```
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Module: Cache.B1
;>
;>      BANK 1 MODULE
;>
;>      This module contains those Cache routines that must be located
;>      in Bank 1.
;>
;>      PROCEDURE Load_Cache
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     External
                .LSTON
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Procedure: Load_Cache
;>
;>      This procedure loads the 'cache-table' with the next 7
;>      sequential logical blocks ( from the current logical block )
;>      and status' associated with each block ( i.e., if no seek is
;>      required, if a head change is required, etc. ) and the
;>      head and sector number of each logical block ( assuming that
;>      all the blocks are on the same track ). It should be noted that
;>      the status bytes are kept in a seperate table.
;>
;>      Inputs: { none }
;>
;>      Outputs: { none }
;>
;>      Algorithm:
;>
;>      BEGIN
;>       Data_Type := User_Type
;>       FOR i := 1 TO CacheLength DO
;>        Cur_Block := LogicalBlock + i
;>        TempCyl, TempHead, TempSector,
;>                        SrchStat, SrchPtr := CnvrtLogical( Cur_Block )
;>        CacheStatus[ i - 1 ].Seek := CalcMagDir( TempCyl )
;>        IF ( TempHead <> Head )
;>         THEN CacheStatus[ i - 1 ].HeadChange := True
;>         ELSE CacheStatus[ i - 1 ].HeadChange := False
;>        CacheArray[ i - 1 ].LogicalBlock := Cur_Block
;>        CacheArray[ i - 1 ].Head := TempHead
;>        CacheArray[ i - 1 ].Sector := TempSector
;>       Cache_Index := 0
;>      END
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                .LSTOFF
                .FIN
                .DO     Internal
                .LSTON
                .Page
                .FIN

Load_Cache:
```

```
                Ld      !r4,#CacheLength
                Ld      !r5,#0 ;clear seekneeded, headchange booleans
                Ld      Data_Type,#User_Type

                Ld      !r8,#.HIBYTE. CachStat
                Ld      !r9,#.LOWBYTE. CachStat
                Ld      !rA,#.HIBYTE. CacheArray
                Ld      !rB,#.LOWBYTE. CacheArray

                Srp     #Wrk_Sys2       ;init logical block
                Call    Load_Logical
                Srp     #Wrk_Sys

                Call    Ld_C_Srch
                 Push   Flags   ;save result of spare table search
                 Push   !r0
                 Ld     Wrk_Sys2+$09,!rC ;store physical block
                 Ld     Wrk_Sys2+$0A,!rD ;store physical block
                 Ld     Wrk_Sys2+$0B,!rE ;store physical block
                Call    Cache_Cnvrt
                 Pop    !r0
                 Pop    Flags
                Clr     !r1 ;SeekNeeded,HdChg := False
                Jr      Nz,Ld_Cache_In
                Jp      Ld_C_Enter

Ld_Cache_Lp:    Srp     #Wrk_Sys2
                Add     !rE,#1          ;inc logical block
                Adc     !rD,#0
                Adc     !rC,#0
                Add     !rB,#1          ;inc physical block
                Adc     !rA,#0
                Adc     !r9,#0
                Srp     #Wrk_Sys
                Call    Ld_C_Srch
                Jr      Z,Cache_NoSpare

                 Ld     !r1,#CachSeek
                 Or     !r5,!r1 ;set seekneeded
Ld_Cache_In:    Call    Enter_Cache
                Jr      Ld_Cache_More

Cache_NoSpare:  Srp     #Wrk_Sys2
                Ld      !r0,Wrk_Sys+$0C ;nondestructve subtract
                Ld      !r1,Wrk_Sys+$0D
                Ld      !r2,Wrk_Sys+$0E
                Sub     !r2,!rB ;IF ( TPBlock <> PBlock )
                Sbc     !r1,!rA
                Sbc     !r0,!r9
                Ld      !rF,#0 ;check distance
                Or      !rF,!r0
                Or      !rF,!r1
                Or      !rF,!r2
                Jr      Z,Ld_Cach_NoSeek

                Or      !r0,!r1 ;check for spare block
                Jr      Nz,Ld_Cach_Seek
                Cp      !r2,#1 ;should be distance 1
                Jr      Nz,Ld_Cach_Seek

                Add     !rB,#1 ;inc PBlock to account for spare
                Adc     !rA,#0
                Adc     !r9,#0
```

```
                Inc     !r8 ;bump the sector address

Ld_Cach_NoSeek: Inc     !r8 ;bump the sector address
                Cp      !r8,#NbrSctrs
