFLD(LONG,                       0),             ! STV
                $RMS_VALFLD(LONG,                       ALQ),           ! ALQ
                $RMS_VALFLD(WORD,                       DEQ),           ! DEQ
                $RMS_BITFLD(BYTE,       FAB$M_,         FAC),           ! FAC
                $RMS_BITFLD(BYTE,       FAB$M_,         SHR),           ! SHR
                $RMS_VALFLD(LONG,                       CTX),           ! CTX
                $RMS_VALFLD(BYTE,                       RTV),           ! RTV
                $RMS_CODFLD(BYTE,       FAB$C_,         ORG),           ! ORG
                $RMS_BITFLD(BYTE,       FAB$M_,         RAT),           ! RAT
                $RMS_CODFLD(BYTE,       FAB$C_,         RFM),           ! RFM
                $RMS_VALFLD(LONG,                       JNL),           ! JNL
                $RMS_VALFLD(LONG,                       XAB),           ! XAB
                $RMS_VALFLD(LONG,                       NAM),           ! NAM
%IF %NULL(FNM) %THEN
                $RMS_VALFLD(LONG,                       FNA),           ! FNA
```

```
%ELSE
                $RMS_VALFLD(LONG,               UPLIT BYTE(FNM)),       ! FNA
%FI
%IF %NULL(DNM) %THEN
                $RMS_VALFLD(LONG,                       DNA),           ! DNA
%ELSE
                $RMS_VALFLD(LONG,               UPLIT BYTE(DNM)),       ! DNA
%FI
%IF %NULL(FNM) %THEN
                $RMS_VALFLD(BYTE,                       FNS),           ! FNS
%ELSE
                $RMS_VALFLD(BYTE,               %CHARCOUNT(FNM)),       ! FNS
%FI
%IF %NULL(DNM) %THEN
                $RMS_VALFLD(BYTE,                       DNS),           ! DNS
%ELSE
                $RMS_VALFLD(BYTE,               %CHARCOUNT(DNM)),       ! DNS
%FI
                $RMS_VALFLD(WORD,                       MRS),           ! MRS
                $RMS_VALFLD(LONG,                       MRN),           ! MRN
                $RMS_VALFLD(WORD,                       BLS),           ! BLS
                $RMS_VALFLD(BYTE,                       BKS),           ! BKS
                $RMS_VALFLD(BYTE,                       FSZ),           ! FSZ
                $RMS_VALFLD(LONG,                       0),             ! DEV
                $RMS_VALFLD(LONG,                       0),             ! SDC
                $RMS_VALFLD(WORD,                       GBC),           ! GBC
                BYTE(                                                   ! ACMODES
                        ( LNM_MODE  ^ $BITPOSITION(FAB$V_LNM_MODE) )
                      + ( CHAN_MODE ^ $BITPOSITION(FAB$V_CHAN_MODE) )
                      + ( FILE_MODE ^ $BITPOSITION(FAB$V_FILE_MODE) ) ),
                $RMS_VALFLD(BYTE,                       0),             ! RCF
                $RMS_VALFLD(LONG,                       0)              ! SPARE
        ) %;


!++
! $FAB_INIT
!
!       Used to dynamically initialize
!       a FAB control block.
!--

KEYWORDMACRO

        $FAB_INIT(

                FAB,

                FAC=GET,        SHR,            FNA,            FNS,
                DNA,            DNS,            RTV,            ORG,
                RAT,            FOP,            XAB,            MRS,
                JNL,            MRN,            ALQ,            DEQ,
                BLS,            NAM,            RFM=VAR,        FSZ,
                FNM,            DNM,            BKS,            CTX,
```

```
                GBC,            LNM_MODE,       CHAN_MODE,      FILE_MODE
                )=

        ( BIND $RMS_PTR = FAB;
          CH$FILL(0,FAB$C_BLN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_BID),                   FAB$C_BID);     ! BID
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_BLN),                   FAB$C_BLN);     ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$W_IFI),                   0);             ! IFI
                $RMS_BITFLD_INI($RMS_PTR,(FAB$L_FOP),   FAB$M_,         FOP);           ! FOP
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$L_STS),                   0);             ! STS
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$L_STV),                   0);             ! STV
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_ALQ),                   ALQ);           ! ALQ
                $RMS_VALFLD_INI($RMS_PTR,(FAB$W_DEQ),                   DEQ);           ! DEQ
                $RMS_BITFLD_INI($RMS_PTR,(FAB$B_FAC),   FAB$M_,         FAC);           ! FAC
                $RMS_BITFLD_INI($RMS_PTR,(FAB$B_SHR),   FAB$M_,         SHR);           ! SHR
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_CTX),                   CTX);           ! CTX
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_RTV),                   RTV);           ! RTV
                $RMS_CODFLD_INI($RMS_PTR,(FAB$B_ORG),   FAB$C_,         ORG);           ! ORG
                $RMS_BITFLD_INI($RMS_PTR,(FAB$B_RAT),   FAB$M_,         RAT);           ! RAT
                $RMS_CODFLD_INI($RMS_PTR,(FAB$B_RFM),   FAB$C_,         RFM);           ! RFM
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_JNL),                   JNL);           ! JNL
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_XAB),                   XAB);           ! XAB
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_NAM),                   NAM);           ! NAM
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_FNA),
                        %IF %NULL(FNM) %THEN FNA %ELSE UPLIT BYTE(FNM) %FI);            ! FNA
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_DNA),
                        %IF %NULL(DNM) %THEN DNA %ELSE UPLIT BYTE(DNM) %FI);            ! DNA
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_FNS),
                        %IF %NULL(FNM) %THEN FNS %ELSE %CHARCOUNT(FNM) %FI);            ! FNS
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_DNS),
                        %IF %NULL(DNM) %THEN DNS %ELSE %CHARCOUNT(DNM) %FI);            ! DNS
                $RMS_VALFLD_INI($RMS_PTR,(FAB$W_MRS),                    MRS);          ! MRS
                $RMS_VALFLD_INI($RMS_PTR,(FAB$L_MRN),                    MRN);          ! MRN
                $RMS_VALFLD_INI($RMS_PTR,(FAB$W_BLS),                    BLS);          ! BLS
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_BKS),                    BKS);          ! BKS
                $RMS_VALFLD_INI($RMS_PTR,(FAB$B_FSZ),                    FSZ);          ! FSZ
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$L_DEV),                    0);            ! DEV
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$L_SDC),                    0);            ! SDC
                $RMS_VALFLD_INI($RMS_PTR,(FAB$W_GBC),                    GBC);          ! GBC
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$B_ACMODES),                0);            ! ACMODES
                $RMS_VALFLD_INI($RMS_PTR,(FAB$V_LNM_MODE),               LNM_MODE);     ! LNM_MODE
                $RMS_VALFLD_INI($RMS_PTR,(FAB$V_CHAN_MODE),              CHAN_MODE);    ! CHAN_MODE
                $RMS_VALFLD_INI($RMS_PTR,(FAB$V_FILE_MODE),              FILE_MODE);    ! FILE_MODE
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$B_RCF),                    0);            ! RCF
!               $RMS_VALFLD_INI($RMS_PTR,(FAB$L_SPARE),                  0);            ! SPARE
        0 ) %;
```

```
!++
! $NAM_DECL
!
!       Permits the declaration of the NAM control block
!       where initialization is not required.
!--
MACRO
        $NAM_DECL = BLOCK[NAM$C_BLN,BYTE] %;

