```
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Module: Read Routines
;>
;>      This module contains all but the very most primitive of
;>      of procedures ( the routines that are resident are listed
;>      in the external spec's ) that concern themselves with performing
;>      a read operation.
;>
;>      FUNCTION ReadBlock( Parent : BYTE { !r8 } ) :
;>                          BOOLEAN
;>                          Status : BYTE { !r0 }
;>                          RdErrCnt : BYTE { !r1 }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: ReadBlock
;>
;>      This function assumes that the heads are positioned over the
;>      correct track and reads the block of data whose header
;>      matches the header that is made up of the cylinder, head
;>      and sector information that is stored in memory.
;>
;>      Inputs:
;>              Parent : BYTE { !r8 }
;>
;>      Outputs:
;>              ReadBlock : BOOLEAN { Zero flag, true if error in ReadBlock }
;>              Status    : BYTE { !r0 }
;>              RdErrCnt  : BYTE { !r1 }
;>
;>      Global Variables Used:
;>              Cylinder, Head, Sector, Recovery
;>
;>      Local Variables Used:
;>              RdRetryCnt : BYTE { !r8 }
;>              RdError    : BOOLEAN { !r9/bit 7 }
;>              RdExcept   : BOOLEAN { !r9/bit 6 }
;>              RdSuccess  : BOOLEAN { !r9/bit 5 }
;>              NoHdrFound : BOOLEAN { !r9/bit 4 }
;>              SectorsRead: BYTE { !rA }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       SetDeadManTimer( ReadBlock, Parent )
;>       RdRetryCnt := 10
;>       RdErrCnt := 0
;>       RdError := False
;>       RdExcept := False
;>       NoHeaderFound := False
;>       SectorsRead := 2 * NbrSctrs { try to find header for two rotations }
;>       REPEAT
;>         RHeader[ 1 ] := HiCylinder
```

```
;>         RHeader[ 2 ] := LoCylinder
;>         RHeader[ 3 ]/bits 7:6 := Head
;>         RHeader[ 3 ]/bits 5:0 := Sector
;>         RHeader[ 4 ] := Invert( RHeader[ 1 ] )
;>         RHeader[ 5 ] := Invert( RHeader[ 2 ] )
;>         RHeader[ 6 ] := Invert( RHeader[ 3 ] )
;>         ReadArray[ RDummy-1 ] := 0
;>       _
;>      /
;>   R  |  Set-up external ram address counter for READ
;>   E  |  MselO:1 := Disk <--> Mem
;>   S  |  WHILE SectorMark DO BEGIN END
;>   I  |  StartL := True
;>   D  |  WHILE NOT( SectorDnL ) DO BEGIN END
;>   E  |  Status := Status_Port
;>   N  |  StartL := False
;>   T  |  MselO:1 := Z8 <--> Mem
;>      \_
;>
;>         CASE Status.State OF
;>              NormalEndState        : RdSuccess := True
;>
;>              NoMatchingHeaderFound : RdSuccess := False
;>                                      RdError := True
;>                                      NoHeaderFound := True
;>
;>              AbnormalState         : Reset_StateMachine
;>                                      Abort
;>
;>         IF Status.ServoErr OR NOT( Status.ServoRdy )
;>          THEN
;>               RdError := True
;>               RdExcept := True
;>
;>         IF Status.CrcErr AND RdSuccess
;>          THEN
;>              RdError := True
;>              RdErrCnt := RdErrCnt + 1
;>              RdRetryCnt := RdRetryCnt - 1
;>          ELSE
;>              IF RdError
;>               THEN
;>                   BlockMove( Buffer2, RBuffer1 )
;>                   RdRetryCnt := RdRetryCnt - 1
;>
;>
;>       UNTIL NOT( Recovery ) OR NOT( RdError ) OR ( RdRetryCnt = 0 ) OR
;>                     RdExcept OR ( NoHeaderFound )
;>       ClearDeadManTimer
;>       Status := !r9
;>      END
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

ReadBlock:
                Clr     !r9     ;clear booleans
                Ld      !r8,#10 ;RdRetryCnt := 10
                Clr     !rB     ;ErrCnt := 0
```

```
                And     DiskStat,#$FF-Wr_Op ;make certain we are reading

RdBlk_Rpt:       Ld     !r2,#.HIBYTE. RHeader ;initialize gaps
                 Ld     !r3,#.LOWBYTE. RHeader
                Call    Load_Header
                Ld      !r0,#0
                Lde     @!!r2,!r0

                Ld      !r2,#.HIBYTE. ( RDummy-1 )
                Ld      !r3,#.LOWBYTE. ( RDummy-1 )
                Lde     @!!r2,!r0
                Incw    !!r2
                Lde     @!!r2,!r0
                Jr      Do_Read

Read_Fast:      Clr     !r9     ;clear booleans
                Ld      !r8,#10 ;RdRetryCnt := 10
                Clr     !rB     ;ErrCnt := 0
                And     DiskStat,#$FF-Wr_Op ;make certain we are reading

                Ld      !r2,#.HIBYTE. ( RHeader+2 ) ;get location of Sector
                Ld      !r3,#.LOWBYTE. ( RHeader+2 )
                Lde     !r0,@!!r2 ;get current Head/Sector value
                And     !r0,#$C0 ;mask out old sector value
                Or      !r0,Sector ;merge in new sector
                Lde     @!!r2,!r0 ;and store it
                Add     !r3,#3 ;get location of Inverse Head/Sector
                Com     !r0
                Lde     @!!r2,!r0 ;and store it, too!

