```
;-----------------------------------------------------------
;    Program Number: 68030 Monitor for S100Computers.com board
;    Written by     : John Monahan
;    Date Created   : 11/11/2011, updated to 68030 on 8/26/2017
;    Description    : Basic monitor for 68030 S-100 board
;
;   'A=Memmap    C=XMODEM    D=Display RAM  E=Echo Keyboard  F=Fill (Byte)'
;   'G=Goto RAM  H=Math      I=Test Ints    J=Test RAM       K=This Menu'
;   'M=Move RAM  N=IDE Menu  Q=Port I/O     S=Subs RAM       T=Type RAM'
;   'U=Serial    V=Verify    W=Fill (Word)  X=Signals        Y=Exec in RAM '
;   '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 00
;       V2.4    04/30/2014      ;Cleanup console I/O routines, add serial port I/O
;       V2.5    05/02/2014      ;Added XMODEM .bin file download capabilities over serial port
;       V2.6    06/12/2014      ;Corrected QO/QI port bug
;       V2.7    08/26/2017      ;Used with 68030 Board
;       V2.8    08/26/2017      ;MEM MAP for > 16M. Abort with ESC for Display RAM and MEM Map
;       V2.9    09/10/2017      ;Fill RAM with WORD or BYTE option, RAM test, RAM Map update, Port
;       V3.0    10/03/2017      ;Rearrange menu so items are approximately the same as for the 803
;       V3.1    11/13/2017      ;Cleanup of some minor issues.
;       V3.2    11/16/2017      ;Serial port test now also outputs to Propeller Console I/O Board.
;       V3.3    21/01/2021      ;IDE Fixes
;       V3.4    2/6/2021        ;Fix Sec Write error, Drive ID, changed IDE menu options, and many
;       V3.42   2/8/2021        ;Corrected Drive ID display of Serial #, Cyl,Head,Sector count
;
;
;   >>>>>> Please note the IDE Board Diagnostic section is NOT complete. All the code is there
;   >>>>>> it just needs to be checked out/debugged. It is from the 8086 MSDOS ROM BIOS code "conv
;
;   >>>>>> The stack is currently set to 0000FF00H in RAM.  So we assume only a 64K RAM board.
;   >>>>>> It can be located anywhere in teh 3GB address space.  Also the IDE board needs some RAM
;   >>>>>> buffers.  Currently they start at 007FE000H (BeginRAM).  See the end section of this mo
;
;Programming a Wellon VP-290 with GLS29EE010 EEPROMS.
;Assemble and make a S68 file (Project Menu for EASy68K)
;For "From File Address(Hex) enter FC0000 (Note "To Buffer Address (HEX) is 0)
;For "Auto Format Detect" use Motorola S
;Remember if you use the XMODEM command to load a file in RAM it must be in .bin format (not S68).
;

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


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


KEYSTAT         EQU     $FFFF0000
KEYIN           EQU     $FFFF0001       ;Console input port. Normally the Propeller Driven S-100 C
KEYOUT          EQU     $FFFF0001       ;Console output port. Normally the Propeller Driven S-100
```

```
CMOS_VALID       EQU     $FFFF000D       ;To check DS12887 CMOS chip is present and OK (Note AT-BIO
CMOS_REGA        EQU     $FFFF000A       ;CMOS REGISTER A

TIMER            EQU     $FFFF0040       ;Base port of 8254
TIM_CTL          EQU     $FFFF0043
COUNTS_SEC       EQU     $18
COUNTS_MIN       EQU     1092
COUNTS_HOUR      EQU     $07             ;Seems this value is used with AT/CMOS chip (was 65543 on

UPDATE_TIMER     EQU     $80

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


;-------------- 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     $FFFF0030       ;lower 8 bits of IDE interface
IDEportB         EQU     $FFFF0031       ;upper 8 bits of IDE interface
IDEportC         EQU     $FFFF0032       ;control lines for IDE interface
IDECtrlPort      EQU     $FFFF0033       ;8255 configuration port
IDEDrivePort     EQU     $FFFF0034       ;To select the 1st or 2nd CF card/drive

IDE_Reset_Delay EQU      $20             ;Time delay for reset/initilization (~66 uS, with 8MHz 808

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
```

```
        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.L  #$4,D2

        MOVE.B  #BLANK,D1
        BSR     PUTCHAR

START1: MOVE.L  (A3),D1                 ;Is there RAM/ROM there
        NOT.L   D1
        NOT.L   (A3)                    ;See if we can flip bits
        MOVE.L  (A3),D6
        CMP.L   D1,D6
        BNE     NOT_RAM
        NOT.L   (A3)                    ;Put back origional data
        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 ROM if FFFFFFFF's!
        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 with total data display yet