!++
! $NAM
!
!       Macro to allocate and initialize the NAM control block.
!--

KEYWORDMACRO

        $NAM(
                RSA=0,          RSS=0,          ESA=0,          ESS=0,
                RLF=0,          NOP)=

        $NAM_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       NAM$C_BID),     ! BID
                $RMS_VALFLD(BYTE,                       NAM$C_BLN),     ! BLN
                $RMS_VALFLD(BYTE,                       RSS),           ! RSS
                $RMS_VALFLD(BYTE,                       0),             ! RSL
                $RMS_VALFLD(LONG,                       RSA),           ! RSA
                $RMS_BITFLD(BYTE,       NAM$M_,         NOP),           ! NOP
                $RMS_VALFLD(BYTE,                       0),             ! RFS
                $RMS_VALFLD(BYTE,                       ESS),           ! ESS
                $RMS_VALFLD(BYTE,                       0),             ! ESL
                $RMS_VALFLD(LONG,                       ESA),           ! ESA
                $RMS_VALFLD(LONG,                       RLF),           ! RLF
                REP 8 OF $RMS_VALFLD(WORD,              0),             ! DVI
                REP 3 OF $RMS_VALFLD(WORD,              0),             ! FID
                REP 3 OF $RMS_VALFLD(WORD,              0),             ! DID
                $RMS_VALFLD(LONG,                       0),             ! WCC
                $RMS_VALFLD(LONG,                       0),             ! FNB
                $RMS_VALFLD(BYTE,                       0),             ! NODE
                $RMS_VALFLD(BYTE,                       0),             ! DEV
                $RMS_VALFLD(BYTE,                       0),             ! DIR
                $RMS_VALFLD(BYTE,                       0),             ! NAME/QUOTED
                $RMS_VALFLD(BYTE,                       0),             ! TYPE
                $RMS_VALFLD(BYTE,                       0),             ! VER
                REP 2 OF $RMS_VALFLD(BYTE,              0),             ! SPARE
                $RMS_VALFLD(LONG,                       0),             ! NODE
                $RMS_VALFLD(LONG,                       0),             ! DEV
                $RMS_VALFLD(LONG,                       0),             ! DIR
                $RMS_VALFLD(LONG,                       0),             ! NAME/QUOTED
```

```
                $RMS_VALFLD(LONG,                       0),             ! TYPE
                $RMS_VALFLD(LONG,                       0),             ! VER
                REP 2 OF $RMS_VALFLD(LONG,              0)              ! SPARE
        ) %;


!++
!
! $NAM_INIT
!
!       Used to dynamically initialize
!       a NAM control block.
!
!--

KEYWORDMACRO

        $NAM_INIT(

                NAM,

                RSA,            RSS,            ESA,            ESS,
                RLF,            NOP)=

        ( BIND $RMS_PTR = NAM;
          CH$FILL(0,NAM$C_BLN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(NAM$B_BID),                   NAM$C_BID);     ! BID
                $RMS_VALFLD_INI($RMS_PTR,(NAM$B_BLN),                   NAM$C_BLN);     ! BLN
                $RMS_VALFLD_INI($RMS_PTR,(NAM$B_RSS),                   RSS);           ! RSS
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_RSL),                   0);             ! RSL
                $RMS_VALFLD_INI($RMS_PTR,(NAM$L_RSA),                   RSA);           ! RSA
                $RMS_BITFLD_INI($RMS_PTR,(NAM$B_NOP),   NAM$M_,         NOP);           ! NOP
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_RFS),                   0);             ! RFS
                $RMS_VALFLD_INI($RMS_PTR,(NAM$B_ESS),                   ESS);           ! ESS
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_ESL),                   0);             ! ESL
                $RMS_VALFLD_INI($RMS_PTR,(NAM$L_ESA),                   ESA);           ! ESA
                $RMS_VALFLD_INI($RMS_PTR,(NAM$L_RLF),                   RLF);           ! RLF
!               REP 8 OF $RMS_VALFLD_INI($RMS_PTR,(NAM$W_DVI),          0);             ! DVI
!               REP 3 OF $RMS_VALFLD_INI($RMS_PTR,(NAM$W_FID),          0);             ! FID
!               REP 3 OF $RMS_VALFLD_INI($RMS_PTR,(NAM$W_DID),          0);             ! DID
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_WCC),                   0);             ! WCC
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_FNB),                   0);             ! FNB
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_NODE),                  0);             ! NODE
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_DEV),                   0);             ! DEV
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_DIR),                   0);             ! DIR
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_NAME),                  0);             ! NAME/QUOTED
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_TYPE),                  0);             ! TYPE
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$B_VER),                   0);             ! VER
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(NAM$B_SPARE),        0),             ! SPARE
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_NODE),                  0);             ! NODE
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_DEV),                   0);             ! DEV
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_DIR),                   0);             ! DIR
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_NAME),                  0);             ! NAME/QUOTED
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_TYPE),                  0);             ! TYPE
```

```
!               $RMS_VALFLD_INI($RMS_PTR,(NAM$L_VER),                   0);             ! VER
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(NAM$L_SPARE),        0);             ! SPARE
        0 ) %;
```

```
!++
! $XABDAT_DECL
!
!       Permits the declaration of the XABDAT control block
!       where initialization is not required.
!--

MACRO
        $XABDAT_DECL = BLOCK[XAB$C_DATLEN,BYTE] %;


!++
! $XABDAT
!
!       Macro to allocate and initialize the XABDAT control block.
!--

KEYWORDMACRO
        $XABDAT(
                NXT=0,  RVN=0,  EDT0=0, EDT4=0)=

        $XABDAT_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_DAT),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_DATLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_VALFLD(WORD,                       RVN),           ! RVN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                REP 2 OF $RMS_VALFLD(LONG,              0),             ! RDT
                REP 2 OF $RMS_VALFLD(LONG,              0),             ! CDT
                $RMS_VALFLD(LONG,                       EDT0),          ! EDT0
                $RMS_VALFLD(LONG,                       EDT4),          ! EDT4
                REP 2 OF $RMS_VALFLD(LONG,              0)              ! BDT
        ) %;

!++
! $XABDAT_INIT
!
!       Macro to dynamically initialize the XABDAT control block.
!--

KEYWORDMACRO
        $XABDAT_INIT(
                XAB,
```

```
                NXT=0,  RVN=0,  EDT0=0, EDT4=0)=

        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_DATLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_DAT);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_DATLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_RVN),                   RVN);           ! RVN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(XAB$L_RDT),          0);             ! RDT
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CDT),          0);             ! CDT
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_EDT0),                  EDT0);          ! EDT0
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_EDT4),                  EDT4);          ! EDT4
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(XAB$L_BDT),          0);             ! BDT
          0 ) %;


!++
! $XABRDT_DECL
!
!       Permits the declaration of the XABRDT control block
!       where initialization is not required.
!--

MACRO
        $XABRDT_DECL = BLOCK[XAB$C_RDTLEN,BYTE] %,


!++
! $XABRDT
!
!       Macro to allocate and initialize the XABRDT control block.
!--

KEYWORDMACRO

        $XABRDT(
                NXT=0,  RVN=0 )=

        $XABRDT_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_RDT),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_RDTLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_VALFLD(WORD,                       RVN),           ! RVN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                REP 2 OF $RMS_VALFLD(LONG,              0)              ! RDT
```

```
        ) %;

!++
! $XABRDT_INIT
!       Macro to dynamically initialize the XABRDT control block.
!--
KEYWORDMACRO
        $XABRDT_INIT(
                XAB,
                NXT=0,  RVN=0 )=
        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_RDTLEN,CH$PTR($RMS_PTR));
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_RDT);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_RDTLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_RVN),                   RVN);           ! RVN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
!               REP 2 OF $RMS_VALFLD_INI($RMS_PTR,(XAB$L_RDT),          0);             ! RDT
          0 ) %;

!++
! $XABPRO_DECL
!       Permits the declaration of the XABPRO control block
!       where initialization is not required.
!--
MACRO
        $XABPRO_DECL = BLOCK[XAB$C_PROLEN,BYTE] %;

