

_5:

Syr

NT
NT
NT
NT
NT

NT
NT
NT
NT

NT
NT
NT
NT

NT
NT
NT
NT

NT
NT
NT
NT

NT
NT
NT
NT
NT

NT
NT
NT
NT
NT
PI

```

RRRRRRRRRRRR    MMM        MMM    SSSSSSSSSSSSS
RRRRRRRRRRRR    MMM        MMM    SSSSSSSSSSSSS
RRRRRRRRRRRR    MMM        MMM    SSSSSSSSSSSSS
RRR              RRR  MMMMMM  MMMMMM  SSS
RRR              RRR  MMMMMM  MMMMMM  SSS
RRR              RRR  MMMMMM  MMMMMM  SSS
RRR              RRR  MMM  MMM  MMM  SSS
RRR              RRR  MMM  MMM  MMM  SSS
RRR              RRR  MMM  MMM  MMM  SSS
RRRRRRRRRRRR    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    SSS
RRR              RRR  MMM        MMM  SSSSSSSSSSSSS
RRR              RRR  MMM        MMM  SSSSSSSSSSSSS
RRR              RRR  MMM        MMM  SSSSSSSSSSSSS

```


\$BEGIN,000,015

```
*****
* 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: RMS32

ABSTRACT:

RMS-32 user control block allocation and initialization macros.
 Note that changes made to any of the structure definitions must
 be reflected also in the file RMSMAC.REQ which contains the
 BLISS macros.

ENVIRONMENT:

USER PROGRAMS RUNNING VAX/VMS

AUTHOR: L F LAVERDURE,

CREATION DATE: 7-NOV-77

MODIFIED BY:

V03-015 RAS0325 Ron Schaefer 11-Jul-1984
 Convert the longword alignment check in the \$xxx macros,
 to a informational message since alignment is not required and
 it potentially screws up programs if any data buffer
 above (such as NAMSC_MAXRSS) is not longword sized.
 Unfortunately, MACRO-32 doesn't have a ".INFO" directive
 so the message is hand-crafted using the ".PRINT" directive.

V03-014 DGB0036 Donald G. Blair 26-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_ALSTS

V03-013 DAS0004 David Solomon 28-feb-1984

Add support for FAB\$V_LNM_MODE, FAB\$V_CHAN_MODE,
FAB\$V_FILE_MODE, and XAB\$B_PROF_MODE.

V03-012	RAS0169	Ron Schaefer	12-Jul-1983
	Add NOP parameter, delete \$XABACE macros and fix GBC parameter bugs in \$FAB macros.		
V03-011	LJA0065	Laurie J. Anderson	01-Mar-1983
	Add initialization of CXRBFZ in XABCXR.		
V03-010	JWH0190	Jeffrey W. Horn	21-Feb-1983
	Add \$XABACE macros for the Access Control Entry ACE.		
V03-009	DAS0002	David Solomon	09-Feb-1983
	Add \$XABTRM macros for the terminal XAB.		
V03-008	LJA0052	Laurie J. Anderson	11-Jan-1983
	Fix LJA0046 - add the missing .ENDC		
V03-007	LJA0046	Laurie J. Anderson	21-Dec-1982
	Add quad word key data types.		
V03-006	RAS0104	Ron Schaefer	23-Nov-1982
	Correct \$STYPE macro for registers AP, FP and SP.		
V03-005	LJA0035	Laurie J. Anderson	27-Oct-1982
	Allow initialization of the key buffer address in XABCXR		
V03-004	RAS0100	Ron Schaefer	14-Oct-1982
	Add the \$XABSUM_STORE macro that had never previously existed.		
V03-003	JWH0001	Jeffrey W. Horn	7-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. Fixed the COMPAT and STRUCT XAB fields which were still in.		
V02-012	RAS0061	Ron Schaefer	15-Jan-1982
	Add support for FAB\$W_GBC.		
V02-011	KBT0002	Keith B Thompson	8-Jan-1982
	Remove COMP, change STRUCT to PROLOG and add MTACC		
V02-010	RAS0048	Ron Schaefer	19-Nov-1981
	Change the \$xxx assembly-time initialization macros from using .LONG to using .ADDRESS when initializing longword fields in the control blocks to allow these macros to be used in a PIC shareable image.		
V02-009	CDS0001	C Saether	3-Nov-1981

Change name of xab field to "struct" in key, xab.
Change declaration of "struct" and "compat" within
key xab to use symbolic values.

V02-008 PSK0001 Paulina S Knibbe 22-Jun-1981
Add support for the long XABKEYs

V02-007 SPR38242 Ron Schaefer 16-Jun-1981
Fix the run-time initialization macros that operate on
symbolic bit values and constants to work correctly if
the user has specified .DEFAULT DISPLACEMENT WORD/BYTE.

V02-006 MCN0007 Maria del C. Nasr 12-May-1981
Use old symbol for new length of backup date and time XAB.

V02-005 SPR35529 P Lieberwirth 12-Feb-1981 13:15
Fix bug in \$\$TYPE macro that led to bad code when user
specified displacement mode with displacement=0. This
bug occurred in \$xxx_STORE macros.

V004 MCN0004 Maria del C. Nasr 17-Dec-1980
Change length of \$XABDAT block to include backup date
and time.

V003 HJ0003 Herb Jacobs 8-Jul-80
Simplify \$FAB,\$RAB,\$NAM for assembly performance now
that they are stable.

V002 RAS0001 R SCHAEFER 06-Sep-79 14:28
Correct spurious error from \$XABKEY for FLG=<CHG>

```

SFAB MACRO TO INITIALIZE A FAB

```

```

.MACRO SFAB      FAC=,SHR=,FNA=0,FNS=0,DNA=0,-
                 DNS=0,FNM=,DNM=,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,BKS=0,CTX=0,BSZ=8,GBC=0,-
                 LNM_MODE=0,CHAN_MODE=0,FILE_MODE=0
SFABDEF
$$R_TABINIT     FABSC BID, FABSC_BLN
$$R_VBFSET      FAB,<FOP>
.= $$ .TAB+FAB$L_FOP
.ADDRESS        $$ .TMP
.= $$ .TAB+FAB$L_ALQ
.ADDRESS        ALQ
.WORD           DEQ
$$R_VBFSET      FAB,<FAC>
.BYTE          $$ .TMP
$$R_VBFSET      FAB,<SHR>
.BYTE          $$ .TMP
.ADDRESS        CTX
.BYTE          RTV
.IF DF FABSC 'ORG
.BYTE          FABSC_'ORG
.IFF
.BYTE
.ERROR          ; UNDEFINED VALUE FOR FIELD : ORG;
.ENDC
$$R_VBFSET      FAB,<RAT>
.BYTE          $$ .TMP
.IF DF FABSC 'RFM
.BYTE          FABSC_'RFM
.IFF
.BYTE
.ERROR          ; UNDEFINED VALUE FOR FIELD : RFM;
.ENDC
.ADDRESS        JNL
.ADDRESS        XAB
.ADDRESS        NAM
.ADDRESS        FNA
.ADDRESS        DNA
.BYTE          FNS
.BYTE          DNS
.WORD          MRS
.ADDRESS        MRN
.WORD          BLS
.BYTE          BKS
.BYTE          FSZ
.= $$ .TAB+FAB$W_GBC
.WORD          GBC
.BYTE          <<LNM_MODE@FAB$V_LNM_MODE> + <CHAN_MODE@FAB$V_CHAN_MODE> + -
               <<FILE_MODE@FAB$V_FILE_MODE>>
.IIF NE BSZ-8, .ERROR ; INVALID BYTE SIZE;
.IF NB <FNM>
.SAVE

```



```

:
: $RAB MACRO TO INITIALIZE A RAB
:
.MACRO $RAB      RAC=SEQ,ROP=,UBF=0,USZ=0,-
                 RBF=0,RSZ=0,BKT=0,KBF=0,PBF=-
                 KSZ=0,PSZ,KRF=0,RHB=0,FAB=0,MBF=0,-
                 MBC=0,TMO=0,CTX=0,KRF=0,XAB=0

$RABDEF
$$R_TABINIT     RAB$C_BID,RAB$C_BLN
$$R VBFSET RAB,<ROP>
.= $$ .TAB+RAB$$_ROP
.ADDRESS        $$ .TMP
.= $$ .TAB+RAB$$_CTX
.ADDRESS        CTX
.= $$ .TAB+RAB$$_RAC
.IF DF RAB$$_RAC
.BYTE          RAB$$_RAC
.IFF
.BYTE
.ERROR          ; UNDEFINED VALUE FOR FIELD: CNST;
.ENDC
.BYTE          TMO
.WORD          USZ
.WORD          RSZ
.ADDRESS        UBF
.ADDRESS        RBF
.ADDRESS        RHB
.ADDRESS        KBF
.IF NB <PBF>
.= $$ .TAB+RAB$$_PBF
.ADDRESS        PBF
.ENDC
.BYTE          KSZ
.IF NB <PSZ>
.= $$ .TAB+RAB$$_PSZ
.BYTE          PSZ
.ENDC
.BYTE          KRF
.BYTE          MBF
.BYTE          MBC
.ADDRESS        BKT
.ADDRESS        FAB
.ADDRESS        XAB
.= $$ .TABEND
.ENDM $RAB

```



```
: $XABDAT MACRO TO INITIALIZE XAB OF DATE/TIME TYPE
:
:MACRO $XABDAT EDT=0,NXT=0
$XABDEF
$XABDATDEF
$SR_TABINIT      XABSC_DAT,XABSC_DATLEN
$SR_XSET          _EDI,EDT
$SR_XSET          _NXT,NXT
.=$$TABEND
.ENDM $XABDAT

:           $XABRDT MACRO TO INITIALIZE XAB OF REVISION DATE/TIME TYPE
:
:MACRO $XABRDT NXT=0
$XABDEF
$XABRDTDEF
$SR_TABINIT      XABSC_RDT,XABSC_RDTLEN
$SR_XSET          _NXT,NXT
.=$$TABEND
.ENDM $XABRDT

:           $XABPRO MACRO TO INITIALIZE XAB OF PROTECTION TYPE
:
:MACRO $XABPRO UIC=<0,0>,PRO=,NXT=0,MTACC=0,PROT_MODE=0-
PROT_OPT=,ACLBUF=0,ACLSIZ=0,ACLCTR=0
$XABDEF
$XABPRODEF
$SR_TABINIT      XABSC_PRO,XABSC_PROLEN
$$TMP=0
.IRP X,<UIC>
.IIF GE $$TMP-2,.MEXIT
.IIF EQ $$TMP,$$SR_XSET_GRP,<^0'X>
.IIF NE $$TMP,$$SR_XSET_MBM,<^0'X>
$$TMP=$$TMP+1
.ENDM
.IIF NE $$TMP-2,.ERROR : INVALID UIC_FIELD;

