```
;       MS-DOS Test program for CMOS Clock on S100Computers/N8VEM board.
;
;       V0.1                ;Original version 11/19/2011
;
; The IBM-AT BIOS has  AH=2 to read time
;       Return CH = HOURS IN BCD
;              CL = Minutes in BCD
;              DH = Seconds in BCD

; The IBM-AT BIOS has  AH=3 to set time
;              CH = HOURS IN BCD
;              CL = Minutes in BCD
;              DH = Seconds in BCD

; The IBM-AT BIOS has  AH=4 to read date
;       Return CH = CENTURY IN BCD
;              CL = Year in BCD
;              DH = Month in BCD
;              DL = Day in BCD

; The IBM-AT BIOS has  AH=5 to Set date
;              CH = CENTURY IN BCD
;              CL = Year in BCD
;              DH = Month in BCD
;              DL = Day in BCD

; The IBM-AT BIOS has  AH=6 to Set Alarm
;              CH = HOURS IN BCD
;              CL = Minutes in BCD
;              DH = Seconds in BCD

; The IBM-AT BIOS has  AH=7 to Reset Alarm

BELL          EQU    07H
SPACE         EQU    20H
TAB           EQU    09H          ; TAB ACROSS (8 SPACES FOR SD-BOARD)
CR            EQU    0DH
LF            EQU    0AH
FF            EQU    0CH
ESC           EQU    1BH

TRUE          equ    1
FALSE         equ    TRUE-TRUE

MSDOS         EQU    TRUE         ;TRUE = Use MS-DOS for Console I/O, False direct to hardware

CMOS_PORT     EQU    70H          ;Base Port for CMOS Chip

KEYSTAT              EQU    0H           ;Propeller Console IO S-100 board or SD SYSTEMS VIDIO BOARD FOR
CONSOLE
KEYIN         EQU    01H          ;Console input port. Normally the Propeller Driven S-100 Console-IO
Board
KEYOUT        EQU    01H          ;Console output port. Normally the Propeller Driven S-100 Console-IO
Board


       ORG    100H

START: MOV    BX,RTC_MENU         ;Print a simple menu
       CALL   PRINT_STRING

       CALL   CI                  ;Get a character
       CMP    AL,ESC
       JZ     T1                  ;Back to DOS if ESC


   %if   MSDOS
   %else
       MOV    CL,AL
       CALL   CO                  ;Echo character if direct to hardware
   %endif
```

```
       MOV    AH,AL
       CMP    AL,'2'              ;Check for menu item.
       JNZ    NOT_RT
       JMP    READ_TIME
NOT_RT:       CMP    AL,'3'
       JNZ    NOT_ST
       JMP    SET_TIME
NOT_ST:       CMP    AL,'4'
       JNZ    NOT_RD
       JMP    READ_DATE
NOT_RD:       CMP    AL,'5'
       JNZ    NOT_SD
       JMP    SET_DATE
NOT_SD:       CMP    AL,'6'
       JNZ    NOT_SA
       JMP    SET_ALARM
NOT_SA:       CMP    AL,'7'
       JNZ    NOT_RA
       JMP    RESET_ALARM
NOT_RA:       CMP    AL,'8'
       JNZ    NOT_DUMP
       JMP    DUMP_RAM
NOT_DUMP:JMP  START



READ_TIME:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE IN PROCESS
       JNC    RTC_2A              ;GO AROUND IF OK
       JMP    ERROR               ;IF ERROR

RTC_2A CLI                        ;INTERRUPTS OFF DURING READ
       MOV    DL,-2               ;-2 goes to 0 for PORT_INC_2
       CALL   PORT_INC_2          ;SET ADDRESS OF SECONDS
       IN     AL,CMOS_PORT+1             ;Get BCD value returned
       MOV    DH, AL              ;SAVE IN DH
       CALL   PORT_INC_2          ;SET ADDRESS OF MINUTES
       IN     AL,CMOS_PORT+1             ;Get BCD value returned
       MOV    CL,AL               ;SAVE IN CL
       CALL   PORT_INC_2          ;SET ADDRESS OF HOURS
       IN     AL,CMOS_PORT+1             ;Get BCD value returned
       MOV    CH,AL               ;SAVE
       MOV    DL,0                ;SET DL TO ZERO
       STI
       CALL   DisplayTime
       CALL   CRLF
       JMP    START               ;BACK TO START

T1:    STI
       CALL   CRLF
       MOV    AH,4CH
       INT    21H                 ;Back to MS-DOS


ERROR: MOV    BX,TIME_ERROR_MSG
       CALL   PRINT_STRING
       JMP    START

SET_TIME:
       CALL   InputTime           ;Return      CH = HOURS IN BCD, CL = Minutes in BCD, DH = Seconds in BCD
       CALL   CRLF                ;No registers changed

