```
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Module: Exception Handling
;>
;>      PROCEDURE: Abort
;>      PROCEDURE: Status_Call
;>      PROCEDURE: Reset_StMach
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Abort
;>
;>      This procedure is the 'garbage-collector' for this set of
;>      firmware. Abort is the place where all routines 'goto' if
;>      things get so screwed up that they can no longer cope.
;>
;>      Inputs: { none; address of caller on stack }
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       Except_Stat.Recovery := False
;>       DiskStat.Parked := False
;>       DiskStat.On_Track := False
;>       Clr_BankSwitch
;>       Set_RamBank( Ram0 )
;>       Registers !rE and !rF get address of  caller
;>       FOR i := 0 TO $F DO
;>        Abort_Stat[ i ] := Register[ Rp+i ]
;>       Initialize internal and external stacks
;>       ZeroBlock( ReadArray )
;>       Initialize Write_Buffer Fence
;>       SetStatus( Abort )
;>       GoTo RdLeave
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Abort:
                Di

                Pop     !rE ;remember who called us
                Pop     !rF
                Push    Rp  ;save context
                Srp     #Wrk_Scr
```

```
                Call    Clr_BankSwitch

                Pop     !r0 ;get caller's context from stack
                Ld      !r1,#16 ;load 16 locations
                Ld      !rE,#.HIBYTE. Abort_Stat
                Ld      !rF,#.LOWBYTE. Abort_Stat

Abort_Lp1:      Ldei    @!!rE,@!r0
                Djnz    !r1,Abort_Lp1

                Srp     #Wrk_Sys
                Clr     Sph ;clean up the stack
                Ld      Spl,#Stack_Top

                Call    ExtStk_Vector

                Call    ZrRd_Vector

                 Ld     !rE,#.HIBYTE. WBlkFence ;re-write the write block fence
                 Ld     !rF,#.LOWBYTE. WBlkFence
                Call    LdPw_Vector

                Tm      DiskStat,#MultiBlk
                Jr      Z,Abt_Stat_Ld

                Call    UpDate_Logical

Abt_Stat_Ld:    Call    SS_Abort

                Tm      Port2,#Bsy ;check if BSY is set
                Jp      Nz,RdL_Vector

                Srp     #Wrk_Io
                Ld      !rA,#Init_Response
                Ld      !rB,#00
                Jp      Clr_Bsy1


UpDate_Logical:
                Call    LL_Vector
                Add     !rE,!r5
                Adc     !rD,#0
                Adc     !rC,#0
                Ld      !r0,#Wrk_Sys+$C
                Ld      !r1,#3
                Ld      !r2,#.HIBYTE. LogicalBlock
                Ld      !r3,#.LOWBYTE. LogicalBlock

UpD_Lgcl_Lp:    Ldei    @!!r2,@!r0
                Djnz    !r1,UpD_Lgcl_Lp

                And     DiskStat,#$FF-MultiBlk
                Ret

                .LSTOFF
                .DO    External
                .LSTON
                .Page
;>>>
```

```
;>
;>      SS_Abort: SetStatus( Abort )
;>                SetStatus( OperationFailed )
;>
;>      SS_OpFail: SetStatus( OperationFailed )
;>
;>      SS_ReadErr: SetStatus( ReadErrCount )
;>
;>      SS_NoHdr: SetStatus( NoHeaderFound )
;>
;>      SS_SprWarn: SetStatus( SprBlk_Warn )
;>
;>>>
```

```
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       Z8TestL := True
;>       FOR i := 16 DOWNTO 0 DO
;>              Zrwck := True
;>              Zrwck := False
;>       Z8TestL := False
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Reset_StMach:
                Push    Rp ;save context
                Srp     #Wrk_lo

                And     !Port0,#$FF-Not_ZTestL ;asset Z8TestL

                Ld      !rE,#16 ;i := 16

Res_StM_Lp:     Or      !Port3,#Zrwck ;set clock
                And     !Port3,#$FF-Zrwck ;clear clock
                Djnz    !rE,Res_StM_Lp

                Or      !Port0,#ZTestL ;deassert Z8TestL
                Or      !Port0,#Zrwck ;set clock

                Pop     Rp
                Ret

                .LSTOFF
                .DO     Internal
                .LSTON
Int_Version:    .DB     $00, $00
                .LSTOFF
```