: HANDLE PROTECTION
:
$$TMP=-1
$$TMP1=0
.IRP X,<PRO>
.IF NB,\X>
.IRPC Y,<X>
$$TMP2=4
.IIF IDN,<Y>,<R>,$$TMP2=0
.IIF IDN,<Y>,<W>,$$TMP2=1
.IIF IDN,<Y>,<E>,$$TMP2=2
.IIF IDN,<Y>,<D>,$$TMP2=3
.IF NE $$TMP2-4
$$TMP=$$TMP \ <1@$$TMP1+$$TMP2>>
.IIF NE $$TMP & <1@$$TMP1+$$TMP2>>,$$TMP1=16
.IFF
$$TMP1=16 ;CAUSE ERROR MESSAGE
```

```

      .ENDC
      .ENDM
.ENDC
SS.TMP1=SS.TMP1+4
.IIF GT SS.TMP1-16, .MEXIT
.ENDM
.IIF GT SS.TMP1-16, .ERROR      ;INVALID PRO_FIELD SPECIFICATION;
SSR_XSET      PRO,SS.TMP
SSR_XSET      MTACC,MTACC
SSR_XSET      -PROT_MODE,PROT_MODE
SSR_VBFSET    XAB,<PROT_OPTS>
SSR_XSET      PROT_OPT,SS.TMP
SSR_XSET      -ACLBUF,ACLBUF
SSR_XSET      -ACLSIZ,ACLSIZ
SSR_XSET      -ACLCTX,ACLCTX
SSR_XSET      -NXT,NXT
.=SS.TABEND
.ENDM $XABPRO
:
: $XABFHC MACRO TO INITIALIZE XAB OF FHC TYPE
:
.MACRO $XABFHC NXT=0
$XABDEF
$XABFHCDEF
SSR_TABINIT  XAB$C_FHC,XAB$C_FHCLEN
SSR_XSET     NXT,NXT
.=SS.TABEND
.ENDM $XABFHC
:
: $XABSUM MACRO TO INITIALIZE XAB OF SUMMARY TYPE
:
.MACRO $XABSUM NXT=0
$XABDEF
$XABSUMDEF
SSR_TABINIT  XAB$C_SUM,XAB$C_SUMLEN
SSR_XSET     -NXT,NXT
.=SS.TABEND
.ENDM $XABSUM

```

...
: \$XABCXF MACRO TO INITIALIZE XAB OF CONTEXT TYPE ASSOCIATED WITH THE FAB
...

```
.MACRO $XABCXF NXT=0
$XABDEF
$XABCXFDEF
$SR_TABINIT      XABSC_CXF,XABSC_CXFLEN
$SR_XSET         _NXT,NXT
.=$$TABEND
.ENDM $XABCXF
```

...
: \$XABCXR MACRO TO INITIALIZE XAB OF CONTEXT TYPE ASSOCIATED WITH THE RAB
...

```
.MACRO $XABCXR NXT=0,CXRBUF=0,CXRBFZ=0
$XABDEF
$XABCXRDEF
$SR_TABINIT      XABSC_CXR,XABSC_CXRLEN
$SR_XSET         _NXT,NXT
$SR_XSET         _CXRBUF,CXRBUF
$SR_XSET         _CXRBFZ,CXRBFZ
.=$$TABEND
.ENDM $XABCXR
```

```

:
: $XABKEY MACRO TO INITIALIZE XAB OF KEY DEFINITION TYPE
:
: MACRO $XABKEY DAN=0,DFL=0,DTP=STG,FLG=,IAN=0-
:           IFL=0,KNM=0,LAN=0,NUL=0,REF=0-
:           POS=<0>,SIZ,NXT=0,PROLOG=0
$XABDEF
$XABKEYDEF
$$R_TABINIT      XAB$C_KEY,XAB$C_KEYLEN
$$R_XSET         DAN,DAN
$$R_XSET         DFL,DFL
$$R_XSET         IAN,IAN
$$R_XSET         IFL,IFL
$$R_XSET         KNM,KNM
$$R_XSET         LAN,LAN
$$R_XSET         NUL,NUL
$$R_XSET         REF,REF
$$R_XSET         PROLOG,PROLOG
: IF NB,<FLG>
:   $$R_VBFSET   XAB,<FLG>
:   IF EQ,REF
:     .IIF NE,<$$TMP&XAB$M_CHG>,.ERROR ; PRIMARY KEY MAY NOT CHANGE;
:   .ENDC
: .IFF
:   .IIF EQ,REF,$$TMP=0
:   .IIF NE,REF,$$R_VBFSET      XAB,<CHG,DUP>
: .ENDC
$$R_XSET         _FLG,$$TMP
$$DTPTMP=-1
: IF DF,XAB$C_'DTP
:   $$R_XSET     DTP,XAB$C_'DTP
:   $$DTPTMP=XAB$C_'DTP
: .IFF
: .ERROR       ;UNDEFINED KEY FIELD DATA TYPE;
: .ENDC
$$R_XSPSET      POS,<POS>
: .IIF GE,<$$TMP-8>,.ERROR ;MAXIMUM OF 8 SEGMENTS EXCEEDED;
$$SPTMP=$$TMP
: IF NB,<SIZ>
$$R_XSPSET      SIZ,<SIZ>
: .IIF GE,<$$TMP-8>,.ERROR ;MAXIMUM OF 8 SEGMENTS EXCEEDED;
: .IIF NE,<$$SPTMP-$$TMP>,.ERROR ;UNBALANCED POS AND SIZ FIELDS;
: .IIF EQ,$$STMP,.ERROR ;SIZE OF KEY FIELD MUST BE NON ZERO;
: .IIF GE,<$$STMP-256>,.ERROR ;MAXIMUM SIZE OF KEY FIELD EXCEEDED;
: IF GT,$$DTPTMP
: .IIF NE,$$TMP,.ERROR ;SEGMENTED KEYS ONLY ALLOWED FOR STG DATA TYPE;
: IF EQ,<$$DTPTMP-XAB$C_PAC>
: .IIF GT,<$$STMP-16>,.ERROR ;MAXIMUM KEY SIZE (16) EXCEEDED FOR PAC DATA TYPE;
: .IFF
:   IF LE,<$$DTPTMP-XAB$C_BN2>
:     .IIF NE,<$$STMP-2>,.ERROR ;SIZE MUST BE 2 BYTE FOR BN2 AND IN2 DATA TYPES;
:   .IFF
:     IF LE,<$$DTPTMP-XAB$C_BN4>
:       .IIF NE,<$$STMP-4>,.ERROR ;SIZE MUST BE 4 BYTES FOR IN4 AND BN4 DATA TYPES;
:     .IFF
:       IF LE,<$$DTPTMP-XAB$C_BN8>

```

```
.IIF NE,<$$STMP-8>,.ERROR ;SIZE MUST BE 8 BYTE FOR BNB AND IN8 DATA TYPES;  
.ENDC  
.ENDC  
.ENDC  
.ENDC  
.ENDC  
.IFF  
    SSR_XSPSET      _SIZ,<0>  
.ENDC  
SSR_XSET          _NXT,NXT  
.=$$TABEND  
.ENDM $XABKEY
```

```

:
: $XABALL MACRO TO INITIALIZE XAB OF ALLOCATION TYPE
:
: MACRO $XABALL VOL=0,ALN=,AOP=,LOC=0,RFI=<0,0,0>,-
: ALQ=0,AID=0,BKZ=0,DEQ=0,NXT=0
$XABDEF
$XABALLDEF
$$R_TABINIT XABSC_ALL,XABSC_ALI,LEN
$$R_XSET _VOL,VOL
. IF NB <ALN>
. IF DF XABSC 'ALN
$$R_XSET _ALN,XABSC_'ALN
. IFF
. ERROR : UNDEFINED VALUE FOR ALIGNMENT FIELD;
. ENDC
. ENDC
$$R_VBFSET XAB,<AOP>
$$R_XSET _AOP,$$.TMP
$$R_XSET _LOC,LOC
$$$.TMP=-1
. IRP X,<RFI>
. IIF GE $$$.TMP-2,.MEXIT
. IIF LT $$$.TMP,$$R_XSET _RF10,X
. IIF EQ $$$.TMP,$$R_XSET _RF12,X
. IIF GT $$$.TMP,$$R_XSET _RF14,X
$$$.TMP=$$.TMP+1
. ENDM
. IIF NE $$$.TMP-2,.ERROR ; INVALID FILE ID;
$$R_XSET _ALQ,ALQ
$$R_XSET _AID,AID
$$R_XSET _BKZ,BKZ
$$R_XSET _DEQ,DEQ
$$R_XSET _NXT,NXT
.= $$$.TABEND
. ENDM $XABALL

```

```
.. $XABTRM macro to initialize XAB of terminal type (used when ROP ETO set).  
..  
..MACRO $XABTRM NXT=0,ITMLST=0,ITMLST_LEN=0  
$XABDEF  
$XABTRMDEF  
$$R_TABINIT XABSC_TRM,XABSC_TRMLEN  
$$R_XSET NXT,NXT  
$$R_XSET ITMLST,ITMLST  
$$R_XSET ITMLST_LEN,ITMLST_LEN  
.= $$TABEND  
..ENDM $XABTRM
```



```

:
: $XABJNL Macro to initialize XAB of Journal Type
:

```

```

.MACRO $XABJNL JOP=, -
             BIS=0,AIS=0,ATS=0, -
             BIA=0,AIA=0,ATA=0, -
             BIN=, AIN=, ATN=, -
             NXT= 0

$XABDEF
$XABJNLDEF
$$R_TABINIT XABSC_JNL,XABSC_JNLLEN
$$R_XSET    NXT,NXT
$$R_VBFSET  XAB,<JOP>
$$R_XSET    _JOP,$$.TMP
$$R_XSET    _BIS,BIS
$$R_XSET    _AIS,AIS
$$R_XSET    _ATS,ATS
$$R_XSET    _BIA,BIA
$$R_XSET    _AIA,AIA
$$R_XSET    _ATA,ATA

. IF NB <BIN>
  .SAVE
  .PSECT $RMSNAM
  $$TMPX=.
  .ASCII %BIN%
  $$TMPX1=-$$TMPX
  .RESTORE
  $$R_XSET    _BIS,$$.TMPX1
  $$R_XSET    _BIA,$$.TMPX
. ENDC

