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

```
;>        Head := Temp DIV Sectors
;>        Sector := Temp MOD Sectors
;>       END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

CnvrtLogical:
CnvrtSrch:      Call    SrchSpTabl      ;check if logical block in spare table
                Push    Flags           ;save results
                Push    !r0             ;save status
                Push    !r1             ;save ptr

                 Ld     !r2,#.HIBYTE. Get_Cyl_H_S
                 Ld     !r3,#.LOWBYTE. Get_Cyl_H_S
                Call    Bank_Call     ;get cylinder, head,and sector
                 Ld     !r0,!rF         ;pass physical sector
                 Ld     !r2,#.HIBYTE. ReMap_Sector
                 Ld     !r3,#.LOWBYTE. ReMap_Sector
                Call    Bank_Call
                 Ld     !rF,!r0         ;            !rF := logical sector

                Pop     !r1             ;Element.Ptr
                Pop     !r0             ;Search Status
                Pop     Flags

                Jp      Bank_Ret

                .LSTOFF
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Function: SrchCache
;>
;>      This funtion searches the block cache and returns both a
;>      boolean variable ( indicating whether the block is in the
;>      cache AND if there is a seek associated with it ), as well
;>      as a info as to whether a head select may be needed. The
;>      cache may be searched sequentially, or randomly ( as is the
;>      case for multiblock commands. ).
;>
;>      Inputs:
;>              LogicalBlock : 3 BYTES { !rC:E }
;>              Offset       : 3 BYTE  { !r0 }
;>              WORKING REGISTER SET MUST BE Wrk_Sys
;>
;>      Outputs:
;>              SrchCache : BOOLEAN { zero flag is set if block is not
;>                                    in the cache OR if there is a seek
;>                                    needed to get to the block }
;>              HeadSector  : BYTE { !r0 }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       i := 1
```

```
;>        SrchCache := False
;>        IF ( CacheArray[ Offset ].LogicalBlock = Logical Block )
;>         THEN
;>              Found := True
;>              CachStat := CacheStatus[ Offset ]
;>              IF NOT( CachStat.Seek )
;>               THEN SrchCache := True
;>       END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

SrchCache:
                Srp     #Wrk_Scr
                Ld      !r4,#CacheLength

Srch_C_Lp:      Ld      !rA,#.HIBYTE. CacheArray
                Ld      !rB,#.LOWBYTE. CacheArray
                Ld      !r0,Cache_Index
                Rl      !r0 ;multiply by 4
                Rl      !r0
                Add     !rB,!r0 ;index into array
                Adc     !rA,#0

                Ld      !r8,#ScrReg0

                Ldei    @!r8,@!!rA ;load CacheArray.Logical
                Ldei    @!r8,@!!rA
                Ldei    @!r8,@!!rA
                Xor     !r0,Wrk_Sys+$C ;compare values
                Xor     !r1,Wrk_Sys+$D
                Xor     !r2,Wrk_Sys+$E
                Or      !r0,!r1 ;test for zero
                Or      !r0,!r2
                Jr      Nz,Srch_More

                Ld      !r8,#.HIBYTE. CachStat ;get CacheStatus
                Ld      !r9,#.LOWBYTE. CachStat
                Add     !r9,Cache_Index
                Inc     Cache_Index
                Lde     !r1,@!!r8
                Ld      BlkStat,!r1 ;save search result
                Lde     !r0,@!!rA ;get Head/Sector info

Srch_C_End:     Srp     #Wrk_Sys ;get back to normal system context
                Ld      !r0,ScrReg0 ;pass head/sector info back
                Tm      BlkStat,#CachSeek ;set seekneeded flag
                Ret

Srch_More:      Inc     Cache_Index
                Cp      Cache_Index,#CacheLength ;check for cache overflow
                Jr      Lt,Srch_M_1

                Clr     Cache_Index
Srch_M_1:       Djnz    !r4,Srch_C_Lp

                Ld      BlkStat,#CachSeek
                Clr     Cache_Index
```

```
Jr      Srch_C_End

.LSTOFF
```