```
*-----------------------------------------------------------
* Program Number: 68K Monitor for S100Computers.com
* Written by    : John Monahan
* Date Created  : 11/11/2011
* Description   : Basic monitor for 68K S-100 board
*
*   'A=Memmap    C=XMODEM(Bin) D=Disp RAM    E=Echo        F=Fill RAM'
*   'G=Goto RAM  H=Math        I=Time        K=Menu        L=Test Ints'
*   'M=Move RAM  N=IDE Menu    Q=Port I/O    U=Serial Test S=Subs RAM'
*   'T=Type RAM  V=Verify RAM  X=Signals     Y=Patch       Z=Back to Z80'
*
*-----------------------------------------------------------
;           V1.5    03/07/2012              ;Corrected line length display of RAM (D & T Commands)
;           V1.6    03/07/2012              ;Added initilization of Interrupt routines in low RAM
;           V1.7    03/08/2012              ;Add test interrupts routine, "L" CMD.
;           V1.8    03/09/2012              ;Code to switch back to Z80, and hardware signals anal
;           V1.9    03/18/2012              ;Added IDE Board Diagnostic Section
;           V1.91   03/27/2012              ;Substitute RAM redone
;           V2.0    04/02/2012              ;Added IDE Menu Items and Y command
;           V2.1    04/26/2013              ;Fixed numerous small bugs, RAM display map,D,F,M X co
;           V2.2    04/27/2013              ;Display RAM (D CMD) also displays ASCII
;           V2.3    04/23/2014              ;Allow output to 16 bit ports (>0FFH), DMA1* port swit
;           V2.4    04/30/2014              ;Cleanup console I/O routines, add serial port I/O
;           V2.5    05/2/2014               ;Added XMODEM .bin file download capabilities over ser
;           V2.6    06/12/2014              ;Corrected QO/QI port bug
;           V3.0    7/27/2020               ;Damian Wildie corrections and addition of working IDE
;           V3.1    2/1/2021                ;Corrected numerous small errors. Sec Writes corercted
;           V3.11    2/1/2021               ;Made commands the same as for 68030 Monitor..
;           V3.12   2/8/2021                ;Corrections to Drive ID info
;
;
;
;Programming a Wellon VP-290 with 28C256 EEPROMS.
;Assemble and make a S68 file (Project Menu for EASy68K)
;Load first, even byte in dropdown menu for "File Mode"
;For "From File Address(Hex) enter FD0000 (Note "To Buffer Address (HEX) is 0)
;File Size is 8000 (for X28C256's)
;For "Auto Format Detect" use Motorola S
;Repeat, for second EEPROM using Odd Bytes for "File Mode"
;(Note for the 68030 Board only one PROM is needed).
;
;
BELL                EQU     $07
BLANK               EQU     $20
CR                  EQU     $0D
LF                  EQU     $0A
ESC                 EQU     $1B
TAB                 EQU     $09
SOH                 EQU     1               ; For Modem etc.
EOT                 EQU     4
ACK                 EQU     6
NAK                 EQU     $15

                                            ;Note, only one of the following 3 equates can be 1
SIMMULATOR          EQU     0               ;Set to 1 if using EASy68K Simmulator (Console I/O wil
S100_TEST           EQU     0               ;Set to 1 if using S100 RAM (We will not use INT's for
ROM_CODE            EQU     1               ;Set to 1 for ROM code (We will not use INT's for cons


;Propeller Console IO S-100 board or SD SYSTEMS VIDIO BOARD FOR CONSOLE I/O(<---These must configu

KEYSTAT             EQU     $00FF0000
KEYIN               EQU     $00FF0001       ;Console input port. Normally the Propeller Driven S-1
KEYOUT              EQU     $00FF0001       ;Console output port. Normally the Propeller Driven S-



;-------------- THIS IS MY PORT TO OUTPUT DATA TO HP 4050T LASAR PRINTER  (IMSAI 8PIO Board)

PRINTER_STATUS      EQU     $00FF0005       ;IN, HP PARRELL PORT
PRINTER_OUT         EQU     $00FF0005       ;OUT
```

```
COUNTS_SEC          EQU     $18
COUNTS_MIN          EQU     1092
COUNTS_HOUR         EQU     $07             ;Seems this value is used with AT/CMOS chip (was 65543

UPDATE_TIMER                                EQU     $80

CMOS_SECONDS                                EQU     $0  ;RAM offsets for CMOS Registers
CMOS_MINUTES                                EQU     $2
CMOS_HOURS          EQU     $4


   IFEQ SIMMULATOR                          ;If SIMMULATOR = 0 (Normal EEPROM)
Patch_RAM           EQU    $00F40000        ;Location of Patch code area
   ENDC

   IFNE SIMMULATOR                          ;If SIMMULATOR = 1 (For testing here within EASy68K &
Patch_RAM           EQU    $00008000
   ENDC


;-------------- S100Computers IDE BOARD PORT ASSIGNMENTS  (30-34H)

;Ports for 8255 chip. Change these to specify where the 8255 is addressed,
;and which of the 8255's ports are connected to which IDE signals.
;The first three control which 8255 ports have the IDE control signals,
;upper and lower data bytes.  The forth one is for mode setting for the
;8255 to configure its ports, which must correspond to the way that
;the first three lines define which ports are connected.

IDEportA            EQU     $00FF0030       ;lower 8 bits of IDE interface
IDEportB            EQU     $00FF0031       ;upper 8 bits of IDE interface
IDEportC            EQU     $00FF0032       ;control lines for IDE interface
IDECtrlPort         EQU     $00FF0033       ;8255 configuration port
IDEDrivePort                                EQU     $00FF0034   ;To select the 1st or 2nd CF card/

IDE_Reset_Delay     EQU     $80             ;Time delay for reset/initilization (~66 uS, with 8MHz

READcfg8255         EQU     %10010010       ;Set 8255 IDEportC out, IDEportA/B input
WRITEcfg8255        EQU     %10000000       ;Set all three 8255 ports output

;IDE control lines for use with IDEportC.

IDEa0line           EQU     $01             ;direct from 8255 to IDE interface
IDEa1line           EQU     $02             ;direct from 8255 to IDE interface
IDEa2line           EQU     $04             ;direct from 8255 to IDE interface
IDEcs0line          EQU     $08             ;inverter between 8255 and IDE interface
IDEcs1line          EQU     $10             ;inverter between 8255 and IDE interface
IDEwrline           EQU     $20             ;inverter between 8255 and IDE interface
IDErdline           EQU     $40             ;inverter between 8255 and IDE interface
IDErstline          EQU     $80             ;inverter between 8255 and IDE interface
;
;Symbolic constants for the IDE Drive registers, this makes the
;code more readable than always specifying the address pins

REGdata             EQU     IDEcs0line
REGerr              EQU     IDEcs0line+IDEa0line
REGseccnt           EQU     IDEcs0line+IDEa1line
REGsector           EQU     IDEcs0line+IDEa1line+IDEa0line
REGcylinderLSB      EQU     IDEcs0line+IDEa2line
REGcylinderMSB      EQU     IDEcs0line+IDEa2line+IDEa0line
REGshd              EQU     IDEcs0line+IDEa2line+IDEa1line  ;(0EH)
REGcommand          EQU     IDEcs0line+IDEa2line+IDEa1line+IDEa0line    ;(0FH)
REGstatus           EQU     IDEcs0line+IDEa2line+IDEa1line+IDEa0line
REGcontrol          EQU     IDEcs1line+IDEa2line+IDEa1line
REGastatus          EQU     IDEcs1line+IDEa2line+IDEa1line+IDEa0line

;IDE Command Constants.  These should never change.