. IF NB <AIN>
  .SAVE
  .PSECT $RMSNAM
  $$TMPY=.
  .ASCII %AIN%
  $$TMPY1=-$$TMPY
  .RESTORE
  $$R_XSET    _AIS,$$.TMPY1
  $$R_XSET    _AIA,$$.TMPY
. ENDC

. IF NB <ATN>
  .SAVE
  .PSECT $RMSNAM
  $$TMPZ=.
  .ASCII %ATN%
  $$TMPZ1=-$$TMPZ
  .RESTORE
  $$R_XSET    _ATS,$$.TMPZ1
  $$R_XSET    _ATA,$$.TMPZ
. ENDC

.=$$.TABEND
.ENDM $XABJNL

```

```

:
: $RMSDEFEND
: MACRO TO RELEASE SPACE USED BY RMS MACROS
: CALL ONLY AFTER ALL STRUCTURES SET UP
:
: .MACRO $RMSDEFEND
: .MDELETE $FAB,$RAB,$NAM,$XABDAT,$XABPRO,$XABSUM,$XABKEY
: .MDELETE $XABRDT,$XABTRM,$XABCF,$XABCXR,$XABJNL
: .MDELETE $XABALL,$XABFAC,$$R TABINIT,$$R XSET,$$R FSET
: .MDELETE $$R RSET,$$R NSET,$$R XSET,$$R VBFSET,$$R FVSET
: .MDELETE $$R RVSET,$$R FCSET,$$R RCSET,$$R_XSPSET,$$R_XSP2SET
: .MDELETE $RMSDEFEND
: .ENDM $RMSDEFEND
:
:
:
:
: MACRO TO DEFINE ALL RMS SYMBOLS
: (LOCALLY IF GBL IS BLANK ELSE GLOBALLY)
:
: .MACRO $RMSALLDEF GBL
$RABDEF GBL
$FABDEF GBL
$NAMDEF GBL
$XABDEF GBL
$RMSDEF
: .MACRO $RMSALLDEF
: .ENDM
: .ENDM $RMSALLDEF

```

```

: LEVEL - 2 MACROS TO PROCESS RMS STRUCTURES
:

```

```

: $$R_TABINIT MACRO TO PERFORM COMMON RMS STRUCTURE INITIALIZATION
: AND GIVE DIAGNOSTIC INFO IF STRUCTURE NOT LONGWORD ALIGNED
:

```

```

.MACRO $$R_TABINIT ID,LEN
.IIF NE .B3, .print ;%MACRO-I-GENINFO, Generated INFO: RMS BLOCK NOT LONGWORD ALIGNED;
$$TAB=

```

```

    .BYTE ID
    .BYTE LEN
    .BLKB LEN-2

```

```

$$TABEND=
.ENDM $$R_TABINIT

```

```

: $$R_SET MACRO TO INITIALIZE A SPECIFIC FIELD
: TO A VALUE
:

```

```

.MACRO $$R_SET FLD,VAL,SYM

```

```

.IF DF, SYM'SB'FLD
.= $$TAB+SYM'SB'FLD
    .BYTE <VAL>
    .MEXIT

```

```

.ENDC
.IF DF, SYM'SL'FLD
.= $$TAB+SYM'SL'FLD
    .ADDRESS <VAL>
    .MEXIT

```

```

.ENDC
.IF DF, SYM'SW'FLD
.= $$TAB+SYM'SW'FLD
    .WORD <VAL>
    .MEXIT

```

```

.ENDC
.IF DF, SYM'SQ'FLD
.= $$TAB+SYM'SQ'FLD
    .QUAD <VAL>
    .MEXIT

```

```

.ENDC
    .ERROR ; UNKNOWN SYM FIELD: FLD;
.ENDM $$R_SET

```

```

: $$R_XSP2SET LEVEL 2 MACRO TO STORE INTO SIZ/POS FIELD OF KEY XAB
:

```

```

.MACRO $$R_XSP2SET FLD,VAL,ELEM
$$R_SET FLD'ELEM,VAL,XAB
.ENDM $$R_XSP2SET

```

```

: INTERMEDIATE LEVEL MACROS TO CALL $$R_SET MACRO
:

```

```

.MACRO $$R_FSET X,Y
$$R_SET X,Y,FAB
.ENDM $$R_FSET

```

```

.MACRO $$R_RSET X,Y

```

```

      $$R SET X,Y,RAB
      .ENDM $$R_RSET

      .MACRO $$R_NSET X,Y
      $$R SET X,Y,NAM
      .ENDM $$R_NSET

      .MACRO $$R_XSET      X,Y
      $$R SET X,<?>,XAB
      .ENDM $$R_XSET

:
: $$R_XS. SET MACRO TO STORE 8 ELEMENT SIZ/POS FIELD IN KEY XAB
:
.MACRO $$R_XSPSET      FLD,VAL
$$S.TMP=0
$$S.TMP=-1
.IRP X,<VAL>
  $$S.TMP=$$S.TMP+1
  .IIF GE,<$$S.TMP-8>,.MEXIT
  .IF IDN <FLD>,< SIZ>
    .IIF GE,<X-256>,.ERROR ;MAXIMUM VALUE FOR SIZE ELEMENT EXCEEDED: X;
    $$S.TMP=$$S.TMP+X
  .ENDC
  $$R_XSP2SET FLD,X,\$$S.TMP
.ENDM
.ENDM $$R_XSPSET

:
: $$R_FVSET MACRO TO STORE COMPLEX BIT MASK IN FAB
:
      .MACRO $$R_FVSET FLD,BITS
      $$R_VBFSET FAB,<BITS>
      $$R_FSET FLD,$$S.TMP
      .ENDM $$R_FVSET

:
: $$R_VBFSET MACRO TO CREATE A BIT MASK FROM
: A LIST OF BIT NAMES
:
      .MACRO $$R_VBFSET BLK,BITS
      $$S.TMP=0
      .IRP X,<BITS>
      .IF DF BLK '$V_' X
        $$S.TMP=$$S.TMP!<1@BLK '$V_' X>
      .IIF
        .ERROR ; UNDEFINED BIT VALUE CODE: X;
      .ENDC
      .ENDR
      .ENDM $$R_VBFSET

:
: $$R_FCSET MACRO TO STORE A NAMED CONSTANT VALUE IN A
: FIELD FOR FAB
:
      .MACRO $$R_FCSET FLD, CNST
      .IF DF FAB$C_'CNST
      $$R_FSET FLD,FAB$C_'CNST

```

```
.IFF  
.ERROR ; UNDEFINED VALUE FOR FIELD : CNST;  
.ENDC  
.ENDM $$R_FCSET
```

```
..  
: $$R_RCSET MACRO TO STORE A NAMED CONSTANT VALUE  
: IN A FIELD OF THE RAB
```

```
..  
: .MACRO $$R_RCSET FLD,CNST  
: .IF DF RAB$C_'CNST  
: $$R_RSET FLD,RAB$C_'CNST  
: .IFF  
: .ERROR ; UNDEFINED VALUE FOR FIELD: CNST;  
: .ENDC  
: .ENDM $$R_RCSET
```

```
..  
: $$R_RVSET MACRO TO STORE COMPOUND BIT MASK IN RAB
```

```
..  
: .MACRO $$R_RVSET FLD,BITS  
: $$R_VBFSET RAB,<BITS>  
: $$R_RSET FLD,$$.TMP  
: .ENDM $$R_RVSET
```

THESE ARE THE MACROS TO STORE INTO FIELDS OF RMS DATA STRUCTURES AT RUN-TIME

MACRO TO STORE INTO THE FIELDS OF A FAB

```
MACRO $FAB_STORE FAB=RO,FAC=,SHR=,FNA=,FNS=,DNA=,-
DNS=,RTV=,ORG=,RAT=,-
FOP=,XAB=,MRS=,JNL=,MRN=,-
ALQ=,DEQ=,BLS=,NAM=,RFM=,-
FSZ=,BKS=,GBC=,CTX=,BSZ=,-
LNM_MODE=,CHAN_MODE=,FILE_MODE=,-
;FIELDS THAT
;AREN'T IN $FAB MACRO

```

\$FABDEF

\$SSTYPE FAB

..AFLG=0

;ZERO ADDRESSING FLAG

..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM

```
.IIF NB <FAC>, $$RMS_FVSET FLD= FAC , BITS=<FAC> , REG='FAB'
.IIF NB <SHR>, $$RMS_FVSET FLD= SHR , BITS=<SHR> , REG='FAB'
.IIF NB <FNA>, $$RMS_FASET FLD= FNA , SRC=FNA , REG='FAB'
.IIF NB <FNS>, $$RMS_FSET FLD= FNS , SRC=FNS , REG='FAB'
.IIF NB <DNA>, $$RMS_FASET FLD= DNA , SRC=DNA , REG='FAB'
.IIF NB <DNS>, $$RMS_FSET FLD= DNS , SRC=DNS , REG='FAB'
.IIF NB <RTV>, $$RMS_FSET FLD= RTV , SRC=RTV , REG='FAB'
.IIF NB <ORG>, $$RMS_FCSET FLD= ORG , CNST=ORG , REG='FAB'
.IIF NB <RAT>, $$RMS_FVSET FLD= RAT , BITS=<RAT> , REG='FAB'
.IIF NB <FOP>, $$RMS_FVSET FLD= FOP , BITS=<FOP> , REG='FAB'
.IIF NB <XAB>, $$RMS_FASET FLD= XAB , SRC=XAB , REG='FAB'
.IIF NB <MRS>, $$RMS_FSET FLD= MRS , SRC=MRS , REG='FAB'
.IIF NB <JNL>, $$RMS_FASET FLD= JNL , SRC=JNL , REG='FAB'
.IIF NB <MRN>, $$RMS_FSET FLD= MRN , SRC=MRN , REG='FAB'
.IIF NB <ALQ>, $$RMS_FSET FLD= ALQ , SRC=ALQ , REG='FAB'
.IIF NB <DEQ>, $$RMS_FSET FLD= DEQ , SRC=DEQ , REG='FAB'
.IIF NB <BLS>, $$RMS_FSET FLD= BLS , SRC=BLS , REG='FAB'
.IIF NB <NAM>, $$RMS_FASET FLD= NAM , SRC=NAM , REG='FAB'
.IIF NB <RFM>, $$RMS_FCSET FLD= RFM , CNST=RFM , REG='FAB'
.IIF NB <FSZ>, $$RMS_FSET FLD= FSZ , SRC=FSZ , REG='FAB'
.IIF NB <BKS>, $$RMS_FSET FLD= BKS , SRC=BKS , REG='FAB'
.IIF NB <GBC>, $$RMS_FSET FLD= GBC , SRC=GBC , REG='FAB'
.IIF NB <LNM_MODE>, $$RMS_FVBSET FLD= LNM_MODE , SRC=LNM_MODE , REG='FAB', FLD2=B_ACMODES
.IIF NB <CHAN_MODE>, $$RMS_FVBSET FLD= CHAN_MODE , SRC=CHAN_MODE , REG='FAB', FLD2=B_ACMODES
.IIF NB <FILE_MODE>, $$RMS_FVBSET FLD= FILE_MODE , SRC=FILE_MODE , REG='FAB', FLD2=B_ACMODES
.IIF NB <CTX>, $$RMS_FSET FLD= CTX , SRC=CTX , REG='FAB'
.IIF NB <BSZ>, $$RMS_FSET FLD= BSZ , SRC=BSZ , REG='FAB'
.IIF NB <BID>, $$RMS_FCSET FLD= BID , CNST=BID , REG='FAB'
.IIF NB <BLN>, $$RMS_FCSET FLD= BLN , CNST=BLN , REG='FAB'
.IIF NB <STS>, $$RMS_FSET FLD= STS , SRC=STS , REG='FAB'
.IIF NB <STV>, $$RMS_FSET FLD= STV , SRC=STV , REG='FAB'
.ENDM $FAB_STORE

```