                Jr      Ge,Ld_Cach_HdChg

                 Clr    Wrk_Sys+$01 ;CacheSeek, CachHdChg := False
                Jr      Ld_C_Enter

Ld_Cach_HdChg:  Ld      !r8,#0 ;start at sector 0
                Cp      !r7,#0 ;check for head 0
                Jr      Nz,Ld_Cach_Seek

                 Ld     !r7,#1 ;otherwise go to head 1
                Ld      Wrk_Sys+$01,#CachHdChg
                Jr      Ld_C_Enter

Ld_Cach_Seek:   Ld      !r8,#0 ;start at head 0
                Ld      !r7,#0 ;and sector 0
                Add     !r6,#1 ;bump the track count by 1
                Adc     !r5,#0
                Ld      Wrk_Sys+$01,#CachSeek

Ld_C_Enter:     Srp     #Wrk_Sys
                 Or     !r5,!r1 ;set seekneeded or headchange
                Call    Ld_BlkStat

Ld_Cache_More:  Dec     !r4
                Jp      Nz,Ld_Cache_Lp

Ld_Cache_End:   Clr     Cache_Index
                Jp      Bank_Ret


;*************************************

Enter_Cache:    Tm      !r0,#Spare ;check if block is a spare
                Jr      Z,Ld_Blk_BB

                Or      !r1,#S_Block ;set Spare Block status
                Jr      Ld_BlkStat

Ld_Blk_BB:      Or      !r1,#B_Block ;otherwise it must be a Bad Block

Ld_BlkStat:     Or      !r1,!r5 ;merge in global seekneeded or headchange
                Lde     @!!r8,!r1

Ld_Cach_Lgcl:   Ld      !r0,#Wrk_Sys2+$0C ;load CacheArray.Logical
                Ldei    @!!rA,@!r0
                Ldei    @!!rA,@!r0
                Ldei    @!!rA,@!r0

                Incw    !!r8 ;set up for the next go 'round

                Ld      !r0,Wrk_Sys2+$08 ;get the latest sector value
                Ld      !r2,#.HIBYTE. Map_Table ;logically remap the sector
                Ld      !r3,#.LOWBYTE. Map_Table
                Add     !r3,!r0
                Adc     !r2,#0
                Lde     !r0,@!!r2
                Ld      !r1,Wrk_Sys2+$07 ;get latest head value
                Swap    !r1 ;merge Head with sector
```

```
                Rcf
                Rl      !r1
                Rl      !r1
                Or      !r1,!r0
                Lde     @!!rA,!r1

                Incw    !!rA ;point to next cache entry

                Ret


Cache_Cnvrt:    Ld      !rC,Wrk_Sys2+$09 ;get latest Physical block
                Ld      !rD,Wrk_Sys2+$0A
                Ld      !rE,Wrk_Sys2+$0B

                 Ld     !r2,#.HIBYTE. Get_Cyl_H_S
                 Ld     !r3,#.LOWBYTE. Get_Cyl_H_S
                Call    Bank_Call

                Ld      Wrk_Sys2+$05,!rC ;store latest seek address
                Ld      Wrk_Sys2+$06,!rD
                Ld      Wrk_Sys2+$07,!rE
                Ld      Wrk_Sys2+$08,!rF
                Ret


                .PAGE

;*************************************
;
;       Fast search of spare table
;
;*************************************

Ld_C_Srch:      Ld      !rC,Wrk_Sys2+$0C ;get last logical block
                Ld      !rD,Wrk_Sys2+$0D
                Ld      !rE,Wrk_Sys2+$0E

                Ld      !r1,!rD ;check if head ptr is NIL
                Ld      !r0,!rC ;but first form a headptr structure
                Rrc     !r0     ;and index into HeadPtr Array
                Rrc     !r1
                Rcf
                Rrc     !r1
                Ld      !r2,#.HIBYTE. SegPtrArray
                Ld      !r3,#.LOWBYTE. SegPtrArray
                Add     !r3,!r1
                Adc     !r2,#0
                Lde     !r0,@!!r2       ;get headptr and check for NIL
                Tm      !r0,#Nil
                Jr      Z,Ld_C_Long ;do a real search if not NIL

                .DO     W_10MB
                Ld      !r0,!rD ;save for possible rollover
                Add     !rE,!rD ;Physical Block := LBlock + LBlock DIV 256
                Adc     !rD,#0
                Adc     !rC,#0
                .FIN

                Cp      !r0,!rD ;check for rollover
                Jr      Z,Ld_C_S_End

                Add     !rE,#1 ;otherwise bump Physical Block
```

```
                Adc     !rD,#0
                Adc     !rC,#0

Ld_C_S_End:     Xor     !r0,!r0 ;return zero status
Ld_C_S_Ret:     Ret

Ld_C_Long:       Ld     !r2,#.HIBYTE. SrchSpTabl
                 Ld     !r3,#.LOWBYTE. SrchSpTabl
                Call    Bank_Call
                Jr      Ld_C_S_Ret

                .LSTOFF
```