```
                .DO     ListDefs
                .LSTON
                .Page
                .FIN
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Controller Status Port
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

Status_Port     .Equ    $1F00

CrcErrL         .Equ    $80     ;status bit 7: Crc Error ( active lo )
WrtNvldL        .Equ    $40     ;status bit 6: Write Not Valid ( active lo )
ServoRdy        .Equ    $20     ;status bit 5: Servo Ready ( active hi )
ServoErr        .Equ    $10     ;status bit 4: Servo Error ( active hi )
Ymask           .Equ    $0F     ;mask for status bits 0:3: state
 Norm_State      .Equ    $02     ;normal end state for Read,Write,ReadHdr
 NormFmt_State   .Equ    $0A     ;normal end state for Format
 NoHdr_State     .Equ    $00     ;last state if no matching header found
 Start_State     .Equ    $00     ;beginning state machine state

                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: Miscellaneous
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

Stack_Top       .Equ    $80             ;Register $7F + 1
Init_Pc         .Equ    $0c             ;Initial Pc setting after RESET

IntL_Dflt       .Equ    #02             ;assume drive interleaved 1:1
Map_Dflt        .Equ    #$0C            ;Map 5:1 onto 2:1 as default

RegLpTimes      .Equ    2               ;Loop twice ( 0's and 1's )
RegCount        .Equ    128             ;Total number of regs z8
RegUsed         .Equ    5               ;this test uses r4,r5,r6,r7,r8
I_OregUsed      .Equ    4               ;Ports 0,1,2,3 can't be tested here
HiRegAdr        .Equ    RegCount-1

RamSize         .Equ    2048            ;Physical length of 6116 ( 2kx8 )
HiRamAdr        .Equ    RamSize-1
RamLpTimes      .Equ    2               ;loop twice ( 0's and 1's )

EpromSize       .Equ    4096
MaxEpromAddress .Equ    EpromSize - 1
EpromStartAdr   .Equ    $0001

Max_Cmnd_Types  .Equ    $02             ;currently only three types of cmnds
CmndType        .Equ    $F0             ;mask for TYPE of command
Cmnd_Len        .Equ    $07
Servo_Len       .Equ    $05

                .DO     W_10MB
HiMaxLogical    .Equ    $00     ;highest user block = $04BFF
MidMaxLogical   .Equ    $4B
LoMaxLogical    .Equ    $FF
HiMaxCyl        .Equ    $02     ;highest cylinder = $220 ( 544 )
LoMaxCyl        .Equ    $20
MaxSeek         .Equ    $185    ;seeks longer than this are broken into 2
Mid_Cyl         .Equ    $101    ;middle of data area
Init_HiCyl      .Equ    $01     ;initial setting for data cylinder
```

```
Init_LoCyl      .Equ    $F9
Tst_HiCyl       .Equ    $02     ;Cyl, Head, and sector of R/W test area
Tst_LoCyl       .Equ    $05
Tst_Head        .Equ    $00
Tst_Sctr        .Equ    $00
NbrSctrs        .Equ    19      ;number of sectors = 19
NbrHds          .Equ    2       ;number of heads = 2
NbrTracks       .Equ    514     ;number of tracks = 514
HiSpr0          .Equ    $00     ;logical block of Spr( 0 )
MidSpr0         .Equ    $19
LoSpr0          .Equ    $55
HiSpr1          .Equ    $00     ;logical block of Spr( 1 )
MidSpr1         .Equ    $32
LoSpr1          .Equ    $AA
ParkCyl         .Equ    $0235     ;cylinder to park heads at
                .FIN
                .DO     W_20MB
HiMaxLogical    .Equ    $00     ;highest user block = $0984B
MidMaxLogical   .Equ    $98
LoMaxLogical    .Equ    $4B
HiMaxCyl        .Equ    $02     ;highest cylinder = $220 ( 544 )
LoMaxCyl        .Equ    $20
MaxSeek         .Equ    $185    ;seeks longer than this are broken into 2
Mid_Cyl         .Equ    $101    ;middle of data area
Init_HiCyl      .Equ    $01     ;initial setting for data cylinder
Init_LoCyl      .Equ    $F9