```
: MACRO TO STORE INTO THE FIELDS OF A RAB
```

```
: MACRO $RAB_STORE RAB=RO,RAC=,ROP=,UBF=,USZ=,-
: RBF=,RSZ=,BKT=,KBF=,PBF=,-
: KSZ=,PSZ=,RHB=,FAB=,MBF=,-
: MBC=,TMO=,CTX=,KRF=,-
: BID=,BLN=,STS=,STV=,RFA=,XAB= : FIELDS
: THAT AREN'T IN $RAB
```

```
$RABDEF
```

```
$$TYPE RAB
```

```
..AFLG=0 :ZERO ADDRESSING FLAG
```

```
: ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
```

```
.IIF NB <RAC>, $$RMS_RCSET FLD=_RAC , CNST=RAC , REG='RAB'
.IIF NB <ROP>, $$RMS_RVSET FLD=_ROP , BITS=<ROP> , REG='RAB'
.IIF NB <UBF>, $$RMS_RASET FLD=_UBF , SRC=UBF , REG='RAB'
.IIF NB <USZ>, $$RMS_RSET FLD=_USZ , SRC=USZ , REG='RAB'
.IIF NB <RBF>, $$RMS_RASET FLD=_RBF , SRC=RBF , REG='RAB'
.IIF NB <RSZ>, $$RMS_RSET FLD=_RSZ , SRC=RSZ , REG='RAB'
.IIF NB <BKT>, $$RMS_RSET FLD=_BKT , SRC=BKT , REG='RAB'
.IIF NB <KBF>, $$RMS_RASET FLD=_KBF , SRC=KBF , REG='RAB'
.IIF NB <PBF>, $$RMS_RASET FLD=_PBF , SRC=PBF , REG='RAB'
.IIF NB <KSZ>, $$RMS_RSET FLD=_KSZ , SRC=KSZ , REG='RAB'
.IIF NB <PSZ>, $$RMS_RSET FLD=_PSZ , SRC=PSZ , REG='RAB'
.IIF NB <RHB>, $$RMS_RASET FLD=_RHB , SRC=RHB , REG='RAB'
.IIF NB <FAB>, $$RMS_RASET FLD=_FAB , SRC=FAB , REG='RAB'
.IIF NB <XAB>, $$RMS_RASET FLD=_XAB , SRC=XAB , REG='RAB'
.IIF NB <MBF>, $$RMS_RSET FLD=_MBF , SRC=MBF , REG='RAB'
.IIF NB <MBC>, $$RMS_RSET FLD=_MBC , SRC=MBC , REG='RAB'
.IIF NB <TMO>, $$RMS_RSET FLD=_TMO , SRC=TMO , REG='RAB'
.IIF NB <CTX>, $$RMS_RSET FLD=_CTX , SRC=CTX , REG='RAB'
.IIF NB <KRF>, $$RMS_RSET FLD=_KRF , SRC=KRF , REG='RAB'
.IIF NB <BID>, $$RMS_RCSET FLD=_BID , CNST=BID , REG='RAB'
.IIF NB <BLN>, $$RMS_RCSET FLD=_BLN , CNST=BLN , REG='RAB'
.IIF NB <STS>, $$RMS_RSET FLD=_STS , SRC=STS , REG='RAB'
.IIF NB <STV>, $$RMS_RSET FLD=_STV , SRC=STV , REG='RAB'
.IF 'R' <RFA>
.IIF EQ ..FLG, $$TRI_WORD_MOVE RFA , <RAB$W_RFA'RAB'> , _RFA
.IIF EQ ..FLG-1, $$TRI_WORD_MOVE RFA , <RAB$W_RFA(RO)> , _RFA
.IIF EQ ..FLG-2, $$TRI_WORD_MOVE RFA , <RAB$W_RFA('RAB')> , _RFA
```

```
.ENDC
```

```
.ENDM $RAB_STORE
```

```
; MACRO TO STORE INTO THE FIELDS OF A NAME BLOCK
```

```
.MACRO $NAM_STORE NAM=RO, RSA=, RSS=, RSL=, ESA=, ESS=, ESL=, RLF=, NOP=, -
      BID=, BLN=, FID=, DID=, WCC=, FNB=, DVI=          : FIELDS
      ; THAT AREN'T IN $NAM
```

```
$NAMDEF
```

```
$STYPE NAM
```

```
..AFLG=0 ; ZERO ADDRESSING FLAG
```

```
..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
```

```
.IIF NB <RSA>, $$RMS_NASET FLD= RSA , SRC=RSA , REG='NAM'
.IIF NB <RSS>, $$RMS_NASET FLD= RSS , SRC=RSS , REG='NAM'
.IIF NB <RSL>, $$RMS_NASET FLD= RSL , SRC=RSL , REG='NAM'
.IIF NB <NOP>, $$RMS_NVSET FLD= NOP , BITS=<NOP> , REG='NAM'
.IIF NB <ESA>, $$RMS_NASET FLD= ESA , SRC=ESA , REG='NAM'
.IIF NB <ESS>, $$RMS_NASET FLD= ESS , SRC=ESS , REG='NAM'
.IIF NB <ESL>, $$RMS_NASET FLD= ESL , SRC=ESL , REG='NAM'
.IIF NB <RLF>, $$RMS_NASET FLD= RLF , SRC=RLF , REG='NAM'
.IIF NB <BID>, $$RMS_NCSET FLD= BID , CNST=BID , REG='NAM'
.IIF NB <BLN>, $$RMS_NCSET FLD= BLN , CNST=BLN , REG='NAM'
.IIF NB <WCC>, $$RMS_NASET FLD= WCC , SRC=WCC , REG='NAM'
.IIF NB <FNB>, $$RMS_NVSET FLD= FNB , BITS=<FNB> , REG='NAM'
.IF NB <FID>
  .IIF EQ ..FLG, $$TRI_WORD_MOVE FID , <NAMSW_FID'NAM'> , _FID
  .IIF EQ ..FLG-1, $$TRI_WORD_MOVE FID , <NAMSW_FID(RO)> , _FID
  .IIF EQ ..FLG-2, $$TRI_WORD_MOVE FID , <NAMSW_FID('NAM')>, _FID
.ENDC
.IF NB <DID>
  .IIF EQ ..FLG, $$TRI_WORD_MOVE DID , <NAMSW_DID'NAM'> , _DID
  .IIF EQ ..FLG-1, $$TRI_WORD_MOVE DID , <NAMSW_DID(RO)> , _DID
  .IIF EQ ..FLG-2, $$TRI_WORD_MOVE DID , <NAMSW_DID('NAM')>, _DID
.ENDC
.IF NB <DVI>
  .IIF EQ ..FLG, $$TWO_QUAD_MOVE DVI , <NAMST_DVI'NAM'> , _DVI
  .IIF EQ ..FLG-1, $$TWO_QUAD_MOVE DVI , <NAMST_DVI(RO)> , _DVI
  .IIF EQ ..FLG-2, $$TWO_QUAD_MOVE DVI , <NAMST_DVI('NAM')>, _DVI
.ENDC
.ENDM $NAM_STORE
```



```
; MACRO TO STORE INTO THE FIELDS OF A DATE XAB
```

```
.MACRO $XABDAT_STORE XAB=RO,EDT=,NXT=,-  
                    COD=,BLN=,-      ;FIELDS  
                    RVN=,RDT=,CDT=   ;THAT AREN'T IN $XABDAT
```

```
$XABDEF
```

```
$XABDATDEF
```

```
$$TYPE XAB
```

```
..AFLG=0                      ;ZERO ADDRESSING FLAG
```

```
..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
```

```
.IIF NB <EDT>, $$RMS_XSET FLD=_EDT , SRC=EDT , REG='XAB'  
.IIF NB <NXT>, $$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'  
.IIF NB <COD>, $$RMS_XCSET FLD=_COD , CNST=COD , REG='XAB'  
.IIF NB <BLN>, $$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'  
.IIF NB <RVN>, $$RMS_XSET FLD=_RVN , SRC=RVN , REG='XAB'  
.IIF NB <RDT>, $$RMS_XSET FLD=_RDT , SRC=RDT , REG='XAB'  
.IIF NB <CDT>, $$RMS_XSET FLD=_CDT , SRC=CDT , REG='XAB'  
.ENDM $XABDAT_STORE
```

```
; MACRO TO STORE INTO THE FIELDS OF A DATE XAB
```

```
.MACRO $XABRDT_STORE XAB=RO,NXT=-,  
                    COD=-,BLN=-,  
                    RVN=-,RDT=-
```

```
$XABDEF
```

```
$XABRDTDEF
```

```
$$TYPE XAB
```

```
..AFLG=0
```

```
;ZERO ADDRESSING FLAG
```

```
..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
```

```
.IIF NB <NXT>, $$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'
```

```
.IIF NB <COD>, $$RMS_XCSET FLD=_COD , CNST=COD , REG='XAB'
```

```
.IIF NB <BLN>, $$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'
```

```
.IIF NB <RVN>, $$RMS_XSET FLD=_RVN , SRC=RVN , REG='XAB'
```

```
.IIF NB <RDT>, $$RMS_XSET FLD=_RDT , SRC=RDT , REG='XAB'
```

```
.ENDM $XABRDT_STORE
```

```
; MACRO TO STORE INTO THE FIELDS OF A PROTECTION XAB
```

```
MACRO $XABPRO_STORE XAB=RO,UIC=,PRO=,NXT=,MTACC=,PROT_MODE=,-
    PROT_OPT=,ACLBUF=,ACLSIZ=,ACLCTX=,-
    COD=,BLN=,ACLEN=,ACLSTS= ; FIELDS THAT AREN'T
    ; IN $XABPRO
```

