```
                .DO     External
                .LSTON
                .FIN
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>     This file describes all the definitions ( Constants, Vars, I/O,
;>  etc. ) that is used in the Widget Firmware.
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

                .Page

;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Z8 Working Register/Internal Ram Conventions
;>
;>>>>>>>>>>>>>>>>>>>>>>>>


;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Working Register Set $0: Registers 0:3 are used as I/O ports
;>
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>              Register        Use
;>              -------         ---
;>                 0            I/O: Port 0 { Output }
;>                                Bit 0: Adr8
;>                                Bit 1: Adr9
;>                                Bit 2: Adr10/RWI-
;>                                       RWI: Reduced Write Cuurent
;>                                Bit 3: Adr11/PC-
;>                                       PC: Pre-Compensation
;>                                Bit 4: Servo_Reset-
;>                                Bit 5: Format_Enable-
;>                                Bit 6: Z8_Test-
;>                                Bit 7: Read_Header
;>
;>                 1            I/O: Port 1 { Input/Output }
;>                                Bit 0: Adr0/Apple_Bus0
;>                                Bit 1: Adr1/Apple_Bus1
;>                                Bit 2: Adr2/Apple_Bus2
;>                                Bit 3: Adr3/Apple_Bus3
;>                                Bit 4: Adr4/Apple_Bus4
;>                                Bit 5: Adr5/Apple_Bus5
;>                                Bit 6: Adr6/Apple_Bus6
;>                                Bit 7: Adr7/Apple_Bus7
;>
;>                 2            I/O: Port 2 { Input/Output }
;>                                Bit 0: Start-
;>                                Bit 1: EccError-
;>                                Bit 2: Command { Input }
;>                                Bit 3: Busy
;>                                Bit 4: Msel0
;>                                Bit 5: Msel1
;>                                Bit 6: SioRdy { Input }
;>                                Bit 7: Disk_RW-
;>
;>                 3            I/O: Port 3 { Input/Output }
;>                                Bit 0: SioDataIn { Input }
;>                                Bit 1: Sector { Input }
;>                                Bit 2: Index { Input }
```

```
;>                              Bit 3: Sector_Done
;>                              Bit 4: Data_Memory_Select
;>                              Bit 5: Head_Select
;>                              Bit 6: Z8_RW_Clock
;>                              Bit 7: SioDataOut
;>
;>>>>>>>>>>>>>>>>>>>>>>>>
                .Page
;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Misc. Conventions
;>
;>              1. Each major routine is preceded by a psuedo-english
;>                 "Pascal" like explanation of the inputs, outputs,
;>                 and algorithm, as well as register assignments.
;>
;>              2. KeyWords:
;>                  BOOLEAN : True/False values: $01 if true, $00 if false
;>                  BIT     : Single bit values: 1 if true, 0 if false
;>                  BYTE    : 8 bit values
;>                  WORD    : 16 bit values
;>                  PTR     : 16 bit address
;>
;>>>>>>>>>>>>>>>>>>>>>>>>
                .DO     ListDefs
                .LSTON
                .Page
                .FIN
RamOffset       .Equ    $1000
EpromOffset     .Equ    $1000

;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: I/O Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

Port0           .Equ    0               ;I/O Port definitions
Port1           .Equ    1               ;       |
Port2           .Equ    2               ;       |
Port3           .Equ    3               ;      \|/

;> Port 0 Defs:

TimeOut         .Equ    $01             ;hardware timeout flag
EccTest         .Equ    $02             ;Active Low to Ecc gate array chip
RWI             .Equ    $04             ;Reduced Write Current { active low }
PC              .Equ    $08             ;Write Pre-compensation { active low }
ServoRst        .Equ    $00             ;Servo Reset { active low }
Not_ServoRst    .Equ    $10
Not_FmenL       .Equ    $20             ;Format Enable { active low }
FmenL           .Equ    $00
AcEraseL        .Equ    $20
Not_AcEraseL    .Equ    $00
ZTestL          .Equ    $00             ;Z8 Test switch { 1=normal, 0=test }