Tst_HiCyl       .Equ    $02     ;Cyl, Head, and sector of R/W test area
Tst_LoCyl       .Equ    $05
Tst_Head        .Equ    $00
Tst_Sctr        .Equ    $00
NbrSctrs        .Equ    38      ;number of sectors = 38
NbrHds          .Equ    2       ;number of heads = 2
NbrTracks       .Equ    514     ;number of tracks = 514
HiSpr0          .Equ    $00     ;logical block of Spr( 0 )
MidSpr0         .Equ    $32
LoSpr0          .Equ    $AA
HiSpr1          .Equ    $00     ;logical block of Spr( 1 )
MidSpr1         .Equ    $65
LoSpr1          .Equ    $54
ParkCyl         .Equ    $0235     ;cylinder to park heads at
                .FIN
                .DO     W_40MB
HiMaxLogical    .Equ    $01     ;highest user block = $130E3
MidMaxLogical   .Equ    $30
LoMaxLogical    .Equ    $E3
HiMaxCyl        .Equ    $04     ;highest cylinder = $440 ( 1088 )
LoMaxCyl        .Equ    $40
MaxSeek         .Equ    $30A    ;seeks longer than this are broken into 2
Mid_Cyl         .Equ    $202    ;middle of data area
Init_HiCyl      .Equ    $03     ;initial setting for data cylinder
Init_LoCyl      .Equ    $F9
Tst_HiCyl       .Equ    $04     ;Cyl, Head, and sector of R/W test area
Tst_LoCyl       .Equ    $0A
Tst_Head        .Equ    $00
Tst_Sctr        .Equ    $00
NbrSctrs        .Equ    38      ;number of sectors = 38
NbrHds          .Equ    2       ;number of heads = 2
NbrTracks       .Equ    1028    ;number of tracks = 1028
HiSpr0          .Equ    $00     ;logical block of Spr( 0 )
MidSpr0         .Equ    $65
LoSpr0          .Equ    $54
HiSpr1          .Equ    $00     ;logical block of Spr( 1 )
MidSpr1         .Equ    $CA
```

```
LoSpr1          .Equ    $A8
ParkCyl         .Equ    $046A     ;cylinder to park heads at
                .FIN

SprThresh       .Equ    3 ;this many CRC errors to be candidate for sparing

Write_Op        .Equ    $00     ;current operation is a write
Read_OP         .Equ    $80     ;current operation is a read

Wait            .Equ    $40     ;wait for seek-end

Nil             .Equ    $80     ;Nil Ptr
Used            .Equ    $40     ;Spare Block in table is used
Useable         .Equ    $20     ;Spare Block in table is useable
Spare           .Equ    $10     ;element is a true spare
BadBlock        .Equ    $00     ;element is a bad block

SprTbl_Type     .Equ    $08     ;type of element is spare table
ID_Type         .Equ    $04     ;type of element is ID block
User_Type       .Equ    $02     ;type of element is User data block

Found           .Equ    $01     ;Block was found in spare table

Random          .Equ    $80     ;search cache by index

TestBitMap      .Equ    $80     ;check bit map location for 0 or 1
SetBitMap       .Equ    $40     ;set a bit map location
ClearBitMap     .Equ    $20     ;clear a bit map location

BlockLength     .Equ    1 + 532 + 2 + 6 ; Length = Dummy + Data + CRC + ECC

RdError         .Equ    $80
RdSrvoErr       .Equ    $40
RdSuccess       .Equ    $20
RdNoHdrFnd      .Equ    $10
RdCrcErr        .Equ    $08

RdHError        .Equ    $80
RdHSrvoErr      .Equ    $40

WrError         .Equ    $80
WrSrvoErr       .Equ    $40
WrSuccess       .Equ    $20
WrNoHdrFnd      .Equ    $10

FmtError         .Equ   $80
FmtSrvoErr       .Equ   $40
FmtSuccess       .Equ   $20
Max_InterLeave   .Equ   $06

Cmnd_Pending    .Equ    $80     ;Wait_Cmnd was called from a command
!Bsy            .Equ    $40     ;set BSY
MultiWr         .Equ    $20     ;current operation is MultiBlockWrite

Frst_SprTbl     .Equ    $80     ;First spare table has been found

;> Data Exception Error Codes
Error           .Equ    $80
Ex_Undetermined .Equ    $00     ;Undetermined error ( recovery off? )
Ex_SprBlock     .Equ    $02     ;Spare this block!