!++
! $XABPRO
!       Macro to allocate and initialize the XABPRO control block.
!--
KEYWORDMACRO
        $XABPRO(
```

```
                NXT=0,          UIC=<0,0>,      PRO,            MTACC=0,
                PROT_OPT,       PROT_MODE=0,    ACLBUF=0,       ACLSIZ=0,
                ACLCTX=0 )=

        $XABPRO_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_PRO),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_PROLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_VALPRO(%REMOVE(PRO)),                              ! PRO
                $RMS_VALFLD(BYTE,                       MTACC),         ! MTACC
                $RMS_BITFLD(BYTE,       XAB$M_,         PROT_OPT),      ! PROT_OPT
                $RMS_VALUIC(%REMOVE(UIC)),                              ! UIC
                $RMS_VALFLD(BYTE,                       PROT_MODE),     ! PROT_MODE
                $RMS_VALFLD(BYTE,                       0),             ! SPARE
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       ACLBUF),        ! ACLBUF
                $RMS_VALFLD(WORD,                       ACLSIZ),        ! ACLSIZ
                $RMS_VALFLD(WORD,                       0),             ! ACLLEN
                $RMS_VALFLD(LONG,                       ACLCTX),        ! ACLCTX
                $RMS_VALFLD(LONG,                       0)              ! ACLSTS

        ) %;


!++
! $XABPRO_INIT
!       Macro to dynamically initialize the XABPRO control block.
!--

KEYWORDMACRO

        $XABPRO_INIT(

                XAB,

                NXT=0,          UIC=<0,0>,      PRO,            MTACC=0,
                PROT_OPT,       PROT_MODE=0,    ACLBUF=0,       ACLSIZ=0,
                ACLCTX=0 )=

        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_PROLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_PRO);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_PROLEN);  ! BLN
!               $RMS_VALFLD_INI(SPARE);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
                $RMS_VALPRO_INI($RMS_PTR,%REMOVE(PRO));                                 ! PRO
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_MTACC),                 MTACC);         ! MTACC
                $RMS_BITFLD_INI($RMS_PTR,(XAB$B_PROT_OPT),      XAB$M_, PROT_OPT);      ! PROT_OPT
```

```
                $RMS_VALUIC_INI($RMS_PTR,%REMOVE(UIC));                         ! UIC
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_PROT_MODE),             PROT_MODE);     ! PROT_MODE
!               $RMS_VALFLD_INI(SPARE);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ACLBUF),                ACLBUF);        ! ACLBUF
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_ACLSIZ),                ACLSIZ);        ! ACLSIZ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_ACLLEN),                0);             ! ACLLEN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ACLCTX),                ACLCTX);        ! ACLCTX
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ACLSTS),                0);             ! ACLSTS
        0 ) %;
!++
! $XABALL_DECL
!       Permits the declaration of the XABALL control block
!       where initialization is not required.
!--

MACRO
        $XABALL_DECL = BLOCK[XAB$C_ALLLEN,BYTE] %;


!++
! $XABALL
!       Macro to allocate and initialize the XABALL control block.
!--

KEYWORDMACRO

        $XABALL(
                NXT=0,          AID=0,          ALN=ANY,        ALQ=0,
                AOP,            BKZ=0,          DEQ=0,
                LOC=0,          RFI=<0,0,0>,    VOL=0)=

        $XABALL_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_ALL),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_ALLLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_BITFLD(BYTE,       XAB$M_,         AOP),           ! AOP
                $RMS_CODFLD(BYTE,       XAB$C_,         ALN),           ! ALN
                $RMS_VALFLD(WORD,                       VOL),           ! VOL
                $RMS_VALFLD(LONG,                       LOC),           ! LOC
                $RMS_VALFLD(LONG,                       ALQ),           ! ALQ
                $RMS_VALFLD(WORD,                       DEQ),           ! DEQ
                $RMS_VALFLD(BYTE,                       BKZ),           ! BKZ
                $RMS_VALFLD(BYTE,                       AID),           ! AID
                $RMS_VALRFI(%REMOVE(RFI)),                              ! RFI
                $RMS_VALFLD(WORD,                       0)              ! SPARE
        ) %;
```

```
!++
! $XABALL_INIT
!
!       Macro to dynamically initialize the XABALL control block.
!--

KEYWORDMACRO

        $XABALL_INIT(

                XAB,

                NXT=0,          AID=0,          ALN,            ALQ=0,
                AOP,            BKZ=0,          DEQ=0,
                LOC=0,          RFI=<0,0,0>,    VOL=0)=

        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_ALLLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_ALL);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_ALLLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
                $RMS_BITFLD_INI($RMS_PTR,(XAB$B_AOP),   XAB$M_,         AOP);           ! AOP
                $RMS_CODFLD_INI($RMS_PTR,(XAB$B_ALN),   XAB$C_,         ALN);           ! ALN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_VOL),                   VOL);           ! VOL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_LOC),                   LOC);           ! LOC
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ALQ),                   ALQ);           ! ALQ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_DEQ),                   DEQ);           ! DEQ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BKZ),                   BKZ);           ! BKZ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_AID),                   AID);           ! AID
                $RMS_VALRFI_INI($RMS_PTR,%REMOVE(RFI));                                 ! RFI
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
          0 ) %;


!++
! $XABFHC_DECL
!
!       Permits the declaration of the XABFHC control block
!       where initialization is not required.
!--

MACRO
        $XABFHC_DECL = BLOCK[XAB$C_FHCLEN,BYTE] %;


!++
! $XABFHC
!
```

```
!       Macro to allocate and initialize the XABFHC control block.
!
!--

KEYWORDMACRO

        $XABFHC(
                NXT=0)=

        $XABFHC_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_FHC),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_FHCLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                REP 9 OF $RMS_VALFLD(LONG,              0)              ! REMAINING FIELDS
        ) %;


!++
! $XABFHC_INIT
!       Macro to dynamically initialize the XABFHC control block.
!--

KEYWORDMACRO

        $XABFHC_INIT(

                XAB,

                NXT=0)=

        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_FHCLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_FHC);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_FHCLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
!               REP 9 OF $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),        0);             ! REMAINING FIELDS
          0 ) %;


!++
! $XABKEY_DECL
!       Permits the declaration of the XABKEY control block where
!       initialization is not required.
!--
```

```
MACRO
        $XABKEY_DECL = BLOCK[XAB$C_KEYLEN,BYTE] %;

!++
! $XABKEY
!
!       Macro to allocate and initialize the XABKEY control block.
!--

KEYWORDMACRO

        $XABKEY(
                DAN=0,          DFL=0,          DTP=STG,        FLG,
                IAN=0,          IFL=0,          KNM=0,          LAN=0,
                NXT=0,          NUL=0,          POS,            POS0=0,
                POS1=0,         POS2=0,         POS3=0,         POS4=0,
                POS5=0,         POS6=0,         POS7=0,         KREF=0,
                SIZ,            SIZ0=0,         SIZ1=0,         SIZ2=0,
                SIZ3=0,         SIZ4=0,         SIZ5=0,         SIZ6=0,
                SIZ7=0,         PROLOG=0)=