Do_Read:        Call    Rd_Resident     ;go internal to the Z8

                Ld      !r1,!r0         ;CASE Status.State
                And     !r1,#$0F
                Or      !rA,!rA
                Jr      Z,RdBlk_NoHdr
                Cp      !r1,#Norm_State
                Jr      Nz,RdBlk_AbNorm

RdBlk_Norm:     Or      !r9,#RdSuccess

                Ld      !r1,!r0         ;IF ServorErr OR NOT( ServoRdy )
                Tm      !r1,#ServoErr
                Jr      Nz,Rd_ServoErr
                Tm      !r1,#ServoRdy
                Jr      Z,Rd_ServoErr

Rd_ServoOk:     Tm      !r0,#CrcErrL    ;IF Status.CrcErr
                Jr      Z,Rd_BadCrc
                Tm      !r0,#WrtNvldL   ; OR Status.EccErr
                Jr      Z,Rd_BadEcc

Rd_NoCrcErr:    Tm      !r9,#RdError    ; THEN IF RdError
                Jr      Nz,RdBlk_Bmove

RdBlk_Until:    Tm      Expt_Status,#Recovery
                Jr      Z,RdBlk_End
                Tm      !r9,#RdError
                Jr      Nz,RdB_Rpt1

RdBlk_End:      Ld      RdErrCnt,!rB
                Ld      !r0,!r9 ;send status back to caller
                Ld      RdStat,!r9
```

```
                Tcm     !r0,#RdError ;set zero flag if error
                Ret


RdB_Rpt1:        Ld     !r2,#.HIBYTE. ZeroHeader
                 Ld     !r3,#.LOWBYTE. ZeroHeader
                Call    Bank_Call
                Jp      RdBlk_Rpt

RdBlk_AbNorm:    Call   Reset_StMach
                 Ld     !rA,!r0
                Call    Abort

RdBlk_NoHdr:    And     !r9,#$FF-RdSuccess
                Or      !r9,#Error
                Tm      !rB,#Hdr_MisMatch
                Jr      Nz,Rd_HdrErr
                Or      !rB,#Hdr_MisMatch

                Tm      Excpt_Stat,#Recovery ;auto offset ONLY if recovery on
                Jr      Z,RdBlk_Until

                Call    ReSeek              ;set auto_offset
                Jr      RdBlk_Until

Rd_HdrErr:      Or      !r9,#RdNoHdrFnd + RdError
                Jr      RdBlk_End

Rd_ServoErr:    Or      !r9,#RdError + RdSrvoErr ; THEN RdError AND RdServoErr
                Jr      RdBlk_End

Rd_BadCrc:      Or      !r9,#RdError+RdCrcErr    ; ELSE
                Inc     !rB
                Or      !rB,#CrcStat
                Tm      !r0,#WrtNvldL   ;check for Ecc error too
                Jr      Nz,Rd_B_Crc_1
RD_Bad1:        Or      !rB,#EccStat

                Tm      Excpt_Stat,#Recovery    ;ReSeek only if recovery is on
                Jr      Z,Rd_B_Crc_1

                Tm      DiskStat,#Offset_On ;set auto offset if needed
                Jr      Nz,Rd_B_Crc_1

                Call    ReSeek

Rd_B_Crc_1:     Djnz    !r8,RdBlk_Until
                Jr      RdBlk_End

Rd_BadEcc:      Or      !r9,#RdError+RdCrcErr
                Inc     !rB
                Jr      Rd_Bad1

RdBlk_Bmove:     Ld     !r2,#.HIBYTE. RBuf_To_Buf2
                 Ld     !r3,#.LOWBYTE. RBuf_To_Buf2
                Call    Bank_Call
                Jr      Rd_B_Crc_1

                .LSTOFF
```