```
$XABDEF
```

```
$XABPRODEF
```

```
$$TYPE XAB
```

```
..AFLG=0
```

```
;ZERO ADDRESSING FLAG
```

```
..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
```

```
.IIF NB <NXT>, $$RMS_XASET FLD=NXT , SRC=NXT , REG='XAB'
.IIF NB <COD>, $$RMS_XCSET FLD=COD ; CNST=COD , REG='XAB'
.IIF NB <BLN>, $$RMS_XCSET FLD=BLN , CNST=BLN , REG='XAB'
.IIF NB <MTACC>, $$RMS_XSET FLD=MTACC , SRC=MTACC , REG='XAB'
.IIF NB <PROT_MODE>, $$RMS_XSET FLD=PROT_MODE , SRC=PROT_MODE , REG='XAB'
.IIF NB <PROT_OPT>, $$RMS_XVSET FLD=PROT_OPT , BITS=<PROT_OPT> , REG='XAB'
.IIF NB <ACLBUF>, $$RMS_XASET FLD=ACLBUF , SRC=ACLBUF , REG='XAB'
.IIF NB <ACLSIZ>, $$RMS_XSET FLD=ACLSIZ , SRC=ACLSIZ , REG='XAB'
.IIF NB <ACLCTX>, $$RMS_XSET FLD=ACLCTX , SRC=ACLCTX , REG='XAB'
.IIF NB <ACLEN>, $$RMS_XSET FLD=ACLEN , SRC=ACLEN , REG='XAB'
.IIF NB <ACLSTS>, $$RMS_XSET FLD=ACLSTS , SRC=ACLSTS , REG='XAB'
.IIF NB <PRO>, $$PRO PROT=<PRO>, REGISTER='XAB'
```

```
.IF NB <UIC>
```

```
    $$NARG UIC
```

```
    .IF EQ ..N-1
```

```
        $$RMS_XSET FLD=_UIC, SRC=UIC, REG='XAB'
```

```
    .IFF
```

```
        .IF GT ..N-2
```

```
            .ERROR ;INVALID _UIC;
```

```
        .IFF
```

```
            $$TMP5=0
```

```
            .IRP X,<UIC>
```

```
                .NTYPE ..TYP,X
```

```
                .IIF GE $$TMP5-2, .MEXIT
```

```
                .IF EQ $$TMP5
```

```
                    ..MOD = 0
```

```
                    .IIF EQ <..TYP-^XEF>, ..MOD = 1
```

```
                    .IIF EQ <..TYP-^XCF>, ..MOD = 1
```

```
                    .IIF EQ <..TYP-^XAF>, ..MOD = 1
```

```
                    .IF NE ..MOD
```

```
                        $$RMS_XSET FLD=_GRP, SRC=#^O'X, REG='XAB'
```

```
                    .IFF
```

```
                        $$RMS_XSET FLD=_GRP, SRC=X, REG='XAB'
```

```
                .ENDC
```

```
            .ENDC
```

```
            .IF NE $$TMP5
```

```
                ..MOD = 0
```

```
                .IIF EQ <..TYP-^XEF>, ..MOD = 1
```

```
                .IIF EQ <..TYP-^XCF>, ..MOD = 1
```

```
                .IIF EQ <..TYP-^XAF>, ..MOD = 1
```

```
                .IF NE ..MOD
```

```
                    $$RMS_XSET FLD=_MBM, SRC=#^O'X, REG='XAB'
```

```
            .IFF
```

```

                                $$$RMS_XSET FLD=_MBM, SRC=X, REG='XAB'
                                .ENDC
                                .ENDC
                                $$$TMP5=$$TMP5 + 1
                                .ENDC
                                .ENDM
                                .ENDC
                                .ENDC
                                .ENDC
                                .ENDM $XABPRO_STORE

.MACRO $$$PRO PROT,REGISTER
$$$TMP=-1
$$$TMP1=0
.IPP X,<PROT>
.IF NB,<X>
.IRPC Y,<X>
$$$TMP2=4
.IIF IDN,<Y>,<R>, $$$TMP2=0
.IIF IDN,<Y>,<W>, $$$TMP2=1
.IIF IDN,<Y>,<E>, $$$TMP2=2
.IIF IDN,<Y>,<D>, $$$TMP2=3
.IF NE $$$TMP2-4
$$$TMP=$$TMP \ <1@<$$$TMP1+$$$TMP2>>
.IIF NE $$$TMP & <1@<$$$TMP1+$$$TMP2>>, $$$TMP1=16

.IFF
$$$TMP1=16 ;CAUSE ERROR MESSAGE

.ENDC
.ENDM

.ENDC
$$$TMP1=$$TMP1+4
.IIF GT $$$TMP1-16, .MEXIT
.ENDM
.IF GT $$$TMP1-16
$$NARG PROT
.IF EQ ..N-1
$$$RMS_XSET FLD=_PRO, SRC=PROT, REG=REGISTER
.IFF
.ERROR ;INVALID PRO_FIELD SPECIFICATION: PROT;
.ENDC
.IFF
$$$RMS_XSET FLD=_PRO,SRC=#$$$TMP,REG=REGISTER
.ENDC
.ENDM $$$PRO

```

```

; MACRO TO STORE INTO THE FIELDS OF AN ALLOCATION XAB
; MACRO $XABALL_STORE XAB=RO,VOL=,ALN=,AOP=,LOC=,RFI=,ALQ=,NXT=,-
;                   DEQ=,AID=,BKZ=,-
;                   COD=,BLN=           : FIELDS
;                   ; THAT AREN'T IN $XABALL

```

```

$XABDEF
$XABALLDEF
$$TYPE XAB
..AFLG=0                               ; ZERO ADDRESSING FLAG
;
; ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
;
.IIF NB <VOL>, $$RMS_XSET FLD=VOL , SRC=VOL , REG='XAB'
.IIF NB <ALN>, $$RMS_XCSET FLD=ALN , CNST=ALN , REG='XAB'
.IIF NB <AOP>, $$RMS_XVSET FLD=AOP , BITS=<AOP> , REG='XAB'
.IIF NB <LOC>, $$RMS_XSET FLD=LOC , SRC=LOC , REG='XAB'
.IIF NB <ALQ>, $$RMS_XSET FLD=ALQ , SRC=ALQ , REG='XAB'
.IIF NB <NXT>, $$RMS_XASET FLD=NXT , SRC=NXT , REG='XAB'
.IIF NB <COD>, $$RMS_XCSET FLD=COD , CNST=COD , REG='XAB'
.IIF NB <BLN>, $$RMS_XCSET FLD=BLN , CNST=BLN , REG='XAB'
.IIF NB <DEQ>, $$RMS_XSET FLD=DEQ , SRC=DEQ , REG='XAB'
.IIF NB <AID>, $$RMS_XSET FLD=AID , SRC=AID , REG='XAB'
.IIF NB <BKZ>, $$RMS_XSET FLD=BKZ , SRC=BKZ , REG='XAB'
.IF NB <RFI>
.IIF EQ ..FLG, $$TRI_WORD_MOVE RFI , <XAB$W_RFI'XAB'> , _RFI
.IIF EQ ..FLG-1, $$TRI_WORD_MOVE RFI , <XAB$W_RFI(RO)> , _RFI
.IIF EQ ..FLG-2, $$TRI_WORD_MOVE RFI , <XAB$W_RFI('XAB')> , _RFI
.ENDC
.ENDM $XABALL_STORE

```

```
; MACRO TO STORE INTO THE FIELDS OF A FILE HEADER XAB
;
;MACRO $XABFHC_STORE XAB=RO,NXT=-,
;                   COD=-,BLN=
;                   :FIELDS
;                   ;THAT AREN'T IN $XABFHC
```

```
$XABDEF
$XABFHCDEF
$$TYPE XAB
..AFLG=0 ;ZERO ADDRESSING FLAG
;
; ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
;
; .IIF NB <NXT>, $$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'
; .IIF NB <COD>, $$RMS_XCSET FLD=_COD ; CNST=COD , REG='XAB'
; .IIF NB <BLN>, $$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'
; .ENDM $XABFHC_STORE
```

; MACRO TO STORE INTO THE FIELDS OF A SUMMARY XAB

.MACRO \$XABSUM_STORE XAB=RO,NXT=-,
 COD=-,BLN= ;FIELDS
 ;THAT AREN'T IN \$XABSUM

\$XABDEF
\$XABSUMDEF
\$\$TYPE XAB
..AFLG=0 ;ZERO ADDRESSING FLAG

; ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM

.IIF NB <NXT>, \$\$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'
.IIF NB <COD>, \$\$RMS_XCSET FLD=_COD , CNST=COD , REG='XAB'
.IIF NB <BLN>, \$\$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'
.ENDM \$XABSUM_STORE

```

; MACRO TO STORE INTO THE FIELDS OF A FAB CONTEXT XAB
;MACRO $XABCXF_STORE XAB=RO,NXT=-,
;                COD=-,BLN=
;                ;FIELDS
;                ;THAT AREN'T IN $XABCXF

```

```

$XABDEF
$XABCXFDEF
$$TYPE XAB
..AFLG=0
;ZERO ADDRESSING FLAG
;
; ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
;
; .IIF NB <NXT>, $$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'
; .IIF NB <COD>, $$RMS_XCSET FLD=_COD , CNST=COD , REG='XAB'
; .IIF NB <BLN>, $$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'
;ENDM $XABCXF_STORE

```

```

; MACRO TO STORE INTO THE FIELDS OF A RAB CONTEXT XAB
;MACRO $XABCXR_STORE XAB=RO,NXT=-,CXRBUF=-,CXKBFZ=-
;                COD=-,BLN=
;                ;FIELDS
;                ;THAT AREN'T IN $XABCXR

```

```

$XABDEF
$XABCXRDEF
$$TYPE XAB
..AFLG=0
;ZERO ADDRESSING FLAG
;
; ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM
;
; .IIF NB <NXT>, $$RMS_XASET FLD=_NXT , SRC=NXT , REG='XAB'
; .IIF NB <COD>, $$RMS_XCSET FLD=_COD , CNST=COD , REG='XAB'
; .IIF NB <BLN>, $$RMS_XCSET FLD=_BLN , CNST=BLN , REG='XAB'
; .IIF NB <CXKBFZ>, $$RMS_XCSET FLD=_CXKBFZ , SRC=CXKBFZ , REG='XAB'
; .IIF NB <CXRBUF>, $$RMS_XCSET FLD=_CXRBUF , SRC=CXRBUF , REG='XAB'
;ENDM $XABCXR_STORE

```