Ex_BadBlock     .Equ    $04     ;make this block a bad block
Ex_ReadErr      .Equ    $06     ;non-catastophic error during read
Ex_HdrBad       .Equ    $08     ;make this block bad because no header
```

```
Ex_HdrSpr       .Equ    $0A     ;spare this block because no header

Ex_Case_Max     .Equ    $0A     ;maximum case values

;> Interface Responses
Read_Response   .Equ    $02     ;Interface Response to a Read command
Wr_Response     .Equ    $03     ;Interface Response to a Write command
WrVer_Response  .Equ    $04     ;Interface Response to a Write Verify command
End_Wr_Response .Equ    $06     ;Interface response after receiving wr data
D_R_ID_Response .Equ    $02
Rd_Stat_Resp    .Equ    $03
Rd_SStat_Resp   .Equ    $04
Sd_S_C_Respone  .Equ    $05
S_Seek_Response .Equ    $06
S_Rstr_Response .Equ    $07
Set_Rcvr_Resp   .Equ    $08
S_Park_Respone  .Equ    $0A
D_Read_Response .Equ    $0B
D_RdHdr_Resp    .Equ    $0C
D_Write_Resp    .Equ    $0D
St_Map_Response .Equ    $0E
D_R_Spr_Resp    .Equ    $0F
Wr_Spr_Resp     .Equ    $10
Fmt_Response    .Equ    $11
I_Spr_Response  .Equ    $12
Rd_Abrt_Resp    .Equ    $13
RstSrvo_Resp    .Equ    $14
D_Scan_Response .Equ    $15

Sys_Rd_Resp     .Equ    $22
Sys_Wr_Resp     .Equ    $23
Sys_WrEx_Resp   .Equ    $A3
Sys_WrEnd_Resp  .Equ    $27
Sys_WrVer_Resp  .Equ    $24

Profile         .Equ    $00     ;Device Interface Type is Profile
Widget          .Equ    $01     ;Device Interface Type is Widget

Pro_Log_Offset  .Equ    $01     ;offset from start of command to blocknumber
Sys_Log_Offset  .Equ    $03

Inc_SprCnt      .Equ    $00
Inc_BadCnt      .Equ    $01
Dec_BadCnt      .Equ    $02

;> System Status Definitions

;> Byte 0
Bad_55          .Equ    $80     ;Status: Other than 55 response from host
WrBuf_OR        .Equ    $40     ;Status: Write Buffer OverFlow
Stat_Rd_Err     .Equ    $08     ;Status: Read Error
Stat_No_Hdr     .Equ    $04     ;Status: No Header Found
Stat_Srvo       .Equ    $02     ;Status: Unrecoverable Servo Error
Op_Failed       .Equ    $01     ;Status: Operation Failed

;> Byte 1
SprBlk_Hard     .Equ    $40     ;Status: No room left in spare table
SprBlk_Warn     .Equ    $20     ;Status: 5 or less spare blocks available
Stat_SlfTst     .Equ    $08     ;Status: Abort caused by self test failure
Stat_Spare      .Equ    $04     ;Status: Sparing has occured
Stat_Seek       .Equ    $02     ;Status: Seek to wrong track
Stat_Abort      .Equ    $01     ;Status: Controller aborted
```

```
;> Byte2
Power_Reset     .Equ    $80     ;Widget was power reset
Illegal_Block   .Equ    $40     ;Status: Block number out of bounds

;> Byte3
EccStat         .Equ    $80     ;read error detected by ecc
CrcStat         .Equ    $40     ;read error detected by crc
Hdr_MisMatch    .Equ    $20     ;timeout waiting for header

;> Bank Switch Registers Defs

BankReg         .Equ    $1800 ;address of first bank bit

Led             .Equ    $1800 ;address of Led bit
Led_Mask        .Equ    $FE   ;mask for Led bit

RamBank0        .Equ    $1900 ;address of first Ram_Bank bit
Ram0            .Equ    $0    ;offset of ram bank zero
Ram1            .Equ    $1
Ram2            .Equ    $2
Ram3            .Equ    $3

RomBank0        .Equ    $1E00
RomBank2        .Equ    $1D00
Eprom0          .Equ    $0