Not_ZTestL      .Equ    $40
RdHdrH          .Equ    $80             ;State Machine: Ignore Header
Not_RdHdrH      .Equ    $00             ;                { active hi }

;> Port 1 Defs:
;> { none }

;> Port 2 Defs:
```

```
StartL          .Equ    $00             ;State Machine: Start Execution
Not_StartL      .Equ    $01             ;               { active low }
EccError        .Equ    $00             ;Flags Ecc circuitry { active low }
Not_EccError    .Equ    $02
Cmd             .Equ    $04             ;Apple i/o signal: CMD { active hi }
Bsy             .Equ    $08             ;Apple i/o signal: BSY { active hi }
Not_Bsy         .Equ    $00
Msel0           .Equ    $10             ;Memory Select bit 0
Msel1           .Equ    $20             ;Memory Select bit 1

 Apple_Mem      .Equ    $00             ;Mem <--> Apple
 Z8_Mem         .Equ    $10             ;Mem <--> Z8
 Disk_Mem       .Equ    $20             ;Mem <--> State Machine ( Disk )
 Z8_Apple       .Equ    $30             ;Z8 <--> Apple

SioRdy          .Equ    $40             ;Servo's Sio is ready { active hi }
DrwL_Read       .Equ    $80             ;State Machine: Disk Read/Write
DrwL_Write      .Equ    $00             ;               { Write active lo }

P21_In          .Equ    $02             ;Port 2 Bit 1 is an input
P22_In          .Equ    $04             ;Port 2 Bit 2 is an input
P26_In          .Equ    $40             ;Port 2 Bit 6 is an input

;> Port 3 Defs:

SectorMark      .Equ    $02             ;Sector Mark from drive { active hi }
IndexMark       .Equ    $04             ;Index Mark from drive { active hi }
SectDnL         .Equ    $08             ;Sector Done { active low }
Dm              .Equ    $10             ;Data Memory selected { active low }
Hs0             .Equ    $20             ;Head select { 0=head_0, 1=head_1 }
Zrwck           .Equ    $40             ;Z8 test clock for state machine

                .Page

;>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: Registers
;>
;>>>>>>>>>>>>>>>>>>>>>>>>

Wrk_Io          .Equ    $00             ;Working regs set 0 = I/O and scratch
Wrk_Sys         .Equ    $10             ;Working regs set 1 = System
Wrk_Except      .Equ    $20             ;Working regs set 2 = Exception set
Wrk_Sys2        .Equ    $30             ;Working regs set 3 = System2
Wrk_Scr         .Equ    $40             ;Working regs set 4 = Scratch set
Wrk_Cntrl       .Equ    $F0             ;Working regs set F = control set

r0              .Equ    0               ;working register 0
r1              .Equ    1               ;working register 1
r2              .Equ    2               ;working register 2
r3              .Equ    3               ;working register 3
r4              .Equ    4               ;working register 4
r5              .Equ    5               ;working register 5
r6              .Equ    6               ;working register 6
r7              .Equ    7               ;working register 7
r8              .Equ    8               ;working register 8
r9              .Equ    9               ;working register 9
rA              .Equ    $A              ;working register A
rB              .Equ    $B              ;working register B
rC              .Equ    $C              ;working register C
rD              .Equ    $D              ;working register D
rE              .Equ    $E              ;working register E
```

```
rF              .Equ    $F              ;working register F

Hi_Rwi_Reg      .Equ    $20     ;holds cylinder value for RWI and PC
Lo_Rwi_Reg      .Equ    $21

PSector         .Equ    $22             ;last phsyical sector
Free_SlfTst     .Equ    $23             ;pointer to self test routine

Excpt_Status    .Equ    $24             ;exception status register
 Recovery        .Equ    $80            ;global recovery bit { active hi }
 SprTbl_Warn     .Equ    $40            ;spare table has is almost full
 Buf_Damage      .Equ    $20            ;StMach entry buffer is not zero
 PwrRst          .Equ    $10            ;Widget was power reset
 NZero_Stat      .Equ    $08            ;last system status was non-zero
 Led_Stat        .Equ    $01            ;state of LED on controller

SlfTst_Result   .Equ    $25             ;results of self test
 Ram_Fail        .Equ    $80            ;writeable memory failure
 Eprom_Fail      .Equ    $40            ;Eprom checkbyte mismatch
 Disk_Speed      .Equ    $20            ;Disk not up to speed
 Servo_Fail      .Equ    $10            ;Servo is not healthy
 Sector_Cnt      .Equ    $08            ;Not enough sectors!