        $XABKEY_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_KEY),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$C_KEYLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_VALFLD(BYTE,                       IAN),           ! IAN
                $RMS_VALFLD(BYTE,                       LAN),           ! LAN
                $RMS_VALFLD(BYTE,                       DAN),           ! DAN
                $RMS_VALFLD(BYTE,                       0),             ! LVL
                $RMS_VALFLD(BYTE,                       0),             ! IBS
                $RMS_VALFLD(BYTE,                       0),             ! DBS
                $RMS_VALFLD(LONG,                       0),             ! RVB
                $RMS_BITFLD(BYTE,       XAB$M_,         FLG),           ! FLG
                $RMS_CODFLD(BYTE,       XAB$C_,         DTP),           ! DTP
                $RMS_VALFLD(BYTE,                       0),             ! NSG
                $RMS_VALFLD(BYTE,                       NUL),           ! NUL
                $RMS_VALFLD(BYTE,                       0),             ! TKS
                $RMS_VALFLD(BYTE,                       KREF),          ! REF
                $RMS_VALFLD(WORD,                       0),             ! MRL
                $RMS_VALFLD(WORD,                       IFL),           ! IFL
                $RMS_VALFLD(WORD,                       DFL),           ! DFL
%IF %NULL(POS)
%THEN
                $RMS_VALFLD(WORD,                       POS0),          ! POS0
                $RMS_VALFLD(WORD,                       POS1),          ! POS1
                $RMS_VALFLD(WORD,                       POS2),          ! POS2
                $RMS_VALFLD(WORD,                       POS3),          ! POS3
                $RMS_VALFLD(WORD,                       POS4),          ! POS4
                $RMS_VALFLD(WORD,                       POS5),          ! POS5
                $RMS_VALFLD(WORD,                       POS6),          ! POS6
```

```
                $RMS_VALFLD(WORD,                       POS7),          ! POS7
        %ELSE
                $RMS_8FLD(WORD,                 %REMOVE(POS)),          ! POS
        %FI
        %IF %NULL(SIZ)
        %THEN
                $RMS_VALFLD(BYTE,                       SIZ0),          ! SIZ0
                $RMS_VALFLD(BYTE,                       SIZ1),          ! SIZ1
                $RMS_VALFLD(BYTE,                       SIZ2),          ! SIZ2
                $RMS_VALFLD(BYTE,                       SIZ3),          ! SIZ3
                $RMS_VALFLD(BYTE,                       SIZ4),          ! SIZ4
                $RMS_VALFLD(BYTE,                       SIZ5),          ! SIZ5
                $RMS_VALFLD(BYTE,                       SIZ6),          ! SIZ6
                $RMS_VALFLD(BYTE,                       SIZ7),          ! SIZ7
        %ELSE
                $RMS_8FLD(BYTE,                 %REMOVE(SIZ)),          ! SIZ
        %FI
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       KNM),           ! KNM
                $RMS_VALFLD(LONG,                       0),             ! DVB
                $RMS_VALFLD(BYTE,                       0),             ! TYP0
                $RMS_VALFLD(BYTE,                       0),             ! TYP1
                $RMS_VALFLD(BYTE,                       0),             ! TYP2
                $RMS_VALFLD(BYTE,                       0),             ! TYP3
                $RMS_VALFLD(BYTE,                       0),             ! TYP4
                $RMS_VALFLD(BYTE,                       0),             ! TYP5
                $RMS_VALFLD(BYTE,                       0),             ! TYP6
                $RMS_VALFLD(BYTE,                       0),             ! TYP7
                $RMS_VALFLD(BYTE,                       PROLOG),        ! PROLOG
                $RMS_VALFLD(BYTE,                       0)              ! SPARE

        ) %;

!++
! $XABKEY_INIT
!
!       Macro to dynamically initialize the XABKEY control block.
!--

KEYWORDMACRO

        $XABKEY_INIT(

                XAB,
                DAN=0,          DFL=0,          DTP=STG,        FLG,
                IAN=0,          IFL=0,          KNM=0,          LAN=0,
                NXT=0,          NUL=0,          POS,            POS0=0,
                POS1=0,         POS2=0,         POS3=0,         POS4=0,
                POS5=0,         POS6=0,         POS7=0,         KREF=0,
                SIZ,            SIZ0=0,         SIZ1=0,         SIZ2=0,
                SIZ3=0,         SIZ4=0,         SIZ5=0,         SIZ6=0,
                SIZ7=0,         PROLOG=0)=
```

```
        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_KEYLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),                   XAB$C_KEY);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),                   XAB$C_KEYLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_IAN),                   IAN);           ! IAN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_LAN),                   LAN);           ! LAN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_DAN),                   DAN);           ! DAN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_LVL),                   0);             ! LVL
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_IBS),                   0);             ! IBS
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_DBS),                   0);             ! DBS
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_RVB),                   0);             ! RVB
                $RMS_BITFLD_INI($RMS_PTR,(XAB$B_FLG),   XAB$M_,         FLG);           ! FLG
                $RMS_CODFLD_INI($RMS_PTR,(XAB$B_DTP),   XAB$C_,         DTP);           ! DTP
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_NSG),                   0);             ! NSG
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_NUL),                   NUL);           ! NUL
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_TKS),                   0);             ! TKS
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_REF),                   KREF);          ! REF
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_MRL),                   0);             ! MRL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_IFL),                   IFL);           ! IFL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_DFL),                   DFL);           ! DFL
        %IF %NULL(POS)
        %THEN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS0),                  POS0);          ! POS0
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS1),                  POS1);          ! POS1
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS2),                  POS2);          ! POS2
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS3),                  POS3);          ! POS3
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS4),                  POS4);          ! POS4
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS5),                  POS5);          ! POS5
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS6),                  POS6);          ! POS6
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_POS7),                  POS7);          ! POS7
        %ELSE
                $RMS_8FLD_INI($RMS_PTR, %STRING('XAB$W_POS'),   %REMOVE(POS));          ! POS
        %FI
        %IF %NULL(SIZ)
        %THEN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ0),                  SIZ0);          ! SIZ0

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ1),                  SIZ1);          ! SIZ1
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ2),                  SIZ2);          ! SIZ2
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ3),                  SIZ3);          ! SIZ3
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ4),                  SIZ4);          ! SIZ4
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ5),                  SIZ5);          ! SIZ5
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ6),                  SIZ6);          ! SIZ6
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SIZ7),                  SIZ7);          ! SIZ7
        %ELSE
                $RMS_8FLD_INI($RMS_PTR, %STRING('XAB$B_SIZ'),   %REMOVE(SIZ));          ! SIZ
        %FI
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),                 0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_KNM),                   KNM);           ! KNM
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_DVB),                   0);             ! DVB
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_PROLOG),                PROLOG);        ! PROLOG
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SPARE),                 0);             ! SPARE
```

```
        0 ) %;

!++
! $XABSUM_DECL
!       Permits the declaration of the XABSUM control block where
!       initialization is not required.
!--

MACRO
        $XABSUM_DECL = BLOCK[XAB$C_SUMLEN,BYTE] %;

!++
! $XABSUM
!       Macro to allocate and initialize the XABSUM control block.
!--

KEYWORDMACRO
        $XABSUM(
                NXT=0)=

        $XABSUM_DECL
        INITIAL(
                $RMS_VALFLD(BYTE,       XAB$C_SUM),     ! COD
                $RMS_VALFLD(BYTE,       XAB$K_SUMLEN),  ! BLN
                $RMS_VALFLD(WORD,       0),             ! SPARE
                $RMS_VALFLD(LONG,       NXT),           ! NXT
                $RMS_VALFLD(BYTE,       0),             ! NOA
                $RMS_VALFLD(BYTE,       0),             ! NOK
                $RMS_VALFLD(WORD,       0)              ! PVN
        ) %;

!++
! $XABSUM_INIT
!       Macro to dynamically initialize the XABSUM control block.
!--

KEYWORDMACRO
        $XABSUM_INIT(
                XAB,
                NXT=0)=
```

```
        (BIND $RMS_PTR = XAB;
         CH$FILL(0, XAB$C_SUMLEN, CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),          XAB$C_SUM);      ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),          XAB$C_SUMLEN);   ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),        0);              ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),          NXT);            ! NXT
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_NOA),          NOA);            ! NOA
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$B_NOK),          NOK);            ! NOK
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_PVN),          PVN);            ! PVN
        0 ) %;
```

