```
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Module: Resident Misc.
;>
;>      PROCEDURE ReSeek
;>      PROCEDURE New_Seek( Cylinder : WORD { !!rC }
;>                          Head     : BYTE { !rD }
;>                          Sector   : BYTE { !rE }
;>                        )
;>      PROCEDURE Load_Header( VAR BufferPtr : PTR { !!r2 } )
;>      FUNCTION Load_Logical : LogicalBlock : 3 BYTES { !rC:E }
;>      FUNCTION Chk_Chk_Byte( SourcePtr : PTR { !!rE }
;>                             SourceLength : BYTE { !!r8 }
;>                           ) :
;>                           BOOLEAN
;>      PROCEDURE Gen_Chk_Byte( SourcePtr : PTR { !!rE }
;>                              SourceLength : BYTE { !!r8 }
;>                            )
;>      PROCEDURE ClearStatus
;>      PROCEDURE Ld_LgclBlk( Offset : BYTE { !r0 } )
;>      PROCEDURE Set_SeekNeeded
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN


ReSeek:         Ld      SeekType,#Access_Offset
                Decw    SeekCount ;account for zero track seek

                Ld      !rC,Cylinder
                Ld      !rD,Cylinder+1
                Ld      !rE,Head
                Ld      !rF,Sector

New_Seek:        Ld     !r2,#.HIBYTE. Seek_Vector
                 Ld     !r3,#.LOWBYTE. Seek_Vector
                Call    Bank_Call
                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Load_Header
;>
;>      This procedure is responsible for generating ( and loading
;>      into the disk buffer ) the header that will be used to identify
;>      the correct block to operate on.
;>
;>      Inputs:
;>              BufferPtr: PTR { !!r2 }
;>
;>      Outputs:
```

```
;>              BufferPtr + 5 : PTR { !!r2 }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       @BufferPtr( 0 ) := HiCylinder
;>       @BufferPtr( 1 ) := LoCylinder
;>       @BufferPtr( 2 )/bits 7:6 := Head
;>       @BufferPtr( 2 )/bits 5:0 := Sector
;>       @BufferPtr( 3 ) := Invert( @BufferPtr( 0 ) )
;>       @BufferPtr( 4 ) := Invert( @BufferPtr( 1 ) )
;>       @BufferPtr( 5 ) := Invert( @BufferPtr( 2 ) )
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Load_Header:
                Ld      !r0,#Cylinder     ;load header: hi cylinder
                Ldei    @!!r2,@!r0

                Ldei    @!!r2,@!r0        ;load lo cylinder

                Ld      !r1,@!r0         ;Head
                Swap    !r1
                Rl      !r1
                Rl      !r1

                Inc     !r0
                Or      !r1,@!r0         ;merge in Sector

                Lde     @!!r2,!r1        ;load Head/Sector
                Incw    !!r2

                Ld      !r0,Cylinder     ;Not( hi cylinder )
                Com     !r0
                Lde     @!!r2,!r0
                Incw    !!r2

                Ld      !r0,Cylinder+1   ;Not( lo cylinder )
                Com     !r0
                Lde     @!!r2,!r0
                Incw    !!r2

                Com     !r1              ;Not( Head/Sector )
                Lde     @!!r2,!r1
                Incw    !!r2

                Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: Load_Logical
;>
```

```
;>      This function returns the last logical block number
;>      requested by the host.