```
: Macro to store into the fields of a terminal XAB.  
: .MACRO $XABTRM_STORE XAB=RO,NXT=,COD=,BLN=,ITMLST=,ITMLST_LEN=  
$XABDEF  
$XABTRMDEF  
$$TYPE XAB  
..AFLG=0 ;ZERO ADDRESSING FLAG  
: ..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM  
: .IIF NB <NXT>,$$RMS_XASET FLD=_NXT,SRC=NXT,REG='XAB'  
: .IIF NB <COD>,$$RMS_XCSET FLD=_COD,CNST=COD,REG='XAB'  
: .IIF NB <BLN>,$$RMS_XCSET FLD=_BLN,CNST=BLN,REG='XAB'  
: .IIF NB <ITMLST>,$$RMS_XASET FLD=_ITMLST,SRC=ITMLST,REG='XAB'  
: .IIF NB <ITMLST_LEN>,$$RMS_XSET FLD=_ITMLST_LEN,SRC=ITMLST_LEN,REG='XAB'  
.ENDM $XABTRM_STORE
```

; MACRO TO STORE INTO THE FIELDS OF A KEY XAB

```

;MACRO $XABKEY_STORE XAB=RO,DAN=,DFL=,DTP=,FLG=,IAN=,IFL=,
KNM=,LAN=,NUL=,POS=,POS0=,POS1=,POS2=,POS3=,
POS4=,POS5=,POS6=,POS7=,REF=,SIZ=,SIZ0=,SIZ1=,
SIZ2=,SIZ3=,SIZ4=,SIZ5=,SIZ6=,SIZ7=,PROLOG=,COD=,BLN=,NXT=

```

\$XABDEF

\$XABKEYDEF

\$STYPE XAB

..AFLG=0

;ZERO ADDRESSING FLAG

..FLG DESCRIBES WHICH ADDRESSING MODE IS DESIRED AS A PARAM

```

.IIF NB <PROLOG>, $$RMS_XSET FLD=PROLOG , SRC=PROLOG , REG='XAB'
.IIF NB <DAN>, $$RMS_XSET FLD=DAN , SRC=DAN , REG='XAB'
.IIF NB <DFL>, $$RMS_XSET FLD=DFL , SRC=DFL , REG='XAB'
.IIF NB <DTP>, $$RMS_XCSET FLD=DTP , CNST=DTP , REG='XAB'
.IIF NB <FLG>, $$RMS_XVSET FLD=FLG , BITS=<FLG> , REG='XAB'
.IIF NB <IAN>, $$RMS_XSET FLD=IAN , SRC=IAN , REG='XAB'
.IIF NB <IFL>, $$RMS_XSET FLD=IFL , SRC=IFL , REG='XAB'
.IIF NB <KNM>, $$RMS_XASET FLD=KNM , SRC=KNM , REG='XAB'
.IIF NB <LAN>, $$RMS_XSET FLD=LAN , SRC=LAN , REG='XAB'
.IIF NB <NUL>, $$RMS_XSET FLD=NUL , SRC=NUL , REG='XAB'
.IIF NB <POS>, $$POS_FLAG=..FLG , SRC=<POS> , REG='XAB'
.IIF NB <POS0>, $$RMS_XSET FLD=POS0 , SRC=POS0 , REG='XAB'
.IIF NB <POS1>, $$RMS_XSET FLD=POS1 , SRC=POS1 , REG='XAB'
.IIF NB <POS2>, $$RMS_XSET FLD=POS2 , SRC=POS2 , REG='XAB'
.IIF NB <POS3>, $$RMS_XSET FLD=POS3 , SRC=POS3 , REG='XAB'
.IIF NB <POS4>, $$RMS_XSET FLD=POS4 , SRC=POS4 , REG='XAB'
.IIF NB <POS5>, $$RMS_XSET FLD=POS5 , SRC=POS5 , REG='XAB'
.IIF NB <POS6>, $$RMS_XSET FLD=POS6 , SRC=POS6 , REG='XAB'
.IIF NB <POS7>, $$RMS_XSET FLD=POS7 , SRC=POS7 , REG='XAB'
.IIF NB <REF>, $$RMS_XSET FLD=REF , SRC=REF , REG='XAB'
.IIF NB <SIZ>, $$SIZ_FLAG=..FLG , SRC=<SIZ> , REG='XAB'
.IIF NB <SIZ0>, $$RMS_XSET FLD=SIZ0 , SRC=SIZ0 , REG='XAB'
.IIF NB <SIZ1>, $$RMS_XSET FLD=SIZ1 , SRC=SIZ1 , REG='XAB'
.IIF NB <SIZ2>, $$RMS_XSET FLD=SIZ2 , SRC=SIZ2 , REG='XAB'
.IIF NB <SIZ3>, $$RMS_XSET FLD=SIZ3 , SRC=SIZ3 , REG='XAB'
.IIF NB <SIZ4>, $$RMS_XSET FLD=SIZ4 , SRC=SIZ4 , REG='XAB'
.IIF NB <SIZ5>, $$RMS_XSET FLD=SIZ5 , SRC=SIZ5 , REG='XAB'
.IIF NB <SIZ6>, $$RMS_XSET FLD=SIZ6 , SRC=SIZ6 , REG='XAB'
.IIF NB <SIZ7>, $$RMS_XSET FLD=SIZ7 , SRC=SIZ7 , REG='XAB'
.IIF NB <COD>, $$RMS_XCSET FLD=COD , CNST=COD , REG='XAB'
.IIF NB <BLN>, $$RMS_XCSET FLD=BLN , CNST=BLN , REG='XAB'
.IIF NB <NXT>, $$RMS_XASET FLD=NXT , SRC=NXT , REG='XAB'
.ENDM $XABKEY_STORE

```

; Macro to store into the fields of a Journal XAB

```
.MACRO $XABJNL_STORE XAB=RO,JOP=, -  
                    BIS=,AIS=,ATS=,BIA=,AIA=,ATA=, -  
                    NXT=
```

```
$XABDEF  
$XABJNLDEF  
$$TYPE XAB  
..AFLG=0
```

; ..FLG Describes which addressing mode is desired as a param

```
.IIF NB <NXT>, $$RMS_XASET FLD=_NXT,SRC=NXT,REG='XAB'  
.IIF NB <COD>, $$RMS_XCSET FLD=_COD,CNST=COD,REG='XAB'  
.IIF NB <BLN>, $$RMS_XCSET FLD=_BLN,CNST=BLN,REG='XAB'  
.IIF NB <JOP>, $$RMS_XVSET FLD=_JOP,BITS=<JOP>,REG='XAB'  
.IIF NB <BIS>, $$RMS_XSET FLD=_BIS,SRC=BIS,REG='XAB'  
.IIF NB <AIS>, $$RMS_XSET FLD=_AIS,SRC=AIS,REG='XAB'  
.IIF NB <ATS>, $$RMS_XSET FLD=_ATS,SRC=ATS,REG='XAB'  
.IIF NB <BIA>, $$RMS_XASET FLD=_BIA,SRC=BIA,REG='XAB'  
.IIF NB <AIA>, $$RMS_XASET FLD=_AIA,SRC=AIA,REG='XAB'  
.IIF NB <ATA>, $$RMS_XASET FLD=_ATA,SRC=ATA,REG='XAB'  
.ENDM $XABJNL_STORE
```

```

: LEVEL-2-AND-3 MACROS TO STORE INTO FIELDS OF RMS DATA STRUCTURES AT RUN-TIME
:

```

```

; THIS IS A GENERAL PROCEDURE TO ACTUALLY GENERATE THE 'MOV' OR 'CLR' CODE

```

```

; MACRO $$RMS MOVE PTR,BLOCK,FIELD,SOURCE,FIELD2

```

```

    .LEN=0
    .IIF NB <FIELD2>, .LEN=5
    .IIF DF 'BLOCK'B'FIELD', .LEN=1
    .IIF DF 'BLOCK'W'FIELD', .LEN=2
    .IIF DF 'BLOCK'L'FIELD', .LEN=4
    .IF DF 'BLOCK'Q'FIELD'
        .NTYPE ..TYP,SOURCE
        .IF EQ <..TYP&^XFJ>-^X50
            ..TMP=..TYP&^XF
            .IF GT ..TMP-11
                .ERROR
                ..AFLG=0
                .MEXIT
            ;QUADWORD?
            ;IF REGISTER
            ;GET REG #
            ;ILLEGAL USE OF REG : SOURCE;
        .ENDC
    .ENDC
    .LEN=8
    .ENDC
; 'FIELD' IS UNDEFINED;
    .IF EQ .LEN
        .ERROR
        ..AFLG=0
        .MEXIT
    .ENDC
; Move bit field
    .IF EQ .LEN-5
        INSV 'SOURCE',#,'BLOCK'V'FIELD',#,'BLOCK'S'FIELD','BLOCK''FIELD2'PTR'
    .IFF
    .IF IDN <SOURCE><#0>
        ;IMMEDIATE ZERO
        .IIF EQ .LEN-1, CLRB 'BLOCK'B'FIELD'PTR'
        .IIF EQ .LEN-2, CLRW 'BLOCK'W'FIELD'PTR'
        .IIF EQ .LEN-4, CLRL 'BLOCK'L'FIELD'PTR'
        .IIF EQ .LEN-8, CLRQ 'BLOCK'Q'FIELD'PTR'
    .IFF
    ;ORDINARY STORE
    .IF EQ ..AFLG
        .IIF EQ .LEN-1, MOVB 'SOURCE','BLOCK'B'FIELD'PTR'
        .IIF EQ .LEN-2, MOVW 'SOURCE','BLOCK'W'FIELD'PTR'
        .IIF EQ .LEN-4, MOVL 'SOURCE','BLOCK'L'FIELD'PTR'
        .IIF EQ .LEN-8, MOVQ 'SOURCE','BLOCK'Q'FIELD'PTR'
    .IFF
    ;STORE ADDRESS
        .IIF EQ .LEN-1, MOVAB 'SOURCE','BLOCK'B'FIELD'PTR'
        .IIF EQ .LEN-2, MOVAW 'SOURCE','BLOCK'W'FIELD'PTR'
        .IIF EQ .LEN-4, MOVAL 'SOURCE','BLOCK'L'FIELD'PTR'
        .IIF EQ .LEN-8, MOVAAQ 'SOURCE','BLOCK'Q'FIELD'PTR'
    .ENDC
    ..AFLG=0
    .MEXIT
    .ENDC
    .ENDC
    ..AFLG=0

```

RMS32MAC.MAR;1

16-SEP-1984 17:06:28.^{N 6}33 Page 35

.ENDM &SRMS_MOVF