COMMANDrecal        EQU     $10
COMMANDread         EQU     $20
COMMANDwrite        EQU     $30
```

```
START:
            LEA     Signon,A2               ;Show we are alive
            BSR     PRINT_STRING

            BSR     SERIAL_INITILIZE_A      ;Initilize Consoel-IO board Serial Port A
            BSR     SERIAL_INITILIZE_B      ;Initilize Consoel-IO board Serial Port B
            LEA     SMSG,A2
;           BSR     SPEAK_STRING            ;Speak out signon the message

            BSR     LOW_RAM_INITILIZE       ;Initilize low RAM Int Vectors (for all modes)

                                            ;Initilize HIGH RAM to 0  (Used by IDE Routines)
            LEA     BeginRAM,A2             ;START OF WORK RAM (PAST STACK)
            MOVE.L  #(EndRAM-BeginRAM),D0   ;BYTES TO ZERO
            CLR.L   D1

ZERO_RAM:   MOVE.B  D1,(A2)+                ;ZERO MEMORY
            SUBQ.L  #1,D0
            BNE     ZERO_RAM


loop:       LEA     Prompt,A2               ;Show CR,LF,'>'
            BSR     PRINT_STRING
            CLR.L   D1                      ;Just to be on the safe side
            BSR     GETCHAR                 ;Get a menu character (WITH ECHO)
            AND.B   #$7F,D1                 ;Just to be safe, strip any potential parity bit
            BSR     TOUPPER                 ;Lower case to Upper case for lookup table

            CMP.B   #'A',D1
            BLT     ERR
            CMP.B   #'Z',D1
            BGT     ERR
            SUB.B   #'A',D1
            LSL.L   #2,D1                   ;X4 for offset into table
            LEA     ctable,A2               ;Start of cmd table
            MOVE.L  (A2,D1),A3              ;Add X4 offset
            JMP     (A3)

;-------------------------------------------------------------------------------------------------
ERR:        CMP.B   #CR,D1                  ;If CR just return
            BEQ     loop
            MOVE.L  D1,-(A7)                ;> Save D1
            LEA     BadCmdMsg,A2            ;Non menu selection
            BSR     PRINT_STRING
            MOVE.L  (A7)+,D7                ;Put D1 in D7
            BSR     PUTLONG_D7
            LEA     H_MSG_CRLF,A2           ;H, then CR,LF
            BSR     PRINT_STRING
            BRA     loop                    ;Back to start for next command

;-------------------------------------------------------------------------------------------------
SHOW_MENU:                                  ;Display this monitors commands on CRT
            LEA     Menu,A2                 ;Menu string
            BSR     PRINT_STRING
            BRA     loop                    ;Back to start for next command


;-------------------------------------------------------------------------------------------------
MEM_MAP:    MOVE.L  #0,A3                   ;A Command.   Do Memory Map. Pointer to RAM area A3=0
NEWLINE:    BSR     CRLF
            MOVE.L  A3,D7
            BSR     PUTLONG_D7              ;Print long value of D7
            MOVE.L  #64,D3                  ;64 characters across per line
            MOVE.L  #$FFFFFFFF,D5

            MOVE.B  #BLANK,D1
            BSR     PUTCHAR

START1:     MOVE.L  (A3),D1                 ;Is there RAM/ROM there
            NOT.L   D1
            MOVE.L  D1,(A3)                 ;See if we can flip bits
```

```
            MOVE.L  (A7)+,A4            ;Next Show ASCII for this line, Back to origional RAM
            MOVE.L  (A7)+,A3            ;Were stored above
            BSR     SPACE
            BSR     SPACE
            MOVE.B  #16,D4              ;Count of characters across

PARMS_OK4:  MOVE.B  (A3),D1             ;Get RAM byte to D1
            CMP.B   #' ',D1
            BLT     PRINT_DOT
            CMP.B   #$7F,D1
            BGE     PRINT_DOT
PARMS_OK6   BSR     PUTCHAR             ;Print character

            ADDQ.L  #1,A3
            SUBQ.B  #1,D4
            TST.B   D4                  ;Have we done 16 characters across
            BNE     PARMS_OK4

            CMP.L   A3,A4               ;Are we done wit total data display yet
            BLE     LOOP
            BRA     PARMS_OK5

PRINT_DOT:
            MOVE.B  #'.',D1
            BRA     PARMS_OK6


;-------------------------------------------------------------------------------------------

FILL_RAM:                               ;F Command. Fill RAM with one byte value
            BSR     GETLONG_D7          ;Get start address
            CMP.B   #',',D2             ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A3               ;Save in A3

            BSR     GETLONG_D7          ;End address
            CMP.B   #',',D2             ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A4               ;Save in A4

            CMP.L   A3,A4
            BEQ     LOOP                ;If the same nothing to display
            BGE     FILL_OK
            MOVE.L  A3,A5               ;Else swap values
            MOVE.L  A4,A3
            MOVE    A5,A4

FILL_OK:    ADD.L   #1,A4               ;End + 1
            BSR     GETBYTE_D7          ;get Hex value in D7 (0-FF)
            CMP.B   #CR,D2              ;Is it valid
            BNE     ERROR

FILL_OK1:   MOVE.B  D7,(A3)             ;D7 to RAM
            ADDQ.L  #1,A3
            CMP.L   A3,A4               ;Are we done yet
            BLE     LOOP
            BRA     FILL_OK1



;-------------------------------------------------------------------------------------------

SUBS_RAM:                               ;S Command. Substitute RAM with one byte values
            BSR     GETLONG_D7          ;Get start address
            CMP.B   #CR,D2              ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A3               ;--- Save in A3 (also leave in A7)

SUBS_RAM2:  BSR     CRLF                ;New line
```

```
QUERY_PORT:
            CLR.L   D1                      ;Just to be on the safe side
            BSR     GETCHAR                 ;get a menu character
            BSR     TOUPPER                 ;Lower case to Upper case

            CMP.B   #'I',D1                 ;Is it a port input request
            BEQ     QUERY_IN
            CMP.B   #'O',D1                 ;Is it a port output request
            BEQ     QUERY_OUT
            BRA     ERROR                   ;Must be an error

QUERY_IN:   BSR     GETLONG_D7              ;Get (Byte only) Port Hex value in D7 (0-FF)
            TST.B   D3                      ;Byte count > 0
            BEQ     LOOP
            CMP.B   #ESC,D2                 ;If ESC then we abort
            BEQ     LOOP
            CMP.B   #CR,D2                  ;If CR then we also abort
            BNE     ERROR
            MOVE.B  D7,D6                   ;store in D6 (also in D7)

            LEA     PortMsg,A2              ;'Port xx'
            BSR     PRINT_STRING
            BSR     PUTBYTE_D6              ;Display Port value
            MOVE.B  #'H',D1                 ;'H'
            BSR     PUTCHAR
            MOVE.B  #'=',D1                 ;'='
            BSR     PUTCHAR

            MOVE.L  #$00FF0000,D6           ;Point to Port RAM area
            OR.B    D7,D6                   ;OR in the hardware value
            MOVE.L  D6,A2                   ;A2 now has port address
            MOVE.B  (A2),D6                 ;Get value at port

            BSR     PUTBYTE_D6              ;Display Byte value
            MOVE.B  #'H',D1                 ;'H'
            BSR     PUTCHAR
            MOVE.B  #' ',D1                 ;' '
            BSR     PUTCHAR
            MOVE.B  #'(',D1                 ;'('
            BSR     PUTCHAR
            BSR     PUTBITS_D6
            MOVE.B  #')',D1                 ;')'
            BSR     PUTCHAR
            BRA     LOOP


QUERY_OUT:
            BSR     GETLONG_D7              ;Get Port value (value in D7, 0-FFFF)
            TST.B   D3                      ;Byte count > 0
            BEQ     LOOP
            CMP.B   #ESC,D2                 ;If ESC then we abort
            BEQ     LOOP
            CMP.B   #',',D2                 ;If CR then we also abort
            BNE     ERROR
            MOVE.L  D7,D4                   ;<<< Store (WORD) port # in D4

            BSR     GETBYTE_D7              ;Get data in D7 (0-FF)
            TST.B   D3                      ;Byte count > 0
            BEQ     LOOP
            CMP.B   #ESC,D2                 ;If ESC then we are done
            BEQ     LOOP
            CMP.B   #CR,D2                  ;If CR then we are done
            BNE     LOOP
            MOVE.L  D7,D5                   ;<<< Store data (BYTE) to send to port in D5

            LEA     PortMsg2,A2             ;'Send to Port xxxx'
            BSR     PRINT_STRING
            MOVE.L  D4,D6
            BSR     PUTWORD_D6              ;Display Port value (as a word)
            MOVE.B  #'H',D1                 ;'H'
            BSR     PUTCHAR
```

```
;-------------------------------------------------------------------------------------------

VERIFY_RAM:                             ;M Command. Verify two RAM locations have the same dat
            BSR     GETLONG_D7          ;Get start address
            CMP.B   #',',D2             ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A3               ;--- Save in A3

            BSR     GETLONG_D7          ;End address
            CMP.B   #',',D2             ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A4               ;--- Save in A4

            CMP.L   A3,A4
            BEQ     LOOP                ;If the same nothing to display
            BGE     VERIFY_OK
            MOVE.L  A3,A5               ;Else swap values
            MOVE.L  A4,A3
            MOVE    A5,A4

VERIFY_OK:  ADD.L   #1,A4               ;End + 1
            BSR     GETLONG_D7          ;End address
            CMP.B   #CR,D2              ;Is it valid
            BNE     ERROR
            MOVE.L  D7,A5               ;--- Save in A5

VERIFY_OK1: CMP.B   (A3)+,(A5)+
            BNE     BAD_MATCH
VERIFY_OK2: CMP.L   A3,A4               ;Are we done yet
            BGE     VERIFY_OK1
            BRA     LOOP

BAD_MATCH:  BSR     CRLF                ;New line
            MOVE.L  A3,D7
            SUBQ.L  #1,D7               ;Backup to problem
            MOVE.L  D7,A3
            BSR     PUTLONG_D7          ;Show first address
            MOVE.B  #BLANK,D1
            BSR     PUTCHAR
            MOVE.B  (A3)+,D6
            BSR     PUTBYTE_D6
            MOVE.B  #BLANK,D1

            BSR     PUTCHAR
            BSR     PUTCHAR

            MOVE.L  A5,D7
            SUBQ.L  #1,D7               ;Backup to problem
            MOVE.L  D7,A5
            BSR     PUTLONG_D7          ;Show first address
            MOVE.B  #BLANK,D1
            BSR     PUTCHAR
            MOVE.B  (A5)+,D6
            BSR     PUTBYTE_D6
            BSR     CRLF
            BRA     VERIFY_OK2



;-------------------------------------------------------------------------------------------

ECHO_ASCII:                             ;E Command. Get ASCII typed on keyboard and display on
            LEA     ECHO_MSG,A2         ;"Will echo each keyboard char on screen"
            BSR     PRINT_STRING

            BSR     CRLF                ;New line
ECHO2:      CLR.L   D1                  ;Just to be on the safe side
            BSR     GETCHAR             ;get a character

            CMP.B   #ESC,D1             ;If ESC then we abort
            BEQ     LOOP
```

```
            BSR     CRLF
            MOVE.W  #$2000,SR               ;Allow ALL INTERRUPTS, SUPERVISOR MODE
INT_LOOP:   MOVE.B  #'.',D1                 ;Continously print .......
            BSR     PUTCHAR                 ;Echo character
            BSR     GETSTAT                 ;Is there a keyboard character ready
            BEQ     INT_LOOP
            BSR     GETCHAR                 ;Get a keyboard character
            CMP.B   #ESC,D1                 ;ESC to abort test
            BEQ     INT_LOOP1
            BSR     TOUPPER                 ;Lower case to Upper case
            BSR     PUTCHAR
            BRA     INT_LOOP
INT_LOOP1:  MOVE.W  #$2700,SR               ;MASK OFF INTERRUPTS
            BSR     CRLF
            BRA     START                   ;Reload Monitor



LOW_RAM_INITILIZE:
            MOVE.L  #$8,A2                  ;Skip (Reset vectors) STACK & ADDRESS vectors etc.