        BLE     LOOP

        BSR     GETSTAT         ;Is there a keyboard character ready
        BEQ     PARMS_OK5
        BSR     GETCHAR         ;Get a keyboard character
        CMP.B   #ESC,D1         ;ESC to abort test
        BNE     PARMS_OK5
        BSR     CRLF
        BRA     LOOP

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


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

FILL_RAM_W:                     ;W Command. Fill RAM with one WORD 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_OKW
        MOVE.L  A3,A5           ;Else swap values
        MOVE.L  A4,A3
        MOVE    A5,A4

FILL_OKW:       ADD.L   #2,A4   ;End + 1
        BSR     GETLONG_D7      ;get Hex value in D7 (0-FFFF)
        CMP.B   #CR,D2          ;Is it valid
        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
        MOVE.L  #$FFFF0000,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.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  (test for example 00FC0008H)
        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!



RUN_AT:
                                ;J Command . Execute code in OTT RAM test (IO back and forth with
        LEA     RUN_AT_MSG,A2   ;"To test running code in OTT RAM"
        BSR     PRINT_STRING

        BSR     GETLONG_D7      ;Go to address in D7  (test for example 00FC0008H)
        CMP.B   #CR,D2          ;Is it valid
        BNE     ERROR
        MOVE.L  D7,A3           ;Save in A3
        MOVE.L  D7,A4
        MOVE.L  (TEST_OTT_RAM),(A3)+
        MOVE.L  (TEST_OTT_RAM+4),(A3)+
        MOVE.L  (TEST_OTT_RAM+8),(A3)+
        MOVE.L  (TEST_OTT_RAM+12),(A3)+
        MOVE.L  (TEST_OTT_RAM+16),(A3)+
        BSR     CRLF            ;New line
        MOVE.L  #KEYSTAT,A0
        MOVE.L  #KEYOUT,A1
        JMP     (A4)            ;That's all there is to it! JUMP to it.

 TEST_OTT_RAM:
        MOVE.B  (A0),D5         ;Check CRT status is ready to recieve character
        AND.B   #$04,D5
        TST.B   D5
        BEQ     TEST_OTT_RAM
        MOVE.B  #$33,(A1)       ;Output ASCII (in D1) to hardware port 01H
        BRA     TEST_OTT_RAM

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

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  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)+


        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
        MOVE.L  D5,A3

        MOVE.B  #00,(A2)                ;If we use use TMA line #1 to switch in/out the 68K board
        NOP                             ;<-- 68K Is held in HALT mode here until released again by
        NOP
        NOP
        NOP
        MOVE.B  (A3),D5         ;If we use TMA line #0 to switch in/out the 68K board
        NOP                     ;For testing using the default GAL0 code if we input from port EDH
        NOP                     ;If we input a second time we will switch control back to the Z80
        NOP                     ;Note the "Proper" Z80 Master command is the 'B' command with jump
        NOP                     ;If you do it this way the Intel CPU's can also reside in teh bus
        BRA     LOOP

;--------------------------------------------------------------------------------------------------
SIGNALS:                                ;Setup hardware signal tests to look at S-100 signals pDBI
        LEA     SIGNALS_MSG,A2          ;Put CPU in hardware loop to test (pDBIN or pWR*), Enter t
        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
        LEA     RAM_Test_Location,A2    ;Show location (in A5)
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     H_MSG,A2                ; 'H',0
        BSR     PRINT_STRING
TEST_RAM2:
        CMP.L   A5,A4                   ;Are we done yet
        BGE     TEST_RAM0


                                        ;:::::::::::::::
        LEA     CHK_FILL_JMSG,A2        ;Inform test,  Check zero fill,  then fill with 55H
        BSR     PRINT_STRING
        MOVE.L  A3,A5                   ;<<<-------  A5 will contain RAM location under test
TEST_RAM3:
        CMP.B   #$0,(A5)                ;Is the RAM location 0
        BEQ     TEST_RAM3A
        BSR     RAM_ERROR_0             ;Show Error and current value

TEST_RAM3A:
        MOVE.B  #$55,(A5)+              ;Fill everything with 55H's
        MOVE.W  A5,D2                   ;;Time for an ESC check yet?, will check for every xxxx000