RTC_3: CALL   UPD_IN_PR           ;CHECK FOR UPDATE IN PROCESS
       JNC    RTC_3A              ;GO AROUND IF CLOCK OPERATING
       CALL   INITIALIZE_STATUS
RTC_3A:
       CLI                        ;INTERRUPTS OFF DURING SET
       PUSH   DX                  ;SAVE for below
       MOV    DL,-2               ;-2 goes to 0 for PORT_INC_2
       CALL   PORT_INC_2          ;UPDATE ADDRESS
       MOV    AL,DH                 ;GET TIME BYTE - SECONDS
       OUT    CMOS_PORT+1,AL               ;STORE TIME BYTE
       CALL   PORT_INC_2            ;UPDATE ADDRESS
       MOV    AL,CL                 ;GET TIME BYTE - MINUTES
       OUT    CMOS_PORT+1,AL               ;STORE TIME BYTE
       CALL   PORT_INC_2            ;UPDATE ADDRESS
       MOV    AL,CH                 ;GET TIME BYTE - HOURS
       OUT    CMOS_PORT+1,AL               ;STORE TIME BYTE
       MOV    DL,0AH
       CALL   PORT_INC
       POP    DX                    ;RESTORE

       IN     AL,CMOS_PORT+1               ;GET CURRENT VALUE
       AND    AL,23H                ;MASK FOR VALID BIT POSITIONS
       OR     AL,DL                 ;GET DST BIT
       OR     AL,02H                ;TURN ON 24 HR MODE (For compatability with AT)
       PUSH   AX                    ;
       MOV    DL,0AH                ;
       CALL   PORT_INC              ;
       POP    AX                    ;
       OUT    CMOS_PORT+1,AL
       STI
       JMP    START                 ;DONE



READ_DATE:
       CALL   UPD_IN_PR
       JNC    RTC_4A
       JMP    ERROR                 ;ON ERROR

RTC_4A:       CLI                          ;INTERRUPTS OFF DURING READ
       MOV    DL,6
       CALL   PORT_INC              ;POINT TO DAY
       IN     AL,CMOS_PORT+1
       MOV    CH,AL                 ;SAVE
       CALL   PORT_INC              ;POINT TO MONTH
       IN     AL,CMOS_PORT+1
       MOV    DH,AL                 ;SAVE
       CALL   PORT_INC              ;POINT TO YEAR
       IN     AL,CMOS_PORT+1
       MOV    CL,AL                 ;SAVE
       MOV    DL,31H                ;POINT TO CENTURY BYTE SAVE AREA
       CALL   PORT_INC              ;
       IN     AL,CMOS_PORT+1               ;GET VALUE
       MOV    DL,CH                 ;GET DAY BACK
       MOV    CH,AL
       STI                          ;
       CALL   DisplayDate
       CALL   CRLF
       JMP    START                 ;FINISHED


SET_DATE:
       CALL   InputDate             ;Return      CH = HOURS IN BCD, CL = Minutes in BCD, DH = Seconds in BCD
       CALL   CRLF                  ;No registers changed
RTC_5:
       CALL   UPD_IN_PR             ;CHECK FOR UPDATE  IN PROCESS
       JNC    RTC_5A                ;GO AROUND  IF CLOCK UPDATING
       CALL   INITIALIZE_STATUS
RTC_5A:
       CLI                          ;INTERRUPTS OFF DURING SET
       PUSH   CX                    ;SAVE
       MOV    CH,DL                 ;SAVE DAY OF MONTH
       MOV    DL,5                  ;ADDRESS OF DAY OF WEEK REGISTER
       CALL   PORT_INC
       MOV    AL,0H
       OUT    CMOS_PORT+1,AL               ;LOAD ZEROS TO 'DAY OF WEEK' BYTE
       CALL   PORT_INC              ;ADDRESS OF DAY OF MONTH REGISTER
       MOV    AL,CH                 ;GET DAY OF MONTH BYTE
       OUT    CMOS_PORT+1,AL               ;STORE IT
```

```
        CALL    PORT_INC                ;ADDRESS MONTH REGISTER
        MOV     AL,DH                   ;GET MONTH BYTE
        OUT     CMOS_PORT+1,AL                  ;STORE IT
        CALL    PORT_INC                ;ADDRESS OF YEAR REGISTER
        MOV     AL,CL                   ;GET YEAR BYTE
        OUT     CMOS_PORT+1,AL                  ;STORE IT
        MOV     DL,0AH
        CALL    PORT_INC
        IN      AL,CMOS_PORT+1                  ;GET CURRENT SET ING
        AND     AL,07FH                         ;CLEAR  'SET BIT'
        OUT     CMOS_PORT+1,AL                  ;AND START CLOCK UPDATING
        POP     CX                      ;GET BACK
        MOV     DL,31H                  ;POINT TO SAVE AREA
        CALL    PORT_INC                ;