            LEA     BUS_ERROR,A3            ;8H = BUS ERROR
            MOVE.L  A3,(A2)+
            LEA     ADDRESS_ERROR,A3        ;CH = ADDRESS
            MOVE.L  A3,(A2)+
            LEA     ILLEGAL_ERROR,A3
            MOVE.L  A3,(A2)+
            LEA     ZERO_ERROR,A3
            MOVE.L  A3,(A2)+
            LEA     ILLEGAL_ERROR,A3
            MOVE.L  A3,(A2)+
            LEA     ILLEGAL_ERROR,A3
            MOVE.L  A3,(A2)+
            LEA     PRIVILEGE_ERROR,A3
            MOVE.L  A3,(A2)+
            LEA     TRACE_ERROR,A3
            MOVE.L  A3,(A2)+

            MOVE.L  #$28,A2                 ;General Error, Starting at 1010 Illegal Opcode
            LEA     ABORTE,A3               ;Use default Error message
INIT0:      MOVE.L  A3,(A2)+
            CMPA.L  #$60,A2                 ;Continue up to Spurious Interrupt (60H)
            BMI.S   INIT0

            LEA     SPURIOUS_INT,A3         ;Spurious Interrupt vector
            MOVE.L  A3,(A2)+
            LEA     L1_INTERRUPT,A3         ;Hardware Interrupts
            MOVE.L  A3,(A2)+
            LEA     L2_INTERRUPT,A3
            MOVE.L  A3,(A2)+
            LEA     L3_INTERRUPT,A3
            MOVE.L  A3,(A2)+
            LEA     L4_INTERRUPT,A3
            MOVE.L  A3,(A2)+
            LEA     L5_INTERRUPT,A3
            MOVE.L  A3,(A2)+
            LEA     L6_INTERRUPT,A3
            MOVE.L  A3,(A2)+
            LEA     L7_INTERRUPT,A3
            MOVE.L  A3,(A2)+


   IFEQ SIMMULATOR                          ;If SIMMULATOR = 0, i.e. we are in S100 or ROM mode, w
            LEA     TRAPS,A3                ;16 Trap vectors
INIT1:      MOVE.L  A3,(A2)+
            CMPA.L  #$C0,A2                 ;Up to C0H
            BMI.S   INIT1
    ENDC
            MOVE.L  #$C0,A2                 ;Just to be sure we are at the correct place
            LEA     ABORTE,A3               ;Use default Error message

INIT2:      MOVE.L  A3,(A2)+                ;INITIALIZE VECTORS
```

```
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

L3_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     L3_INTERRUPT_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

L4_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     L4_INTERRUPT_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

L5_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     L5_INTERRUPT_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

L6_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     L6_INTERRUPT_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

L7_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     L7_INTERRUPT_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

ABORTE:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     INT_ERR_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

TRAPS:
            MOVEM.L D0-D7/A0-A6,-(A7)       ;SAVE ALL REGISTERS
            LEA     TRAPS_ERR_MSG,A2
            BSR     PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6       ;POP ALL REGISTERS
            RTE

;-------------------------------------------------------------------------------------------
TIME:
            LEA     TIME_MSG,A2             ;Time determination module not written yet
            BSR     PRINT_STRING
            BRA     LOOP


;-------------------------------------------------------------------------------------------

JMP_Z80:                                    ;Switch back to Z80 Master CPU
            MOVE.L  #SW68K ,D5              ;Point to status Port 0ECH for DMA1 switch line
            MOVE.L  D5,A2

            MOVE.B  #00,(A2)                ;Currently will use TMA line #1 to switch in/out the 6
            NOP                             ;<-- 68K Is held in HALT mode here until released agai
            NOP
            NOP
            NOP
            BRA     LOOP
```

```
            MOVE.W  D2,(A3)
            MOVE.W  D2,(A3)
            BRA     WR_TEST1                ;Must Hit Reset button to abort




PATCH:                                      ;Y Command, Quick patch to move RAM 4000H-9000H to F40
            LEA     PATCH_MSG,A2            ;"Moving Code 4000H-9000H to F4000H, Then jump to that
            BSR     PRINT_STRING            ;The CPM3/Z80 SID program will place any test .bin fil
                                            ;in RAM (no matter what its final ORG is).
            BSR     GETCHAR                 ;Get character in D1
            BSR     TOUPPER
            CMP.B   #CR,D1
            BEQ     PATCH0
            BRA     LOOP

PATCH0:     BSR     CRLF                    ;Start patch
            MOVE.L  #$4000,A3               ;This command will then move test versions of this mon
            MOVE.L  #$F4000,A5              ;program) up out of the way to F4000H and execute it f
            MOVE.L  #$8000,D1               ;This shold be large enough for a monitor copy

PATCH1:     MOVE.B  (A3)+,(A5)+
            SUBQ.L  #1,D1
            BNE     PATCH1
            JMP     Patch_RAM               ;Jump to this loction ($00F40000)



;-------------------------------------------------------------------------------------------
;-------------------------------------------------------------------------------------------
;------------- MAIN IDE DRIVE DIAGNOSTIC MENU ----------------------------------------------
;       Normally the DMA buffers will reside in the RAM on the 68K board itself at 00FD9000H
;-------------------------------------------------------------------------------------------

MY_IDE:     BSR     CLEAR_ID_BUFFER         ;Clear ID Buffer

            BSR     SEL_DRIVE_A             ;Select the first Drive/CF card
            BSR     IDEinit                 ;Initialize the board and drive 0. If there is no driv
            BEQ     INIT1_OK

            LEA     INIT_1_ERROR,A2
            BSR     PRINT_STRING
            BRA     LOOP

INIT1_OK:
            BSR     SEL_DRIVE_B             ;Select the second Drive/CF card (Do not mess with CPM
            BSR     IDEinit                 ;Initialize drive 1. If there is no drive abort
            BEQ     INIT2_OK

            BSR     CLEAR_ID_BUFFER         ;Clear ID Buffer

            LEA     INIT_2_ERROR,A2         ;Warn second IDE drive did not initilize
            BSR     PRINT_STRING

INIT2_OK:
            BSR     SEL_DRIVE_A             ;Back to first drive/CF Card

            BSR     DRIVE_ID                ;Get the drive 0 id info. If there is no drive just ab
            BEQ     INIT3_OK

            LEA     BAD_DRIVE,A2
            BSR     PRINT_STRING
            BRA     LOOP

INIT3_OK:                                   ;Set default position will be first sector block
            LEA     IDE_Buffer+12,A2
            CMP.W   #0,(A2)                 ;Is it non zero
            BNE     INIT4_OK                ;If there are zero sectors then something wrong
```

```
            BRA     IDE_LOOP                ;Back to IDE Menu

SET_DRIVE_B:                                ;Select First Drive
            BSR     SEL_DRIVE_B
            BRA     IDE_LOOP                ;Back to IDE Menu

SEL_DRIVE_A:                                ;Select First Drive
            LEA     IDE_SEL_A,A2            ;Say so
            BSR     PRINT_STRING
            CLR.B   D1
SELECT_DRIVE:
            MOVE.B  D1,CURRENT_IDE_DRIVE
            MOVE.B  D1,IDEDrivePort         ;Select Drive 0 or 1
            RTS

SEL_DRIVE_B:                                ;Select Drive 1
            LEA     IDE_SEL_B,A2            ;Say so
            BSR     PRINT_STRING
            MOVE.B  #1,D1
            JMP     SELECT_DRIVE



DRIVE_ID:
            BSR     IDEwaitnotbusy
            BGE     L_5
            CLR     D1
            SUBQ.B  #1,D1                       ;NZ if error
            RTS                                 ;If Busy return NZ

L_5:        MOVE.B  #COMMANDid,D4
            MOVE.B  #REGcommand,D5
            BSR     IDEwr8D                     ;Issue the command

            BSR     IDEwaitdrq                  ;Wait for Busy=0, DRQ=1
            BGE     L_6
            BRA     SHOWerrors

L_6:        CLR.B   D6                          ;256 words

            LEA     IDE_Buffer,A2               ;Store data here
            BSR     MoreRD16                    ;Get 256 words of data from REGdata port to IDE_Bu

                                                ;Decode Drive INFO
            LEA     msgmdl,A2                   ;Drive/CF Card Information:-
            BSR     PRINT_STRING

            LEA     IDE_Buffer+54,A2            ;@ Word 27
            MOVE.B  #20,D3                      ;Character count in words
            BSR     Print_ID_Info               ;Print [A2], [D3] X 2 characters
            BSR     CRLF
                                                ;print the drive's serial number
            LEA     msgsn,A2                    ;'S/N:   '
            BSR     PRINT_STRING
            LEA     IDE_Buffer+20,A2            ;@ Word 10
            MOVE.B  #10,D3                      ;Character count in words
            BSR     Print_ID_Info
            BSR     CRLF

            LEA     msgrev,A2                   ;'Rev:   '
            BSR     PRINT_STRING
            LEA     IDE_Buffer+46,A2            ;@ Word 23
            MOVE.B  #4,D3
            BSR     Print_ID_Info               ;Character count in words
            BSR     CRLF
                                                ;Print the drive's cylinder, head, and sector spec
            LEA     msgcy,A2                    ;'Cylinders: '
            BSR     PRINT_STRING
            LEA     IDE_Buffer+2,A2             ;@ Word 1
            BSR     Print_ID_Hex
```

```
            CMP.B   #'Y',D1
            BEQ     WR_SEC_OK1
            BSR     CRLF                    ;Here if there was a problem
            BRA     IDE_LOOP                ;Back to IDE Menu

WR_SEC_OK1: BSR     CRLF
            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)            ;DMA initially to IDE_Buffer

            BSR     WRITESECTOR             ;Will write whatever is in the IDE_Buffer

            BEQ     Main2B
            BSR     CRLF                    ;Here if there was a problem
            BRA     IDE_LOOP                ;Back to IDE Menu
Main2B:
            LEA     msgwr,A2                ;Sector written OK
            BSR     PRINT_STRING

            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)            ;DMA initially to IDE_Buffer
            BSR     DISPLAY_SECTOR
            LEA     CR_To_Continue,A2
            BSR     PRINT_STRING
            BSR     GETCHAR
            BSR     CRLF
            BRA     IDE_LOOP                ;Back to IDE Menu



;------------------ Fill a sector with a Byte Value (in D5)

FILL_SEC:   LEA     FILL_BYTE_MSG,A2        ;Enter sector Fill byte
            BSR     PRINT_STRING
            BSR     GETBYTE_D7              ;Get data in D7 (0-FF)
            CMP.B   #ESC,D2                 ;If ESC then we are done
            BEQ     IDE_LOOP
            CMP.B   #CR,D2                  ;If CR then we are done
            BNE     IDE_LOOP
            MOVE.L  D7,D5                   ;<<< Store data (BYTE) in D5

            LEA     CONFIRM_WR_MSG,A2       ;Are you sure?