```
!++
! $XABCXF_DECL
!
!       Permits the declaration of the XABCXF control block where
!       initialization is not required.
!--

MACRO
        $XABCXF_DECL = BLOCK[XAB$C_CXFLEN,BYTE] %;

!++
! $XABCXF
!
!       Macro to allocate and initialize the XABCXF control block.
!--

KEYWORDMACRO

        $XABCXF(
                NXT=0)=

        $XABCXF_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,               XAB$C_CXF),     ! COD
                $RMS_VALFLD(BYTE,               XAB$K_CXFLEN),  ! BLN
                $RMS_VALFLD(WORD,               0),             ! SPARE
                $RMS_VALFLD(LONG,               NXT),           ! NXT
                $RMS_VALFLD(LONG,               0),             ! STS
                $RMS_VALFLD(LONG,               0),             ! STV
                $RMS_VALFLD(LONG,               0),             ! COP
                $RMS_VALFLD(LONG,               0),             ! BKPBITS
                $RMS_VALFLD(WORD,               0),             ! IFI
                $RMS_VALFLD(BYTE,               0),             ! CXFVER
                $RMS_VALFLD(BYTE,               0),             ! SPARE
                $RMS_VALFLD(LONG,               0),             ! SPARE
!               Above in common with XABCXR
                $RMS_VALFLD(WORD,               0),             ! DEQ
                $RMS_VALFLD(BYTE,               0),             ! FAC
                $RMS_VALFLD(BYTE,               0),             ! SHR
                $RMS_VALFLD(WORD,               0),             ! RTDEQ
                $RMS_VALFLD(BYTE,               0),             ! SPARE
                $RMS_VALFLD(BYTE,               0),             ! ORGCASE
                $RMS_VALFLD(WORD,               0),             ! GBC
                $RMS_VALFLD(BYTE,               0),             ! RTV
                $RMS_VALFLD(BYTE,               0),             ! SPARE
                $RMS_VALFLD(LONG,               0),             ! SPARE
                $RMS_VALFLD(LONG,               0),             ! SPARE
                $RMS_VALFLD(LONG,               0),             ! SPARE
                $RMS_VALFLD(LONG,               0)              ! SPARE
        ) %;
```

```
!++
! $XABCXF_INIT
!
!       Macro to dynamically initialize the XABCXF control block.
!
!--

KEYWORDMACRO

        $XABCXF_INIT(

                XAB,

                NXT=0)=

        (BIND $RMS_PTR = XAB;
         CH$FILL(0, XAB$C_CXFLEN, CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),           XAB$C_CXF);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),           XAB$C_CXFLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),           NXT);           ! NXT
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_STS),           STS);           ! STS
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_STV),           STV);           ! STV
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_COP),           COP);           ! COP
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$L_BKPBITS),       BKPBITS);       ! BKPBITS
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_IFI),           IFI);           ! IFI
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXFVER),        0);             ! CXFVER
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0),             ! SPARE
                All of above in common with XABCXR
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_DEQ),           DEQ);           ! DEQ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_FAC),           FAC);           ! FAC
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SHR),           SHR);           ! SHR
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_RTDEQ),         RTDEQ);         ! RTDEQ
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_ORGCASE),       ORGCASE);       ! ORGCASE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXFGBC),        0);             ! GBC
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXFRTV),        0);             ! RETRIEVAL WINDOW
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);             ! SPARE
        0 ) %;
!++
! $XABCXR_DECL
!
!       Permits the declaration of the XABCXR control block where
!       initialization is not required.
!--
```

```
MACRO
        $XABCXR_DECL = BLOCK[XAB$C_CXRLEN,BYTE] %;

!++
!
! $XABCXR
!
!       Macro to allocate and initialize the XABCXR control block.
!
!--

KEYWORDMACRO

        $XABCXR(
                NXT=0,CXRBUF=0,CXFBFZ=0)=

        $XABCXR_DECL
        INITIAL(

                $RMS_VALFLD(BYTE,                       XAB$C_CXR),     ! COD
                $RMS_VALFLD(BYTE,                       XAB$K_CXRLEN),  ! BLN
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       NXT),           ! NXT
                $RMS_VALFLD(LONG,                       0),             ! STS
                $RMS_VALFLD(LONG,                       0),             ! STV
                $RMS_VALFLD(LONG,                       0),             ! COP
                $RMS_VALFLD(LONG,                       0),             ! BKPBITS
                $RMS_VALFLD(WORD,                       0),             ! ISI
                $RMS_VALFLD(BYTE,                       0),             ! CXRVER
                $RMS_VALFLD(BYTE,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       0),             ! SPARE
!               Above in common with XABCXF
                $RMS_VALFLD(BYTE,                       0),             ! CXRMBF
                $RMS_VALFLD(BYTE,                       0),             ! CXRMBC
                $RMS_VALFLD(WORD,                       CXRBFZ),        ! CXRBFZ
                $RMS_VALFLD(LONG,                       0),             ! CXRVBN
                $RMS_VALFLD(LONG,                       0),             ! CXROFF
                $RMS_VALFLD(LONG,                       0),             ! CXRPOS0
                $RMS_VALFLD(WORD,                       0),             ! CXRPOS4
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       0),             ! CXRCUR0
                $RMS_VALFLD(WORD,                       0),             ! CXRCUR4
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       0),             ! CXRSID0
                $RMS_VALFLD(WORD,                       0),             ! CXRSID4
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(WORD,                       0),             ! CXRCNT
                $RMS_VALFLD(BYTE,                       0),             ! CXRKRF
                $RMS_VALFLD(BYTE,                       0),             ! CXRKLEN
                $RMS_VALFLD(LONG,                       CXRBUF),        ! CXRBUF
                $RMS_VALFLD(LONG,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       0)              ! SPARE
        ) %;

!++
```

```
!
! $XABCXR_INIT
!
!       Macro to dynamically initialize the XABCXR control block.
!
!--

KEYWORDMACRO

        $XABCXR_INIT(

                XAB,

                NXT=0,CXRBUF=0,CXRBFZ=0)=

        (BIND $RMS_PTR = XAB;
         CH$FILL(0, XAB$C_CXRLEN, CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),           XAB$C_CXR);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),           XAB$C_CXRLEN);  ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),           NXT);           ! NXT
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_STS),           STS);           ! STS
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_STV),           STV);           ! STV
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_COP),           COP);           ! COP
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_BKPBITS),       BKPBITS);       ! BKPBITS
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_ISI),           ISI);           ! ISI
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXRVER),        0);             ! CXRVER
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_SPARE),         0);             ! SPARE
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);             ! SPARE
                All of above in common with XABCXF
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXRMBF),        0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXRMBC),        0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXRBFZ),        CXRBFZ);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXRVBN),        CXRVBN);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXROFF),        CXROFF);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXRPOS0),       CXRPOS0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXRPOS4),       CXRPOS4);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),         0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXRCUR0),       CXRCUR0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXRCUR4),       CXRCUR4);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),         0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXRSID0),       CXRSID0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXRSID4),       CXRSID4);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_SPARE),         0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$W_CXRCNT),        0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXRKRF),        0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_CXRKLEN),       0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_CXRBUF),        CXRBUF);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_SPARE),         0);
        0 ) %;
```

```
!++
! $XABJNL_DECL
!
!       Permits the declaration of the XABJNL control bock where
!       initialization is not required
!--

MACRO
        $XABJNL_DECL = BLOCK[XAB$C_JNLLEN,BYTE] %;

!++
! $XABJNL
!
!       Macro to allocate and initialize the XABJNL control block
!--