        TST.W   D2
        BNE     TEST_RAM5
        BSR     GETSTAT                 ;Is there a keyboard character ready
        BEQ     TEST_RAM4
        BSR     GETCHAR                 ;Get a keyboard character
        CMP.B   #ESC,D1                 ;ESC to abort test
        BNE     TEST_RAM4
        BRA     BEGIN                   ;Finished RAM test
TEST_RAM4:
        LEA     RAM_Test_Location,A2    ;Show location (in A4)
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     H_MSG,A2                ;'H',0
        BSR     PRINT_STRING
TEST_RAM5:
        CMP.L   A5,A4                   ;Are we done yet
        BGE     TEST_RAM3

                                        ;::::::::::::::::::::::::
        LEA     CHK_WORD_JMSG,A2        ;'Checking RAM was filled with BYTE 55H, replacing with WO
        BSR     PRINT_STRING
        MOVE.L  A3,A5                   ;<<<-------  A5 will contain RAM location under test
TEST_RAM6:
        CMP.B   #$55,(A5)               ;Is the RAM location 55H
        BEQ     TEST_RAM6A
        BSR     RAM_ERROR_55            ;Show Error and current value
TEST_RAM6A:
        MOVE.W  #$1234,(A5)+            ;Fill everything with 1234H's
        MOVE.W  A5,D2                   ;;Time for an ESC check yet?, will check for every xxxx000
        TST.W   D2
        BNE     TEST_RAM8
        BSR     GETSTAT                 ;Is there a keyboard character ready
        BEQ     TEST_RAM7
        BSR     GETCHAR                 ;Get a keyboard character
        CMP.B   #ESC,D1                 ;ESC to abort test
        BNE     TEST_RAM7
        BRA     BEGIN                   ;Finished RAM test
TEST_RAM7:
        LEA     RAM_Test_Location,A2    ;Show location (in A4)
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     H_MSG,A2                ;'H',0
        BSR     PRINT_STRING
TEST_RAM8:
        CMP.L   A5,A4                   ;Are we done yet
        BGE     TEST_RAM6
        LEA     RAM_Error1_Location,A2  ;Show location of error (in A4)
RAM_ERROR0:
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     ShowValueMsg,A2         ;'H  RAM Byte value =   ' (in A4)
        BSR     PRINT_STRING
        MOVE.B  (A5),D6
        BSR     PUTBYTE_D6
        LEA     H_MSG_CRLF,A2           ; 'H',CR,LF,0
        BSR     PRINT_STRING
        RTS

RAM_ERROR_55:
        LEA     RAM_Error2_Location,A2  ;Show location of error (in A4)
        BRA     RAM_ERROR0

RAM_ERROR_W:
        LEA     RAM_Error3_Location,A2  ;Show location of error (in A4)
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     ShowValueMsg1,A2        ;'H  RAM Word value =   ' (in A4)
        BSR     PRINT_STRING
        MOVE.W  (A5),D6
        BSR     PUTWORD_D6
        LEA     H_MSG_CRLF,A2           ; 'H',CR,LF,0
        BSR     PRINT_STRING
        RTS

RAM_ERROR_L:
        LEA     RAM_Error4_Location,A2  ;Show location of error (in A4)
        BSR     PRINT_STRING
        MOVE.L  A5,D7
        BSR     PUTLONG_D7
        LEA     ShowValueMsg2,A2        ;'H  RAM Long value =   ' (in A4)
        BSR     PRINT_STRING
        MOVE.L  (A5),D7
        BSR     PUTLONG_D7
        LEA     H_MSG_CRLF,A2           ; 'H',CR,LF,0
        BSR     PRINT_STRING
        RTS

;*****************************************************************************************************
;
;       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: BSR     CLEAR_ID_BUFFER         ;Clear ID Buffer
        BSR     CRLF

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

        LEA     INIT_1_ERROR,A2
        BSR     PRINT_STRING
        BRA     LOOP

INIT1_OK:
        BSR     SEL_DRIVE_B             ;Select the second Drive/CF card (Do not mess with CPM Dri
        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
        CMP.B   #'A',D1
        BLT     ERR
        CMP.B   #'Z',D1
        BGT     ERR
        SUBI.B  #'A',D1
        EXT.W   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)

        CMP.B   #'A',D1
        BLT     ERR
        CMP.B   #'Z',D1
        BGT     ERR
        SUBI.B  #'A',D1
        EXT.W   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
        BSR     SEL_DRIVE_A
        BRA     IDE_LOOP                ;Back to IDE Menu

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

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

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


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