;>
;>      Inputs: { none }
;>
;>      Outputs:
;>              LogicalBlockNumber : 3 BYTES { !rC:E }
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Load_Logical:
                Ld      !r2,#.HIBYTE. LogicalBlock
                Ld      !r3,#.LOWBYTE. LogicalBlock
                Ld      !r1,#3
                Ld      !r0,Rp
                Or      !r0,#$0C

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

                Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: Chk_Chk_Byte
;>      Inputs:
;>              SourcePtr : PTR, { !!rE }
;>              SourceLength : BYTE { !r8 }
;>
;>      Outputs:
;>              If CheckBytes Match
;>               Then Zero_Flag := True
;>               Else Zero_Flag := False
;>
;>      Algorithm:
;>       Begin
;>        Temp := 0
;>        For i := 1 To SourceLength Do
;>         Temp := Temp + @SourcePtr
;>         SourcePtr := SourcePtr + 1
;>        Compare ( Temp, @SourcePtr )
;>       End
;>
;>      Temp = !r0
;>
;>      Scratch Register = !r2
;>
;>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
```

```
                .FIN

Chk_Chk_Byte:
                Clr     !r0

ChkB_Lp1:       Lde     !r2,@!!rE
                Add     !r0,!r2
                Incw    !!rE
                Djnz    !r8,ChkB_Lp1

                Com     !r0
                Lde     !r1,@!!rE
                Cp      !r0,!r1         ;set/clear zero flag
                Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Gen_Chk_Byte
;>
;>      Inputs:
;>              SourcePtr : PTR, { !!rE }
;>              SourceLength : BYTE { !r8 }
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>       Begin
;>        Temp := 0
;>        For i := 1 To SourceLength Do
;>         Temp := Temp + @SourcePtr
;>         SourcePtr := SourcePtr + 1
;>        Temp := NOT( Temp )
;>        @SourcePtr := Temp
;>       End
;>
;>      Temp = !r0
;>
;>      Scratch Register = !r2
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Gen_Chk_Byte:
                Clr     !r0

Gen_ChkB_Lp1:   Lde     !r2,@!!rE
                Add     !r0,!r2
                Incw    !!rE
                Djnz    !r8,Gen_ChkB_Lp1

                Com     !r0

                Lde     @!!rE,!r0
                Ret
```

```
                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: ClearStatus
;>
;>      This procedure clears all the bytes within the status array.
;>
;>      Inputs: { none }
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       FOR i := 1 TO Length( StatusArray ) DO
;>        StatusArray[ i ] := 0
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

ClearStatus:
                Ld      !r2,#.HIBYTE. CStatus0
                Ld      !r3,#.LOWBYTE. CStatus0
                Clr     !r0
                Ld      !r1,#( End_CStatus - CStatus0 ) ;clear all status bytes

Clr_Stat:       Lde     @!!r2,!r0
                Incw    !!r2
                Djnz    !r1,Clr_Stat
                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Ld_LgclBlk   { Load Logical Block }
;>
;>      This procedure is used to extract the LogicalBlockNumber
;>      information from a command string ( sent by the host ) and
;>      move it into the global variable LogicalBlock.
;>
;>      Inputs:
;>              Offset : BYTE { !r0 }
;>
;>      Outputs: { none }
;>
;>      Global Variables Changed:
;>              LogicalBlock
;>
;>      Algorithm:
;>
```

```
;>      BEGIN
;>       FOR i := 1 TO Length( LogicalBlock ) DO
;>        LogicalBlock[ i ] := CmndArray.LogicalBlock[ i ]
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Ld_LgclBlk:
                Ld      ScrReg0,!r0 ;pass parameter
                Push    Rp ;save context
                Srp     #Wrk_Scr

                 Ld     !r2,#.HIBYTE. CStatus4
                 Ld     !r3,#.LOWBYTE. CStatus4
                 Add    !r3,!r0 ;add offset to BlockNumber
                 Adc    !r2,#0
                 Ld     !rE,#.HIBYTE. LogicalBlock
                 Ld     !rF,#.LOWBYTE. LogicalBlock

                 Ld     !r1,#4 ;move 4 bytes
Ld_Lgcl_Lp:      Lde    !r0,@!!r2
                 Lde    @!!rE,!r0
                 Incw   !!r2
                 Incw   !!rE
                 Djnz   !r1,Ld_Lgcl_Lp

                Pop     Rp ;context switch
                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Set_SeekNeeded
;>
;>      This procedure acts in much the same fashion for the
;>      LogicalBlockCache as DiskStat.On_Track does for simple
;>      seeks. It keeps a legitimate seeks between tow requests
;>      for the same block number from returning a bogus result.
;>
;>      Inputs: { none }
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       FOR i := 1 TO CacheLength DO
;>        CachStat[ i ].SeekNeeded := True
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
```

```
                .LSTON
                .Page
                .FIN

Set_SkNeeded:
                Ld      !r2,#.HIBYTE. CachStat
                Ld      !r3,#.LOWBYTE. CachStat
                Ld      !r1,#CacheLength

S_SeekN_Lp:     Lde     !r0,@!!r2 ;get array value
                Or      !r0,#CachSeek
                Lde     @!!r2,!r0
                Incw    !!r2
                Djnz    !r1,S_SeekN_Lp

                Jp      Bank_Ret

                .LSTOFF
```