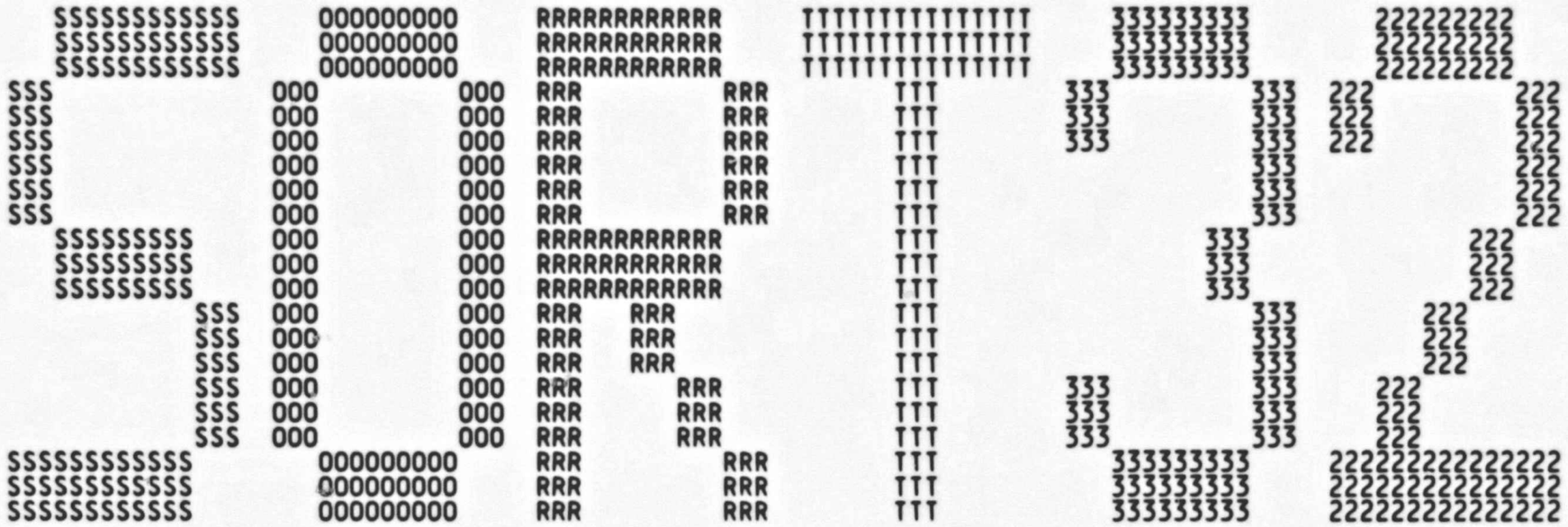
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSS 000000000 RRRRRRRRRRR TTTTTTTTTTTTTTT 333333333 222222222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 222 222
SSS 000 000 RRR RRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 222
SSSSSSSSS 000 000 RRRRRRRRRRR TTT 333 222
SSSSSSSSS 000 000 RRRRRRRRRRR TTT 333 222
SSSSSSSSS 000 000 RRRRRRRRRRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 222
SSS 000 000 RRR RRR TTT 333 333 222
SSS 000 000 RRR RRR TTT 333 333 222
SSS 000 000 RRR RRR TTT 333 333 222
SSSSSSSSSSS 000000000 RRR RRR TTT 333333333 222222222222222
SSSSSSSSSSS 000000000 RRR RRR TTT 333333333 222222222222222
SSSSSSSSSSS 000000000 RRR RRR TTT 333333333 222222222222222