```
;-------------- 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: BSR     CRLF
        LEA     msgrd,A2                ;Sector read OK
        BSR     PRINT_STRING

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



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

        BSR     WRITESECTOR             ;Will write whatever is in the IDE_Buffer

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

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


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

FILL_SEC:
        LEA     FILL_BYTE_MSG,A2        ;Enter sector Fill byte
        BSR     PRINT_STRING
        BSR     GETBYTE_D7              ;Get data in D7 (0-FF)
        CMP.B   #ESC,D2                 ;If ESC then we are done
        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#

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

        BSR     DISPLAY_SECTOR

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

NO_WAIT:
```

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

N_WR_SEC:
        LEA     WILL_WR_MSG,A2          ;Enter RAM location for the start of sector writes
        BSR     PRINT_STRING

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

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

        LEA     CONFIRM_WR_MSG,A2       ;Are you sure?
        BSR     PRINT_STRING
        BSR     GETCHAR
        BSR     TOUPPER
        CMP.B   #'Y',D1
        BEQ     NextWSec1
        BRA     IDE_LOOP                ;Back to IDE Menu
NextWSec1:
        BSR     CRLF

NextWSec:
        BSR     CRLF
        MOVE.L  (RAM_DMA_STORE),D1      ;DMA initially to IDE_Buffer
        MOVE.L  D1,(RAM_DMA)
        MOVE.L  D1,D7
        BSR     PUTLONG_D7              ;Show current address
        LEA     WRITEN_MSG,A2           ;' ----> ',0
        BSR     PRINT_STRING
        BSR     DISPLAY_POSITION        ;Display current Track,sector


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

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

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

NEXT_SEC_NWR:
        BSR     GET_NEXT_SECT
        BEQ     NextWSec

        LEA     AT_END_MSG,A2           ;Tell us we are at end of disk
        BSR     PRINT_STRING
DoneWSec:
        BSR     CRLF
        MOVE.B  #0,(RAM_SEC)            ;Back to CPM sector 0
        MOVE.B  #0,(RAM_TRK)
        MOVE.B  #0,(RAM_TRK+1)
        BSR     WR_LBA                  ;Update LBA on drive
        BSR     CRLF
        BRA     IDE_LOOP                ;Back to IDE Menu
        BSR     PRINT_STRING
FORMAT_DONE:
        MOVE.B  #0,D1                   ;Login drive A:
        BSR     SELECT_DRIVE
        MOVE.B  D1,(CURRENT_IDE_DRIVE)
        MOVE.B  #0,(RAM_SEC)            ;Back to CPM sector 0
        MOVE.B  #0,(RAM_TRK)
        MOVE.B  #0,(RAM_TRK+1)
        BSR     WR_LBA                  ;Update LBA on drive
        BSR     CRLF
        BRA     IDE_LOOP                ;Back to IDE Menu



;---------------- 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
        MOVE.B  #0,(RAM_TRK+1)
        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:
        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           ;'Verify Error.
        BSR     PRINT_STRING
        BSR     SHOW_TRACK_SEC          ;Show current location of error
        BRA     VERIFYT1                ;Ask for a continue message here

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


;-------------------- Back to parent 68K Monitor commands

QUIT_IDE:
        BRA     LOOP                    ;Back to main Menu






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

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


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

        LEA     ENTERRAM_TRKL,A2        ;Enter low byte track number
        BSR     PRINT_STRING
        BSR     GETBYTE_D7              ;Get 8 bit value (2 digits) to D7
        LEA     H_MSG_CR,A2
        BSR     PRINT_STRING
        RTS



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

SF172:  MOVE.L  A2,D7
        BSR     PUTLONG_D7              ;Show current address
        MOVE.B  #BLANK,D1
        BSR     PUTCHAR
        BEQ     AT_DISK_END
        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
        BGE.B   Sloop3
        MOVE.B  #'.',D6
        BRA     Sloop4
Sloop3: CMP.B   #'~',D6
        BLE.B   Sloop4
        MOVE.B  #'.',D6
Sloop4: MOVE.B  D6,D1
        BSR     PUTCHAR
        SUBQ.B  #1,D4
        BNE     Sloop2
        BSR     CRLF

        SUBQ.B  #1,D3
        BNE     SF172
        RTS




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

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

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