        MOV     AL,CH                   ;GET CENTURY BYTE
        OUT     CMOS_PORT+1,AL                  ;SAVE IT
        JMP     START                   ;Done


SET_ALARM:
        MOV     BX,SetAlarmMsg                  ;Print set alarm msg
        CALL    PRINT_STRING

        CALL    InputTime               ;Return      CH = HOURS IN BCD, CL = Minutes in BCD, DH = Seconds in BCD
        CALL    CRLF                    ;No registers changed
RTC_6:
        MOV     DL,0AH                  ;CHECK FOR ALARM ALREADY ENABLED
        CALL    PORT_INC                ;
        IN      AL,CMOS_PORT+1                  ;GET CURRENT SETTING OF ALARM ENABLE
        TEST    AL,20H
        JZ      RTC_6A                  ;ALARM NOT SET - GO PROCESS
        MOV     BX,AlarmBusyMsg                 ;Print set alarm msg
        CALL    PRINT_STRING
        XOR     AX,AX                   ;
        JMP     ERROR                   ;RETURN   IF ERROR
RTC_6A:
        CALL    UPD_IN_PR               ;CHECK FOR UPDATE  IN PROCESS
        JNC     RTC_6B                  ;
        CALL    INITIALIZE_STATUS
RTC_6B:
        CLI                             ;INTERRUPTS OFF DURING SET
        MOV     DL,-1
        CALL    PORT_INC_2
        MOV     AL,DH                   ;GET SECONDS BYTE
        OUT     CMOS_PORT+1,AL                  ;LOAD ALARM BYTE - SECONDS

        CALL    PORT_INC_2
        MOV     AL,CL                   ;GET MINUTES PARAMETER
        OUT     CMOS_PORT+1,AL                  ;LOAD ALARM BYTE - MINUTES
        CALL    PORT_INC_2
        MOV     AL,CH                   ;GET HOURS PARAMETER
        OUT     CMOS_PORT+1,AL                  ;LOAD ALARM BYTE - HOURS
        IN      AL,0A1H                         ;ENSURE INTERRUPT UNMASKED
        AND     AL,0FEH                         ;
        OUT     0A1H,AL                         ;
        MOV     DL,0AH
        CALL    PORT_INC
        IN      AL,CMOS_PORT+1                  ;GET CURRENT VALUE
        AND     AL,07FH                         ;ENSURE SET BIT TURNED OFF
        OR      AL,20H                  ;TURN ON ALARM ENABLE
        PUSH    AX                      ;
        MOV     DL,0AH                  ;
        CALL    PORT_INC                ;
        POP     AX                      ;
        OUT     CMOS_PORT+1,AL                  ;ENABLE ALARM

        MOV     BX,AlarmSetMsg                  ;Print set alarm msg
        CALL    PRINT_STRING
        JMP     START

RESET_ALARM:
```

```
RTC_7:
       CLI                       ;INTERRUPTS MASKED DURING RESET
       MOV    DL,0AH
       CALL   PORT_INC
       IN     AL,CMOS_PORT+1             ;GET STATUS BYTE
       AND    AL,57H             ;TURN OFF ALARM ENABLE
       PUSH   AX                 ;SAVE
       MOV    DL,0AH
       CALL   PORT_INC           ;
       POP    AX                 ;
       OUT    CMOS_PORT+1,AL             ;RESTORE
       MOV    BX,AlarmResetMsg   ;Print set alarm msg
       CALL   PRINT_STRING
        JMP   START


DUMP_RAM:
       MOV    BX,DumpRamMsg      ;Print Dump RAM msg
       CALL   PRINT_STRING
       MOV    DL,-1              ;-1 goes to 0 for PORT_INC
       MOV    BX,LookupTable
       MOV    CX,1AH             ;Count of registers
       PUSH   CX                 ;Save it
       PUSH   BX                 ;Save it

RAM2:  CALL   PORT_INC           ;SET ADDRESS OF Register/Ram area
       IN     AL,CMOS_PORT+1             ;Get BCD value returned
       CALL   AL_HEXOUT          ;Show Hex data
       POP    BX
       CALL   PRINT_STRING
       POP    CX
       DEC    CX
       JZ     DONE_RAM
       PUSH   CX
       PUSH   BX                 ;Save for next time
       JMP    RAM2
DONE_RAM:CALL CRLF
       JMP    START


PORT_INC:
       INC    DL                 ;INCREMENT ADDRESS
       MOV    AL,DL
       OUT    CMOS_PORT,AL
       RET

PORT_INC_2:
       ADD    DL,2                       ;INCREMENT ADDRESS
       MOV    AL,DL
       OUT    CMOS_PORT,AL
       RET