```

.MACRO $$STRI_WORD_MOVE SRC,DATA,FLD          ;MACRO TO HANDLE 3-WORD FIELDS
  .IF IDN <SRC><#0>                            ;JUST CLEAR FIELD
;SOURCE IS #0
  CLRL 'DATA'
  CLRW 4+'DATA'
  .MEXIT
  .ENDC
  .NTYPE ..TYP,SRC                            ;CHECK TYPE OF SRC
  .IF EQ <..TYP&^XFO>-^X50
;SOURCE IS A REGISTER
  ..TMP=<..TYP-^X50>+1                        ;NEXT REGISTER
  .IF GT ..TMP-12                            ;REG TOO BIG
  .ERROR                                     ;ILLEGAL USE OF REGISTER : SRC ;
  .MEXIT
  .ENDC
  MOVL 'SRC','DATA'
  $SEMIT \..TMP , <4+'DATA'>
  .MEXIT
  .ENDC
  ..TMP=..TYP&^XFO                            ;DONE W/ SRC=REG
;SOURCE IS AN AUTO DECREMENT
  .IF EQ <..TMP-^X70>                        ;ISOLATE ADDR MODE
  MOVL 'SRC',2+'DATA'
  MOVW 'SRC','DATA'
  .MEXIT
  .ENDC
;SOURCE IS AN AUTO INCREMENT
  .IF EQ <..TMP-^X80>
  MOVL 'SRC','DATA'
  MOVW 'SRC',4+'DATA'
  .MEXIT
  .ENDC
;SOURCE IS DISPLACEMENT MODE
  .LEN=0
  .IIF EQ <..TMP-^XA0>, $$STRI_DISP SRC,DATA,FLD ;BYTE DISPLACEMENT
  .IIF EQ <..TMP-^X60>, $$STRI_DISP SRC,DATA,FLD ;DEFERRED MODE
  .IIF EQ <..TMP-^XC0>, $$STRI_DISP SRC,DATA,FLD ;WORD DISPLACEMENT
  .IIF NE .LEN, .MEXIT
  .ERROR ; ** SRC ** -- ILLEGAL ADDRESSING MODE FOR FLD;
  .ENDM $$STRI_WORD_MOVE

.MACRO $SEMIT REG,DEST
  MOVW R'REG','DEST'
.ENDM $SEMIT

.MACRO $$STRI_DISP SRC,DATA,FLD
  .IRPC ..TMP,<SRC>
  .IIF NE .LEN, .MEXIT
  .IF IDN <..TMP><^>
  .ERROR ; ** SRC ** -- ILLEGAL ADDRESSING MODE FOR FLD;
  .LEN=1
  .MEXIT
  .ENDC
  .ENDR
  .IF EQ .LEN
  MOVL 'SRC','DATA'

```

.ENDM SSTR_DISP
.ENDC
.LEN=1

MOVW 4+'SRC',4+'DATA'

.....

```

.MACRO $$TWO_QUAD_MOVE SRC,DATA,FLD          ;MACRO TO HANDLE 16-BYTE FIELDS
  .IF IDN <SRC><R#0>                          ;JUST CLEAR FIELD
;SOURCE IS #0
  CLRQ 'DATA'
  CLRQ 8+'DATA'
  .MEXIT
  .ENDC
  .NTYPE ..TYP,SRC
  ..TMP=..TYP&^XFO                          ;ISOLATE ADDR MODE
  .IF EQ <..TMP-^X70>
;SOURCE IS AN AUTO DECREMENT
  MOVQ 'SRC',8+'DATA'
  MOVQ 'SRC','DATA'
  .MEXIT
  .ENDC
  .IF EQ <..TMP-^X80>
;SOURCE IS AN AUTO INCREMENT
  MOVQ 'SRC','DATA'
  MOVQ 'SRC',8+'DATA'
  .MEXIT
  .ENDC
;SOURCE IS A DISPLACEMENT MODE
  .LEN=0
  .IIF EQ <..TMP-^XA0>, $$SQUAD_DISP SRC,DATA,FLD ;BYTE DISPLACEMENT
  .IIF EQ <..TMP-^X60>, $$SQUAD_DISP SRC,DATA,FLD ;DEFERRED
  .IIF EQ <..TMP-^XC0>, $$SQUAD_DISP SRC,DATA,FLD ;WORD DISPLACEMENT
  .IIF NE .LEN, .MEXIT
  .ERROR ; ** SRC ** -- ILLEGAL ADDRESSING MODE FOR FLD;
.ENDM $$TWO_QUAD_MOVE

.MACRO $$SQUAD_DISP SRC,DATA,FLD
  .IRPC ..TMP,<SRC>
    .IIF NE .LEN, .MEXIT
    .IF IDN <..TMP><^>
      .ERROR ; ** SRC ** -- ILLEGAL ADDRESSING MODE FOR FLD;
      .LEN=1
      .MEXIT
    .ENDC
  .ENDR
  .IF EQ .LEN
    MOVQ 'SRC','DATA'
    MOVQ 8+'SRC',8+'DATA'
  .ENDC
  .LEN=1
.ENDM $$SQUAD_DISP

```



```
.MACRO $$POS SRC,REG
```

```
    $$NARG SRC
```

```
    .IF EQ ..N-1
```

```
        .IIF EQ ..FLG-2, $$TWO_QUAD_MOVE SRC , <XABS$W_POS('REG')> , _POS
        .IIF EQ ..FLG-1, $$TWO_QUAD_MOVE SRC , <XABS$W_POS(RO)> , _POS
        .IIF EQ ..FLG,   $$TWO_QUAD_MOVE SRC , <XABS$W_POS'REG'> , _POS
```

```
    .IFF
```

```
        $$TMP=0
```

```
        .IRP X,<SRC>
```

```
            .IF NB <X>
```

```
                $PEMIT X , REG , \$$TMP
```

```
            .ENDC
```

```
            $$TMP=$$TMP+1
```

```
        .ENDM
```

```
    .ENDC
```

```
.ENDM $$POS
```

```
.MACRO $PEMIT SRC,REG,N
```

```
    .IIF EQ ..FLG-2, MOVW 'SRC',XABS$W_POS'N'('REG')
```

```
    .IIF EQ ..FLG-1, MOVW 'SRC',XABS$W_POS'N'(RO)
```

```
    .IIF EQ ..FLG,   MOVW 'SRC',XABS$W_POS'N'('REG')
```

```
.ENDM $PEMIT
```

```
.MACRO $$$SIZ SRC,REG
```

```
  $$NARG SRC  
  .IF EQ ..N-1
```

```
    .IF IDN <SRC><#0>
```

```
      .IIF EQ ..FLG-2, CLRQ XAB$B_SIZ('REG')
```

```
      .IIF EQ ..FLG-1, CLRQ XAB$B_SIZ(RO)
```

```
      .IIF EQ ..FLG, CLRQ XAB$B_SIZ'REG'
```

```
    .IFF
```

```
      .IIF EQ ..FLG-2, MOVQ 'SRC',XAB$B_SIZ('REG')
```

```
      .IIF EQ ..FLG-1, MOVQ 'SRC',XAB$B_SIZ(RO)
```

```
      .IIF EQ ..FLG, MOVQ 'SRC',XAB$B_SIZ'REG'
```

```
    .ENDC
```

```
  .IFF
```

```
    $$TMP=0
```

```
    .IRP X,<SRC>
```

```
      .IF NB <X>
```

```
        $SEMIT X , REG , \$$TMP
```

```
      .ENDC
```

```
      $$TMP=$$TMP+1
```

```
    .ENDM
```

```
  .ENDC
```

```
.ENDM $$$SIZ
```

```
.MACRO $SEMIT SRC,REG,N
```

```
  .IIF EQ ..FLG-2, MOVB 'SRC',XAB$B_SIZ'N'('REG')
```

```
  .IIF EQ ..FLG-1, MOVB 'SRC',XAB$B_SIZ'N'(RO)
```

```
  .IIF EQ ..FLG, MOVB 'SRC',XAB$B_SIZ'N'('REG')
```

```
.ENDM $SEMIT
```

```

: THESE ARE THE INTERNAL MACROS TO HANDLE THE FAB

```

```

: SETUP FOR $$RMS_MOVE

```

```

.MACRO $$RMS_FSET FLD, SRC, REG
    .IIF EQ ..FLG-2, $$RMS_MOVE PTR=('REG') , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG-1, $$RMS_MOVE PTR=(RO) , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG, $$RMS_MOVE PTR='REG' , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC
.ENDM $$RMS_FSET

```

```

: Insert variable length bit field into a field.

```

```

.MACRO $$RMS_FVBSET FLD, SRC, REG, FLD2
    .IIF EQ ..FLG-2, $$RMS_MOVE PTR=('REG') , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC , FIELD2=FLD2
    .IIF EQ ..FLG-1, $$RMS_MOVE PTR=(RO) , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC , FIELD2=FLD2
    .IIF EQ ..FLG, $$RMS_MOVE PTR='REG' , BLOCK=FAB$ , FIELD=FLD , SOURCE=SRC , FIELD2=FLD2
.ENDM $$RMS_FVBSET

```

```

: MACRO TO DETERMINE IF CONSTANT KEYWORD IS DESIRED

```

```

.MACRO $$RMS_FCSET FLD, CNST, REG
    .NTYPE ..TYP, CNST
    ..MOD = 0
    .IIF EQ <..TYP-^XEF>, ..MOD = 1
    .IIF EQ <..TYP-^XCF>, ..MOD = 1
    .IIF EQ <..TYP-^XAF>, ..MOD = 1
    .IF NE ..MOD
        .NCHR ..TYP, CNST
        .IF LT <..TYP-9>
            .IF DF FAB$C 'CNST'
                $$RMS_FSET FLD, #FAB$C_'CNST', REG
            .MEXIT
        .ENDC
    .ENDC
    .ENDC
    $$RMS_FSET FLD, CNST, REG
.ENDM $$RMS_FCSET

```