```
MoreError:                                  ;Get here if bit 0 of the status register indicted a probl
        MOVE.B  #REGerr,D5                  ;Get error code in REGerr
        BSR     IDErd8D
        MOVE.B  D4,D6

        BTST    #4,D4                       ;Not found bit
        BEQ     NOTE4
        LEA     SEC_NOT_FOUND,A2
        BSR     PRINT_STRING
        BRA     DONEERR

NOTE4:  BTST    #7,D4                       ;Bad block bit
        BEQ     NOTE7
        LEA     BAD_BLOCK,A2
        BSR     PRINT_STRING
        BRA     DONEERR

NOTE7:  BTST    #6,D4                       ;Uncorrectable bit
        BEQ     NOTE6
        LEA     UNRECOVER_ERR,A2
        BSR     PRINT_STRING
        BRA     DONEERR

NOTE6:  BTST    #2,D4                       ;Invalid command bit
        BEQ     NOTE2
        LEA     INVALID_CMD,A2
        BSR     PRINT_STRING
        JMP     DONEERR

NOTE2:  BTST    #1,D4                       ;Not found bit
        BEQ     NOTE1
        LEA     TRK0_ERR,A2
        BSR     PRINT_STRING
        JMP     DONEERR

NOTE1:  LEA     UNKNOWN_ERROR1,A2
        BSR     PRINT_STRING

DONEERR:                                    ;Display Byte bit pattern in D6
        BSR     PUTBITS_D6                  ;Show error bit pattern
        BSR     CRLF
        MOVE.W  #1,D0                       ;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.
;================================================================================================

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

        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
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,hea
;       MOVE.B  #%10100000,D4               ;For Trk,Sec,head (non LBA) use 10100000 (This is the mode
                                            ;Note. Cannot get LBA mode to work with an old Seagate Med
                                            ;have to use the non-LBA mode. (Common for old hard disks)
        MOVE.B  #REGshd,D5                  ;00001110,(0EH) for CS0,A2,A1,
        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)+,IDEportB

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

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

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


                                        ;Write the logical block address to the drive's 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 Hig
WR_LBA:                                 ;numbers etc).
        MOVE.B  (RAM_SEC),D4            ;LBA mode, Low sectors go directly
;       ADDQ.B  #$1,D4                  ;Sectors are numbered 1 -- MAXSEC (even in LBA mode)
        MOVE.B  D4,(DISPLAY_SEC)        ;For Diagnostic Diaplay Only
        MOVE.B  #REGsector,D5           ;Send info to drive
        BSR     IDEwr8D                 ;Write to 8255 A Register
                                        ;Note: For drive we will have 0 - MAXSEC sectors only

        MOVE.B  (RAM_TRK),D4
        MOVE.B  D4,(DISPLAY_TRK)        ;Send Low TRK#
        MOVE.B  #REGcylinderLSB,D5
```

```
                                        ;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  #$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


IDE_ERR:        CMP.B   #CR,D1          ;If CR just return
        BEQ     IDE_LOOP
        LEA     BadIDECmdMsg,A2         ;Non menu selection
        BSR     PRINT_STRING
        BSR     CRLF
        BRA     IDE_LOOP                ;Back to start for next command


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

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

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

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

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

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


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

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

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

        OR.B    #IDEwrline,IDEportC             ;Assert write pin

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

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


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
```

```
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:
        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 PUTCHAR0
;       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


