```
*-----------------------------------------------------------
* Program Number: 68K Monitor for S100Computers.com / N8VEM S-100 board
* 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 analysis
;           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 commands
etc
;           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 switch is now
00EEH
;           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 serial port
;           V2.6    06/12/2014  ;Corrected QO/QI port bug


;   >>>>>> Please note the IDE Board Diagnostic section is not complete yet. Do NOT use <<<<<<<


;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"
;
;

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 will be via
software interrupts)
S100_TEST       EQU     0   ;Set to 1 if using S100 RAM (We will not use INT's for console I/O)
ROM_CODE        EQU     1   ;Set to 1 for ROM code (We will not use INT's for console I/O)


;Propeller Console IO S-100 board or SD SYSTEMS VIDIO BOARD FOR CONSOLE I/O(<---These must
configured for your hardware)
```

```
KEYSTAT         EQU     $00FF0000
KEYIN           EQU     $00FF0001   ;Console input port. Normally the Propeller Driven S-100
Console-IO Board
KEYOUT          EQU     $00FF0001   ;Console output port. Normally the Propeller Driven S-100
Console-IO Board



;-------------- 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
PRINTER_STROBE  EQU     $00FF0004   ;OUT
DIAG_LEDS       EQU     $00FF0005   ;OUT (Will use this port initially for diagnostic LED
display)


;-------------- S100Computers Serial I/O BOARD PORT ASSIGNMENTS  (A0-AC)

BCTL            EQU     $00FF00A0   ;CHANNEL B CONTROL PORT ASSIGNMENTS OF THE ZILOG SCC CHIP
ACTL            EQU     $00FF00A1   ;CHANNEL A CONTROL (Base Port)
BDTA            EQU     $00FF00A2   ;CHANNEL B DATA
ADTA            EQU     $00FF00A3   ;CHANNEL A DATA
MODEM_ERR_LIMIT EQU     8   ;Max number of Modem serial port re-reads aborting (See XMODEM_BIN)
SERIAL_RETRYS   EQU     8000    ;Default number of times to try reading serial port before
aborting. Not critical.
SPEAKER_DELAY   EQU     1000    ;Slight delay for speak sysnthesizer
;
PortA_8255      EQU     $00FF00A8   ;A port of 8255 ;<--- Adjust as necessary
PortB_8255      EQU     $00FF00A9   ;B port of 8255
PortC_8255      EQU     $00FF00AA   ;C Port of 8255
PortCtrl_8255   EQU     $00FF00AB   ;8255 configuration port

AinBout8255cfg  EQU     %10011000   ;Set 8255 ports:- A input, B output,
                    ;C(bits 0-3) output, (bits 4-7)input

USB_DATA        EQU     $00FF00AC   ;PORT ASSIGNEMENT FOR DLP-USB Controller chip
USB_STATUS      EQU     $00FF00AA   ;Status port for USB port (Port C of 8255, bits 6,7)

USB_RXE         EQU     $80 ;If Bit 7 = 0, data available to recieve by S-100 Computer
USB_TXE         EQU     $40 ;If Bit 6 = 0 data CAN be written for transmission to PC

IOBYTE          EQU     $00FF00EF   ;IOBYTE Port on S100Computers SMB Board.

; BIT MAP OF PORT 0EFH:- X X X X  X X X X      (11111111=NORMAL CONFIG)
;                       | | | |  | | | |..For Z80 Monitor,     0=CONSOLE DATA TO PRINTER ALSO
;                       | | | |  | | |....For 8086 Monitor,    0=Force MSDOS Consol output to
CGA/VGA Board instead of Propeller board
;                       | | | |  | |......For 8086 Monitor,    0=Do not initilize extra ROMS
;                       | | | |  |........For 8086 Monitor,    0=Debug data for all MSDOS 10H &
15H INT's to Serial Port and external display
;                       | | | |...........Unused
;                       | | |-------------For Monitor,         0=Consol Output,(not input), via
ACIA Serial port on S100Computers Serial-IO Board
;                       | |...............For CPM3,            0=Force reformat of Memory disk
upon any CPM3 cold re-boot
;                                         |
;                       |.................For CPM3,            0=Write protect Memory disk for
CPM3
;                                       For 8086 Monitor    0=Prevent doing a JMPF to 500H
after 8086 reset (to CPM86 boot)
;                                       Normally a test is made to see if the CPM86 Boot is
already in RAM at 500H
```

```
;                                       If it is, a 8086 reset will bypass the monitor and go
directly there.
;                                       (see Init: in 8086 Monitor)
;
;               Note if IOBYTE = 00xxxxxx,                  This will force the 8086 hardware
diagnostic test after reset.
;                                                               (See code at FFFF0H in this
8086 monitor)



;------------ S100Computers MSDOS BOARD PORT ASSIGNMENTS

NS_EOI          equ     $20 ;Non specific end of interrupt command
MASTER_PIC_PORT equ     $00FF0020   ;Hardware port the 8259A is assigned (two ports 20H & 21H)

MasterICW1      equ     %00010111   ;EDGE triggered, 4 bytes, single Master,ICW4 needed
MasterICW2      equ     $8  ;Base address for 8259A Int Table (IBM-PC uses 8X4 = 20H)
MasterICW3      equ     $0  ;No slave
MasterICW4      equ     %00000011   ;No special mode, non buffer, Auto EOI, 8086.  ;<<<<,

CMOS_PORT       EQU     $00FF0070   ;Base Port for CMOS Chip
CMOS_VALID      EQU     $00FF000D   ;To check DS12887 CMOS chip is present and OK (Note AT-BIOS
uses 0EH)
CMOS_REGA       EQU     $00FF000A   ;CMOS REGISTER A

TIMER           EQU     $00FF0040   ;Base port of 8254
TIM_CTL         EQU     $00FF0043
COUNTS_SEC      EQU     $18
COUNTS_MIN      EQU     1092
COUNTS_HOUR     EQU     $07 ;Seems this value is used with AT/CMOS chip (was 65543 on PC)

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 & S100 RAM)
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/drive
```

```
IDE_Reset_Delay EQU     $20 ;Time delay for reset/initilization (~66 uS, with 8MHz 8086, 1 I/O
wait state)

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
COMMANDinit     EQU     $91
COMMANDid       EQU     $EC
COMMANDspindown     EQU     $E0
COMMANDspinup   EQU     $E1
;
; IDE Status Register:
;  bit 7: Busy  1=busy, 0=not busy
;  bit 6: Ready     1=ready for command, 0=not ready yet
;  bit 5: DF    1=fault occured on the IDE drive
;  bit 4: DSC   1=seek complete
;  bit 3: DRQ   1=data request ready, 0=not ready to xfer yet
;  bit 2: CORR  1=correctable error occured
;  bit 1: IDX   vendor specific
;  bit 0: ERR   1=error occured

MAXSEC          EQU      $3D    ;Sectors per track for CF my Memory drive, Kingston CF 8G. (CPM
format, 0-3CH)
                    ;translates to LBA format of 1 to 3D sectors, for a total of 61
sectors/track.
                    ;This CF card actully has 3F sectors/track. Will use 3D for my CPM86 system
because
                    ;my Seagate drive has 3D sectors/track. Don't want different CPM86.SYS
files around
                    ;so this program will also work with a Seagate 6531 IDE drive
```

```
DOS_MAXSEC      EQU      $3F    ;For MS-DOS BIOS Setting "Hard Disk" to Custom type (CF Card,
63 Sectors/track)
DOS_MAXHEADS    EQU      $10    ;16 head(s)
DOS_MAXCYL_L    EQU      $FF    ;Low Byte maximum cylinder (sent via INT 13H's in CH)
DOS_MAXCYL      EQU      1024   ;Max cylinders
DOS_MAXSEC_CYL  EQU      $FF    ;3FH, maximum sector number (bits 5-0)+ two Cyl High Bits
(Sectors numbered 1....x)



;--------------Other Hardware Equates ------------------------------------------------------

HOLD_STATE      EQU      $80    ;Set Keyboard flag to indicate a Pause is required
NO_HOLD_STATE   EQU      $7F    ;To clear the above flag

SW68K           EQU      $00FF00EE  ;OUTPUT TO THIS PORT 02H, SWITCHES THE 68000 CPU BACK to
the Z80 in hardware
                ;In my system using the V2 SMB.


;------------------ START ------------------------------------------------------------------
-----------



   IFNE ROM_CODE        ;If ROM_CODE = 1 (Use with ROM)
            ORG $00FD0000

            DC.L    $00FD8100   ;SSP = FD8100H
            DC.L    $00FD0020   ;PC = FD1020H

            ORG $00FD0020   ;Will start monitor here in ROM
   ENDC

   IFNE SIMMULATOR      ;If SIMMULATOR = 1 (Use for testing in EASy68K simulator)
            ORG $00000000
            BRA SKIP    ;jump up to 1000H because Int vectors will overwrite code here

            ORG $1000   ;In both EASy68K and S100 test modes will place code at 1000H
SKIP:
   ENDC

     IFNE S100_TEST     ;If SIMMULATOR = 1 (Use for testing, for S100 RAM use the ROM stack)
            ORG $00000000
            BRA SKIP    ;jump up to 1000H because Int vectors will overwrite code here

            ORG $1000   ;In both EASy68K and S100 test modes will place code at 1000H
SKIP:
   ENDC


            MOVE.L  #KEYSTAT,A0 ;Point to status Port 0H (CRT, Propeller Console IO Board)
            MOVE.L  #KEYIN,A1   ;Point to hardware Port 01H (CRT, Propeller Console IO Board)
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  (A3),D6
            CMP.L   D1,D6
            BNE NOT_RAM
            NOT.L   D1  ;Put back origional data
            MOVE.L  D1,(A3)
            MOVE.B  #'R',D1
            BRA DONE_TEST

NOT_RAM:    MOVE.L  (A3),D1 ;Is there RAM/ROM there
            CMP.L   D5,D1   ;Chances are it is empty if FFFFF's!
            BEQ EMPTY
            MOVE.B  #'p',D1
            BRA DONE_TEST
EMPTY:      MOVE.B  #'.',D1
DONE_TEST:  BSR PUTCHAR

            ADD.L   #$2000,A3   ;No matter what point to next 2K byte section

            SUBQ.L  #1,D3
            TST.L   D3  ;Have we done 32X2 characters across
            BEQ NEWLINE

            MOVE.L  A3,D4
            CMP.L   #$00FE0000,D4   ;Have we done all the RAM area
            BLE START1
            BRA LOOP    ;Back to start of menu


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

DISPLAY_RAM:        ;D Command. Display Bytes in RAM
            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   #CR,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 PARMS_OK
            MOVE.L  A3,A5   ;Else swap values
            MOVE.L  A4,A3
            MOVE    A5,A4

PARMS_OK:   ADD.L   #1,A4   ;End + 1
PARMS_OK5:  MOVE.L  A3,-(A7)    ;Save A3
            MOVE.L  A4,-(A7)    ;Save A4

PARMS_OK2:  BSR CRLF    ;New line
            MOVE.L  A3,D7
            BSR PUTLONG_D7  ;Show current address
            BSR SPACE
            MOVE.B  #16,D4  ;Count of characters across

PARMS_OK1:  MOVE.B  (A3),D6 ;Get RAM byte to D6
            BSR PUTBYTE_D6  ;Show Byte value
```

```
            BSR SPACE

            ADDQ.L  #1,A3   ;Shift pointer up one
            SUBQ.B  #1,D4   ;Have we done 16 characters across
            TST.B   D4
            BNE PARMS_OK1   ;Not 16 across, then next byte

            MOVE.L  (A7)+,A4    ;Next Show ASCII for this line, Back to origional RAM location
            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
            MOVE.L  A3,D7
            BSR PUTLONG_D7  ;Show current address
            MOVE.B  #BLANK,D1
            BSR PUTCHAR

            MOVE.B  #8,D4   ;Count of characters across

SUBS_RAM3:  MOVE.B  (A3),D6
            BSR PUTBYTE_D6  ;Display current byte

            BSR GETBYTE_D7  ;Get new Hex value in D7 (0-FF)
            CMP.B   #ESC,D2 ;If ESC then we are done
            BEQ LOOP
            CMP.B   #CR,D2  ;Also CR we are done
            BEQ LOOP
            TST.B   D3  ;Is byte count = 0 from GETBYTE_D7 above then no update
            BEQ SUBS_RAM1   ;Is already on screen

            MOVE.B  D7,(A3) ;Substitute in the byte
            BRA SUBS_RAM4

SUBS_RAM1:  MOVE.B  #' ',D1
            BSR PUTCHAR
            BSR PUTCHAR
SUBS_RAM4:  BSR PUTCHAR

            ADDQ.L  #1,A3   ;Next byte
            SUBQ.B  #1,D4
            BNE SUBS_RAM3
            BRA SUBS_RAM2


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

MOVE_RAM:           ;M Command. Move RAM
            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 MOVE_OK
            MOVE.L  A3,A5   ;Else swap values
            MOVE.L  A4,A3
            MOVE    A5,A4

MOVE_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

MOVE_OK1:   MOVE.B  (A3)+,(A5)+
            CMP.L   A3,A4
            BGE MOVE_OK1
            BRA LOOP


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

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
            MOVE.B  #'-',D1 ;'-'
            BSR PUTCHAR
            MOVE.B  #'>',D1 ;'>'
            BSR PUTCHAR


            MOVE.B  D5,D6
            BSR PUTBYTE_D6  ;Display Byte value
            MOVE.B  #'H',D1 ;'H'
            BSR PUTCHAR

            MOVE.L  #$00FF0000,D6   ;Point to Port RAM area
            OR.B    D4,D6   ;OR in the hardware value
            MOVE.L  D6,A2   ;A2 now has port address
            MOVE.B  D5,(A2) ;Send actual data to port
            BRA LOOP


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

ASCII_RAM:          ;T Command. Display ASCII in RAM

            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   #CR,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 ASCII_OK
            MOVE.L  A3,A5   ;Else swap values
            MOVE.L  A4,A3
            MOVE    A5,A4

ASCII_OK:   BSR CRLF    ;New line
            MOVE.L  A3,D7
            BSR PUTLONG_D7  ;Show current address
            MOVE.B  #BLANK,D1
            BSR PUTCHAR
            MOVE.B  #32,D4  ;Count of characters across

ASCII_OK1:  MOVE.B  (A3),D1 ;Get RAM byte to D6
            CMP.B   #' ',D1
            BLT UNPRINTABLE
            CMP.B   #$7F,D1
            BGE UNPRINTABLE

ASCII_OK2:  BSR PUTCHAR

            SUBQ.B  #1,D4   ;Have we done 64 characters across
            TST.B   D4
            BEQ ASCII_OK

            ADDQ.L  #1,A3
            CMP.L   A3,A4   ;Are we done yet
            BLE LOOP
            BRA ASCII_OK1
UNPRINTABLE:
            MOVE.B  #'.',D1
            BRA ASCII_OK2

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

VERIFY_RAM:         ;M Command. Verify two RAM locations have the same data
            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 Console
            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
            CMP.B   #CR,D1  ;If CR then we also abort
            BEQ LOOP

            CMP.B   #' ',D1
            BLT NOASCII
            CMP.B   #$7F,D1
            BGE NOASCII
ECHO1:      BSR PUTCHAR ;Echo character
            BRA ECHO2

NOASCII:    MOVE.B  #'.',D1
            BRA ECHO1


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

GOTO_RAM:           ;G Command . Go to a location in RAM and start from there.
            BSR GETLONG_D7  ;Go to address in D7
            CMP.B   #CR,D2  ;Is it valid
            BNE ERROR
```

```
            MOVE.L  D7,A3   ;Save in A3
            JMP (A3)    ;That's all there is to it!



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

HEX_MATH:           ;H Command.      Add/subtract two hex mumbers.
            BSR GETLONG_D7  ;Get First number
            CMP.B   #',',D2 ;Is it valid
            BNE ERROR
            MOVE.L  D7,D4   ;Save in D4

            BSR GETLONG_D7  ;Get second number
            CMP.B   #CR,D2  ;Is it valid
            BNE ERROR
            MOVE.L  D7,D5   ;Save in D5
            MOVE    D7,D6

            LEA HEX_Data,A2 ;Hex data = string
            BSR PRINT_STRING
            ADD.L   D7,D6   :Total in D6
            MOVE.L  D6,D7
            BSR PUTLONG_D7

            LEA HEX_Data2,A2    ;Difference =
            BSR PRINT_STRING
            SUB.L   D4,D5
            MOVE.L  D5,D7
            BSR PUTLONG_D7
            LEA H_MSG_CRLF,A2   ;'H'
            BSR PRINT_STRING
            BRA LOOP


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

TEST_INTS:          ;L Command setup interrupt vectors in RAM 0 -- 400H
            BSR LOW_RAM_INITILIZE   ;Make it a callable routine for later use.
            LEA INTS_DONE_MSG,A2    ;'Interrupt vectors in RAM (0-400H) initilized'
            BSR PRINT_STRING
            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, write this, else
skip
            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
            CMPA.L  #$400,A2        ;Up to end of all vectors (3FFH)
            BMI.S   INIT2
            RTS     ;All Done


                    ;Below are the error messages
BUS_ERROR:  MOVEM.L D0-D7/A0-A6,-(A7)   ;SAVE ALL REGISTERS
            LEA BUS_ERROR_MSG,A2
            BSR PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6   ;POP ALL REGISTERS
            RTE
```