KEYWORDMACRO

        $XABJNL(
                JOP,    BIS=0,  AIS=0,  ATS=0,
                        BIA=0,  AIA=0,  ATA=0,
                        BIN,    AIN,    ATN,    NXT= 0 ) =
        $XABJNL_DECL
        INITIAL(
                $RMS_VALFLD(BYTE,                       XAB$C_JNL),     !COD
                $RMS_VALFLD(BYTE,                       XAB$C_JNLLEN),  !BLN
                $RMS_VALFLD(WORD,                       0),             !SPARE
                $RMS_VALFLD(LONG,                       NXT),           !NXT
                $RMS_BITFLD(WORD,       XABSM_,         JOP),           !JOP
                $RMS_VALFLD(WORD,                       0),             !SPARE

%IF %NULL(BIN) %THEN
                $RMS_VALFLD(BYTE,                       BIS),           !BIS
%ELSE
                $RMS_VALFLD(BYTE,               %CHARCOUNT(BIN)),       !BIS
%FI
                $RMS_VALFLD(BYTE,                       0),             !BIL
                $RMS_VALFLD(WORD,                       0),             !SPARE
%IF %NULL(BIN) %THEN
                $RMS_VALFLD(LONG,                       BIA),           !BIA
%ELSE
                $RMS_VALFLD(LONG,               UPLIT BYTE(BIN)),       !BIA
%FI

%IF %NULL(AIN) %THEN
                $RMS_VALFLD(BYTE,                       AIS),           !AIS
%ELSE
                $RMS_VALFLD(BYTE,               %CHARCOUNT(AIN)),       !AIS
%FI
                $RMS_VALFLD(BYTE,                       0),             !AIL
                $RMS_VALFLD(WORD,                       0),             !SPARE
%IF %NULL(AIN) %THEN
```

```
                $RMS_VALFLD(LONG,                       AIA),           !AIA
%ELSE
                $RMS_VALFLD(LONG,               UPLIT BYTE(AIN)),       !AIA
%FI

%IF %NULL(ATN) %THEN
                $RMS_VALFLD(BYTE,                       ATS),           !ATS
%ELSE
                $RMS_VALFLD(BYTE,               %CHARCOUNT(ATN)),       !ATS
%FI
                $RMS_VALFLD(BYTE,                       0),             !ATL
                $RMS_VALFLD(WORD,                       0),             !SPARE
%IF %NULL(ATN) %THEN
                $RMS_VALFLD(LONG,                       ATA)            !ATA
%ELSE
                $RMS_VALFLD(LONG,               UPLIT BYTE(ATN))        !ATA
%FI
        ) %;

!++
! $XABJNL_INIT
!
!       Macro to dynamically initialize the XABJNL control block
!--

KEYWORDMACRO
        $XABJNL_INIT(
                XAB,
                NXT = 0,        JOP,    BIS=0,  AIS=0,  ATS=0,
                                        BIA=0,  AIA=0,  ATA=0,
                                        BIN,    AIN,    ATN) =
        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_JNLLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),   XAB$C_JNL);     !COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),   XAB$C_JNLLEN);  !BLN
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),   NXT);           !NXT

                $RMS_BITFLD_INI($RMS_PTR,(XAB$W_JOP),   JOP);           !JOP

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BIS),
                        %IF %NULL(BIN) %THEN BIS %ELSE %CHARCOUNT(BIN) %FI); !BIS
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BIL),   0);             !BIL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_BIA),
                        %IF %NULL(BIN) %THEN BIA %ELSE UPLIT BYTE(BIN) %FI); !BIA

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_AIS),
                        %IF %NULL(AIN) %THEN AIS %ELSE %CHARCOUNT(AIN) %FI); !AIS
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_AIL),   0);             ! AIL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_AIA),
                        %IF %NULL(AIN) %THEN AIA %ELSE UPLIT BYTE(AIN) %FI); !AIA

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_ATS),
                        %IF %NULL(ATN) %THEN ATS %ELSE %CHARCOUNT(ATN) %FI); !ATS
```

```
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_ATL),   0);             ! ATL
                $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ATA),
                        %IF %NULL(ATN) %THEN ATA %ELSE UPLIT BYTE(ATN) %FI); !ATA
        0 ) %;
```

```
!++
! $XABTRM_DECL
!
!       Permits the declaration of the XABTRM control block
!       where initialization is not required.
!--

MACRO
        $XABTRM_DECL = BLOCK[XAB$C_TRMLEN,BYTE] %;

!++
! $XABTRM
!
!       Macro to allocate and initialize the XABTRM control block.
!--

KEYWORDMACRO

        $XABTRM(
                NXT=0, ITMLST=0, ITMLST_LEN=0
                ) =

        $XABTRM_DECL
        INITIAL(
                $RMS_VALFLD(BYTE,       XAB$C_TRM),     ! COD
                $RMS_VALFLD(BYTE,       XAB$C_TRMLEN),  ! BLN
                $RMS_VALFLD(WORD,       0),             ! SPARE
                $RMS_VALFLD(LONG,       NXT),           ! NXT
                $RMS_VALFLD(LONG,       0),             ! ITMLST
                $RMS_VALFLD(WORD,       0)              ! ITMLST_LEN
        ) %;

!++
! $XABTRM_INIT
!
!       Macro to dynamically initialize the XABTRM control block.
!--

KEYWORDMACRO

        $XABTRM_INIT(
                XAB, NXT=0, ITMLST=0, ITMLST_LEN=0
                ) =

        ( BIND $RMS_PTR = XAB;
          CH$FILL(0,XAB$C_TRMLEN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_COD),           XAB$C_TRM);     ! COD
                $RMS_VALFLD_INI($RMS_PTR,(XAB$B_BLN),           XAB$C_TRMLEN);  ! BLN
```

```
        $RMS_VALFLD_INI($RMS_PTR,(XAB$L_NXT),                   NXT);           ! NXT
        $RMS_VALFLD_INI($RMS_PTR,(XAB$L_ITMLST),                ITMLST);        ! ITMLST
        $RMS_VALFLD_INI($RMS_PTR,(XAB$W_ITMLST_LEN),            ITMLST_LEN);    ! ITMLST_LEN
    0 ) %;
```

```
!++
! $RAB_DECL and $RAB_DECL_ASYN
!
!       Permit the declaration of the RAB control block
!       where initialization is not required.
!
!       Note that a RAB intended for ASYNCHRONOUS I/O
!       differs from one intended for SYNCHRONOUS I/O
!       in some RMS implementations.
!       USE:
!               LOCAL RAB_01 : $RAB_DECL_ASYN;  ! for ASYNCHRONOUS I/O
!               LOCAL RAB_02 : $RAB_DECL;       ! for SYNCHRONOUS I/O
!               MAP RAB_03 : $RAB_DECL;
!               EXTERNAL RAB_04 : $RAB_DECL_ASYN;
!
!--

MACRO
        $RAB_DECL =
                BLOCK[RAB$C_BLN,BYTE]  %,
        $RAB_DECL_ASYN =
                BLOCK[RAB$C_BLN,BYTE]  %;
!++
! $RAB
!
!       Macro to allocate and initialize the RAB control block.
!       To indicate that the RAB may be used for ASYNCHRONOUS I/O,
!       indicate a value for the ASYN keywordparameter as in
!       the example:
!
!       OWN RAB_01 : $RAB ( ASYN=YES , ....);    ! ASYNCHRONOUS
!--

KEYWORDMACRO
        $RAB(
                RAC=SEQ,        ROP,            UBF=0,          USZ=0,
                RBF=0,          RSZ=0,          BKT=0,          KBF=0,
                PBF=0,          KSZ=0,          PSZ=0,          RHB=0,
                FAB=0,          MBF=0,          MBC=0,          TMO=0,
                CTX=0,          KRF=0,          ASYN=NO,        XAB=0)=