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

                                            ;A0 has console status port, A1 has console data port
GETCHAR:
        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
        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  #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_B      ;F  ;Fill memory contents with a Byte
                dc.l    GOTO_RAM        ;G  ;Jump to a ADDRESS location
                dc.l    HEX_MATH        ;H  ;Add & Subtract two Hex numbers
                dc.l    TEST_INTS       ;I  ;Test Interrupt hardware
                dc.l    TEST_RAM        ;J  ;Extensive OTT RAM test
                dc.l    SHOW_MENU       ;K  ;Display this menu
                dc.l    ERR             ;L
                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    FILL_RAM_W      ;W  ;Fill memory contents with a word
                dc.l    SIGNALS         ;X  ;Setup for hardware S-100 bus signals test
                dc.l    RUN_AT          ;Y  ;Execute code in OTT RAM test
                dc.l    JMP_Z80         ;Z  ;Return back to Z80 master


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


                                        ;BOTH CONSOLE IO BOARD's SSC's are set for 38,400 BAUD
SCCINIT_A:      dc.b    $04             ;Point to WR4
```

```
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 abo
SERIAL_TEST_MSG         dc.b   CR,LF,'Enter characters on Serial Board IN port from your PC/TTY Te
                        dc.b   CR,LF,'They should echo on the Serial Board OUT port. (38400 Baud,
                        dc.b   CR,LF,'Enter ESC to abort and return back here.',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.    ',0
IDE_SIGNON1             dc.b   'A=Select Drive A   B=Select Drive B  E=Fill Sec        F=Format Di
                        dc.b   'N=Next Sec         P=Previous Sec    L=Set LBA Value   O=Disk ID',
                        dc.b   'R=Read Sector      S=Seq Sec Rd      X=Sectors to RAM  W=Write Sec
                        dc.b   'Y=RAM to Sectors   C=Copy A->B       V=Verify A=B      (ESC) Main
                        dc.b   CR,LF,'Current settings:- ',0

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

```
RAM_Error4_Location     dc.b    CR,'Error (not 12345678H) at RAM Location = ',0
RAM_Test_Done           dc.b    CR,LF,LF,'RAM test finished.',CR,LF,LF,0
ShowValueMsg            dc.b    'H  RAM Byte value = ',0
ShowValueMsg1           dc.b    'H  RAM Word value = ',0
ShowValueMsg2           dc.b    'H  RAM Long value = ',0
ZERO_FILL_JMSG          dc.b    CR,LF,'Filling RAM with Zeros. Hit ESC any time to abort',CR,LF,0
CHK_FILL_JMSG           dc.b    CR,LF,'Checking RAM was filled with BYTE 0H, replacing with BYTE 5
CHK_WORD_JMSG           dc.b    CR,LF,'Checking RAM was filled with BYTE 55H, replacing with WORD
CHK_DWORD_JMSG          dc.b    CR,LF,'Checking RAM was filled with WORD 1234H, replacing with DWO
FILLED_DWORD_JMSG       dc.b    CR,LF,'Checking RAM was filled with DWORD 12345678H.',CR,LF,0
FILL_BYTE_MSG           dc.b    CR,LF,'Enter Fill byte (+CR):-',0
WILL_RD_MSG             dc.b    CR,LF,'Enter RAM location where data will be placed (+CR) ',0
SEC_COUNT_MSG           dc.b    CR,LF,'Enter sector count (+CR) ',0
WILL_WR_MSG             dc.b    CR,LF,'Enter RAM location where data will be copied from (+CR) ',0
RUN_AT_MSG              dc.b    CR,LF,'Test running code at a valid RAM location (XXXXXXXX+CR):',0
FORMAT_STARTED_MSG      dc.b    CR,LF,'The current drive is being formatted. Use Esc to abort.',0
BadIDECmdMsg            dc.b    CR,LF,BELL,'Invalid IDE Command!',0


;-----------------------------------------------------------------------------------------------------
        ORG     $007FE000       ;<--------- NOTE ASSUMES AT LEAST 8MB OF RAM (Our 16MG RAM board w

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
DISPLAY_TRK             dc.w     0
DISPLAY_SEC             dc.w     0

RAM_SEC:                dc.b     0
RAM_TRK:                dc.w     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

        END     BEGIN
```