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

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

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

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

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

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

L1_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)   ;SAVE ALL REGISTERS
            LEA L1_INTERRUPT_MSG,A2
            BSR PRINT_STRING
            MOVEM.L (A7)+,D0-D7/A0-A6   ;POP ALL REGISTERS
            RTE
            RTE

L2_INTERRUPT:
            MOVEM.L D0-D7/A0-A6,-(A7)   ;SAVE ALL REGISTERS
            LEA L2_INTERRUPT_MSG,A2
            BSR PRINT_STRING
            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

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

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 68K board
            NOP     ;<-- 68K Is held in HALT mode here until released again by the Z80/master
CPU
            NOP
            NOP
            NOP
            BRA LOOP

;-------------------------------------------------------------------------------------------
----
SIGNALS:            ;Setup hardware signal tests to look at S-100 signals pDBIN, pWR*
```

```
            LEA SIGNALS_MSG,A2  ;Put CPU in hardware loop to test (pDBIN or pWR*), Enter test
RAM Location:
            BSR PRINT_STRING

            BSR GETLONG_D7  ;Get RAM location
            TST.B   D3  ;Byte count > 0
            BEQ LOOP
            CMP.B   #ESC,D2 ;If ESC then we abort
            BEQ LOOP
            CMP.B   #CR,D2  ;If not CR then we also abort
            BNE ERROR
            MOVE.L  D7,A3   ;Store in A3 (also in D7)

            LEA Menu_1or2_MSG,A2    ;Enter 1=pDBIN, 2=pWR* :
            BSR PRINT_STRING

            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   #'1',D1 ;Is it a pDBIN request
            BEQ DBIN_TEST
            CMP.B   #'2',D1 ;Is it a pWR* request
            BEQ WR_TEST
            BRA ERROR   ;Must be an error



DBIN_TEST:  MOVE.B  #CR,D1  ;Read test pDBIN
            BSR PUTCHAR
            MOVE.B  #LF,D1
            BSR PUTCHAR
            MOVE.B  #'r',D1
            BSR PUTCHAR

DBIN_TEST1: MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            MOVE.W  (A3),D2
            BRA DBIN_TEST1  ;Must Hit Reset button to abort


WR_TEST:    MOVE.W  #$AAAA,D2   ;Write test pWR*
            MOVE.B  CR,D1
            BSR PUTCHAR
            MOVE.B  #LF,D1
            BSR PUTCHAR
            MOVE.B  #'w',D1
            BSR PUTCHAR

WR_TEST1:   MOVE.W  D2,(A3)
            MOVE.W  D2,(A3)
            MOVE.W  D2,(A3)
            MOVE.W  D2,(A3)
            MOVE.W  D2,(A3)
            MOVE.W  D1,(A3)
            MOVE.W  D2,(A3)
```