            BSR     PRINT_STRING
            BSR     GETCHAR
            BSR     TOUPPER
            CMP.B   #'Y',D1
            BEQ     CLEAR_BUFFER
            BSR     CRLF                        ;Here if abort
            BRA     IDE_LOOP                ;Back to IDE Menu


CLEAR_BUFFER:
            LEA     IDE_Buffer,A2
            MOVE.W  #512,D2                 ;512 bytes total to fill
CLEAR0:     MOVE.B  D5,(A2)+
            SUBQ.W  #1,D2
            BNE     CLEAR0

            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)            ;DMA initially to IDE_Buffer

            BSR     WRITESECTOR             ;Will write whatever is in the IDE_Buffer

            BEQ     CLEAR2
            BSR     CRLF                    ;Here if there was a problem
            BRA     IDE_LOOP                ;Back to IDE Menu
CLEAR2:
            LEA     msgwr,A2                ;Sector written OK
            BSR     PRINT_STRING

            LEA     IDE_BUFFER,A4
```

```
            LEA     CONTINUE_MSG,A2         ;If an error ask if we wish to continue
            BSR     PRINT_STRING
            BSR     GETCHAR
            BSR     TOUPPER
            CMP.B   #ESC,D1                 ;Abort if ESC
            BNE     SEQOK
            BSR     CRLF
            BRA     IDE_LOOP                ;Back to IDE Menu

SEQOK:      BSR     DISPLAY_POSITION        ;Display current Track,sector,head#

            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)            ;DMA initially to IDE_Buffer

            BSR     DISPLAY_SECTOR

            BSR     GETSTAT                 ;Any keyboard character will stop display
            BEQ     NO_WAIT
            BSR     GETCHAR
            LEA     CONTINUE_MSG,A2
            BSR     PRINT_STRING
            BSR     GETCHAR
            BSR     TOUPPER
            CMP.B   #ESC,D1
            BNE     NO_WAIT
            BSR     CRLF
            BRA     IDE_LOOP                ;Back to IDE Menu

NO_WAIT:
            BSR     GET_NEXT_SECT           ;Point LBA to next sector
            BEQ     MORE_SEC                ;Note will go to last sec on disk unless stopped
            BSR     CRLF
            BRA     IDE_LOOP                ;Back to IDE Menu




;--------------- Read N Sectors to disk ----------------------------------------------------
;Note unlike the normal sector read, this routine increments the DMA address after each sector rea

N_RD_SEC:
            LEA     WILL_RD_MSG,A2              ;Enter RAM location where sector data will be plac
            BSR     PRINT_STRING

            BSR     GETLONG_D7                  ;Get start address
            CMP.B   #CR,D2                      ;Is it valid
            BNE     IDE_LOOP
            MOVE.L  D7,(RAM_DMA_STORE)          ;<--- Save in RAM_DMA_STORE

            LEA     SEC_COUNT_MSG,A2            ;Enter sector count
            BSR     PRINT_STRING
            BSR     GETBYTE_D7                  ;Get data in D7 (0-FF)
            CMP.B   #ESC,D2                     ;If ESC then we are done
            BEQ     IDE_LOOP
            CMP.B   #CR,D2                      ;If CR then we are done
            BNE     IDE_LOOP
            MOVE.W  D7,(SECTOR_COUNT)           ;store sector count
            BSR     CRLF
            BSR     CRLF

NextRSec:
            BSR     DISPLAY_POSITION            ;Display current Track,sector
            LEA     READN_MSG,A2                ;' ----> ',0
            BSR     PRINT_STRING

            MOVE.L  (RAM_DMA_STORE),D1          ;DMA initially to IDE_Buffer
            MOVE.L  D1,(RAM_DMA)
            MOVE.L  D1,D7
            BSR     PUTLONG_D7                  ;Show current address
```

```
            LEA     WRITEN_MSG,A2               ;' ----> ',0
            BSR     PRINT_STRING
            BSR     DISPLAY_POSITION            ;Display current Track,sector


            BSR     WRITESECTOR                 ;Sector/track values are sent to board in WRITESEC

            MOVE.L  (RAM_DMA),D1
            ADD.L   #$200,D1
            MOVE.L  D1,(RAM_DMA_STORE)

            SUBQ.W  #1,(SECTOR_COUNT)
            BNE     NEXT_SEC_NWR
            BRA     DoneWSec

NEXT_SEC_NWR:
            BSR     GET_NEXT_SECT
            BEQ     NextWSec

            LEA     AT_END_MSG,A2               ;Tell us we are at end of disk
            BSR     PRINT_STRING
DoneWSec:
            BSR     CRLF
            MOVE.B  #0,(RAM_SEC)                ;Back to CPM sector 0
            MOVE.B  #0,(RAM_TRK)
            MOVE.B  #0,(RAM_TRK+1)
            BSR     WR_LBA                      ;Update LBA on drive
            BSR     CRLF
            BRA     IDE_LOOP                    ;Back to IDE Menu



;-------------- Format current disk-------------------------------------------------------

FORMAT:     CMP.B   #0,(CURRENT_IDE_DRIVE)
            BNE     FORM_B
            LEA     FORMAT_MSG_A,A2
            BRA     FORM_X
FORM_B:     LEA     FORMAT_MSG_B,A2
FORM_X:
            BSR     PRINT_STRING
            LEA     CONFIRM_WR_MSG,A2       ;Are you sure?
            BSR     PRINT_STRING
            BSR     GETCHAR
            BSR     TOUPPER
            CMP.B   #'Y',D1
            BEQ     FORMAT_BUFFER
            BSR     CRLF
            BRA     IDE_LOOP                    ;Back to IDE Menu

FORMAT_BUFFER:
            LEA     FORMAT_STARTED_MSG,A2   ;The current drive is being formatted. Esc to aborot
            BSR     PRINT_STRING
            BSR     CRLF
            MOVE.B  #0,(RAM_SEC)                ;Back to CPM sec_TRK)
            MOVE.B  #0,(RAM_TRK+1)

            BSR     WR_LBA                          ;Update LBA on drive

            MOVE.W  #$0E5E5,D5                  ;First set Sector pattern to E5's
            LEA     IDE_Buffer,A2
            MOVE.W  #512,D2                         ;512 bytes total to fill
CLEARF      MOVE.B  D5,(A2)+
            SUBQ.W  #1,D2
            BNE     CLEARF

FORMAT_LOOP:
            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)                ;DMA initially to IDE_Buffer

            BSR     WRITESECTOR                 ;Will write whatever is in the IDE_Buffer
```

```
            MOVE.L  A4,(RAM_DMA)                ;DMA initially to IDE_Buffer

            BSR     READSECTOR                  ;Get sector data from A: drive to buffer

            MOVE.B  #1,D1                           ;Login drive B:
            BSR     SELECT_DRIVE

            BSR     WR_LBA                          ;Update LBA on "B:" drive

            LEA     IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)

            BSR     WRITESECTOR                 ;Write buffer data to sector on B: drive
            BEQ     COPY_OK1

            LEA     COPY_ERR,A2                 ;Indicate an error
            BSR     PRINT_STRING
            BSR     SHOW_TRACK_SEC              ;Show current location of error
            BSR     CRLF
            BRA     COPY_OK3

COPY_OK1:
            CMP.B   #0,(RAM_SEC)                ;Get Current Sector
            BNE     COPY_OK2

            BSR     SHOW_TRACK

COPY_OK2:
            BSR     GETSTAT                         ;Any keyboard character will stop display
            BEQ     C_NEXTSEC1
            BSR     GETCHAR                         ;Flush character
COPY_OK3:
            LEA     CONTINUE_MSG,A2
            BSR     PRINT_STRING
            BSR     GETCHAR
            CMP.B   #ESC,D1
            BNE     C_NEXTSEC1
C_DONE:     MOVE.B  #0,D1                           ;Login drive A:
            BSR     SELECT_DRIVE
            MOVE.B  D1,(CURRENT_IDE_DRIVE)
            MOVE.B  #0,(RAM_SEC)                ;Start with CPM sector 0
            MOVE.B  #0,(RAM_TRK)                ;Start with CPM Track 0
            MOVE.B  #0,(RAM_TRK+1)
            BSR     WR_LBA                          ;Update LBA on drive
            BSR     CRLF
            BRA     IDE_LOOP                    ;Back to IDE Menu

C_NEXTSEC1:
            BSR     GET_NEXT_SECT               ;Update to next sector/track
            BNE     C_NEXTSEC2
            BRA     NextDCopy

C_NEXTSEC2:
            LEA     CopyDone,A2                 ;Tell us we are all done.