Eprom1          .Equ    $1
Eprom2          .Equ    $2
Eprom3          .Equ    $3
Eprom4          .Equ    $4

                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Servo Definitions
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

S_Cmnd_Byte     .Equ    $00     ;Servo Command Byte
S_Diff_Byte     .Equ    $01     ;Servo Lo Difference Byte
S_Off_Byte      .Equ    $02     ;Servo Offset Byte
S_Stat_Byte     .Equ    $03     ;Servo Status Byte

;> BYTE 0       .Equ    Command, Direction, and Hi Difference Byte

ReadStatus      .Equ    $00     ;Servo Command: Read Status
Offset          .Equ    $10     ;Servo Command: Offset, Track Following
Diagnostic      .Equ    $20     ;Servo Command: Diagnostic
DataRecal       .Equ    $40     ;Servo Command: Recalibrate heads, Data
FrmtRecal       .Equ    $70     ;Servo Command: Recalibrate hedas, Format
Access          .Equ    $80     ;Servo Command: Access only
Access_Offset   .Equ    $90     ;Servo Command: Seek with Offset
Home            .Equ    $C0     ;Servo Command: Home the heads

Hd_Dir_Rev      .Equ    $00     ;Direction: Reverse
Hd_Dir_Frwd     .Equ    $04     ;Direction: Forward

;> Bits 0:1     .Equ    Magnitude: Hi Difference

;> BYTE 1       .Equ    Magnitude: Lo Difference

;> BYTE 2       .Equ    Offset Byte

Off_Dir_Rev     .Equ    $00     ;Offset Direction Reverse
```

```
Off_Dir_Frwd    .Equ    $80     ;Offset Direction Forward
Off_Auto        .Equ    $40     ;Auto-Offset On
Ld_Off_Val      .Equ    $20     ;Load Offset Value from DAC on Servo
;> Off_Val      .Equ    $1F     ;5-bit value Loaded/Stored into DAC

;> BYTE 3       .Equ    Status Byte

S_Rate_19_2     .Equ    $00     ;Servo/Controller baud rate = 19.2
S_Rate_57_6     .Equ    $80     ;Servo/Controller baud rate = 57.6
Pwr_On_Reset    .Equ    $40     ;Servo Power-On Reset

S_Stat_0        .Equ    $00     ;Servo Status 0
S_Stat_1        .Equ    $01     ;Servo Status 1
S_Stat_2        .Equ    $02     ;Servo Status 2
S_Stat_3        .Equ    $03     ;Servo Status 3
S_Stat_4        .Equ    $04     ;Servo Status 4
S_Stat_5        .Equ    $05     ;Servo Status 5
S_Stat_6        .Equ    $06     ;Servo Status 6
S_Stat_7        .Equ    $07     ;Servo Status 7
S_Stat_8        .Equ    $08     ;Servo Status 8

S_Norm_Status   .Equ    S_Stat_1        ;Normal Servo Status

;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: Exception Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

Apl_Ack         .Equ    $55 ;Acknowledge from Host
Init_Response   .Equ    $01 ;Controller Ready for new command
Free_Proc       .Equ    $69 ;Host willing to give up the bus

Apl_Excpt       .Equ    $00             ;Apple_Interface
 ChkB_Mismatch  .Equ    $00             ;Check Byte Mismatch
 Bad_Cmnd       .Equ    $01             ;Illegal Command

Cmnd_Excpt      .Equ    $01             ;Command_Driver
 Strt_Cmnd      .Equ    $00              ; Start_Command

LBlk_Bounds     .Equ    $02             ;Logical Block out-of-bounds

S_Rst_Abort     .Equ    $03             ;Servo Reset Abort
 Serr_NotReady  .Equ    $00      ;abort because NOT( ServoRdy ) OR ServoErr
 Comm_Err       .Equ    $01      ;abort because communication error
  S_Store        .Equ    $00     ;comm error in Servo Store