```
            MOVE.W  D2,(A3)
            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 F4000H & JUMP to it
            LEA PATCH_MSG,A2    ;"Moving Code 4000H-9000H to F4000H, Then jump to that
location"
            BSR PRINT_STRING    ;The CPM3/Z80 SID program will place any test .bin file from a
disk to 4000H
                    ;in RAM (no matter what its final ORG is).
            MOVE.L  #$4000,A3   ;This command will then move test versions of this monitor (or
any other
            MOVE.L  #$F4000,A5  ;program) up out of the way to F4000H and execute it from
there.
            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



;*****************************************************************************************
************
;
;       Module to Test and diagnose the www.S100Computers.com IDE Board
;       Normally the DMA buffers will reside in the RAM on the 68K board itself at 00FD9000H
;
;*****************************************************************************************
************

MY_IDE:     LEA DISPLAY_FLAG,A2 ;Do we have detail sector data display flag on or off
            MOVE.B  $FF,(A2)    ;Set default to detailed sector display


            BSR CLEAR_ID_BUFFER ;Clear ID Buffer

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

            LEA INIT_1_ERROR,A2
            BSR PRINT_STRING
            BRA LOOP

INIT1_OK:
            BSR SET_DRIVE_B ;Select the second Drive/CF card (Do not mess with CPM Drive 0)
;           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 SET_DRIVE_A ;Back to first drive/CF Card
```

```
;           BSR DRIVE_ID    ;Get the drive 0 id info. If there is no drive just abort
;           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


            LEA BAD_DRIVE,A2    ;"Error obtaining first Drive ID"
            BSR PRINT_STRING
;           BRA LOOP

INIT4_OK:   CLR.B   (RAM_SEC)   ;Sec 0
            MOVE.B  #0,(RAM_TRK)    ;Track 0

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

;           BSR IDEinit ;For some reason this need to be here after getting the drive ID.
                    ;otherewise sector #'s are off by one! (Probably because on non-LBA reads)
;           BSR WR_LBA  ;Update LBA on "1st" drive



;------------- MAIN IDE DRIVE DIAGNOSTIC MENU ---------------------------

IDE_LOOP:   LEA IDE_SIGNON0,A2  ;List IDE command options
            BSR PRINT_STRING

            CMP.B   #0,(CURRENT_IDE_DRIVE)
            BNE SIGN_B
            LEA CURRENT_MSG_A,A2
            BRA IDE_LOOP0
SIGN_B:     LEA CURRENT_MSG_B,A2
IDE_LOOP0:
            BSR PRINT_STRING

            CMP.B   #0,(DISPLAY_FLAG)   ;Do we have detail sector data display flag ON or OFF
            BNE IDE_LOOP1
            LEA IDE_SIGNON1,A2  ;"ON"
            BRA IDE_LOOP2
IDE_LOOP1:
            LEA IDE_SIGNON2,A2  ;"OFF"
IDE_LOOP2:
            BSR PRINT_STRING


            LEA IDE_SIGNON3,A2  ;List IDE command options
            BSR PRINT_STRING

            BSR DISPLAY_POSITION    ;Display current Track,sector,head#

            BSR CRLF
            LEA IDE_MENU_CMD,A2 ;Enter a command
            BSR PRINT_STRING

            BSR GETCHAR ;Get a command from Console
```

```
            CMP.B   #ESC,D1 ;Abort if ESC
            BNE NOT_ESC
            BRA START   ;Back to start of Monitor

NOT_ESC:    BSR TOUPPER ;Lower case to Upper case

   IFNE SIMMULATOR      ;If SIMMULATOR = 1
            BSR PUTCHAR ;Echo character
   ENDC
            CMP.B   #'A',D1
            BLT ERR
            CMP.B   #'Z',D1
            BGT ERR
            SUBI.B  #'A',D1
            LSL.L   #2,D1   ;X4 for offset into table
            LEA IDE_TABLE,A2    ;Start of cnd table
            MOVE.L  (A2,D1),A3  ;Add in X4 offset
            JMP (A3)

;             INDIVIDUAL IDE DRIVE MENU COMMANDS

;---------------Select Drive/CF card ----------------------------------------
SET_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
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

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


;---------------- Do the IDEntify drive command, and display the IDE_Buffer ------------
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,A4   ;Store data here
            BSR MoreRD16    ;Get 256 words of data from REGdata port to IDE_Buffer

            LEA msgmdl,A2   ;Print the drive's model number
            BSR PRINT_STRING
```

```
            LEA IDE_Buffer+52,A2
            MOVE.B  #10,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
            BSR PRINT_STRING
            LEA IDE_Buffer+20,A2
            MOVE.B  #5,D3   ;Character count in words
            BSR Print_ID_Info
            BSR CRLF
                    ;PRINT_STRING the drive's firmware revision string
            LEA msgrev,A2
            BSR PRINT_STRING
            LEA IDE_Buffer+46,A2
            MOVE.B  #2,D3
            BSR Print_ID_Info   ;Character count in words
            BSR CRLF
                    ;Print the drive's cylinder, head, and sector specs
            LEA msgcy,A2
            BSR PRINT_STRING
            LEA IDE_Buffer+2,A2
            BSR Print_ID_Hex
            LEA msghd,A2
            BSR PRINT_STRING
            LEA IDE_Buffer+6,A2
            BSR Print_ID_Hex
            LEA msgsc,A2
            BSR PRINT_STRING
            LEA IDE_Buffer+12,A2
            BSR Print_ID_Hex
            BSR CRLF
            BSR CRLF
            CLR.B   D3  ;Ret Z
            BRA IDE_LOOP    ;Back to IDE Menu

Print_ID_Info:
            ADDQ.L  #1,A2
            MOVE.B  (A2),D6 ;Text is stored high byte then low byte
            BSR PUTBYTE_D6
            MOVE.B  -(A2),D6
            BSR PUTBYTE_D6
            ADDQ.L  #2,A2
            SUBQ.B  #1,D3
            BNE Print_ID_Info
            RTS

                    ; Print a 16 bit number in RAM located @ [A2]
Print_ID_Hex:
            ADDQ.L  #1,A2   ;(Note Special Low Byte First. Used only for Drive ID)
            BSR PUTBYTE_D6
            MOVE.B  -(A2),D6
            BSR PUTBYTE_D6
            RTS



;-------------- Read the current selected sector (based on LBA) to the IDE Buffer
READ_SEC:
            LEA IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)    ;DMA initially to IDE_Buffer

            BSR READSECTOR
```

```
            BEQ Main1B
            BSR CRLF    ;Here if there was a problem
            BRA IDE_LOOP    ;Back to IDE Menu

Main1B:     LEA msgrd,A2    ;Sector read OK
            BSR PRINT_STRING

            MOVE.B  (DISPLAY_FLAG),D1   ;Do we have detail sector data display flag on or off
            OR.B    D1,D1   ;NZ = on
            BNE SHOW_SEC_RDATA
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

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

;----------------- Write the current selected sector (based on LBA) from the IDE Buffer
WRITE_SEC:
            LEA CONFIRM_WR_MSG,A2   ;Are you sure?
            BSR PRINT_STRING
            BSR GETCHAR
            BSR TOUPPER
            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:
            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 msgrd,A2    ;Sector written OK
            BSR PRINT_STRING

            MOVE.B  (DISPLAY_FLAG),D1   ;Do we have detail sector data display flag on or off
            OR.B    D1,D1
            BNE SHOW_SEC_WDATA
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