        %IF %IDENTICAL(%STRING(ASYN),'YES')
        %THEN $RAB_DECL_ASYN
        %ELSE $RAB_DECL
        %FI
        INITIAL(

                $RMS_VALFLD(BYTE,                       RAB$C_BID),     ! BID
                $RMS_VALFLD(BYTE,                       RAB$C_BLN),     ! BLN
                $RMS_VALFLD(WORD,                       0),             ! ISI
                $RMS_BITFLD(LONG,       RAB$M_,         ROP),           ! ROP
```

```
                $RMS_VALFLD(LONG,                       0),             ! STS
                $RMS_VALFLD(LONG,                       0),             ! STV
                REP 3 OF $RMS_VALFLD(WORD,              0),             ! RFA
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_VALFLD(LONG,                       CTX),           ! CTX
                $RMS_VALFLD(WORD,                       0),             ! SPARE
                $RMS_CODFLD(BYTE,       RAB$C_,         RAC),           ! RAC
                $RMS_VALFLD(BYTE,                       TMO),           ! TMO
                $RMS_VALFLD(WORD,                       USZ),           ! USZ
                $RMS_VALFLD(WORD,                       RSZ),           ! RSZ
                $RMS_VALFLD(LONG,                       UBF),           ! UBF
                $RMS_VALFLD(LONG,                       RBF),           ! RBF
                $RMS_VALFLD(LONG,                       RHB),           ! RHB
                $RMS_VALFLD(LONG,                       (KBF)+(PBF)),   ! KBF and PBF
                $RMS_VALFLD(BYTE,                       (KSZ)+(PSZ)),   ! KSZ and PSZ
                $RMS_VALFLD(BYTE,                       KRF),           ! KRF
                $RMS_VALFLD(BYTE,                       MBF),           ! MBF
                $RMS_VALFLD(BYTE,                       MBC),           ! MBC
                $RMS_VALFLD(LONG,                       BKT),           ! BKT
                $RMS_VALFLD(LONG,                       FAB),           ! FAB
                $RMS_VALFLD(LONG,                       XAB)            ! XAB
        ) %;

!++
! $RAB_INIT
!
!       Macro to dynamically initialize the RAB control block.
!--

KEYWORDMACRO

        $RAB_INIT(

                RAB,

                RAC,            ROP,            UBF=0,          USZ=0,
                RBF=0,          RSZ=0,          BKT=0,          KBF=0,
                PBF=0,          KSZ=0,          PSZ=0,          RHB=0,
                FAB=0,          MBF=0,          MBC=0,          TMO=0,
                CTX=0,          KRF=0,          ASYN=NO,        XAB=0)=

        ( BIND $RMS_PTR = RAB;
          CH$FILL(0,RAB$C_BLN,CH$PTR($RMS_PTR));

                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_BID),                   RAB$C_BID);     ! BID
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_BLN),                   RAB$C_BLN);     ! BLN
!               $RMS_VALFLD_INI($RMS_PTR,(RAB$W_ISI),                   0);             ! ISI
                $RMS_BITFLD_INI($RMS_PTR,(RAB$L_ROP),   RAB$M_,         ROP);           ! ROP
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_STS),                   0);             ! STS
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_STV),                   0);             ! STV
                REP 3 OF $RMS_VALFLD_INI($RMS_PTR,(RAB$W_RFA),          0);             ! RFA
                REP 1 OF $RMS_VALFLD_INI($RMS_PTR,(RAB$L_SPARE),        0);             ! SPARE
```

```
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_CTX),                   CTX);           ! CTX
                REP T OF $RMS_VALFLD_INI($RMS_PTR,(RAB$L_SPARE),        0);             ! SPARE
                $RMS_CODFLD_INI($RMS_PTR,(RAB$B_RAC),   RAB$C_,         RAC);           ! RAC
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_TMO),                   TMO);           ! TMO
                $RMS_VALFLD_INI($RMS_PTR,(RAB$W_USZ),                   USZ);           ! USZ
                $RMS_VALFLD_INI($RMS_PTR,(RAB$W_RSZ),                   RSZ);           ! RSZ
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_UBF),                   UBF);           ! UBF
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_RBF),                   RBF);           ! RBF
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_RHB),                   RHB);           ! RHB
                %IF NOT %IDENTICAL(KBF,0)
                OR  NOT %IDENTICAL(PBF,0) %THEN
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_KBF),                   (KBF)+(PBF));   ! KBF and PBF
                                        %FI
                %IF NOT %IDENTICAL(KSZ,0)
                OR  NOT %IDENTICAL(PSZ,0) %THEN
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_KSZ),                   (KSZ)+(PSZ));   ! KSZ and PSZ
                                        %FI
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_KRF),                   KRF);           ! KRF
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_MBF),                   MBF);           ! MBF
                $RMS_VALFLD_INI($RMS_PTR,(RAB$B_MBC),                   MBC);           ! MBC
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_BKT),                   BKT);           ! BKT
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_FAB),                   FAB);           ! FAB
                $RMS_VALFLD_INI($RMS_PTR,(RAB$L_XAB),                   XAB);           ! XAB
        0 ) %;
```

```
!++
! RMS-11 Compatibility Macros
!
!       $RMS_INIT
!       $RMS_INITIF
!       $RMS_POOL
!       $RMS_ORG
!
!       All these macros, necessary only in RMS-11, are
!       coded as no-ops here, being executable
!       expressions with value 0.
!--

MACRO
        $RMS_INIT(XX) =
                BEGIN
                0
                END %,

        $RMS_INITIF(XX) =
                BEGIN
                0
                END %,

        $RMS_POOL(XX) =
                BEGIN
                0
                END %,

        $RMS_ORG(XX) =
                BEGIN
                0
                END %,

!++
! $RMS_OKSTATUS
! $RMS_OK
! $RMS_SUC
!
!       Macros to examine status (STS) values.
!
!       $RMS_OK(sts_val) examines the value sts_val and
!       yields 1 if this is a good status value, 0 otherwise.
!
!       $RMS_SUC(sts_val) examines the value sts_val and
!       yields 1 if this equals RMS$_SUC.
!
!       $RMS_OKSTATUS(addr) extracts the STS field from
!       the RAB or FAB at addr ard yields
!       1 if this is a good status value, 0 otherwise.
!--
```

```
        $RMS_OKSTATUS(XX) =
                (.BLOCK[(XX),FAB$L_STS;0,BYTE] AND 1) %,

        $RMS_OK(VAL)=
                ((VAL) AND 1) %,

        $RMS_SUC(VAL)=
                (EXTERNAL LITERAL RMS$_SUC;(VAL) EQL RMS$_SUC) %;

!++
! FUNCTIONAL CALLS TO RMS-32
!
        EXAMPLES OF USE:

        status = $RMS_OPEN(FAB=FAB_01);
        status = $RMS_GET(SUC=SUC_01,RAB=RAB_03);


!--

KEYWORDMACRO
        $RMS_CLOSE(FAB,ERR,SUC)=
            $RMS_CALL(SYS$CLOSE,FAB,ERR,SUC) %,

        $RMS_CONNECT(RAB,ERR,SUC)=
            $RMS_CALL(SYS$CONNECT,RAB,ERR,SUC) %,

        $RMS_CREATE(FAB,ERR,SUC)=
            $RMS_CALL(SYS$CREATE,FAB,ERR,SUC) %,

        $RMS_DELETE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$DELETE,RAB,ERR,SUC) %,