            BSR     PRINT_STRING
            BRA     C_DONE



;-------------- Verify Drive A: = B: ------------------------------

VERIFY_AB:
            LEA     DiskVerifyMsg,A2
            BSR     PRINT_STRING

            MOVE.B  #0,(RAM_SEC)                ;Start with CPM sector 0
            MOVE.B  #0,(RAM_TRK)                ;Start with CPM Track 0
            MOVE.B  #0,(RAM_TRK+1)
            BSR     CRLF
            BSR     CRLF
```

```
            MOVE.B  #0,(RAM_SEC)                ;Start with CPM sector 0
            MOVE.B  #0,(RAM_TRK)                ;Start with CPM Track 0
            MOVE.B  #0,(RAM_TRK+1)
            BSR     WR_LBA                          ;Update LBA on drive
            BSR     CRLF
            BRA     IDE_LOOP                    ;Back to IDE Menu



;-------------------- Back to parent 68K Monitor commands
QUIT_IDE:
            BRA     LOOP                    ;Back to main Menu


;==================== Support Routines FOR IDE MODULE ================================

;Generate an LBA sector number with data input from CPM style Track# & Sector#

GEN_HEX32_LBA:
            LEA     ENTERRAM_SECL,A2        ;Enter sector number, low
            BSR     PRINT_STRING
            BSR     GETBYTE_D7              ;Get 8 bit value (2 digits) to D7
            MOVE.B  D7,(RAM_SEC)
            BSR     CRLF

            LEA     ENTERRAM_TRKL,A2        ;Enter low byte track number
            BSR     PRINT_STRING
            BSR     GETBYTE_D7              ;Get 8 bit value (2 digits) to D7
            MOVE.B  D7,RAM_TRK
            BSR     CRLF

            LEA     ENTERRAM_TRKH,A2        ;Enter high byte track number
            BSR     PRINT_STRING
            BSR     GETBYTE_D7              ;Get 8 bit value (2 digits) to D7
            MOVE.B  D7,(RAM_TRK+1)
            CLR.B   D1                      ;To return NC
            RTS


DISPLAY_POSITION:                           ;Display current track,sector & head position
            LEA     msgCPMTRK,A2            ;Display in LBA format
            BSR     PRINT_STRING            ;---- CPM FORMAT ----
            MOVE.B  (RAM_TRK+1),D6
            BSR     PUTBYTE_D6              ;High TRK byte
            MOVE.B  (RAM_TRK),D6
            BSR     PUTBYTE_D6              ;Low TRK byte

            LEA     msgCPMSEC,A2
            BSR     PRINT_STRING            ;SEC = (16 bits)
;           MOVE.B  (RAM_SEC+1),D6          ;High Sec
;           BSR     PUTBYTE_D6
            MOVE.B  (RAM_SEC),D6            ;Low Sec
            BSR     PUTBYTE_D6

                                            ;---- LBA FORMAT ----
            LEA     msgLBA,A2
            BSR     PRINT_STRING            ;(LBA = 00 (<-- Old "Heads" = 0 for these drives).

            MOVE.B  (DISPLAY_TRK+1),D6      ;High "cylinder" byte
            BSR     PUTBYTE_D6
            MOVE.B  (DISPLAY_TRK),D6        ;Low "cylinder" byte
            BSR     PUTBYTE_D6

            MOVE.B  (DISPLAY_SEC),D6
            BSR     PUTBYTE_D6
            LEA     MSGBracket,A2           ;)$
            BSR     PRINT_STRING
            RTS
```

```
Sloop4:     MOVE.B  D6,D1
            BSR     PUTCHAR
            SUBQ.B  #1,D4
            BNE     Sloop2
            BSR     CRLF

            SUBQ.B  #1,D3
            BNE     SF172
            RTS




                                        ;Point to next sector.  Ret Z if all OK,      NZ if at
GET_NEXT_SECT:
            ADDQ.B  #1,(RAM_SEC)        ;Inc Current Sector
            CMP.B   #MAXSEC-1,(RAM_SEC) ;Assumes < 255 sec /track
            BNE     NEXT_SEC_DONE

            MOVE.B  #0,(RAM_SEC)        ;Back to CPM sector 0

            ADDQ.B  #1,(RAM_TRK)        ;Bump to next track
            MOVE.B  #0,(RAM_TRK+1)
            CMP.B   #0,(RAM_TRK)        ;Tracks 0-0FFH only
            BEQ     AT_DISK_END
NEXT_SEC_DONE:
            BSR     WR_LBA              ;Update the LBC pointer
            EOR.B   D1,D1
            RTS                         ;Ret Z if all OK
AT_DISK_END:
            BSR     WR_LBA              ;Update the LBC pointer
            EOR.B   D1,D1
            SUBQ.B  #1,D1
            RTS                         ;;Ret NZ if end of disk



                                        ;Point to previous sector.  Ret Z if all OK
GET_PREV_SECT:
            CMP.B   #0,(RAM_SEC)        ;Get Current Sector
            BEQ     PREVIOUS_TRACK
            SUBQ.B  #1,(RAM_SEC)        ;0 to MAXSEC CPM Sectors
            BRA     PREVIOUS_SEC_DONE

PREVIOUS_TRACK:
            MOVE.B  #MAXSEC-1,(RAM_SEC) ;Back to CPM last sector on previous track

            CMP.B   #0,(RAM_TRK)        ;If On track 0 already then problem
            BEQ     AT_00
            SUBQ.B  #1,(RAM_TRK)
            MOVE.B  #0,(RAM_TRK+1)
PREVIOUS_SEC_DONE:
            BSR     WR_LBA              ;Update the LBC pointer
            EOR.B   D1,D1
            RTS                         ;Ret z if all OK
AT_00:      BSR     WR_LBA              ;Update the LBC pointer
            LEA     ATHOME_MSG,A2
            BSR     PRINT_STRING
            EOR.B   D1,D1
            SUBQ.B  #1,D1
            RTS

;
SHOWerrors:
            BSR     CRLF
            MOVE.B  #REGstatus,D5       ;Get status in status register
            BSR     IDErd8D
            MOVE.B  D4,D6
            BTST    #0,D4               ;Error bit
            BNE     MoreError           ;Go to  REGerr register for more info
                                        ;All OK if 01000000
```

```
;================================================================================================
; IDE Drive BIOS Routines written in a format that can be used with CPM68K throughout we
; will use IDE_BUFFER so the the buffers can reside at the top segment of available RAM.
; Normally this will be FD8100H  (Above the ROM).
;================================================================================================

IDEinit:                                            ;Initilze the 8255 and drive then do a hard re
                                                    ;By default the drive will come up initilized

            MOVE.B  #READcfg8255,IDECtrlPort    ;Config 8255 chip, READ mode

            MOVE.B  #0,IDEportC                 ;No IDE control lines asserted
            MOVE.W  #$20,D1                     ;time delay for reset/initilization
InitDelay:  SUBQ.W  #1,D1
            BNE     InitDelay                   ;Delay

            MOVE.B  #IDErstline,IDEportC        ;Hard reset the disk drive

            MOVE.W  #IDE_Reset_Delay,D1         ;Time delay for reset/initilization (~66 uS, with
ResetDelay:
            SUBQ.W  #1,D1
            BNE     ResetDelay                  ;Delay (IDE reset pulse width)

            MOVE.B  #0,IDEportC                 ;No IDE control lines asserted

            BSR     DELAY_32                    ;Allow time for CF/Drive to recover

            MOVE.B  #%11100000,D4               ;Data for IDE SDH reg (512bytes, LBA mode,single d
;           MOVE.B  #%10100000,D4               ;For Trk,Sec,head (non LBA) use 10100000 (This is
                                                ;Note. Cannot get LBA mode to work with an old Sea
                                                ;have to use the non-LBA mode. (Common for old har
            MOVE.B  #REGshd,D5                  ;00001110,(0EH) for CS0,A2,A1,
            BSR     IDEwr8D                     ;Write byte to select the MASTER device

            MOVE.B  #$FF,D6                     ;<<< May need to adjust delay time
WaitInit:
            MOVE.B  #REGstatus,D5               ;Get status after initilization
            BSR     IDErd8D                     ;Check Status (info in [DH])
            MOVE.B  D4,D1
            AND.B   #$80,D1
            BEQ     DoneInit                    ;Return if ready bit is zero

            MOVE.L  #$0FFFF,D7
DELAY2:     MOVE.B  #2,D5                       ;May need to adjust delay time to allow cold drive
DELAY1:     SUBQ.B  #1,D5                       ;to speed
            BNE     DELAY1
            SUBQ.B  #1,D7
            BNE     DELAY2

            SUBQ.B  #1,D6
            BNE     WaitInit
            BSR     SHOWerrors                  ;Ret with NZ flag set if error (probably no drive)
            RTS
DoneInit:
            EOR     D1,D1
            RTS

DELAY_32:
            MOVE.B  #40,D1                      ;DELAY ~32 MS (DOES NOT SEEM TO BE CRITICAL)
DELAY3:     MOVE.B  #0,D2
M0:         SUBQ.B  #1,D2
            BNE     M0
            SUBQ.B  #1,D1
            BNE     DELAY3
            RTS
                                                ;Read a sector, specified by the 4 bytes in LBA
                                                ;Z on success, NZ BSR error routine if problem
                                                ;Tell which sector we want to read from.
READSECTOR:                                     ;Note: Translate first in case of an error otherew
```

```
            MOVE.B  (A2)+,IDEportB

            MOVE.B  #REGdata,IDEportC
            OR.B    #IDEwrline,IDEportC         ;Send WR pulse
            MOVE.B  #REGdata,IDEportC
            SUBQ.B  #$1,D6
            BNE     WRSEC1_IDE

            MOVE.B  #READcfg8255,IDECtrlPort  ;Set 8255 back to read mode

            MOVE.B  #REGstatus,D5
            BSR     IDErd8D
            MOVE.B  D4,D1
            AND.B   #$1,D1
            BEQ     L_21
            BSR     SHOWerrors                  ;If error display status
L_24:       RTS


                                                ;Write the logical block address to the drive's re
                                                ;Note we do not need to set the upper nibble of th
                                                ;It will always be 0 for these small CPM drives (s
WR_LBA:                                         ;numbers etc).