 State_Fail      .Equ    $04            ;State machine not healthy
 Rw_Fail         .Equ    $02            ;can't read/write!
 No_SprTbl       .Equ    $01            ;No Spare Table found!

RdStat          .Equ    $26             ;status of last read operation
RdErrCnt        .Equ    $27             ;number of retries of last read
WrStat          .Equ    $28             ;status of last write operation
WrErrCnt        .Equ    $29             ;number of retries of last write
SeekCount       .Equ    $2A             ;number of seeks since last arm swing

PwrFlg0         .Equ    $2C             ;power on flag
PwrFlg1         .Equ    $2D
PwrFlg2         .Equ    $2E
PwrFlg3         .Equ    $2F

Dmt_Counter     .Equ    $36     ;double register counter
Scr_Cntr        .Equ    $38     ;double register counter

Cur_Cyl         .Equ    $50

Cylinder        .Equ    $52
Head            .Equ    $54
Sector          .Equ    $55

DiskStatus      .Equ    $56             ;current state of some disk params
 On_Track        .Equ    $80    ;software 'on-track' flag
 RdHdrRecal      .Equ    $40    ;read the header after a recal operation
 Wr_Op           .Equ    $20    ;current access is for a write
 Parked          .Equ    $10    ;heads are currenly parked
 Long_Seek       .Equ    $08    ;do 2 small seeks for 1 long one
 MultiBlk        .Equ    $04    ;MultiBlock semaphore
 SeekComplete    .Equ    $02    ;head/arm is not in motion
 Offset_On       .Equ    $01    ;Auto_Offset is currently on

Seek_Type       .Equ    $57             ;Seek w/wo Auto-Offset
Data_Type       .Equ    $58             ;Spare Table or User Block
Unused_Reg      .Equ    $59             ;reflection of bank register

BlkStat         .Equ    $5A             ;current state of cache logical block
 CachSeek        .Equ    $80      ;a seek is needed for this block
 CachHdChg       .Equ    $40      ;a head change is needed for this block
```

```
 S_Block          .Equ    $01       ;this block is a spare block
 B_Block          .Equ    $02       ;this block is a bad block

Cache_Index       .Equ    $5B     ;Index into cache assuming sequential search

Lst_HiCyl         .Equ    $5C     ;previous seek address
Lst_LoCyl         .Equ    $5D     ; "
Lst_Head          .Equ    $5E     ; "
Lst_Sector        .Equ    $5F     ; "

P01m_Image        .Equ    $04
P3m_Image         .Equ    $05
P01m_StMach       .Equ    $06
P3m_StMach        .Equ    $07
Dm_Mask           .Equ    $08
Start_Mask        .Equ    $09

ScrReg0           .Equ    $40             ;Scratch register 0
ScrReg1           .Equ    $41             ;Scratch register 1
ScrReg2           .Equ    $42             ;Scratch register 2
ScrReg3           .Equ    $43             ;Scratch register 3
ScrReg4           .Equ    $44             ;Scratch register 4
ScrReg5           .Equ    $45             ;Scratch register 5
ScrReg6           .Equ    $46             ;Scratch register 6
ScrReg7           .Equ    $47             ;Scratch register 7
ScrReg8           .Equ    $48             ;Scratch register 8
ScrReg9           .Equ    $49             ;Scratch register 9
ScrRegA           .Equ    $4A             ;Scratch register A
ScrRegB           .Equ    $4B             ;Scratch register B
ScrRegC           .Equ    $4C             ;Scratch register C
ScrRegD           .Equ    $4D             ;Scratch register D
