

```
.DO      External
.LSTON
.Page
>>>>>>>>>>>>>>>>>>>>>>>>>>
>
>   Module: SrvoUtils
>
>   BANK 1 MODULE
>
>   This module holds those routines that make up most of the
>   auxillary servo commands.
>
>   PROCEDURE ServoCmd( CommandString : 4 BYTES { !r0:3 } )
>   PROCEDURE ServoStatus( CommandString : 4 BYTES { !r0:3 }
>                           BufferPtr : PTR { !!re }
>                           )
>   FUNCTION ServoStore : BOOLEAN
>   FUNCTION ServoLoad( BufferPtr : PTR { !!r2 } ) : BOOLEAN
>   FUNCTION Restore( RecallType : BYTE { !r0 } ) : BOOLEAN
>   PROCEDURE SrvoRcvry
>   PROCEDURE Load_SrvoCmd
>   PROCEDURE Park_Heads
>   PROCEDURE ResetServo
>
>>>>>>>>>>>>>>>>>>>>>>>>>>
>       .LSTOFF
>       .FIN
>       .DO      External
>       .LSTON
>       .Page
>>>>>>>>>>>>>>>>>>>>>>>>>>
>
>   Procedure: ServoCmd { Send the Servo a command }
>
>   Inputs:
>       CommandString : 4 BYTES { !r0:3 }
>
>   Outputs: { none }
>
>   Global Variables Changed:
>       SrvoCmdBuffer
>
>   Local Variables:
>       SioRetry : BYTE { !r8 }
>
>   Algorithm:
>
>   BEGIN
>     SrvoCmdBuffer[ CommandByte ] := !r0
>     SrvoCmdBuffer[ LoDiffByte ] := !r1
>     SrvoCmdBuffer[ OffsetByte ] := !r2
>     SrvoCmdBuffer[ StatusByte ] := !r3
>     SioRetry := 4
>     WHILE NOT( ServoStore( CommandString ) ) AND ( SioRetry > 0 ) DO
>       SioRetry := SioRetry - 1
>     IF ( SioRetry = 0 ) ABORT
>   END
>
>>>>>>>>>>>>>>>>>>>>>>>>>>
>       .LSTOFF
>       .FIN
>       .DO      Internal
>       .LSTON
```

.Page
.FIN

```

ServoCmd:      Srp      #Wrk_Sys2
                Call     Load_SrvCmd

Srv_C_Lp1:     Ld        !r8,#4 ;SioRetry := 4

                Ld        !r4,ScrReg0 ;save command
                Ld        !r5,ScrReg1
                Ld        !rA,ScrReg2
                Ld        !rB,ScrReg3

Srv_C_Lp:       Call     ServoStore
                Jr        Nz,Srv_C_End

                Ld        ScrReg0,!r4 ;restore command
                Ld        ScrReg1,!r5
                Ld        ScrReg2,!rA
                Ld        ScrReg3,!rB

                Djnz      !r8,Srv_C_Lp

Srv_C_Abort:    Call     Abort

Srv_C_End:      Srp      #Wrk_Sys
                Jp        Bank_Ret

```

```
.LSTOFF
.DO      External
.LSTON
.Page
```

[illegible]

```

> Procedure: ServoStatus { Read a status location from the servo }
>
> Inputs:
>     CommandString : 4 BYTES { !r0:3 }
>     BufferPtr      : PTR { !!rE }
>
> Outputs: { none }
>
> Global Variables Changed:
>     SrvoCmdBuffer
>     @BufferPtr
>
> Local Variables:
>     SioRetry : BYTE { !r8 }
>
> Algorithm:
>
> BEGIN
>     ServoCmd
>     IF NOT( ServoLoad( BufferPtr ) ) THEN ABORT
> END

```

[illegible]

```
.LSTOFF
.FIN
.DO      Internal
.LSTON
.Page
```

.FIN

```
ServoStatus:      Push      !r8 ;save for caller
```

```
Srv_St_Lp1:      Call    ServoCmdnd
Srv_St_Else:     Call    ServoLoad
                  Jr       Z,Srv_St_End
```

```
Srp      #Wrk_Sys
Call Abort
```

Srv_St_End:	Pop	ln8
	Jp	Bank_Ret

```
S_Reset:      Call    Ext_Push
              Call    ResetServo
              Call    Ext_Pop
              Ld      !r0,*DataRecal
              Call    Restore
              Ret
```

```
.LSTOFF
.DO      External
.LSTON
.Page
```

[illegible]

```

; >
; >      Function: ServoStore

```

```

;>
;> The function of this routine is to transmit a command string
;> to the Servo Processor. The command string is expected to
;> reside in the global variable
;> SrvoCmdBuffer ( thus providing a record of the last command
;> sent to the servo ). ServoStore then attempts to complete the
;> transmission to the Servo, and returns a boolean variable
;> indicating whether the transmission was completed or not.