            MOVE.B  (RAM_SEC),D4                ;LBA mode, Low sectors go directly
;           ADDQ.B  #$1,D4                      ;Sectors are numbered 1 -- MAXSEC (even in LBA mod
            MOVE.B  D4,(DISPLAY_SEC)            ;For Diagnostic Diaplay Only
            MOVE.B  #REGsector,D5               ;Send info to drive
            BSR     IDEwr8D                     ;Write to 8255 A Register
                                                ;Note: For drive we will have 0 - MAXSEC sectors o

            MOVE.B  (RAM_TRK),D4
            MOVE.B  D4,(DISPLAY_TRK)            ;Send Low TRK#
            MOVE.B  #REGcylinderLSB,D5
            BSR     IDEwr8D                     ;Write to 8255 A Register

            MOVE.B  (RAM_TRK+1),D4
            MOVE.B  D4,(DISPLAY_TRK+1)
            MOVE.B  #REGcylinderMSB,D5          ;Send High TRK#
            BSR     IDEwr8D                     ;Send High TRK# (in DH) to IDE Drive
            BSR     IDEwr8D_X                   ;Special write to 8255 B Register (Not A) to updat
                                                ;High 8 bits ignored by IDE drive

            MOVE.B  #$1,D4                      ;For CPM, one sector at a time
            MOVE.B  #REGseccnt,D5
            BSR     IDEwr8D                     ;Write to 8255 A Register
            RTS

                                                ;Special version for MS-DOS system BIOS (see IBM B
                                                ;This will display Head, Cylinder and Sector on th
DOS_WR_LBA:                                     ;instead of LBA sector numbers.
            MOVE.B  CURRENT_HEAD,D4             ;OR in head info to lower 4 bits
            AND.B   #$0F,D4                     ;Just in case
            OR.B    #%10100000,D4               ;Set to >>>>> NON-LBA mode <<<<<
            MOVE.B  #REGshd,D5                  ;Send "Head #" (in DH) to IDE drive
            BSR     IDEwr8D

            MOVE.B  CURRENT_TRACK_HIGH,D4   ;Send High TRK#
            MOVE.B  #REGcylinderMSB,D5
            BSR     IDEwr8D                     ;Send High TRK# (in DH) to IDE Drive

            MOVE.B  CURRENT_HEAD,D4             ;Get head info to lower 8 bits of the special
            AND.B   #$0F,D4                     ;top two LED HEX displays.
            LSL     #4,D4                       ;These 8 (high) data lines are ignored by the IDE
            OR.B    CURRENT_TRACK_HIGH,D4       ;Will display  the Head in top nibble and the two
            MOVE.B  #REGcylinderMSB,D5          ;of the high cylinder in the low nibble.
            BSR     IDEwr8D_X                   ;Special output to 8255 B Register (Not A) to upda

            MOVE.B  CURRENT_TRACK,D4            ;Get low Track #
            MOVE.B  #REGcylinderLSB,D5          ;Send Low TRK# (in DH)
            BSR     IDEwr8D                     ;Special write to 8255 B Register (Not A)
```

```
;-----------------------------------------------------------------
; Low Level 8 bit R/W to the drive controller.  These are the routines that talk
; directly to the drive controller registers, via the 8255 chip.
; Note the 16 bit Sector I/O to the drive is done directly
; in the routines READSECTOR & WRITESECTOR for speed reasons.

IDErd8D:                                        ;READ 8 bits from IDE register @ [DL], return info
            MOVE.B  D5,IDEportC                 ;Select IDE register, drive address onto control l

            OR.B    #IDErdline,IDEportC         ;RD pulse pin (40H), Assert read pin

            MOVE.B  IDEportA,D4                 ;Return with data in [D4]

            MOVE.B  D5,IDEportC                 ;Select IDE register, drive address onto control l

            MOVE.B  #0,IDEportC                 ;Zero all port C lines
            RTS


IDEwr8D:                                            ;WRITE Data in [DH] to IDE register @ [DL]
            MOVE.B  #WRITEcfg8255,IDECtrlPort       ;Set 8255 to write mode

            MOVE.B  D4,IDEportA                     ;Get data put it in 8255 A port

            MOVE.B  D5,IDEportC                     ;Select IDE register, drive address onto contr

            OR.B    #IDEwrline,IDEportC             ;Assert write pin

            MOVE.B  D5,IDEportC                     ;Select IDE register, drive address onto contr

            MOVE.B  #0,IDEportC                     ;Zero all port C lines

            MOVE.B  #READcfg8255,IDECtrlPort        ;Config 8255 chip, read mode on return
            RTS


IDEwr8D_X:                                          ;WRITE Data in [DH] to IDE register @ [DL]
            MOVE.B  #WRITEcfg8255,IDECtrlPort       ;Set 8255 to write mode

            MOVE.B  D4,IDEportB                     ;Get data and put it in 8255 >>>> Port B <<<<

            MOVE.B  D5,IDEportC                     ;Select IDE register, drive address onto contr

            OR.B    #IDEwrline,IDEportC             ;Assert write pin

            MOVE.B  D5,IDEportC                     ;Select IDE register, drive address onto contr

            MOVE.B  #0,IDEportC                     ;Zero all port C lines

            MOVE.B  #READcfg8255,IDECtrlPort        ;Config 8255 chip, read mode on return
            RTS



;---------------------- Routine to download from a PC via XMODEM a .bin file --------------------

XMODEM_BIN:                                 ;Download an XModem .bin file.  Note some registers ch
            LEA     MODEM_BIN_SIGNON,A2     ;Will show message explaining we are about to upload a
            BSR     PRINT_STRING

            MOVE.B  #0,(RECVD_SECT_NO)
            MOVE.B  #0,(SECTNO)
            MOVE.B  #0,(ERRCT)

            BSR     SERIAL_INITILIZE_A      ;RESET THE ZILOG SCC

            LEA     RAM_DESTINATION_MSG,A2  ;Ask for destination
            BSR     PRINT_STRING
            BSR     GETLONG_D7              ;Get start address
            CMP.B   #CR,D2                  ;Is it valid - must end with CR
```

```
RHNTO:      CMP.B   #SOH,D1                 ;GOT CHAR - MUST BE SOH
            BEQ     GOT_SOH
            OR.B    D1,D1                   ;00 FROM SPEED CHECK?
            BNE     L_2
            BRA     RECV_HDR
L_2:        CMP.B   #EOT,D1
            BNE     L_3
            BRA     GOT_EOT
L_3:        MOVE.B  D1,D6
            BSR     PUTBYTE_D6
            LEA     ERRSOH,A2               ;'H Received',CR,LF,'Did not get Correct SOH'
            BSR     PRINT_STRING
            BRA     RECV_SECT_ERR

GOT_SOH:                                    ;We got correct SOH so now get data
            MOVE.L  #SERIAL_RETRYS,D2       ;Number of times to try reading serial port before abo
            BSR     RECV
            CMP.B   #$FF,D2                 ;Return with FF in D2 if all is OK
            BNE     RECV_HDR_TIMEOUT

            MOVE.B  D1,D5                   ;D5=BLOCK #
            MOVE.L  #SERIAL_RETRYS,D2      ;Number of times to try reading serial port before abor
            BSR     RECV
            CMP.B   #$FF,D2                 ;GET CMA'D SECT #
            BNE     RECV_HDR_TIMEOUT
            NOT.B   D1

            CMP.B   D1,D5                   ;GOOD SECTOR #?
            BEQ     RECV_SECTOR
                                            ;GOT BAD SECTOR #
            LEA     MODEM_ERR2,A2           ;'++BAD SECTOR # IN HDR'
            BSR     PRINT_STRING
            BRA     RECV_SECT_ERR

RECV_SECTOR:                                ;Now get 128 Bytes
            MOVE.B  D5,(RECVD_SECT_NO)      ;GET SECTOR #
            CLR.B   D4                      ;INIT CKSUM = 0
            MOVE.B  #$80,D3                 ;128 Byte sectors always

RECV_CHAR:
            MOVE.L  #20*SERIAL_RETRYS,D2    ;Number of times to try reading serial port before abo
            BSR     RECV                    ;GET CHAR
            CMP.B   #$FF,D2                 ;GET CMA'D SECT #
            BNE     RECV_HDR_TIMEOUT
            MOVE.B  D1,(A3)+                ;<<< STORE CHAR >>>
            ADD.B   D1,D4                   ;Add in checksum
            SUB.B   #1,D3                   ;128 Bytes done yet?
            BNE     RECV_CHAR

                                            ;NEXT VERIFY CHECKSUM
            MOVE.L  #SERIAL_RETRYS,D2       ;Number of times to try reading serial port before abo
            BSR     RECV                    ;GET CHECKSUM
            CMP.B   #$FF,D2                 ;Return with FF in D2 if all is OK
            BNE     RECV_HDR_TIMEOUT
MODL_5:
            CMP.B   D1,D4                   ;CHECK IF CHECKSUM IS CORRECT
            BNE     RECV_CKSUM_ERR
            MOVE.B  (RECVD_SECT_NO),D2      ;GOT A SECTOR, WRITE IF = 1+PREV SECTOR
            ADD.B   #1,D5                   ;CALC NEXT SECTOR #
            CMP.B   D5,D2                   ;MATCH?