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

```
;---------------------  Set a new LBA value from inputted Track/Sec info. Send to drive
SET_LBA:
            LEA SET_LBA_MSG,A2  ;Set new LBA and send to drive
            BSR PRINT_STRING
            BSR GEN_HEX32_LBA   ;Get new CPM style Track & Sector number and put them in RAM at
RAM_SEC & RAM_TRK
            BLT main3b  ;Ret C set if abort/error
            BSR WR_LBA  ;Update LBA on drive
main3b:     BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu


;---------------------- Toggle detailed sector display on/off
DISPLAY:
            NOT.B   (DISPLAY_FLAG)  ;Do we have detail sector data display flag on or off
            BSR CRLF
            BRA IDE_LOOP    ;Update menu showing new "D" command


;---------------------- Point current sector to next sector
NEXT_SECT:
            BSR GET_NEXT_SECT
            BNE AT_END
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu
AT_END:
            LEA AT_END_MSG,A2   ;Tell us we are at end of disk
            BSR PRINT_STRING
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

;---------------------- Point current sector to previous sector
PREV_SECT:
            BSR GET_PREV_SECT
            BNE AT_START
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu
AT_START:
            LEA AT_START_MSG,A2 ;Tell us we are at start of disk
            BSR PRINT_STRING
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

;---------------------- Sequentially read sectors from disk starting at current LBA position
SEQ_SEC_RD:
            BSR IDEwaitnotbusy
            BGE MORE_SEC
            BRA SHOWerrors

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


            MOVE.B  #'<',D1
            BSR PUTCHAR
            MOVE.L  D7,A4
            BSR PUTLONG_D7
            MOVE.B  #'>',D1
            BSR PUTCHAR
```

```
            BSR READSECTOR  ;If there are errors they will show up in READSECTOR
            BEQ SEQOK

            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#

            CMP.B   #0,(DISPLAY_FLAG)   ;Do we have detail sector data display flag on or off
            BEQ MORES2  ;NZ = on
            LEA IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)    ;DMA initially to IDE_Buffer

            BSR DISPLAY_SEC

MORES2:     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
read

N_RD_SEC:
            LEA READN_MSG,A2
            BSR PRINT_STRING
            BSR GETLONG_D7  ;Hex to D7

            MOVE.W  D7,(SECTOR_COUNT)   ;Store sector count

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

NextRSec:
            LEA READN_MSG,A2
            BSR PRINT_STRING
            BSR WR_LBA  ;Update LBA on drive
            BSR DISPLAY_POSITION    ;Display current Track,sector,head#
```

```
            MOVE.L  (RAM_DMA_STORE),D1  ;DMA initially to IDE_Buffer
            MOVE.L  D1,(RAM_DMA)

            BSR READSECTOR  ;Actully, Sector/track values are already updated

            MOVE.L  (RAM_DMA),D1
            MOVE.L  D1,(RAM_DMA_STORE)

            SUBQ.W  #1,(SECTOR_COUNT)
            BNE NEXT_SEC_NRD
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

NEXT_SEC_NRD:
            BSR GET_NEXT_SECT
            BEQ NextRSec
            LEA AT_END_MSG,A2   ;Tell us we are at end of disk
            BSR PRINT_STRING
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu



;------------------- Write N Sectors to disk
;Note unlike the normal sector write routine, this routine incriments the DMA address after
each write.

N_WR_SEC:
            LEA CONFIRM_WR_MSG,A2   ;Are you sure?
            BSR PRINT_STRING
            BSR GETCHAR
            BSR TOUPPER
            CMP.B   #'Y',D1
            BEQ WR_SEC_OK2
            BSR CRLF    ;Here if there was a problem
            BRA IDE_LOOP    ;Back to IDE Menu

WR_SEC_OK2:
            LEA WRITEN_MSG,A2
            BSR PRINT_STRING
            BSR GETLONG_D7  ;Hex to D7

            MOVE.W  D7,(SECTOR_COUNT)   ;store sector count

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

NextWSec:
            LEA WRITEN_MSG,A2
            BSR PRINT_STRING
            BSR WR_LBA  ;Update LBA on drive
            BSR DISPLAY_POSITION    ;Display current Track,sector,head#

            MOVE.L  (RAM_DMA_STORE),D1  ;DMA initially to IDE_Buffer
            MOVE.L  D1,(RAM_DMA)

            BSR WRITESECTOR ;Actully, Sector/track values are already updated

            MOVE.L  (RAM_DMA),D1
            MOVE.L  D1,(RAM_DMA_STORE)

            SUBQ.W  #1,(SECTOR_COUNT)
```

```
            BNE NEXT_SEC_NWR
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

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
            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_OK
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

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

            MOVE.W  #$0E5E5,D1  ;First set Sector pattern to E5's
            BSR RAM_FILL
            BSR CRLF

NEXT_FORMAT:
            MOVE.L  IDE_BUFFER,A4
            MOVE.L  A4,(RAM_DMA)

            BSR WRITESECTOR ;Will return error if there was one
            BEQ NEXTF1  ;Z means the sector write was OK

            LEA FORMAT_ERR,A2   ;Indicate an error
            BSR PRINT_STRING
            BSR SHOW_TRACK_SEC  ;Show current location of error
            BSR CRLF
            BRA FNEXTSEC3

NEXTF1:     CMP.B   #0,(RAM_SEC)    ;Get Current Sector
            BNE FNEXTSEC2

            BSR SHOW_TRACK

FNEXTSEC2:
            BSR GETSTAT ;Any keyboard character will stop display
            BEQ FNEXTSEC1
            BSR GETCHAR ;Flush character
FNEXTSEC3:
            LEA CONTINUE_MSG,A2
            BSR PRINT_STRING
            BSR GETCHAR
```

```
            CMP.B   #ESC,D1
            BNE FNEXTSEC1
F_DONE:     MOVE.B  #0,D1   ;Login drive A:
            BSR SELECT_DRIVE
            MOVE.B  D1,(CURRENT_IDE_DRIVE)
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

FNEXTSEC1:
            BSR GET_NEXT_SECT
            BEQ NEXT_FORMAT
            LEA AT_END_MSG,A2   ;Tell us we are at end of disk
            BSR PRINT_STRING
            BRA F_DONE

;---------------- Copy Drive A: to Drive B:  ------------------------------
COPY_AB:
            LEA DiskCopyMsg,A2
            BSR PRINT_STRING
            BSR GETCHAR
            BSR TOUPPER
            CMP.B   #'Y',D1
            BEQ COPY_AB1
            BRA C_DONE

COPY_AB1:
            MOVE.B  #0,(RAM_SEC)    ;Start with CPM sector 0
            MOVE.B  #0,(RAM_TRK)    ;Start with CPM Track 0
            BSR CRLF
            BSR CRLF

NextDCopy:
            MOVE.B  #0,D1   ;Login drive A:
            BSR SELECT_DRIVE

            BSR WR_LBA  ;Update LBA on "A:" drive

            LEA IDE_BUFFER,A4
            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)
            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

            BSR CRLF
            BSR CRLF

NextVCopy:
            MOVE.B  #0,D1   ;Login drive A:
            BSR SELECT_DRIVE

            BSR WR_LBA  ;Update LBA on "A:" drive

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

            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_BUFFER2,A4
            MOVE.L  A4,(RAM_DMA)    ;DMA initially to IDE_Buffer2

            BSR READSECTOR

            LEA IDE_Buffer2,A2
            LEA IDE_Buffer,A3
```

```
            MOVE.W  #512,D1 ;Length of sector in words

NEXT_CMP:
            CMP.B   (A2)+,(A3)+ ;Are they the same
            BNE VER_ERROR
            SUBQ.W  #1,D1
            BNE NEXT_CMP    ;CX will contain count of words done so far, (0 if done OK)
            BRA IDE_VERIFY_OK

VER_ERROR:
            LEA VERIFY_ERR,A2   ;Indicate an error
            BSR PRINT_STRING
            BSR SHOW_TRACK_SEC  ;Show current location of error
            LEA DRIVE1_MSG,A2   ;'  Drive A',CR,LF
            BSR PRINT_STRING

            LEA IDE_Buffer,A2
            MOVE.W  #512,D1 ;Length of sector in words
VER_SOURCE:
            MOVE.B  (A2)+,D6    ;Show source values
            BSR PUTBYTE_D6
            SUBQ.W  #1,D1
            BNE VER_SOURCE
            BSR CRLF
            BSR SHOW_TRACK_SEC  ;Show current location of error
            LEA DRIVE2_MSG,A2   ;'  Drive B',CR,LF
            BSR PRINT_STRING

            LEA IDE_Buffer2,A2
            MOVE.W  #512,D1 ;Length of sector in words
VER_DEST:
            MOVE.B  (A2)+,D6    ;show destination values
            BSR PUTBYTE_D6
            SUBQ.W  #1,D1
            BNE VER_DEST
            BSR CRLF
            BRA VERIFYT ;Do not ask for a continue message here. Just continue
                    ;If you want it change to VERIFYT1
IDE_VERIFY_OK:
            CMP.B   #0,(RAM_SEC)    ;Get Current Sector
            BNE VERIFYT

            BSR SHOW_TRACK

VERIFYT:
            BSR GETSTAT ;Any keyboard character will stop display
            BEQ V_NEXTSEC1
            BSR GETCHAR ;Flush character
VERIFYT1:
            LEA CONTINUE_MSG,A2
            BSR PRINT_STRING
            BSR GETCHAR
            CMP.B   #ESC,D1
            BNE V_NEXTSEC1
            BRA V_NEXTSEC3
V_NEXTSEC1:
            BSR GET_NEXT_SECT   ;Update to next sector/track
            BNE V_NEXTSEC2
            BRA NextVCopy
V_NEXTSEC2:
            LEA VerifyDone,A2   ;Tell us we are all done.
            BSR PRINT_STRING
V_NEXTSEC3:
```

```
            MOVE.B  #0,D1   ;Login drive A:
            BSR SELECT_DRIVE
            MOVE.B  D1,(CURRENT_IDE_DRIVE)
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu




;------------------ Fill RAM buffer with 0's

RAMCLEAR:
            CLR.B   D1
RAM_FILL:
            LEA IDE_Buffer,A2
            MOVE.W  #512,D2 ;512 bytes total
CLEAR1:     MOVE.B  D1,(A2)+
            SUBQ.W  #1,D2
            BNE CLEAR1

            LEA FILL_MSG,A2
            BSR PRINT_STRING
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu

;----------------  Power up a Hard Disk

SPINUP:     MOVE.B  COMMANDspinup,D4
spup2:      MOVE.B  REGcommand,D5
            BSR IDEwr8D
            BSR IDEwaitnotbusy
            BGE L_7
            BRA SHOWerrors
L_7:        CLR.B   D1  ;Clear carry
            BSR CRLF
            BRA IDE_LOOP    ;Back to IDE Menu


;-------------------- Tell the Hard disk to power down

SPINDOWN:
            BSR IDEwaitnotbusy
            BGE L_8
            BRA SHOWerrors
L_8:        MOVE.B  COMMANDspindown,D4
            JMP spup2


;-------------------- Back to parent 8086 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  (RAM_DRIVE_TRK+1),D6    ;High "cylinder" byte
            BSR PUTBYTE_D6
            MOVE.B  (RAM_DRIVE_TRK),D6  ;Low "cylinder" byte
            BSR PUTBYTE_D6

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

SHOW_TRACK_SEC:     ;Display current (CPM) track,sector
            LEA msgCPMTRK,A2
            BSR PRINT_STRING    ;---- CPM FORMAT ----
            MOVE.B  (RAM_TRK+1),D6  ;High TRK byte
            BSR PUTBYTE_D6
            MOVE.B  (RAM_TRK),D6    ;Low TRK byte
            BSR PUTBYTE_D6

            LEA msgCPMSEC,A2
            BSR PRINT_STRING

            LEA RAM_SEC,A2  ;Low Sec (Only)
            BSR PRINT_STRING
            RTS
```

```
SHOW_TRACK:
            LEA msgCPMTRK,A2
            BSR PRINT_STRING    ;---- CPM FORMAT ----
            MOVE.B  (RAM_TRK+1),D6  ;High TRK byte
            BSR PRINT_STRING
            MOVE.B  (RAM_TRK),D6    ;Low TRK byte
            BSR PRINT_STRING
            RTS


DISPLAY_SEC:        ;Print a DISPLAY_SEC of the data in the 512 byte IDE_Buffer (RAM_DMA)
            BSR CRLF    ;Note written so it can be easily converted to a "normal: DS: based"
routine
            LEA RAM_DMA,A2  ;Get Current DMA Address to A2 & A3
            MOVE.L  A2,A3
            MOVE.B  #32,D3  ;print 32 lines

SF172:      BSR CRLF
            MOVE.L  A2,D7
            BSR PUTLONG_D7  ;Show current address
            MOVE.B  #BLANK,D1
            BSR PUTCHAR
            MOVE.B  #BLANK,D1
            BSR PUTCHAR

            MOVE.B  #16,D4  ;16 characters across
SF175:      MOVE.B  (A2)+,D6
            BSR PUTBYTE_D6
            SUBQ.B  #1,D4
            BNE SF175

            MOVE.B  #BLANK,D1
            BSR PUTCHAR
            MOVE.B  #BLANK,D1
            BSR PUTCHAR
            MOVE.B  #BLANK,D1
            BSR PUTCHAR

            MOVE.B  #16,D4  ;16 across again
Sloop2:     MOVE.B  (A3)+,D6
            AND.B   #$7f,D6
            CMP.B   #' ',D6 ;filter out control characters
            BLE.B   Sloop3
Sloop4:     MOVE.B  #'.',D6
Sloop3:     CMP.B   #'~',D6
            BLE.B   Sloop4
            MOVE.B  D6,D1
            BSR PUTCHAR
            SUBQ.B  #1,D4
            BNE SF172   ;--DH has total byte count
            BSR CRLF
            RTS



                    ;Point to next sector.  Ret Z if all OK NZ if at end of disk
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
            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:
            EOR.B   D1,D1
            SUBQ.B  #1,D1
            RTS



                    ;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)
PREVIOUS_SEC_DONE:
            BSR WR_LBA  ;Update the LBC pointer
            EOR.B   D1,D1
            RTS     ;Ret z if all OK
AT_00:      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  D1,D4
            AND.B   #1,D4
            BNE MoreError   ;Go to  REGerr register for more info
                    ;All OK if 01000000

            AND.B   #$80,D1
            BEQ NOT7
            LEA DRIVE_BUSY,A2   ;Drive Busy (bit 7) stuck high.   Status =
            BSR PRINT_STRING
            BRA DONEERR

NOT7:       AND.B   #$40,D1
            BNE NOT6
            LEA DRIVE_NOT_READY,A2  ;Drive Not Ready (bit 6) stuck low.  Status =
            BSR PRINT_STRING
            BRA DONEERR

NOT6:       AND.B   #$20,D1
            BNE NOT5
            LEA DRIVE_WR_FAULT,A2   ;Drive write fault.    Status =
            BSR PRINT_STRING
```

```
            BRA DONEERR

NOT5:       LEA UNKNOWN_ERROR,A2
            BSR PRINT_STRING
            BRA DONEERR

MoreError:          ;Get here if bit 0 of the status register indicted a problem
            MOVE.B  #REGerr,D5  ;Get error code in REGerr
            BSR IDErd8D
            MOVE.B  D4,D6
            MOVE.L  D1,-(A7)    ;Save D1


            AND.B   #$10,D6
            BEQ NOTE4
            LEA SEC_NOT_FOUND,A2
            BSR PRINT_STRING
            BRA DONEERR

NOTE4:      AND.B   #$80,D6
            BEQ NOTE7
            LEA BAD_BLOCK,A2
            BSR PRINT_STRING
            BRA DONEERR

NOTE7:      AND.B   #$40,D6
            BEQ NOTE6
            LEA UNRECOVER_ERR,A2
            BSR PRINT_STRING
            BRA DONEERR

NOTE6:      AND.B   #$4,D6
            BEQ NOTE2
            LEA INVALID_CMD,A2
            BSR PRINT_STRING
            JMP DONEERR

NOTE2:      AND.B   #$2,D6
            BEQ NOTE1
            LEA TRK0_ERR,A2
            BSR PRINT_STRING
            JMP DONEERR

NOTE1:      LEA UNKNOWN_ERROR1,A2
            BSR PRINT_STRING
            BSR CRLF
            MOVE    (A7)+,D1
            OR.B    D1,D1   ;Set NZ flag
            RTS

DONEERR:            ;Display Byte bit pattern in D6
            BSR PUTBITS_D6  ;Show error bit pattern
            BSR CRLF
            MOVE    (A7)+,D1    ;Get origional flags
            OR.B    D1,D1   ;Set NZ flag
            RTS


;=============================================================================================
==
; 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 reset on the drive,
                            ;By default the drive will come up initilized in LBA mode.

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

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

            MOVE.W  #IDE_Reset_Delay,D1         ;Time delay for reset/initilization (~66 uS,
with 8MHz 8086, 1 I/O wait state)
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
drive,head 0000)
;           MOVE.B  #%10100000,D4       ;For Trk,Sec,head (non LBA) use 10100000 (This is the
mode we use for MSDOS)
                            ;Note. Cannot get LBA mode to work with an old Seagate Medalist
6531 drive.
                            ;have to use the non-LBA mode. (Common for old hard disks).
            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 to
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 otherewise we
            BSR WR_LBA      ;will get stuck on bad sector
            BSR IDEwaitnotbusy      ;make sure drive is ready
            BGE L_19
            BRA SHOWerrors      ;Returned with NZ set if error

L_19:       MOVE.B  #COMMANDread,D4
            MOVE.B  #REGcommand,D5
            BSR IDEwr8D     ;Send sec read command to drive.
            BSR IDEwaitdrq      ;wait until it's got the data
            BGE L_20
            JMP SHOWerrors

L_20:       MOVE.L  (RAM_DMA),D1        ;Get Current DMA Address to D2
            MOVE.L  D1,A2
            MOVE.B  #0,D6       ;Read 512 bytes to D6

MoreRD16:
            MOVE.B  #REGdata,(IDEportC)     ;REG regsiter address
            OR.B    #IDErdline,(IDEportC)       ;08H+40H, Pulse RD line

            MOVE.B  (IDEportA),(A2)+        ;Read the lower byte first
            MOVE.B  (IDEportB),(A2)+        ;Then the upper byte first

            MOVE.B  #REGdata,(IDEportC)     ;Deassert RD line
            SUBQ.B  #$1,D6
            BNE MoreRD16

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


                    ;Write a sector, specified by the 3 bytes in LBA (_ IX+0)",
                    ;Z on success, NZ to error routine if problem
                    ;Tell which sector we want to read from.
WRITESECTOR:        ;Note: Translate first in case of an error otherewise we
            BSR WR_LBA      ;will get stuck on bad sector
            BSR IDEwaitnotbusy      ;make sure drive is ready
            BGE L_22
            JMP SHOWerrors

L_22:       MOVE.B  #COMMANDwrite,D4
            MOVE.B  REGcommand,D5
            BSR IDEwr8D     ;tell drive to write a sector
            BSR IDEwaitdrq      ;wait unit it wants the data
            BGE L_23
            JMP SHOWerrors

L_23:       MOVE.L  (RAM_DMA),D1
            MOVE.L  D1,A2       ;Get Current DMA Address
            MOVE.B  #0,D6       ;256X2 bytes

            MOVE.B  #WRITEcfg8255,(IDECtrlPort)

WRSEC1_IDE:
            MOVE.B  (A2)+,(IDEportA)
            MOVE.B  (A2)+,(IDEportA)
```

```
            MOVE.B  #REGdata,(IDEportC)
            OR  #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 registers
                        ;Note we do not need to set the upper nibble of the LBA
                        ;It will always be 0 for these small CPM drives (so no High Cylinder
WR_LBA:                 ;numbers etc).
            MOVE.B  (RAM_SEC),D1        ;LBA mode, Low sectors go directly