  S_Load         .Equ    $01     ;comm error in Servo Load

RdBlk_Abort     .Equ    $04     ;abort in ReadBlock
RdHdr_Abort     .Equ    $05     ;abort in ReadHeader
WrBlk_Abort     .Equ    $06     ;abort in WriteBlock
FmtBlk_Abort    .Equ    $07     ;abort in FormatBlock
 Bad_State      .Equ    $00      ;abort caused by illegal state machine state
FmtTrk_Abort    .Equ    $08     ;abort in FormatTrack
 FmtTrk_PosErr  .Equ    $00      ;abort because unable to position heads

SpareAbort      .Equ    $09     ;abort while trying to deal with spare table
 NoSpace        .Equ    $00      ;abort was caused by request for space

Srvo_R_Abort    .Equ    $0A     ;abort while attempting Servo Recovery

OvrLp_Abort     .Equ    $0B     ;abort while attempting overlapped seek
```

```
Rd_Cmn_Abort    .Equ    $0C     ;abort while executing system read
 Servo_Dead      .Equ    $00     ;aborted because of servo failure

Seek_Abort      .Equ    $0D     ;abort while executing a seek

Wr_Cmn_Abort    .Equ    $0E     ;abort while executing Write_Common

Bad_PassWord    .Equ    $0F     ;command string had wrond password

Bad_Params      .Equ    $10     ;command had illegal parameters

SprCnt_Abort    .Equ    $11     ;illegal spare count command

Host_OvrFlow    .Equ    $12     ;Host sent too much data!

                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: External Eprom Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>

Pro_Cmnds       .Equ    02
Diag_Cmnds      .Equ    19
Sys_Cmnds       .Equ    02

                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: External Ram Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>

                .ORG    $1000

ReadArray:
RSctrGap:       .Block  0,10
RHdrGap:        .BlocK  0,1
RHeader:        .Block  0,6
RDataGap:       .Block  0,7
RDummy:         .Block  0,1
RBuffer1:       .Block  0,532
RBuf1Crc:       .Block  0,2
RBuf1Ecc:       .Block  0,6
REndGap:        .Block  0,1
RBuf1Pw:        .Block  0,4

BlockID         .Equ    512 ;offset of controller useable space
StatusArray     .Equ    RBuffer1 - 4

                .ORG    $1000

ReadHdrArray:
RHSctrGap:      .Block  0,10
RHHdrGap:       .BlocK  0,2
RHHeader:       .Block  0,6
RHDataGap:      .Block  0,6
RHDummy:        .Block  0,1
RHBuffer1:      .Block  0,532
RHBuf1Crc:      .Block  0,2
RHBuf1Ecc:      .Block  0,6
RHEndGap:       .Block  0,1
RHBuf1Pw:       .Block  0,4
```

```
RdH_Stat_Array  .Equ    RHHeader - 4

                .ORG    $1000

WriteArray:
WSctrGap:       .Block  0,10
WHdrGap:        .BlocK  0,1
WHeader:        .Block  0,6
WDataGap:       .Block  0,14
WDataSync:      .Block  0,2
WBuffer1:       .Block  0,532
WBuf1Crc:       .Block  0,2
WBuf1Ecc:       .Block  0,6
WEndGap:        .Block  0,2
WBuf1Pw:        .Block  0,4

                .ORG    $1000 + $20

FormatArray:
FSctrGap:       .Block  0,10
FHdrGap:        .BlocK  0,16
FHdrSync:       .Block  0,2
FHeader:        .Block  0,6
FDataGap:       .Block  0,14
FDataSync:      .Block  0,2
FBuffer1:       .Block  0,532
FBuf1Crc:       .Block  0,2
FBuf1Ecc:       .Block  0,6
FEndGap:        .Block  0,2

WBlkFence:      .Block  0,4 ;fence to check for host write overruns
Cmnd_Ptr        .Equ    WBuffer1-7 ;host passes command here!