            BNE     DO_ACK

            MOVE.B  D5,(SECTNO)             ;UPDATE SECTOR #
DO_ACK:     MOVE.B  #ACK,D1
            BSR     SERIAL_OUT
            BRA     RECV_LOOP


RECV_CKSUM_ERR:
            LEA     MODEM_ERR3,A2
            BSR     PRINT_STRING
```

```
            MOVE.L  (A7)+,D1
DONE_LONG:  RTS                         ;Normal return with FFh in D1 if CR or ',' was entered


GETBYTE_D7: CLR.L   D7                  ;Get a Byte number and place in D7 (1-2 bytes)
            BSR     GETLONG_D7
            AND.L   #$ff,D7             ;Return with just a Byte (D2 will normally be 2)
            RTS


GETNIBBLE:  CLR.B   D2                  ;clear D2 flag byte
            CLR.B   D1                  ;just in case
            BSR     GETCHAR             ;Get a HEX character (0,1,2,3...A,B,C,D,E,F in D1)
            CMP.B   #ESC,D1             ;Was an abort requested
            BEQ     NIBBLE1

            CMP.B   #CR,D1              ;CR terminates data entry
            BEQ     NIBBLE1
            CMP.B   #',',D1             ;',' also terminates data entry
            BEQ     NIBBLE1
            CMP.B   #BLANK,D1           ;A BLANK also terminates data entry
            BEQ     NIBBLE1

            BSR     TOUPPER             ;(D1)Lower case to Upper case

            SUB.B   #$30,D1             ;SEE IF LESS THAN ZERO
            BLT.S   NIBBLE2

            CMP.B   #$09,D1             ;SEE IF GT 9
            BLE.S   NIBBLE1
            SUBQ.B  #7,D1               ;NORMALIZE $A TO 10
            CMP.B   #$10,D1             ;SEE IF TOO LARGE
            BCC.S   NIBBLE2
            RTS                         ;Return with nibble in D1 (0,1,2,3...F)

NIBBLE1:    MOVE.B  D1,D2               ;Store ESC/CR/,/BLANK in D2
            RTS

NIBBLE2:    MOVE.B  #BELL,D1            ;Not a valid HEX character
            BSR     PUTCHAR
            MOVE.B  #'?',D1
            BSR     PUTCHAR
            MOVE.B  #ESC,D2
            RTS




PUTLONG_D7: MOVE.L  D7,D6               ;Print long in D7 on CRT, Note D6 destroyed
            SWAP    D6                  ;Swap down upper word
            BSR     PUTWORD_D6
            MOVE.L  D7,D6
            BSR     PUTWORD_D6
            RTS



PUTWORD_D6: MOVE.W  D6,D1               ;Note D1 is destroyed
            LSR.W   #8,D1               ;Shift upper byte to lower 8 bits
            LSR.W   #4,D1               ;Shift upper byte to lower 4 bits
            AND.B  #$0F,D1              ;SAVE LOWER NIBBLE
            OR.B   #$30,D1              ;CONVERT TO ASCII
            CMP.B  #$39,D1              ;SEE IF IT IS > 9
            BLE.S   HEXOK2
            ADDQ.B   #7,D1              ;ADD TO MAKE 10=>A
HEXOK2:     BSR     PUTCHAR
                                        ;Address lower high byte nibble
            MOVE.W  D6,D1               ;Origional number again to D1
            LSR.W   #8,D1               ;Shift upper byte to lower 8 bits
            AND.B  #$0F,D1              ;SAVE LOWER NIBBLE
```

```
            BSR     SERIAL_OUT              ;Call serial output routine
            MOVE.L  (A7)+,A0                ;Restore A0
            MOVE.L  (A7)+,D5                ;Restore D5
            RTS                             ;Return from subroutine, char in D1

PUTCHAR0:   MOVE.L  (A7)+,A0                ;< Restore A0
PUTCHAR1:   MOVE.B  (A0),D5                 ;Check CRT status is ready to recieve character
            AND.B   #$04,D5
            TST.B   D5
            BEQ     PUTCHAR1
            MOVE.B  D1,(A1)                 ;Output ASCII (in D1) to hardware port 01H
            MOVE.L  (A7)+,D5                ;< Restore D5
            RTS                             ;Return from subroutine
   ENDC



;---------------------------- MAIN ROUTINE TO GET A CHARACTER FROM CONSOLE -----------------------

                                            ;A0 has console status port, A1 has console data port
GETCHAR:
   IFNE SIMMULATOR                          ;If SIMMULATOR = 1, then echo character via software i
            MOVE.B  #5,D0                   ;Get a character from keyboard, put in D1 (NOTE will b
            TRAP    #15
            RTS

  ENDC

   IFEQ SIMMULATOR                          ;If SIMMULATOR = 0, then echod character via PUTCHAR
            MOVE.L  D5,-(A7)                ;> Save D5
            MOVE.L  A0,-(A7)                ;> Save A0
            MOVE.L  #IOBYTE,A0              ;Point to IOBYTE Port on SMB
            MOVE.B  (A0),D5                 ;Check if data is to be sent to the serial port
            AND.B   #$20,D5
            TST.B   D5
            BRA     GETCHAR0                ;Jump to simple Propeller port
;           BNE     GETCHAR0
            BSR     SERIAL_IN               ;Call serial input routine (currently not working!)
            MOVE.L  (A7)+,A0                ;Restore A0
            MOVE.L  (A7)+,D5                ;Restore D5
            RTS                             ;Return from subroutine, char in D1

GETCHAR0:   MOVE.L  (A7)+,A0                ;< Restore A0 (console status port)
GETCHAR1:   MOVE.B  (A0),D5                 ;Get a keyboard character in D1
            AND.B   #$02,D5
            TST.B   D5                      ;Are we ready
            BEQ     GETCHAR1
            MOVE.B  (A1),D1                 ;Get ASCII (in D1) from hardware port 01H
            BSR     PUTCHAR                 ;Echo it on console
            MOVE.L  (A7)+,D5                ;< Restore D5
            RTS                             ;Return from subroutine, char in D1
   ENDC


GETSTAT:    MOVE.B  (A0),D1                 ;Get a keyboard status in D1, Z= nothing, 2 = char pre
            AND.B   #$02,D1
            TST.B   D1
            RTS

;--------------------------------- SERIAL PORT OUTPUT CHARACTER ROUTINE --------------------------

SERIAL_OUT: MOVE.L  D5,-(A7)                ;> Save D5
            MOVE.L  D2,-(A7)                ;> Save D2
            MOVE.L  A0,-(A7)                ;> Save A0
            MOVE.L  #ACTL,A0                ;Point to Control port of Zilog serial chip
            MOVE.W  #512,D2                 ;Will check status 512 times (only)

SERIAL_OUT_STAT:
            MOVE.B  (A0),D5                 ;Check serial port is ready
            AND.B   #$04,D5
            TST.B   D5
```

```
CRLF:
            MOVE.B  #CR,D1                  ;Send CR/LF to CRT
            BSR     PUTCHAR
            MOVE.B  #LF,D1
            BSR     PUTCHAR
            RTS

PUT_TAB:    MOVE.B  #TAB,D1                 ;Send TAB to CRT
            BSR     PUTCHAR
            RTS

SPACE:      MOVE.B  #BLANK,D1               ;SPACE to CRT
            BSR     PUTCHAR
            RTS


TOUPPER:    CMP.B   #$40,D1                 ;LC->UC in D1
            BCS     UPPER_DONE
            CMP.B   #$7B,D1
            BCC     UPPER_DONE
            AND.B   #$5F,D1
UPPER_DONE: RTS


ERROR:      LEA     ErrorMsg,A2             ;Show unknown error
            BSR     PRINT_STRING
            BRA     LOOP

NOT_DONE:   LEA     NotDoneMsg,A2           ;Code not done yet
            BSR     PRINT_STRING
            BRA     LOOP

                                            ;Send character in D1 to Console IO board speaker
SPEAKOUT:   MOVE.L  A3,-(A7)                ;> Save A3
            MOVE.L  D2,-(A7)                ;> Save D2
            MOVE.L  D3,-(A7)                ;> Save D3
            MOVE.L  #255,D2                 ;Will try 255 times, then timeout
            MOVE.L  #BCTL,A3
SOUT1:      MOVE.L  (A3),D3
            AND.B   #$04,D3
            BNE     SENDS
            SUB.B   #1,D2
            BNE     SOUT1
SOUT2:      MOVE.L  (A7)+,D3                ;< Restore D3
            MOVE.L  (A7)+,D2                ;< Restore D2
            MOVE.L  (A7)+,A3                ;< Restore A3
            RTS

SENDS:      MOVE.L  #BDTA,A3
            MOVE.B  D1,(A3)                 ;Send actual character to data port
;           BSR     PUTCHAR                 ;<---- For debugging, display character ---
            MOVE.L  #SPEAKER_DELAY,D3       ;For some reason we need this delay
SENDS1:     SUB.L   #1,D3                   ;If not characters get dropped!