FILEID**CHKPNT

```
  CCCCCCCC  HH      HH  KK      KK  PPPPPPPP    NN      NN  TTTTTTTTTT
  CCCCCCCC  HH      HH  KK      KK  PPPPPPPP    NN      NN  TTTTTTTTTT
CC          HH      HH  KK      KK  PP      PP  NN      NN      TT
CC          HH      HH  KK      KK  PP      PP  NN      NN      TT
CC          HH      HH  KK    KK    PP      PP  NNNN    NN      TT
CC          HH      HH  KK    KK    PP      PP  NNNN    NN      TT
CC          HHHHHHHHHH  KKKKKK      PPPPPPPP    NN  NN  NN      TT
CC          HHHHHHHHHH  KKKKKK      PPPPPPPP    NN  NN  NN      TT
CC          HH      HH  KK    KK    PP          NN    NNNN      TT
CC          HH      HH  KK    KK    PP          NN    NNNN      TT
CC          HH      HH  KK      KK  PP          NN      NN      TT        ....
CC          HH      HH  KK      KK  PP          NN      NN      TT        ....
  CCCCCCCC  HH      HH  KK      KK  PP          NN      NN      TT        ....
  CCCCCCCC  HH      HH  KK      KK  PP          NN      NN      TT        ....

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
```

```
ROUTINE PSH_CHECKPOINT                  ! Post-checkpoint handler
        (
        CONTEXT: REF CTX_BLOCK
        ) =

!
! 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.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called before a checkpoint, and moves the RDBs
!       from WFB_FREE to WFB_NOUSE.
!
!       This is done by a post-checkpoint handler so that user-written calls
!       to CHK$CHKPNT also cause these blocks to be marked as having no use.
!
!       Note that we check that this routine is not called from an AST routine,
!       because the WFB_FREE and WFB_NOUSE lists aren't manipluated carefully.
!
! FORMAL PARAMETERS:
!
!       CONTEXT - Address of the context area
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
```

```
! ROUTINE VALUE:
!
!       SS$_NORMAL
!
! SIDE EFFECTS:
!
!       NONE
!--
    BEGIN
    GLOBAL REGISTER
        CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);
    LOCAL
        WFB:    REF WFB_BLOCK;

    ! Grab the address of the context area
    !
    CTX = CONTEXT[BASE_];

    ! Move the RDBs from WFB_FREE to WFB_NOUSE
    !
    ASSERT_(%FIELDEXPAND(WFB_FREE,0) NEQ %FIELDEXPAND(WFB_NOUSE,0))
    WFB = CTX[S_WRK_WFB] - BBLOCK[0,WFB_NEXT];
    WHILE (WFB = .WFB[WFB_NEXT]) NEQ 0 DO
        MOVE_ALL_RDBS_( WFB[WFB_FREE], WFB_NOUSE );

    RETURN SS$_NORMAL;
    END;
```

```
GLOBAL ROUTINE SOR$$CHECKPOINT: CAL_CTXREG NOVALUE =    ! Take a checkpoint
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine causes a checkpoint to be taken.
!
! FORMAL PARAMETERS:
!
!       NONE
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       NONE
!
! SIDE EFFECTS:
!
!       NONE
!--
    BEGIN
    EXTERNAL REGISTER
        CTX =   COM_REG_CTX:    REF CTX_BLOCK_(S_FIELDS);

! Reset our counter to a very large value.  If we are really checkpointing,
! the post_checkpoint routine will set it to a more reasonable value.
!
CTX[COM_COUNTDOWN] = 1^(%BPVAL-1)-1;        ! Largest positive value

! Are we checkpointing?
!
IF .CTX[COM_NOCHKPNT]
THEN
    RETURN
ELIF CHK$CHKPNT NEQ 0
THEN
    BEGIN
    LOCAL
        STATUS;
    STATUS = CHK$CHKPNT();
    IF .STATUS EQL CHK$_NOTINIT
    THEN
        BEGIN
        !
        ! Warn the user, since he requested checkpointing
        !
        ! Perhaps we should fatal this, since we shouldn't get this far.
        !
```

```
            SOR$$ERROR(SOR$_SHR_SYSERROR AND NOT STS$M_SEVERITY OR STS$K_WARNING,
                    0, .STATUS);
            END
        ELIF .STATUS
        THEN
            BEGIN
            !
            ! Reset our counter, and inform the user
            !
            CTX[COM_COUNTDOWN] = TUN_K_COUNTDOWN;

            %IF %SWITCHES(DEBUG)
            %THEN
                BEGIN
                EXTERNAL ROUTINE LIB$SHOW_TIMER: ADDRESSING_MODE(GENERAL);
                LIB$SHOW_TIMER();
                END;
            %FI
            SOR$$ERROR(SOR$_CHKPNT, 1, 0);

            RETURN;
            END
        ELSE
            SOR$$FATAL(SOR$_SHR_SYSERROR, 0, .STATUS);

        END;

    ! Some error occured.
    ! Call the post_checkpoint routine to move RDBs from WFB_FREE to WFB_NOUSE.
    ! Indicate that we shouldn't mess with any more checkpoints.
    !
    PSH_CHECKPOINT(CTX[BASE_]);
    CTX[COM_NOCHKPNT] = TRUE;

    END;
```

COM
REQ

SMGUSRTRM
LIS

SORLIB
REQ

SORTSHR
MAP

DEFS0
REQ

DKS
REQ

SORT32

SRTSPC
REQ

CDDTYPE
R32

SORTMERGE
MAP

SMGVECTOR
LIS

CHKPNT
REQ

CDDMAC
R32

SFKEYWRD
REQ

RECSYM
R32

SRTTRN
MAR

OPCODES
REQ