            ADDQ.B  #$1,D1      ;Sectors are numbered 1 -- MAXSEC (even in LBA mode)
            MOVE.B  D1,(RAM_DRIVE_SEC)      ;For Diagnostic Diaplay Only
            MOVE.B  D1,D4
            MOVE.B  D4,(REGsector)      ;Send info to drive
            BSR IDEwr8D     ;Write to 8255 A Register
                        ;Note: For drive we will have 0 - MAXSEC sectors only

            MOVE.B  (RAM_TRK),D1
            MOVE.B  D1,(RAM_DRIVE_TRK)      ;Send Low TRK#
            MOVE.B  D1,D4
            MOVE.B  (REGcylinderLSB),D5
            BSR IDEwr8D     ;Write to 8255 A Register

            MOVE.B  (RAM_TRK+1),D1
            MOVE.B  D1,(RAM_DRIVE_TRK+1)
            MOVE.B  D1,D4
            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 update LED HEX
Display
                        ;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 BIOS Section)
                        ;This will display Head, Cylinder and Sector on the LED HEX display
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 drive
            OR.B    (CURRENT_TRACK_HIGH),D4 ;Will display  the Head in top nibble and the two
bits of the HIGH bits
            MOVE.B  (REGcylinderMSB),D5     ;of the high cylinder in the low nibble.
            BSR IDEwr8D_X       ;Special output to 8255 B Register (Not A) to update LED HEX
Display ONLY

            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)

            MOVE.B  (CURRENT_SECTOR),D4     ;Bits 0-5 only  (currently 1-17)
            MOVE.B  (REGsector),D5      ;Send "Sector#"
            BSR IDEwr8D     ;Write to 8255 A Register

            MOVE.B  (SECTORS_TO_DO),D4      ;# of CONTIGOUS sectors to send
            MOVE.B  (REGseccnt),D5
            BSR IDEwr8D     ;Write to 8255 A Register
            RTS



IDEwaitnotbusy:         ;Drive READY if 01000000
            MOVE.W  #$0FFFF,D6
MoreWait:
            MOVE.B  #REGstatus,D5       ;wait for RDY bit to be set
            BSR IDErd8D     ;Note AH or CH are unchanged
            MOVE.B  D4,D1
            AND.B   #%11000000,D1
            EOR.B   #%01000000,D1
            BEQ DONE_NOT_BUSY
            SUBQ.W  #1,D6
            BNE MoreWait
            MOVE.B  #$FF,D0
            LSL.B   #1,D0       ;Set carry to indicate an error
            RTS
DONE_NOT_BUSY:
            CLR.B   D1      ;Clear carry it indicate no error
            RTS

                    ;Wait for the drive to be ready to transfer data.
IDEwaitdrq:             ;Returns the drive's status in Acc
            MOVE.W  #$0FFFF,D6
MoreDRQ:
            MOVE.B  #REGstatus,D5       ;wait for DRQ bit to be set
            BSR IDErd8D     ;Note AH or CH are unchanged
            MOVE.B  D4,D1
            AND.B   #%10001000,D1
            CMP.B   #%00001000,D1
            BEQ DoneDRQ
            SUBQ.W  #1,D6
            BNE MoreDRQ
            MOVE.B  #$FF,D0
            LSL.B   #1,D0       ;Set carry to indicate an error
            RTS
DoneDRQ:
            CLR.B   D1      ;Clear carry it indicate no error
            RTS


CLEAR_ID_BUFFER:            ;Clear the ID Buffer area
```

```
            MOVE.W  #$2020,D1       ;Clear to spaces
            LEA IDE_Buffer,A2
            MOVE.B  #255,D2     ;512 bytes total
CLEAR2:     MOVE.W  D1,(A2)+
            SUBQ.B  #1,D2
            BNE CLEAR2

            MOVE.W  #$0000,D1       ;Clear to 0
            LEA IDE_Buffer,A2
            MOVE.B  #7,D2       ;14 bytes total
CLEAR3:     MOVE.W  D1,(A2)+
            SUBQ.B  #1,D2
            BNE CLEAR3
            RTS



;---------------------------------------------------------------------
; 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 in [DH]
            MOVE.B  D5,(IDEportC)       ;Select IDE register, drive address onto control lines

            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 lines

            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 control
lines

            OR.B    #IDEwrline,(IDEportC)           ;Assert write pin

            MOVE.B  D5,(IDEportC)           ;Select IDE register, drive address onto control
lines

            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 control
lines
```

```
            OR.B    #IDEwrline,(IDEportC)           ;Assert write pin

            MOVE.B  D5,(IDEportC)           ;Select IDE register, drive address onto control
lines

            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 changed (not A0 and A1)
            LEA MODEM_BIN_SIGNON,A2 ;Will show message explaining we are about to upload a .bin
file
            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
            BNE ERROR   ;From there BRA to LOOP
            MOVE.L  D7,A3   ;<<< Save RAM address in A3
            BSR CRLF

            MOVE.L  SERIAL_RETRYS,D2    ;Number of times to try reading serial port before
aborting.
            BSR RECV

            LEA START_B_POINTER_MSG,A2  ;"Will load .bin file data starting at RAM location "
            BSR PRINT_STRING
            MOVE.L  A3,D7
            BSR PUTLONG_D7  ;Show current address
            BSR CRLF

            LEA DOWNLOAD_MSG,A2 ;Speak "Downloading file"
            BSR SPEAK_STRING

RECV_LOOP:          ;<<< MAIN RECIEVE LOOP >>>>
            CLR.B   D2  ;GET 0
            MOVE.B  D2,(ERRCT)  ;Start error count with 0
RECV_HDR:
            LEA RMSG,A2 ;Reading sector message
            BSR PRINT_STRING
            MOVE.B  (SECTNO),D6 ;Show current sector number
            ADD.B   #1,D6
            BSR PUTBYTE_D6
            LEA RAM_MSG,A2  ;"H.  IF OK, will write to RAM location"
            BSR PRINT_STRING
            MOVE.L  A3,D7
            BSR PUTLONG_D7  ;Show current address
            LEA H_MSG,A2
            BSR PRINT_STRING
```

```
            MOVE.L  #20*SERIAL_RETRYS,D2    ;Number of times to try reading serial port before
aborting.
            BSR RECV    ;Get character from modem serial port
            CMP.B   #$FF,D2 ;Return with FF in D2 if all is OK
            BEQ RHNTO   ;WE ARE OK, NO TIMEOUT

RECV_HDR_TIMEOUT:
            LEA TOUTM,A2    ;PRINT TIMEOUT MESSAGE
            BSR PRINT_STRING
            MOVE.B  (ERRCT),D6  ;Show error count as well
            BSR PUTBYTE_D6
            BSR CRLF

RECV_SECT_ERR:      ;PURGE THE LINE OF INPUT CHARS
            MOVE.L  #SERIAL_RETRYS,D2   ;Number of times to try reading serial port before
aborting.
            BSR RECV
            TST.B   D2
            BNE RECV_SECT_ERR   ;LOOP UNTIL SENDER DONE

            MOVE.B  #NAK,D1
            BSR SERIAL_OUT  ;SEND NAK

            MOVE.B  (ERRCT),D6  ;Inc Error Count (ERRCT)
            ADD.B   #1,D6
            MOVE.B  D6,(ERRCT)
            CMP.B   #MODEM_ERR_LIMIT,D6 ;Currently set for 5 trys
            BLE RECV_HDR    ;Go try again

            BSR CHECK_FOR_QUIT
            TST.B   D1
            BEQ RECV_HDR    ;Try again
            LEA BAD_HEADER,A2   ;'Unable to get a valid file header!'
            BSR PRINT_STRING
            BRA MODEM_DONE  ;Abort back to Monitor start

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
aborting.
            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
aborting.
            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
aborting.
            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
aborting.
            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
            BRA RECV_SECT_ERR

GOT_EOT:            ;DONE - CLOSE UP SHOP
            MOVE.B  #ACK,d1 ;ACK THE EOT
            BSR SERIAL_OUT
            BSR CRLF
            LEA FINISH_MSG,A2   ;Speak downloading finished
            BSR SPEAK_STRING
            LEA TRANS_DONE,A2
EXIT2:      BSR PRINT_STRING
MODEM_DONE:
            BRA LOOP

EXIT1:      LEA ABORT_MSG,A2
            JMP EXIT2
```

```
CHECK_FOR_QUIT:     ;MULTIPLE ERRORS, ASK IF TIME TO QUIT
            MOVE.B  #0,(ERRCT)  ;RESET ERROR COUNT
            LEA QUITM,A2
            BSR PRINT_STRING
            BSR GETCHAR
            MOVE.L  D1,-(A7)
            BSR CRLF
            MOVE.L  (A7)+,D1    ;Is D1 = 'R'
            CMP.B   #'R',D1
            BEQ DONE_CHECK
            CMP.B   #'r',D1
            BEQ DONE_CHECK
            CMP.B   #'Q',D1
            BEQ NOT_DONE_CHECK
            CMP.B   #'q',D1
            BEQ NOT_DONE_CHECK
            CMP.B   #ESC,D1
            BEQ NOT_DONE_CHECK
            JMP CHECK_FOR_QUIT
NOT_DONE_CHECK:
            OR.B    D1,D1   ;TURN OFF ZERO FLAG
DONE_CHECK: RTS



;======================== SUPPORT ROUTINES
==========================================================

GETLONG_D7: CLR.L   D7  ;Get a long number and place in D7 (1-8 bytes)
            CLR.B   D3  ;Byte count
GETLONG1:   BSR GETNIBBLE   ;Get a Hex byte in D1
            CMP.B   #ESC,D2
            BEQ ABORT_LONG
            CMP.B   #CR,D2  ;Loop until GETNIBBLE returns $FF = CR or ','
            BEQ DONE_LONG
            CMP.B   #',',D2 ;Loop until GETNIBBLE returns $FF = CR or ','
            BEQ DONE_LONG
            CMP.B   #BLANK,D2   ;Loop until GETNIBBLE returns $FF = CR or ','
            BEQ DONE_LONG
            LSL.L   #4,D7
            OR.B    D1,D7
            ADDQ.B  #1,D3
            BRA GETLONG1

ABORT_LONG: MOVE.L  #0,D7   ;Return 0, (D1 contains ESC)
            MOVE.L  D1,-(A7)
            MOVE.B  #CR,D1  ;Send out a CR/LF before returning
            BSR PUTCHAR
            MOVE.B  #LF,D1
            BSR PUTCHAR