            TST.L   D3
            BNE     SENDS1
            MOVE.B  #5,D3                   ;Sel register 5
            MOVE.L  #BCTL,A3                ;Raise RTS line to prevent the next character arriving
            MOVE.B  D3,(A3)
            MOVE.B  #$E8,D3
            MOVE.B  D3,(A3)
            BRA     SOUT2

                                            ;ROUTINE TO SEND A STRING IN (A2) TO TALKER, terminate
SPEAK_STRING:
            MOVE.B  (A2)+,D1
            CMP.B   #0,D1
            TST.B   D1
            BEQ     SPEAK_DONE
            CMP.B   #CR,D1
```

```
                dc.l    QUERY_PORT      ;Q      ;Query In or Out to a port
                dc.l    ERR             ;R
                dc.l    SUBS_RAM        ;S      ;Substitute byte values in RAM
                dc.l    ASCII_RAM       ;T      ;Show ASCII values in RAM
                dc.l    TEST_SERIAL     ;U      ;Test serial port
                dc.l    VERIFY_RAM      ;V      ;Verify two memory regions are the same
                dc.l    ERR             ;W
                dc.l    SIGNALS         ;X      ;Setup for hardware S-100 bus signals test
                dc.l    PATCH           ;Y      ;Quick patch to move RAM 4000H-9000H to F4000H
                dc.l    JMP_Z80         ;Z      ;Return back to Z80 master


IDE_TABLE   dc.l    SET_DRIVE_A             ; "A"  Select Drive A
                dc.l    SET_DRIVE_B         ; "B"  Select Drive B
                dc.l    COPY_AB             ; "C"  Copy Drive A to Drive B
                dc.l    IDE_ERR             ; "D"
                dc.l    FILL_SEC            ; "E"  Fill a sector with a byte value
                dc.l    FORMAT              ; "F"  Format current disk
                dc.l    IDE_ERR             ; "G"
                dc.l    IDE_ERR             ; "H"
                dc.l    IDE_ERR             ; "I"
                dc.l    IDE_ERR             ; "J"
                dc.l    IDE_LOOP            ; "K"
                dc.l    SET_LBA             ; "L"  Set LBA value (Set Track,sector)
                dc.l    IDE_ERR             ; "M"
                dc.l    NEXT_SECT           ; "N"  Next Sector
                dc.l    DRIVE_ID            ; "O"  Show current Drive ID
                dc.l    PREV_SECT           ; "P"  Previous sector
                dc.l    IDE_ERR             ; "Q"
                dc.l    READ_SEC            ; "R"  Read sector to data buffer
                dc.l    SEQ_SEC_RD          ; "S"  Sequental sec read and display contents
                dc.l    IDE_ERR             ; "T"
                dc.l    IDE_ERR             ; "U"
                dc.l    VERIFY_AB           ; "V"  Verify Drive A:= Drive B:
                dc.l    WRITE_SEC           ; "W"  Write data buffer to current sector
                dc.l    N_RD_SEC            ; "X"  Read N sectors to RAM
                dc.l    N_WR_SEC            ; "Y"  Write N sectors
                dc.l    IDE_ERR             ; "Z"


                                            ;BOTH CONSOLE IO BOARD's SSC's are set for 19,200
SCCINIT_A:      dc.b    $04                 ;Point to WR4
                dc.b    $44                 ;X16 clock,1 Stop,NP
                dc.b    $03                 ;Point to WR3
                dc.b    $C1                 ;Enable reciever, Auto Enable, Recieve 8 bits
;               dc.b    $E1                 ;Enable reciever, No Auto Enable, Recieve 8 bits (
                dc.b    $05                 ;Point to WR5
                dc.b    $EA                 ;Enable, Transmit 8 bits
                dc.b    $0B                 ;Set RTS,DTR, Enable. Point to WR11
                dc.b    $56                 ;Recieve/transmit clock = BRG
                dc.b    $0C                 ;Point to WR12
;               dc.b    $40                 ;Low Byte 2400 Baud
;               dc.b    $1E                 ;Low Byte 4800 Baud
;               dc.b    $0E                 ;Low Byte 9600 Baud
;               dc.b    $06                 ;Low byte 19,200 Baud
                dc.b    $02                 ;Low byte 38,400 Baud <<<<<<<<<<< XModem I/O
;               dc.b    $00                 ;Low byte 76,800 Baud
                dc.b    $0D                 ;Point to WR13
                dc.b    $00                 ;High byte for Baud
                dc.b    $0E                 ;Point to WR14
                dc.b    $01                 ;Use 4.9152 MHz Clock. Note SD Systems uses a 2.45
                dc.b    $0F                 ;Point to WR15
                dc.b    $00                 ;Generate Int with CTS going high

SCCINIT_B:
                dc.b    $04                 ;Point to WR4
                dc.b    $44                 ;X16 clock,1 Stop,NP
                dc.b    $03                 ;Point to WR3
                dc.b    $C1                 ;Enable reciever, Auto Enable, Recieve 8 bits
                dc.b    $05                 ;Point to WR5
                dc.b    $EA                 ;Enable, Transmit 8 bits
```

```
SERIAL_TEST_DONE_MSG dc.b   CR,LF,'Serial test done.',CR,LF,0

IDE_SIGNON0         dc.b   CR,LF,LF,'IDE HDisk Test Menu Routines.    ',0
IDE_SIGNON1         dc.b   'A=Select Drive A   B=Select Drive B  E=Fill Sec        F=Format Disk',
                    dc.b   'N=Next Sec         P=Previous Sec    L=Set LBA Value   O=Disk ID',CR,L
                    dc.b   'R=Read Sector      S=Seq Sec Rd      X=Sectors to RAM  W=Write Sector.
                    dc.b   'Y=RAM to Sectors   C=Copy A->B       V=Verify A=B      (ESC) Main Menu
                    dc.b   CR,LF,'Current settings:- ',0

IDE_MENU_CMD        dc.b   'Enter a Command:- ',0
IDE_HARDWARE        dc.b   CR,LF,'Initilizing IDE Drive hardware.',0
INIT_1_ERROR        dc.b   CR,LF,'Init of First Drive failed.',BELL,CR,LF,LF,0
INIT_2_ERROR        dc.b   CR,LF,'Init of Second Drive failed. (Possibly not present).',BELL,CR,LF

BAD_DRIVE:          dc.b   CR,LF,'First Drive ID Info appears invalid. '
                    dc.b   '(Drive possibly not present).',CR,LF
                    dc.b   'Aborting Command.',BELL,CR,LF,LF,0
BadIDECmdMsg        dc.b   CR,LF,BELL,'Bad Command!',0

NotDoneYet          dc.b   CR,LF,'CMD Not Done Yet',0
CONFIRM_WR_MSG      dc.b   CR,LF,LF,BELL,'Will erase data on the current drive, '
                    dc.b   'are you sure? (Y/N)...',0
msgrd               dc.b   'Sector Read OK',CR,LF,0
msgwr               dc.b   'Sector Write OK',CR,LF,0
SET_LBA_MSG         dc.b   'Enter CPM style TRK & SEC values (in hex).',CR,LF,0
ENTERRAM_SECL       dc.b   'Starting sector number,(xxH) = ',0
ENTERRAM_TRKL       dc.b   'Track number (LOW byte, xxH) = ',0
ENTERRAM_TRKH       dc.b   'Track number (HIGH byte, xxH) = ',0
DRIVE_BUSY          dc.b   'Drive Busy (bit 7) stuck high.   Status = ',0
DRIVE_NOT_READY     dc.b   'Drive Ready (bit 6) stuck low.  Status = ',0
DRIVE_WR_FAULT      dc.b   'Drive write fault.    Status = ',0
UNKNOWN_ERROR       dc.b   'Unknown error in status register.   Status = ',0
BAD_BLOCK           dc.b   'Bad Sector ID.    Error Register = ',0
UNRECOVER_ERR       dc.b   'Uncorrectable data error.  Error Register = ',0
READ_ID_ERROR       dc.b   'Error setting up to read Drive ID',CR,LF,0
SEC_NOT_FOUND       dc.b   'Sector not found. Error Register = ',0
INVALID_CMD         dc.b   'Invalid Command. Error Register = ',0
TRK0_ERR            dc.b   'Track Zero not found. Error Register = ',0
UNKNOWN_ERROR1      dc.b   'Unknown Error. Error Register = ',0
CONTINUE_MSG        dc.b   CR,LF,'To Abort enter ESC. Any other key to continue. ',0
FORMAT_MSG_A        dc.b   CR,LF,'Format Disk A: with E5Hs',0
FORMAT_MSG_B        dc.b   CR,LF,'Format Disk B: with E5Hs',0
ATHOME_MSG          dc.b   CR,LF,BELL,'Already on Track 0, Sector 0',0
AT_START_MSG        dc.b   CR,LF,BELL,'Already at start of disk!',0
AT_END_MSG          dc.b   CR,LF,BELL,'At end of Disk!',0
READN_MSG           dc.b   ' ----> ',0
WRITEN_MSG          dc.b   'H ----> ',0
DiskCopyMsg         dc.b   CR,LF,'Copy Drive A to Drive B (Y/N)? ',0
DiskVerifyMsg       dc.b   CR,LF,'Will verify Drive A = Drive B.',0
CopyDone            dc.b   CR,LF,'Disk Copy Done.',0
VERIFY_ERR          dc.b   CR,LF,BELL,'Verify Error at:- ',0
VerifyDone          dc.b   CR,LF,'Disk Verify Done.',0
CR_To_Continue      dc.b   CR,LF,'Hit any key to continue.',0
OK_CR_MSG           dc.b   ' OK',CR,LF,0
COPY_ERR            dc.b   CR,LF,BELL,'Sector Copy Error.',0
CURRENT_MSG_A       dc.b   'Current Drive = A:',0
CURRENT_MSG_B       dc.b   'Current Drive = B:',0
FORMAT_ERR          dc.b   CR,LF,BELL,'Sector Format Error',0
ERR_MSG             dc.b   CR,LF,BELL,'Invalid Command (or code not yet done)',CR,LF,0
DRIVE1_MSG          dc.b   '  on Drive A',CR,LF,0
DRIVE2_MSG          dc.b   '  on Drive B',CR,LF,0
IDE_SEL_A           dc.b   CR,LF,'Selecting IDE Drive A',0
IDE_SEL_B           dc.b   CR,LF,'Selecting IDA Drive B',0
FORMAT_STARTED_MSG  dc.b   CR,LF,'The current drive is being formatted. Use Esc to abort.',0
FILL_BYTE_MSG       dc.b   CR,LF,'Enter Fill byte (+CR):-',0

MODEM_BIN_SIGNON    dc.b   CR,LF,'Load a .bin File from a PC to RAM using Serial Board',CR,LF
                    dc.b   'Zilog SCC Ports A1H & A3H, 38,400 Baud.',CR,LF,0
RAM_DESTINATION_MSG dc.b   CR,LF,'Enter destination in RAM for data (up to 8 digits): ',0
DOWNLOAD_MSG        dc.b   'Downloading file Started.',0
RMSG                dc.b   CR,LF,'WAITING FOR SECTOR #',0
```

```
IDE_BUFFER          ds.b     512            ;Buffer area for sector data
IDE_BUFFER2         ds.b     512

RAM_DMA:            dc.w     0              ;Storage or DMA address
RAM_DMA_STORE       dc.l     0
SECTOR_COUNT        dc.w     0
DISPLAY_TRK         dc.w     0
DISPLAY_SEC         dc.w     0

RAM_SEC:            dc.b     0
RAM_TRK:            dc.b     0
CURRENT_IDE_DRIVE   dc.b     0
CURRENT_HEAD        dc.b     0
CURRENT_TRACK_HIGH dc.b      0
CURRENT_TRACK       dc.b     0
CURRENT_SECTOR      dc.b     0
SECTORS_TO_DO       dc.b     0

RECVD_SECT_NO       dc.b     0               ;For XMODEM
SECTNO              dc.b     0               ;   "
ERRCT               dc.b     0               ;   "
S_FILE_ADDRESS      dc.l     0               ;Start location in RAM of S file



EndRAM:             dc.b     0              ;End of 0 cleared RAM area

   IFEQ SIMMULATOR                         ;If SIMMULATOR = 0
            END     $00FDFFFE
   ENDC

   IFNE SIMMULATOR                         ;If SIMMULATOR = 1
            END     $0000
   ENDC
```