        $RMS_DISCONNECT(RAB,ERR,SUC)=
            $RMS_CALL(SYS$DISCONNECT,RAB,ERR,SUC) %,

        $RMS_DISPLAY(FAB,ERR,SUC)=
            $RMS_CALL(SYS$DISPLAY,FAB,ERR,SUC) %,

        $RMS_ERASE(FAB,ERR,SUC)=
            $RMS_CALL(SYS$ERASE,FAB,ERR,SUC) %,

        $RMS_EXTEND(FAB,ERR,SUC)=
            $RMS_CALL(SYS$EXTEND,FAB,ERR,SUC) %,

        $RMS_FIND(RAB,ERR,SUC)=
            $RMS_CALL(SYS$FIND,RAB,ERR,SUC) %,

        $RMS_FLUSH(RAB,ERR,SUC)=
            $RMS_CALL(SYS$FLUSH,RAB,ERR,SUC) %,

        $RMS_FREE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$FREE,RAB,ERR,SUC) %,

        $RMS_GET(RAB,ERR,SUC)=
```

```
            $RMS_CALL(SYS$GET,RAB,ERR,SUC) %,

        $RMS_MODIFY(FAB,ERR,SUC)=
            $RMS_CALL(SYS$MODIFY,FAB,ERR,SUC) %,

        $RMS_NXTVOL(RAB,ERR,SUC)=
            $RMS_CALL(SYS$NXTVOL,RAB,ERP,SUC) %,

        $RMS_OPEN(FAB,ERR,SUC)=
            $RMS_CALL(SYS$OPEN,FAB,ERR,SUC) %,

        $RMS_PUT(RAB,ERR,SUC)=
            $RMS_CALL(SYS$PUT,RAB,ERR,SUC) %,

        $RMS_PARSE(FAB,ERR,SUC)=
            $RMS_CALL(SYS$PARSE,FAB,ERR,SUC) %,

        $RMS_READ(RAB,ERR,SUC)=
            $RMS_CALL(SYS$READ,RAB,ERR,SUC) %,

        $RMS_RELEASE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$RELEASE,RAB,ERR,SUC) %,

        $RMS_REWIND(RAB,ERR,SUC)=
            $RMS_CALL(SYS$REWIND,RAB,ERR,SUC) %,

        $RMS_SEARCH(FAB,ERR,SUC)=
            $RMS_CALL(SYS$SEARCH,FAB,ERR,SUC) %,

        $RMS_ENTER(FAB,ERR,SUC)=
            $RMS_CALL(SYS$ENTER,FAB,ERR,SUC) %,

        $RMS_REMOVE(FAB,ERR,SUC)=
            $RMS_CALL(SYS$REMOVE,FAB,ERR,SUC) %,

        $RMS_RENAME(OLDFAB,ERR,SUC,NEWFAB)=
            $RMS_CALL(SYS$RENAME,OLDFAB,ERR,SUC,NEWFAB) %,

        $RMS_SPACE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$SPACE,RAB,ERR,SUC) %,

        $RMS_TRUNCATE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$TRUNCATE,RAB,ERR,SUC) %,

        $RMS_UPDATE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$UPDATE,RAB,ERR,SUC) %,

        $RMS_WAIT(RAB,ERR,SUC)=
            $RMS_CALL(SYS$WAIT,RAB,ERR,SUC) %,

        $RMS_WRITE(RAB,ERR,SUC)=
            $RMS_CALL(SYS$WRITE,RAB,ERR,SUC) %;

!** The rest of these calls are now defined in <VMSLIB.SRC>STARLET.SDL

!**     $CLOSE(FAB,ERR,SUC)=
```

```
!**         $RMS_CALL(SYS$CLOSE,FAB,ERR,SUC) %,
!**
!**     $CONNECT(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$CONNECT,RAB,ERR,SUC) %,
!**
!**     $CREATE(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$CREATE,FAB,ERR,SUC) %,
!**
!**     $DELETE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$DELETE,RAB,ERR,SUC) %,
!**
!**     $DISCONNECT(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$DISCONNECT,RAB,ERR,SUC) %,
!**
!**     $DISPLAY(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$DISPLAY,FAB,ERR,SUC) %,
!**
!**     $ERASE(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$ERASE,FAB,ERR,SUC) %,
!**
!**     $EXTEND(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$EXTEND,FAB,ERR,SUC) %,
!**
!**     $FIND(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$FIND,RAB,ERR,SUC) %,
!**
!**     $FLUSH(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$FLUSH,RAB,ERR,SUC) %,
!**
!**     $FREE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$FREE,RAB,ERR,SUC) %,
!**
!**     $GET(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$GET,RAB,ERR,SUC) %,
!**
!**     $MODIFY(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$MODIFY,FAB,ERR,SUC) %,
!**
!**     $NXTVOL(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$NXTVOL,RAB,ERR,SUC) %,
!**
!**     $OPEN(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$OPEN,FAB,ERR,SUC) %,
!**
!**     $PUT(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$PUT,RAB,ERR,SUC) %,
!**
!**     $PARSE(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$PARSE,FAB,ERR,SUC) %,
!**
!**     $READ(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$READ,RAB,ERR,SUC) %,
!**
!**     $RELEASE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$RELEASE,RAB,ERR,SUC) %,
!**
```

```
!**     $REWIND(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$REWIND,RAB,ERR,SUC) %,
!**
!**     $SEARCH(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$SEARCH,FAB,ERR,SUC) %,
!**
!**     $ENTER(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$ENTER,FAB,ERR,SUC) %,
!**
!**     $REMOVE(FAB,ERR,SUC)=
!**         $RMS_CALL(SYS$REMOVE,FAB,ERR,SUC) %,
!**
!**     $RENAME(OLDFAB,ERR,SUC,NEWFAB)=
!**         $RMS_CALL(SYS$RENAME,OLDFAB,ERR,SUC,NEWFAB) %,
!**
!**     $SPACE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$SPACE,RAB,ERR,SUC) %,
!**
!**     $TRUNCATE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$TRUNCATE,RAB,ERR,SUC) %,
!**
!**     $UPDATE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$UPDATE,RAB,ERR,SUC) %,
!**
!**     $WAIT(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$WAIT,RAB,ERR,SUC) %,
!**
!**     $WRITE(RAB,ERR,SUC)=
!**         $RMS_CALL(SYS$WRITE,RAB,ERR,SUC) %;
!**
!** End of macros defined in STARLET.SDL
!*********************************************************

!**
!
! $RMS_CALL
!
!       Internal macro.  Processes the functional calls
!       by counting the parameters, putting them in
!       the right order, and calling RMS-32 as required.
!
!--

MACRO
    $ITR_RMS_CALL[ARG] =
        %IF %NULL(ARG)
        %THEN
            0
        %ELSE
            ARG
        %FI %,

    $RMS_CALL(ROUT,BLK) =
        BEGIN
        EXTERNAL ROUTINE ROUT: BLISS ADDRESSING_MODE(GENERAL);
        ROUT(
```

```
            %IF NOT %NULL( %REMAINING)
            %THEN
                $ITR_RMS_CALL(BLK, %REMAINING)
            %ELSE
                BLK
            %FI
            )
        END %;

!++
! FIELD DEFINITIONS, NAMED BIT MASKS, AND NAMED VALUES
!
!       Produced automatically by the SDL processor
!       acting on the input file RMSUSR.SDL
!--
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMSFILSTR
SDL

RMSMAC
REQ

RMSINTSTR
SDL

RMSUSR
SDL

NWADEF
MDL

RMSFWADEF
SDL

RMSSHR
SDL

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMSCALLS
MAR

RMSIDXLNK
R32

RMS
LST

RMS32MAC
MAR

RMSMSGMAC
MAR

UTLDEF
R32

RMSIDXMAC
R32

UTLDEFUND
R32

RMSINTDEF
LST

RMSIDXDEF
R32

NT@MACROS
MAR