```

```

; >
; > Inputs:
; > Parent : BYTE { Ir1 }

```

```
>
>
> Outputs:
> ServoStore : BOOLEAN { Zero flag: True if transmission failed }
```

```

>
>      Local Variables:
>      Retry : BYTE { !r6 }
>      i      : BYTE { !r4 }
>      i      : BYTE { !r5 }

```

```

; >
; >
; > Algorithm:

```

[illegible]

```

> Retry := 10
> REPEAT
>   FOR i := 1 TO 5 DO
>     WHILE ( IRQ.SerialOutput = False ) OR NOT( SioReady ) DO
>       BEGIN
>         END
>       Sio.Data := SrvoCmdBuffer[ i ]
>     Retru := Retru - 1

```

[illegible]


```

Call    Set_SseekNeeded

Ld      Cur_Cyl,*HiMaxCyl ;assume FormatRecal
Ld      Cur_Cyl+1,*LoMaxCyl

Pop     !r6 ;load RecalType from storage
Cp      !r6,*FrmtRecal
Jr      Z,Rest_Up2

Ld      Cur_Cyl,*Init_HiCyl ;otherwise DataRecal
Ld      Cur_Cyl+1,*Init_LoCyl

Rest_Up2:  Cln      !r5
           Ld      !r4,$D5 ;max time to wait is two seconds

Restore_Lp: Call    LoadStatus
           Tm      !r0,*ServoRdy
           Jr      Nz,Rest_UpDate
           Decw    !!r4
           Jr      Nz,Restore_Lp

           Cln      !r0 ;pass error status to exit
           Jr      Restore_End

Rest_UpDate: Tm      DiskStat,*RdHdrRecal
           Jr      Z,Rest_Up1
           Cp      !r6,*FrmtRecal ;don't try to read headers here!
           Jr      Z,Rest_Up1

           Call    UpDate_Cur_Cyl
           Jr      Nz,Rest_Up1 ;leave if positioned correctly

           Call    SS_NoHdr ;header err here is bad news!
           Call    Abort

Rest_Up1:  Ld      !r0,*1
Restore_End: Push    !r0 ;save result
           Call    Ext_Pop
           Pop     !r0
           Or      !r0,!r0 ;set zero flag
           Jp      Bank_Ret

```


[illegible]

Park_Heads:

```
.LSTOFF
.DO      External
.LSTON
.Page
```

[illegible]

17

1

13

10

13

11

2

3

22

•

1

[illegible]

```

.LSTOFF
.FIN
.DO      Internal
.LSTON
.Page
.FIN

```

ResetServo:

```
And    Port0,$FF-Not_ServoRst ;set Servo Reset
```

S_Rst_Lp1:

```

Ld      !r2,*.HIBYTE. 5 ;busy wait for 50 msec
Ld      !r3,*.LOWBYTE. 5
Call    MWait

```

```
Or      Port0,*Not_ServoRst ;clear Servo Reset
```

```

Ld      !r2,*.HIBYTE. 100 ;busy wait for 1 sec
Ld      !r3,*.LOWBYTE. 100
Call    MWait

```

S_Rst_1:

```

Call    LoadStatus ;get servo status
Tm      !r0,*ServoErr
Jr      Z,S_Rst_Bd1

```

S_Rst_Abt1:

```

Ld      !rA,!r0
Call    Abort

```

S_Rst_Bd1:

```

And      Tmr,$FF-T0_CntEn      ;halt T0
Ld      Pre0,$D ;Pre0 := Mod-64,continuous;go to 19.2k baud
Ld      T0,*1 ;interrupt after 1 byte
Or      Tmr,*T0_CntEn + T0_Load

```

```

Call    Set_Dmt
Srp      #Wrk_Scr
Ld      !r0,*ReadStatus ;try talking to the Servo
Clr      !r1
Clr      !r2
Ld      !r3,*S_Norm_Status + S_Rate_57_6
Srp      #Wrk_Sys
Call    Load_SrvoCmd
Call    ServoStore
Jr      Z,S_Rst_Comm
Call    Load_Status
Tm      !r0,*ServoErr
Jr      Z,S_Rst_Ld

```

S_Rst_Comm:

```
Ld      !rA,*S_Store
```

S_Rst_Abt2:

```

Ld      !r8,*S_Rst_Abort
Ld      !r9,*Comm_Err
Call    Abort

```

S_Rst_Ld:

```

Call    ServoLoad
Ld      !r7,*S_Load
Jr      Nz,S_Rst_Abt2

```

```

And      Tmr,$FF-T0_CntEn      ;halt T0
Ld      Pre0,$5 ;Pre0 := 1,continuous run ;go to 57.6k baud
Ld      T0,*1
Or      Tmr,*T0_CntEn + T0_Load

```

```

Srp      #Wrk_Scr
Ld      !r0,*DataRecal
Clr      !r1

```

```

Cln      !r2
Ld        !r3,*S_Rate_57_6
Srp       #Wrk_Sys
Call     Load_ServoCmdnd
Call     ServoStore
Ld        !rA,*S_Store
Jr        Z,S_Rst_Abt2


Call     Cln_Dmt
Ld        !r4,$D5 ;wait for max of 2 secs
Cln       !r5


Reset_S_Lp: Call    LoadStatus
           Tm       !r0,*ServoRdy
           Jr        Nz,Reset_S_End
           Decw     !!r4
           Jr        Nz,Reset_S_Lp


Reset_S_End: And     DiskStatus,$FF-On_Track-Offset_On
             Or      DiskStatus,#SeekComplete


           Ld        Cur_Cyl,*Init_HiCyl
           Ld        Cur_Cyl+1,*Init.LoCyl


S_Rst_2:   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_SeekNeeded:
            Ld        !r2,*HIBYTE.CachStat
            hStat
            Ld        !r1,*CacheLength
```

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

              Jp     Bank_Ret

.LSTOFF
```