            MOVE.L  (A7)+,D1
DONE_LONG:  RTS     ;Normal return with FFh in D1 if CR or ',' was entered. D1 = FEh if a space
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
            OR.B    #$30,D1         ;CONVERT TO ASCII
            CMP.B   #$39,D1         ;SEE IF IT IS>9
```

```
            BLE.S   HEXOK3
            ADDQ.B   #7,D1              ;ADD TO MAKE 10=>A
HEXOK3:     BSR PUTCHAR


PUTBYTE_D6:         ;Print HEX value in D6
            MOVE.W  D6,D1   ;Origional number again to D1
            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   HEXOK4
            ADDQ.B   #7,D1              ;ADD TO MAKE 10=>A
HEXOK4:     BSR PUTCHAR
                    ;Address lower high byte nibble
            MOVE.W  D6,D1   ;Origional number again to D1
            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   HEXOK5
            ADDQ.B   #7,D1              ;ADD TO MAKE 10=>A
HEXOK5:     BSR PUTCHAR
            RTS     ;All done



PUTBITS_D6:         ;Display Byte bit pattern in D6
            MOVE.L  D3,-(A7)    ;Save D3
            MOVE.L  D2,-(A7)    ;Save D2
            MOVE.B  #7,D3   ;Bit indicator (7,6,5...0)
            MOVE.B  #8,D2   ;Bit count

PUTBIT1:    BTST    D3,D6
            BEQ SHOW_0
            MOVE.B  #'1',D1
            BSR PUTCHAR
            BRA NEXT_BIT
SHOW_0:     MOVE.B  #'0',D1
            BSR PUTCHAR
NEXT_BIT:   SUBQ.B  #1,D3
            SUBQ.B  #1,D2   ;8 bits total
            BNE PUTBIT1
            MOVE.L  (A7)+,D2    ;Restore D2
            MOVE.L  (A7)+,D3    ;Restore D3
            RTS

;---------------------------- MAIN ROUTINE TO PRINT A CHARACTER ON CONSOLE --------------------
------

                    ;Send ASCII character in D1
PUTCHAR:
   IFNE SIMMULATOR      ;If SIMMULATOR = 1, then send character via software int
            MOVE    #6,D0   ;Send a character (in D1), to CRT
            TRAP    #15
            RTS
   ENDC

  IFEQ SIMMULATOR       ;If SIMMULATOR = 0, then send via console I/O port (or serial port)
            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
            BNE PUTCHAR0    :Jump to simple Propeller port
            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 int
            MOVE.B  #5,D0   ;Get a character from keyboard, put in D1 (NOTE will be echoed)
            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 present
            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
            BNE SEND_SERIAL ;Ready to send
            SUB.W   #1,D2   ;Decraeae loop count
            TST.W   D2
            BNE SERIAL_OUT_STAT
            MOVE.B  #0,D1   ;Return empty just in case
SERIAL_DONE:
            MOVE.L  (A7)+,A0    ;< Restore A0
            MOVE.L  (A7)+,D2    ;< Restore D2
            MOVE.L  (A7)+,D5    ;< Restore D5
            RTS     ;Return from subroutine, char in D1

SEND_SERIAL:
            MOVE.L  #ADTA,A0    ;Point to data port of Zilog serial chip
            MOVE.B  D1,(A0)
            BRA SERIAL_DONE


;--------------------------------- SERIAL PORT INPUT CHARACTER ROUTINE ------------------------
-----
                    ;For XMODEM input
RECV:       MOVE.L  A3,-(A7)    ;> Save A3
            BSR SERIAL_IN_CORE  ;D2 will contain delay countdown
            MOVE.L  (A7)+,A3    ;< Restore A3
            RTS
                    ;For other serial input
SERIAL_IN:  MOVE.L  A3,-(A7)    ;> Save A3
            MOVE.L  #SERIAL_RETRYS,D2   ;Default number of times to try port before aborting
            BSR SERIAL_IN_CORE
            MOVE.L  (A7)+,A3    ;< Restore A3
            RTS

SERIAL_IN_CORE:
            MOVE.L  #ACTL,A3    ;Point to Control port of Zilog serial chip
            MOVE.W  #$5,D1
            MOVE.B  D1,(A3) ;Lower RTS line
            MOVE.W  #$EA,D1
            MOVE.B  D1,(A3)
            nop
            nop
            nop
            nop
SERIAL_IN2: MOVE.B  (A3),D1
            AND.B   #$01,D1
            CMP.B   #$01,D1
            BEQ SERIAL_IN3  ;Get serial data
            SUB.L   #1,D2
            TST.W   D2
            BNE SERIAL_IN2
            MOVE.B  #0,D2   ;Return with 0 in D2 if timeout
            MOVE.B  #0,D1   ;Return with 0 in D1 if timeout
```

```
            RTS

SERIAL_IN3: MOVE.L  #ADTA,A3    ;Point to data port of Zilog serial chip
            MOVE.B  (A3),D1 ;Get byte of data, put in D1
            MOVE.B  #$FF,D2 ;Return with FF in D2 if all is OK
            RTS


;---------------------------- MAIN ROUTINE TO PRINT A STRING ON CONSOLE ----------------------
---

PRINT_STRING:       ;Print string up to terminating \0
            MOVE.B  (A2)+,D1
            TST.B   D1
            BEQ PRINT_DONE
            BSR PUTCHAR
            BRA PRINT_STRING
PRINT_DONE: RTS


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 with 0 (or CR)
SPEAK_STRING:
            MOVE.B  (A2)+,D1
            CMP.B   #0,D1
            TST.B   D1
            BEQ SPEAK_DONE
            CMP.B   #CR,D1
            BEQ SPEAK_DONE
            BSR SPEAKOUT
            BRA SPEAK_STRING
SPEAK_DONE: MOVE.B  #CR,D1  ;Flush speaker que
            BRA SPEAKOUT
            RTS



TEST_SERIAL:
            LEA SERIAL_TEST_MSG,A2  ;Will show mesage explaining the test
            BSR PRINT_STRING
TEST_SERIAL1:
            BSR SERIAL_IN
            CMP.B   #ESC,D1 ;If ESC return to main loop
            BEQ TEST_SERIAL_DONE
            BSR SERIAL_OUT
            BRA TEST_SERIAL1
TEST_SERIAL_DONE:
            LEA SERIAL_TEST_DONE_MSG,A2 ;Will show mesage explaining the test
            BSR PRINT_STRING
            BRA loop    ;Back to start for next command


SERIAL_INITILIZE_A:
            MOVE.L  #ACTL,A2    ;ZILOG SCC base A port
            MOVE.L  #14,D2  ;Byte count (14), for below
            LEA SCCINIT_A,A3    ;Start of SCCINIT table
SCC_1:      MOVE.B  (A3)+,D5    ;Table of Zilog SCC Initilization values
            MOVE.B  D5,(A2) ;Program the SCC Channel B (A1,A3 or 10,12H) for 19K Baud
            SUB.B   #1,D2   ;All 14 values
            TST.B   D2
            BNE SCC_1
            RTS


SERIAL_INITILIZE_B:
            MOVE.L  #BCTL,A2    ;ZILOG SCC base B port
```

```
            MOVE.L  #14,D2  ;Byte count (14), for below
            LEA SCCINIT_B,A3    ;Start of SCCINIT table
SCC_2:      MOVE.B  (A3)+,D5    ;Table of Zilog SCC Initilization values
            MOVE.B  D5,(A2) ;Program the SCC Channel B (A1,A3 or 10,12H) for 19K Baud
            SUB.B   #1,D2   ;All 14 values
            TST.B   D2
            BNE SCC_2
            RTS


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

ctable          dc.l    MEM_MAP ;A      ;Display Memory Map
                dc.l    ERR ;B
                dc.l    XMODEM_BIN  ;C      ;Upload an XModem .bin file
                dc.l    DISPLAY_RAM ;D      ;Display Memory contents (Read RAM in Bytes)
                dc.l    ECHO_ASCII  ;E      ;Echo ASCII keyboard character to Console
                dc.l    FILL_RAM    ;F      ;Fill memory contents
                dc.l    GOTO_RAM    ;G      ;Jump to a ADDRESS location
                dc.l    HEX_MATH    ;H      ;Add & Subtract two Hex numbers
                dc.l    ERR ;I      ;Put CMOS-RTC Time & Date on CRT
                dc.l    ERR ;J
                dc.l    SHOW_MENU   ;K      ;Display this menu
                dc.l    TEST_INTS   ;L      ;Test Interrupt hardware
                dc.l    MOVE_RAM    ;M      ;Move memory
                dc.l    MY_IDE  ;N      ;Sub-menu to test/diagnose IDE Board
                dc.l    ERR ;O
                dc.l    ERR ;P
                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 & JUMP
to it
                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    ERR     ; "C"  LOAD CPM68K (If present)
                dc.l    DISPLAY     ; "D"  Sector contents display:- ON/OFF
                dc.l    RAMCLEAR        ; "E"  Clear RAM buffer
                dc.l    FORMAT      ; "F"  Format current disk
                dc.l    ERR     ; "G"  Restore backup
                dc.l    ERR     ; "H"  Backup partition
                dc.l    NEXT_SECT       ; "I"  Next Sector
                dc.l    PREV_SECT       ; "J"  Previous sector
                dc.l    IDE_LOOP        ; "K"
                dc.l    SET_LBA     ; "L"  Set LBA value (Set Track,sector)
                dc.l    ERR     ; "M"
                dc.l    SPINDOWN        ; "N"  Power down hard disk command
                dc.l    DRIVE_ID        ; "O"  Show current Drive ID
                dc.l    ERR     ; "P"
                dc.l    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    ERR     ; "T"
                dc.l    SPINUP      ; "U"  Power up hard disk command
```

```
                dc.l    N_RD_SEC        ; "V"  Read N sectors
                dc.l    WRITE_SEC       ; "W"  Write data buffer to current sector
                dc.l    N_WR_SEC        ; "X"  Write N sectors
                dc.l    COPY_AB     ; "Y"  Copy Drive A to Drive B
                dc.l    VERIFY_AB       ; "Z"  Verify Drive A:= Drive B:


                ;BOTH CONSOLE IO BOARD's SSC's are set for 19,200 BAUD
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 (for CTS bit)

                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 2.4576 MHz clock,
enable BRG
                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