ScrRegE           .Equ    $4E             ;Scratch register E
ScrRegF           .Equ    $4F             ;Scratch register F

R240              .Equ     240            ;Sio Register
R241              .Equ     241            ;Timer Mode Register
R242              .Equ     242            ;Counter/Timer Register 1
R243              .Equ     243            ;T1 Prescaler Load Register
R244              .Equ     244            ;Counter/Timer Register 0
R245              .Equ     245            ;T0 Prescaler Load Register
R246              .Equ     246            ;Port 2 Mode Register
R247              .Equ     247            ;Port 3 Mode Register
R248              .Equ     248            ;Port 0 and 1 mode Register
R249              .Equ     249            ;Interrupt Priority Register
R250              .Equ     250            ;Interrupt Request Register
R251              .Equ     251            ;Interrupt Mask Register
R252              .Equ     252            ;Cpu Flags Register
R253              .Equ     253            ;Working Register Set Pointer
R254              .Equ     254            ;Hi-Byte of Stack Pointer
R255              .Equ     255            ;Low-Byte of Stack Pointer

                  .Page

;>>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      These Symbols are Predefined inside the assembler
;>
;>
;> Sio              .Equ     240           ;Sio Register
;> Tmr              .Equ     241           ;Timer Mode Register
;> T1               .Equ     242           ;Counter/Timer Register 1
;> Pre1             .Equ     243           ;T1 Prescaler Load Register
;> T0               .Equ     244           ;Counter/Timer Register 0
```

```
;> Pre0              .Equ     245             ;T0 Prescaler Load Register
;> P2m              .Equ     246             ;Port 2 Mode Register
;> P3m              .Equ     247             ;Port 3 Mode Register
;> P01m             .Equ     248             ;Port 0 and 1 mode Register
;> Ipr              .Equ     249             ;Interrupt Priority Register
;> Irq              .Equ     250             ;Interrupt Request Register
;> Imr              .Equ     251             ;Interrupt Mask Register
;> Flags            .Equ     252             ;Cpu Flags Register
;> Rp               .Equ     253             ;Working Register Set Pointer
;> Sph              .Equ     254             ;Hi-Byte of Stack Pointer
;> Spl              .Equ     255             ;Low-Byte of Stack Pointer
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>

                .Page

;>>>>>>>>>>>>>>>>>>>>>>>>>
;>
;>      Constant Definitions: Control Register Stuff
;>
;>>>>>>>>>>>>>>>>>>>>>>>>>

;> Control Register 240 ( $F0 ): Serial I/O register
;>              { Read: Receive Data }
;>              { Write: Transmit Data }

;> Control Register 241 ( $F1 ): Timer Mode Register

;> Tout Modes: Bits 7 and 6
  T0_Out        .Equ    $40             ;Port 3, bit 6 gets timer 0 out
  T1_Out        .Equ    $80             ;Port 3, bit 6 gets timer 1 out
  Int_Out       .Equ    $C0             ;Port 3, bit 6 gets system clock

;> Tin Mode: Bits 5 and 4
  Ext_Clk       .Equ    $00             ;use external clock on timer
  Gate_Clk      .Equ    $10             ;use internal clock on timer
  Non_Retrig    .Equ    $20             ;load T1 after a high-low trans
  Retrig        .Equ    $30             ;like Non-Retrig, but load on all trans.