Buf2Array:
BufDummy:       .Block  0,1
Buffer2:        .Block  0,532
Buf2Crc:        .Block  0,2
Buf2Ecc:        .Block  0,6
Buf2Pw2:        .Block  0,4

CStatus0:       .Block  0,4     ;Controller Status
CStatus1:       .Block  0,4
CStatus2:       .Block  0,4
CStatus3:       .Block  0,4     ;Last Requested Block Number
CStatus4:       .Block  0,4     ;Last Host Command: word0
CStatus5:       .Block  0,4     ;Last Host Command: word1
CStatus6:       .Block  0,4
End_CStatus     .Equ    .PC.

LogicalBlock    .Equ    CStatus3

SrvoCmndBuf:    .Block  0,5     ;Servo Command Buffer
S_Cmnd_Len      .Equ    5       ;Length of ServoCmndBuffer
SStatus0:       .Block  0,5     ;Servo Status

SpareArray:
SparePw1:       .Block  0,4
SpareTmStmp:    .Block  0,4
FmtOffset:      .Block  0,1
FmtIntrL:       .Block  0,1
SegPtrArray:    .Block  0,128
SprCount:       .Block  0,1
```

```
BadCount:       .Block  0,1
SpareBitMap:    .Block  0,10
SpareTable:     .Block  0,304
Map_Table:      .Block  0,NbrSctrs
SpareCheck:     .Block  0,2
SparePw2:       .Block  0,4

SpareLength     .Equ    .PC.-SpareArray
SpareEnd:       .Block  0,BlockLength-SpareLength

Abort_Stat:     .Block  0,16 ;last abort info

CacheLength     .Equ    NbrSctrs+1
CachStat:       .Block  0,CacheLength
CacheArray:     .Block  0,4*CacheLength

StackPtr:       .Block  0,2
Heap            .Equ    .PC.

StartBlock      .Equ    Heap
BlkOffset       .Equ    StartBlock+3
BlkIndex        .Equ    BlkOffset+1

PBlock          .Equ    Heap
TLBlock         .Equ    PBlock+3
Cur_THS         .Equ    TLBlock+3

TopOfStack      .Equ    $17FF   ;top of external stack

                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Interrupt Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Dmt_PosHeads    .Equ    $00
Dmt_C00         .Equ    $01         ;Command $00
Dmt_C01         .Equ    $02         ;Command $01
Dmt_C02         .Equ    $03         ;Command $02
Dmt_ReadBlock   .Equ    $04         ;Routine: ReadBlock
Dmt_ReadHdr     .Equ    $05         ;Routine: ReadHdr
Dmt_WriteBlock  .Equ    $06         ;Routine: WriteBlock
Dmt_FormatBlock .Equ    $07         ;Routine: FormatBlock
Dmt_FmtTrack    .Equ    $08         ;Routine: FormatTrack
Dmt_LctSctr     .Equ    $09         ;Routine: LocateSector
Dmt_Recal       .Equ    $0A         ;Routine: Restore
Dmt_WV          .Equ    $0B         ;Routine: WriteVerify
Dmt_S_R         .Equ    $0C         ;Routine: ServoRecovery
Dmt_OverLap     .Equ    $0D         ;Routine: OverLappedSeek
Dmt_Rd_Common   .Equ    $0E         ;Routine: Read_Common
Dmt_Seek        .Equ    $0F         ;Routine: Seek
Dmt_ServoOk     .Equ    $10         ;Routine: ServoOk
Dmt_Wr_Common   .Equ    $11         ;Routine: Write_Common
Dmt_TrkCnt      .Equ    $12         ;Routine: TrackCount
Dmt_S_Cmnd      .Equ    $13         ;Routine: ServoCmnd
Dmt_S_Stat      .Equ    $14         ;Routine: ServoStatus
Dmt_S_Store     .Equ    $15         ;Routine: ServoStore
Dmt_S_Load      .Equ    $16         ;Routine: ServoLoad

Dmt_Val         .Equ    500   ;DeadManTimer count

;> IRQ Definitions
```

```
Timer1          .Equ    $20     ;Timer 1 interrupt
Timer0          .Equ    $10     ;Timer 0 interrupt
Serial_Out      .Equ    $10     ;Sio transmitter ready
Serial_In       .Equ    $08     ;Sio receiver ready
Irq_Sector      .Equ    $04     ;Negative edge on sector mark
Irq_Index       .Equ    $01     ;Negative edge on index mark

                .LSTOFF
```