```

: MACRO TO DETERMINE IF BIT KEYWORDS ARE DESIRED

```

```

.MACRO $$RMS_FVSET FLD, BITS, REG
    $$RMS_BITS FAB, <BITS>
    .IF EQ $$TMP
        .IIF EQ ..N-1, $$RMS_FSET FLD, BITS, REG ; ONLY IF ONE ARG
    .IFF
        $$RMS_FSET FLD, #$$TMP, REG
    .ENDC
.ENDM $$RMS_FVSET

```

```

: MACRO TO SIGNAL THAT THE ADDRESS IS DESIRED

```

```

.MACRO $$RMS_FASET FLD, SRC, REG
    ..AFEG=1 ; SET FLAG
    $$RMS_FSET FLD, SRC, REG

```

RMS32MAC.MAR;1

16-SEP-1984 17:06:28.^H33⁷ Page 42

.ENDM SSRMS_FASET

```

; THIS MACRO COUNTS THE NUMBER OF ARGS PASSED IT
;
.MACRO $$NARG ARG1,ARG2,ARG3,ARG4,ARG5,ARG6,ARG7,ARG8,ARG9,ARG10,-
ARG11,ARG12,ARG13,ARG14,ARG15,ARG16,ARG17,ARG18,ARG19,ARG20,-
ARG21,ARG22,ARG23,ARG24,ARG25,ARG26,ARG27,ARG28,ARG29,ARG30,-
ARG31,ARG32,ARG33,ARG34,ARG35,ARG36,ARG37,ARG38,ARG39,ARG40

.NARG ..N
.ENDM $$NARG

.MACRO $$RMS BITS BLK,BITS
$$TMP=0
$$NARG BITS
.IF EQ ..N-1
.NTYPE ..TYP,BITS
..MOD = 0
.IIF EQ <..TYP-^XEF>, ..MOD = 1
.IIF EQ <..TYP-^XCF>, ..MOD = 1
.IIF EQ <..TYP-^XAF>, ..MOD = 1
.IF NE ..MOD
..TYP = 0
.IRPC X,BITS
.IF IDN X,<+>
..TYP = 1
.MEXIT
.ENDC
.ENDM
.IF EQ ..TYP
.NCHR ..TYP,BITS
.IF LE <..TYP-9>
.IF DF BLK'$V_'BITS
$$TMP=<1@BLK'$V_'BITS>
.ENDC
.ENDC
.ENDC
.IFF
.IRPC X,<BITS>
.NTYPE ..TYP,X
..MOD = 0
.IIF EQ <..TYP-^XEF>, ..MOD = 1
.IIF EQ <..TYP-^XCF>, ..MOD = 1
.IIF EQ <..TYP-^XAF>, ..MOD = 1
.IF NE ..MOD
.NCHR ..TYP,X
.IF LE <..TYP-9>
.IF DF BLK'$V_'X
$$TMP=$$TMP!<1@BLK'$V_'X>
.IFF
.ERROR ;UNDEFINED BIT VALUE: X;
.ENDC
.IFF
.ERROR ;BIT VALUE TOO LONG: X;
.ENDC
.IFF

```

;WRONG ADDRESSING MODE: X;

.ENDM \$SRMS_BITS
.ENDC
.ENDM
.ENDC
.ERROR



```

: This is a macro to determine which type of addressing should be used.

```

```

: If user uses displacement mode with a displacement of zero, the .ntype
: directive returns register deferred mode. Therefore we must use r0 to
: avoid problems in the $$rms_move macro, which would otherwise generate
: bad offset definitions.

```

```

.MACRO $$TYPE REG
.NTYPE ..TYP,REG                ;SEE IF PTR IS REGISTER
.IF NE <..TYPE^XFO>-^X50
    .IF EQ <..TYPE^XFO>-^X60    ;IF (RN)
        ..FLG=0                ;JUST WANT REG
        ..N=<..TYPE^XOF>
        .IRP X,<\..N>
            .IF DIFFERENT (R'X) REG ;IF DIFFERENT, ZERO DISP...
                .FLG=1          ;...SO USE R0 INSTEAD
                MOVAL 'REG',R0
            .ENDC
        .ENDR
    .IFF
        .IF EQ ..TYP            ;CHECK IF IMMEDIATE
            MOVL 'REG',R0
        .IFF
            .IF EQ <..TYP-^X1F> ;ALSO IMMEDIATE
                MOVL 'REG',R0
            .IFF
                MOVAL 'REG',R0
            .ENDC
        .ENDC
        ..FLG=1                ;WANT (R0)
    .ENDC
.IFF
    ..FLG=2                    ;WANT ('REG')
.ENDC
.ENDM $$TYPE

```

; THESE ARE THE MACROS TO HANDLE THE RAB

```
.MACRO $$RMS_RSET FLD, SRC, REG
    .IIF EQ ..FLG-2, $$RMS_MOVE PTR=('REG') , BLOCK=RAB$ , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG-1, $$RMS_MOVE PTR=(R0) , BLOCK=RAB$ , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG, $$RMS_MOVE PTR='REG' , BLOCK=RAB$ , FIELD=FLD , SOURCE=SRC
.ENDM $$RMS_RSET
```

```
.MACRO $$RMS_RCSET FLD, CNST, REG
    .NTYPE ..TYP, CNST
    ..MOD = 0
    .IIF EQ <..TYP-^XEF>, ..MOD = 1
    .IIF EQ <..TYP-^XCF>, ..MOD = 1
    .IIF EQ <..TYP-^XAF>, ..MOD = 1
    .IF NE ..MOD
        .NCHR ..TYP, CNST
        .IF LT <..TYP-9>
            .IF DF RAB$C 'CNST'
                $$RMS_RSET FLD, #RAB$C_'CNST', REG
            .MEXIT
        .ENDC
    .ENDC
.ENDC
$$RMS_RSET FLD, CNST, REG
.ENDM $$RMS_RCSET
```

```
.MACRO $$RMS_RVSET FLD, BITS, REG
    $$RMS_BITS RAB, <BITS>
    .IF EQ $$TMP
        .IIF EQ ..N-1, $$RMS_RSET FLD, BITS, REG ; ONLY IF ONE ARG
    .IFF
        $$RMS_RSET FLD, #$$TMP, REG
    .ENDC
.ENDM $$RMS_RVSET
```

; MACRO TO SIGNAL THAT THE ADDRESS IS DESIRED

```
.MACRO $$RMS_RASET FLD, SRC, REG
    ..AF[ G=1 ; SET FLAG
    $$RMS_RSET FLD, SRC, REG
.ENDM $$RMS_RASET
```


; THESE ARE THE MACROS TO HANDLE ALL OF THE XAB'S

```
.MACRO $$RMS_XSET FLD, SRC, REG
    .IIF EQ ..FLG-2, $$RMS_MOVE PTR=('REG') , BLOCK=XABS , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG-1, $$RMS_MOVE PTR=(RO) , BLOCK=XABS , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG, $$RMS_MOVE PTR='REG' , BLOCK=XABS , FIELD=FLD , SOURCE=SRC
.ENDM $$RMS_XSET
```

```
.MACRO $$RMS_XCSET FLD, CNST, REG
    .NTYPE ..TYP, CNST
    ..MOD = 0
    .IIF EQ <..TYP-^XEF>, ..MOD = 1
    .IIF EQ <..TYP-^XCF>, ..MOD = 1
    .IIF EQ <..TYP-^XAF>, ..MOD = 1
    .IF NE ..MOD
        .NCHR ..TYP, CNST
        .IF LT <..TYP-9>
            .IF DF XAB$C 'CNST'
                $$RMS_XSET FLD, #XAB$C_'CNST', REG
            .MEXIT
        .ENDC
    .ENDC
    .ENDC
    $$RMS_XSET FLD, CNST, REG
.ENDM $$RMS_XCSET
```

```
.MACRO $$RMS_XVSET FLD, BITS, REG
    $$RMS_BITS XAB, <BITS>
    .IF EQ $$TMP
        .IIF EQ ..N-1, $$RMS_XSET FLD, BITS, REG ; ONLY IF ONE ARG
    .IFF
        $$RMS_XSET FLD, #$$TMP, REG
    .ENDC
.ENDM $$RMS_XVSET
```

; MACRO TO SIGNAL THAT THE ADDRESS IS DESIRED

```
.MACRO $$RMS_XASET FLD, SRC, REG
    ..AF[G=1 ; SET FLAG
    $$RMS_XSET FLD, SRC, REG
.ENDM $$RMS_XASET
```

```
; THESE ARE THE MACROS TO HANDLE THE NAMBLOCK
```

```
.MACRO $$RMS_NSET FLD, SRC, REG
    .IIF EQ ..FLG-2, $$RMS_MOVE PTR=('REG') , BLOCK=NAMS , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG-1, $$RMS_MOVE PTR=(R0) , BLOCK=NAMS , FIELD=FLD , SOURCE=SRC
    .IIF EQ ..FLG, $$RMS_MOVE PTR='REG' , BLOCK=NAMS , FIELD=FLD , SOURCE=SRC
.ENDM $$RMS_NSET
```

```
.MACRO $$RMS_NCSET FLD, CNST, REG
    .NTYPE ..TYP, CNST
    ..MOD = 0
    .IIF EQ <..TYP-^XEF>, ..MOD = 1
    .IIF EQ <..TYP-^XCF>, ..MOD = 1
    .IIF EQ <..TYP-^XAF>, ..MOD = 1
    .IF NE ..MOD
        .NCHR ..TYP, CNST
        .IF LT <..TYP-9>
            .IF DF NAMSC 'CNST'
                $$RMS_NSET FLD, #NAMSC_'CNST', REG
            .MEXIT
        .ENDC
    .ENDC
.ENDM $$RMS_NCSET
```

```
.MACRO $$RMS_NVSET FLD, BITS, REG
    $$RMS_BITS NAM, <BITS>
    .IF EQ $$TMP
        .IIF EQ ..N-1, $$RMS_NSET FLD, BITS, REG ; ONLY IF ONE ARG
    .IFF
        $$RMS_NSET FLD, #$$TMP, REG
    .ENDC
.ENDM $$RMS_NVSET
```

```
; MACRO TO SIGNAL THAT THE ADDRESS IS DESIRED
```

```
.MACRO $$RMS_NASET FLD, SRC, REG
    .AF[G=1 ; SET FLAG
    $$RMS_NSET FLD, SRC, REG
.ENDM $$RMS_NASET
```


RMSCALLS
MAR

RMSTDXLNK
R32

RM532MAC
MAR

RMSTDXMAC
R32

UTLDEF
R32

UTLDEFUND
R32

RMSTDXDEF
R32

NTOMACROS
MAR

RMSLST

RMSMCMAC
MAR

RMSINTDEF
LST