  T1_CntEn      .Equ    $08             ;enable T1 counting operations
  T1_CntDis     .Equ    $00

  T1_Load       .Equ    $04             ;Load T1

  T0_CntEn      .Equ    $02             ;enable T0 counting operations
  T0_CntDis     .Equ    $00

  T0_Load       .Equ    $01             ;Load T0

;> Control Register 242 ( $F2 ): Counter/Timer Register T1
;>              { Write: Initial Value }
;>              { Read: Current Count Value }

;> Control Register 243 ( $F3 ): T1 Prescaler Load Register ( Pre 1 )
  Single_Pass   .Equ    $00
  Mod_n         .Equ    $01
  T1_Int_Clk    .Equ    $02
  T1_Ext_Clk    .Equ    $00
;> Bits 7:2 are used to provide a 6-bit binary Prescale Value

;> Control Register 244 ( $F4 ): Counter/Timer Register T0
;>              { Write: Initial Value }
;>              { Read: Current Count Value }
```

```
;> Control Register 245 ( $F5 ): T0 Prescaler Load Register ( Pre 0 )
;>      { Same as Pre 1, but no external clock }

;> Control Register 246 ( $F6 ): Port 2 Mode ( P2m)
;>  When a bit of P2m is set, the corresponding bit in Port 2 is defined
;>  as an input, if reset then the corresponding bit for Port 2 is
;>  defined as an output.

;> Control Register 247 ( $F7 ): Port 3 Mode ( P3m )
  Open_Drain    .Equ    $00
  Totem_Pol     .Equ    $01
  B2_5_Io       .Equ    $00             ;Bit 2=input, Bit 5=output
  B2_5_Hs       .Equ    $04             ;Bit 2=DAV0-, Bit 5=RDY0-
  B3_4_Io       .Equ    $00             ;Bit 3=input, Bit 4=output
  B3_4_Idm      .Equ    $10             ;Bit 3=input, Bit 4=DM-
  B3_4_Hs       .Equ    $18             ;Bit 3=DAV1-, Bit 4=RDY1-
  B1_6_Io       .Equ    $00             ;Bit 1=Tin  , Bit 6=Tout
  B1_6_Hs       .Equ    $20             ;Bit 1=DAV2-, Bit 6=RDY2-
  B0_7_Io       .Equ    $00             ;Bit 0=input, Bit 7=output
  B0_7_Ser      .Equ    $40             ;Bit 0=Sio in, Bit 7=Sio out
  Par_On        .Equ    $80             ;Parity On
  Par_Off       .Equ    $00             ;Parity Off

;> Control Register 248 ( $F8 ): Port 0 and 1 Mode ( P01m )
  P0_03_Out     .Equ    $00             ;select Port 0, bits 0:3 as outputs
  P0_03_In      .Equ    $01             ;select Port 1, bits 0:3 as inputs
  P0_03_Adr     .Equ    $02             ;select P0, bits 0:3 as adr bits 8:11
  Stack_Ext     .Equ    $00             ;select external stack
  Stack_In      .Equ    $04             ;select internal stack
  P1_Out        .Equ    $00             ;select Port 1 as outputs
  P1_In         .Equ    $08             ;select Port 1 as inputs
  P1_Adr        .Equ    $10             ;select Port 1 as adr/data bits 0:7
  P1_Tri        .Equ    $18             ;put port 1 into Tri-State
  Mem_Ext       .Equ    $20             ;extended memory timing
  Mem_Norm      .Equ    $00             ;normal memory timing
  P0_47_Out     .Equ    $00             ;select Port 0, bits 4:7 as outputs
  P0_47_In      .Equ    $40             ;select Port 0, bits 4:7 as inputs
  P0_47_Adr     .Equ    $80             ;select P0, bits 4:7 as adr bits 12:15

;> Control Register 249 ( $F9 ): Interrupt Priority Register ( Ipr )
;>              { To Be Established }

;> Control Register 250 ( $FA ): Interrupt Request Register ( Irq )
;>              { To Be Established }

;> Control Register 251 ( $FB ): Interrupt Mask Register ( Imr )
;>              { To Be Established }

;> Control Register 252 ( $FC ): Flag Register ( Flags )
;>              { To Be Established }
                .LSTOFF
```