                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    $06     ;Low byte 19,200 Baud <<<<<<<<<<< Note Speech synthizer
defaults to this value
                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 2.4576 MHz clock,
enable BRG
                dc.b    $0F     ;Point to WR15
                dc.b    $00     ;Generate Int with CTS going high


Signon          dc.b   CR,LF,'68000 Monitor V2.5 John Monahan 05/2/2014',CR,LF,0
Prompt          dc.b   CR,LF,'>',0
Menu            dc.b   CR,LF
                dc.b   'A=Memmap    C=XMODEM(Bin) D=Disp RAM    E=Echo        F=Fill RAM'
                dc.b   CR,LF
                dc.b   'G=Goto RAM  H=Math        I=Time        K=Menu        L=Test Ints'
                dc.b   CR,LF
                dc.b   'M=Move RAM  N=IDE Menu    Q=Port I/O    U=Serial Test S=Subs RAM'
                dc.b   CR,LF
```

```
                dc.b    'T=Type RAM  V=Verify RAM  X=Signals     Y=Patch       Z=Back to Z80'
                dc.b    CR,LF,LF,0

SMSG            dc.b    'THE 68 THOUSAND ROM MONITOR, VERSION 2.5 IS NOW ACTIVE',0

BadCmdMsg       dc.b    CR,LF,BELL,'Bad Command. Got a vlaue of ',0
ErrorMsg        dc.b    CR,LF,BELL,'Invalid entry',CR,LF,0
PortMsg         dc.b    CR,LF,'Port ',0
PortMsg2        dc.b    CR,LF,'Send to Port ',0
NotDoneMsg      dc.b    CR,LF,BELL,'Sorry code for this command has not been written
yet',CR,LF,0
HEX_Data        dc.b    CR,LF,'Sum = ',0
HEX_Data2       dc.b    'H  Difference = ',0

BUS_ERROR_MSG   dc.b    CR,LF,'Bus Error interrupt recieved',CR,LF,0
ADDRESS_ERROR_MSG   dc.b    CR,LF,'Address Error interrupt recieved',CR,LF,0
ILLEGAL_ERROR_MSG   dc.b    CR,LF,'Illegal Opcode interrupt recieved',CR,LF,0
ZERO_ERROR_MSG  dc.b    CR,LF,'Zero Error interrupt recieved',CR,LF,0
PRIVILEGE_ERROR_MSG dc.b    CR,LF,'Priviledge Error interrupt recieved',CR,LF,0
TRACE_ERROR_MSG     dc.b    CR,LF,'Trace Error interrupt recieved',CR,LF,0
SPURIOUS_INT_MSG    dc.b    CR,LF,'Spurious interrupt recieved',CR,LF,0
L1_INTERRUPT_MSG    dc.b    CR,LF,'L1 (or NMI) interrupt recieved',CR,LF,0
L2_INTERRUPT_MSG    dc.b    CR,LF,'L2 interrupt recieved',CR,LF,0
L3_INTERRUPT_MSG    dc.b    CR,LF,'L3 interrupt recieved',CR,LF,0
L4_INTERRUPT_MSG    dc.b    CR,LF,'L4 interrupt recieved',CR,LF,0
L5_INTERRUPT_MSG    dc.b    CR,LF,'L5 interrupt recieved',CR,LF,0
L6_INTERRUPT_MSG    dc.b    CR,LF,'L6 interrupt recieved',CR,LF,0
L7_INTERRUPT_MSG    dc.b    CR,LF,'L7 interrupt recieved',CR,LF,0
INT_ERR_MSG     dc.b    CR,LF,'Undefined interrupt recieved',CR,LF,0
TRAPS_ERR_MSG   dc.b    CR,LF,'Undefined TRAP interrupt recieved',CR,LF,0
INTS_DONE_MSG   dc.b    CR,LF,'Interrupt vectors in RAM (0-400H) initilized',CR,LF,0

SIGNALS_MSG     dc.b    CR,LF,'Put CPU in hardware loop to test (pDBIN or pWR*)'
                dc.b    CR,LF,'Enter test RAM Location: ',0
Menu_1or2_MSG   dc.b    CR,LF,'Enter 1=pDBIN, 2=pWR* :',0
SIG_STARTED_MSG dc.b    CR,LF,'Signal test loop started......(Hit Reset to Abort)',0
ECHO_MSG        dc.b    CR,LF,'Will echo each ASCII character typed on keyboard. ESC to
abort',CR,LF,0
PATCH_MSG       dc.b    CR,LF,'Moving Code at 4000H-9000H to F4000H, Then jump to that
location',CR,LF,0
SERIAL_TEST_MSG dc.b    CR,LF,'Characters entered via serial IN port will appear on serial OUT
port.'
                dc.b    CR,LF,'Press ESC to abort',CR,LF,0
SERIAL_TEST_DONE_MSG dc.b   CR,LF,'Serial test done. Returning to 68000 Monitor.',CR,LF,0

IDE_SIGNON0     dc.b    CR,LF,LF,'IDE HDisk Test Menu Routines.',CR,LF,0
IDE_SIGNON1     dc.b    'A=Select Drive A  B=Select Drive B  C=Boot CPM68K D=Set Sec Display
Mode to On',CR,LF,0
IDE_SIGNON2     dc.b    'A=Select Drive A  B=Select Drive B  C=Boot CPM68K D=Set Sec Display
Mode to Off',CR,LF,0
IDE_SIGNON3     dc.b    'E=Clear Sec Buff  F=Format Disk     I=Next Sec    J=Previous Sec',CR,LF
                dc.b    'L=Set LBA Value   N=Power Down      O=Disk ID     Q=LBA Display
Test',CR,LF
                dc.b    'R=Read Sector     S=Seq Sec Rd      U=Power Up    V=Read N
Sectors',CR,LF
                dc.b    'W=Write Sector    X=Write N Sectors X=Copy A->B   Z=Verify A=B',CR,LF
                dc.b    '(ESC) Back to Main Menu',CR,LF
                dc.b    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,'Initilizing of First Drive failed. Aborting
Command.',BELL,CR,LF,LF,0
```

```
INIT_2_ERROR    dc.b   CR,LF,'Initilizing of Second Drive failed. (Possibly not
present).',BELL,CR,LF,LF,0
BAD_DRIVE:      dc.b   CR,LF,'First Drive ID Infornmation appears invalid. '
                dc.b   '(Drive possibly not present).',CR,LF
                dc.b   'Aborting Command.',BELL,CR,LF,LF,0

msgmdl          dc.b   CR,LF,'Drive/CF Card Information:-',CR,LF
                dc.b   'Model: ',0
msgsn           dc.b   'S/N:   ',0
msgrev          dc.b   'Rev:   ',0
msgcy           dc.b   'Cylinders: ',0
msghd           dc.b   ', Heads: ',0
msgsc           dc.b   ', Sectors: ',0
msgCPMTRK       dc.b   'CPM TRK = ',0
msgCPMSEC       dc.b   ' CPM SEC = ',0
msgLBA          dc.b   '  (LBA = 00',0
MSGBracket      dc.b   ')',0
H_MSG_CRLF      dc.b   'H',CR,LF,0

NotDoneYet      dc.b   CR,LF,'Command 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
SEC_RW_ERROR    dc.b   'Drive Error, Status Register = ',0
ERR_REG_DATA    dc.b   'Drive Error, Error Register = ',0
ENTERRAM_SECL   dc.b   'Starting sector number,(xxH) = ',0
ENTERRAM_HEAD   dc.b   'Starting HEAD number,(xxH) = ',0
ENTERRAM_FTRKL  dc.b   'Enter Starting Track number,(xxH) = ',0
ENTERRAM_TRKL   dc.b   'Track number (LOW byte, xxH) = ',0
ENTERRAM_TRKH   dc.b   'Track number (HIGH byte, xxH) = ',0
ENTER_HEAD      dc.b   'Head number (01-0f) = ',0
ENTER_COUNT     dc.b   'Number of sectors to R/W (xxH) = ',0
ENTERRAM_DMA    dc.b   'Enter DMA Adress (Up to 5 digits, xxxxxH) = ',0
OVER_COUNT_10   dc.b   CR,LF,'1 & 9 sectors. Only!',CR,LF,0
OVER_COUNT_19   dc.b   CR,LF,'1 & 18 sectors. Only!',CR,LF,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   'Fill disk sectors of Disk [A] with 0E5H',0
FORMAT_MSG_B    dc.b   'Fill disk sectors of Disk [B] with 0E5H',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
FILL_MSG        dc.b   CR,LF,'Sector buffer area cleared to 0000....',0
READN_MSG       dc.b   CR,LF,'Read N sectors from disk.',0
WRITEN_MSG      dc.b   CR,LF,'Write N sectors to disk.',0
DiskCopyMsg     dc.b   CR,LF,'Copy CPM Partition on Drive A to Drive B (Y/N)? ',0
DiskVerifyMsg   dc.b   CR,LF,'Will verify CPM Partition on Drive A to Drive B.',0
CopyDone        dc.b   CR,LF,'Disk Copy Done.',0
VERIFY_ERR      dc.b   CR,LF,BELL,'Verify Error. ',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]',CR,LF,0
CURRENT_MSG_B   dc.b    '   Current Drive = [B]',CR,LF,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',CR,LF,0
IDE_SEL_B       dc.b    CR,LF,'Selecting IDA Drive B',CR,LF,0

MODEM_S_SIGNON  dc.b    CR,LF,'Load Motorola .s File from a PC to RAM using the S100Computers IO
Board',CR,LF
                dc.b    'Zilog SCC Ports A1H & A3H. Requires RTS & CTS, 38,400 Baud.',CR,LF,0

MODEM_BIN_SIGNON    dc.b   CR,LF,'Load a .bin File from a PC to RAM using the S100Computers IO
Board',CR,LF
                dc.b    'Zilog SCC Ports A1H & A3H. Requires RTS & CTS, 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
ERRSOH          dc.b    'H Received',CR,LF,'Did not get Correct SOH',CR,LF,0
MODEM_ERR2      dc.b    CR,LF,'Bad Sector # in Header',CR,LF,0
MODEM_ERR3      dc.b    CR,LF,'Bad Checksum for Sector',CR,LF,0
TOUTM           dc.b    CR,LF,'Timeout! Error count = ',0
QUITM           dc.b    CR,LF,'+++ MULTIPLE ERRORS ENCOUNTERED +++'
                dc.b    CR,LF,'Type Q To Quit, R To Retry:',0
RAM_MSG         dc.b    'H.  If OK will write to RAM location ',0
FINISH_MSG      dc.b    'Down loading of file complete.  No Errors',0
TRANS_DONE      dc.b    CR,LF,LF,'Data Transfer Is Complete',CR,LF,LF,0
BAD_HEADER      dc.b    CR,LF,'Unable to get a valid file header!',CR,LF,0
START_B_POINTER_MSG dc.b   CR,LF,'Will load .bin file data starting at RAM location ',0
START_S_POINTER_MSG dc.b   CR,LF,'Will load Motorola .s file data at RAM location ',0
ABORT_MSG       dc.b    CR,LF,LF,'Invalid Character or Program Aborted',CR,LF,0
H_MSG           dc.b    'H',0



;-------------------------------------------------------------------------------------------
-----------
   IFNE ROM_CODE        ;If ROM based Code
            ORG $00FD80F0   ;Start of Supervisor Stack area, (FD80F0H down to FD8000H)
SUPERVISOR_STACK
            DC.W    0
            ORG $00FD8100   ;Start OF RAM Area on CPU Board, (beginning at FD8100 upwards)
   ENDC

   IFNE S100_TEST       ;If S100 RAM based Code
            ORG $00FD80F0   ;Start of Supervisor Stack area, (FD80F0H down to FD8000H)
SUPERVISOR_STACK
            DC.W    0
            ORG $00FD8100   ;Start of RAM Area on CPU Board, (beginning at FD8100 upwards)

   ENDC

   IFNE SIMMULATOR      ;If SIMMULATOR
            ORG $90F0   ;Start of Supervisor Stack area (90F0 down to 9000H)

SUPERVISOR_STACK
            DC.W    0
            ORG $9100   ;Start OF RAM Area on CPU Board, (beginning at FD8100 upwards)
    ENDC
```

```
BeginRAM:
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
RAM_DRIVE_TRK   dc.w     0
RAM_DRIVE_SEC   dc.w     0

DISPLAY_FLAG    dc.b     0  ;Flag it indicate if detail sector info is to be displayed
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
```