INITIALIZE_STATUS:
       PUSH   DX                 ;SAVE
       MOV    DL,09H
       CALL   PORT_INC
       MOV    AL,26H
       OUT    CMOS_PORT+1,AL             ;INITIALIZE 'A' REGISTER
       CALL   PORT_INC
       MOV    AL,82H             ;SET 'SET BIT' FOR CLOCK INITIALIZATION
                                 ;AND 24 HOUR MODE
       OUT    CMOS_PORT+1,AL             ;INITIALIZE 'B' REGISTER
       CALL   PORT_INC
       IN     AL,CMOS_PORT+1             ;READ REGISTER 'C' TO INITIALIZE
       CALL   PORT_INC
       IN     AL,CMOS_PORT+1             ;READ REGISTER  'D'  TO INITIALIZE
       POP    DX                 ;RESTORE
       RET


UPD_IN_PR:                       ;Check we are ready to read clock
```

```
       PUSH   CX
       MOV    CX,600              ;SET LOOP COUNT
UPDATE:
       MOV    AL,0AH              ;ADDRESS OF [A] REGISTER
       OUT    CMOS_PORT,AL
       JMP    $+2                 ;I/O TIME DELAY
       IN     AL,CMOS_PORT+1             ;READ  IN REGISTER [A]
       TEST   AL,80H              ;IF 8XH--> UIP BIT  IS ON (CANNOT READ TIME)
       JZ     UPD_IN_PREND
       LOOP   UPDATE              ;Try again
       XOR    AX,AX               ;
       STC                        ;SET CARRY FOR ERROR
UPD_IN_PREND:
       POP    CX
       RET                        ;RETURN



;Display time
;      Arrive with CH = HOURS IN BCD
;                 CL = Minutes in BCD
;                 DH = Seconds in BCD
DisplayTime:
       PUSH   BX
       PUSH   DX
       PUSH   CX
       PUSH   CX
       MOV    BX,Time_Msg
       CALL   PRINT_STRING

       POP    CX
       MOV    AL,CH
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
       MOV    CL,':'
       CALL   CO
       POP    CX
       MOV    AL,CL
       CALL   PRINT_REG           ;Minutes.  Convert BCD to ASCII
       MOV    CL,':'
       CALL   CO
       POP    DX
       MOV    AL,DH
       CALL   PRINT_REG           ;Seconds.  Convert BCD to ASCII
       MOV    BX,Time1_Msg
       CALL   PRINT_STRING
       POP    BX
       RET



;Input time
;      Return CH = HOURS IN BCD
;             CL = Minutes in BCD
;             DH = Seconds in BCD
InputTime:
       PUSH   BX
       MOV    BX,Input_Hours_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       MOV    CH,AL
       PUSH   CX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       MOV    BX,Input_Minutes_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       POP    CX
       MOV    CL,AL
       PUSH   CX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       MOV    BX,Input_Seconds_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       MOV    DH,AL
       PUSH   DX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       POP    DX
       POP    CX
       POP    BX
       RET


;Display date
;      Return CH = CENTURY IN BCD
;             CL = Year in BCD
;             DH = Month in BCD
;             DL = Day in BCD
DisplayDate:
       PUSH   BX
       PUSH   DX
       PUSH   DX
       PUSH   CX
       PUSH   CX
       MOV    BX,Date_Msg
       CALL   PRINT_STRING

       POP    CX
       MOV    AL,CH
       CALL   PRINT_REG           ;Century (19/20).  Convert BCD to ASCII
       POP    CX
       MOV    AL,CL
       CALL   PRINT_REG           ;Year.  Convert BCD to ASCII
       MOV    CL,'/'
       CALL   CO
       POP    DX
       MOV    AL,DH
       CALL   PRINT_REG           ;Month.  Convert BCD to ASCII
       MOV    CL,'/'
       CALL   CO
       POP    DX
       MOV    AL,DL
       CALL   PRINT_REG           ;Day.  Convert BCD to ASCII
       MOV    BX,Date1_Msg
       CALL   PRINT_STRING
       POP    BX
       RET

PRINT_REG:                        ;Print BCD in [AL]
       PUSH   AX
       MOV    CL,4
       RCR    AX,CL
       AND    AL,0FH
       ADD    AL,30H
       MOV    CL,AL               ;Write high byte mins to CRT
       CALL   CO
       POP    AX
       AND    AL,0FH
       ADD    AL,30H
       MOV    CL,AL
       CALL   CO
       RET


;Input Date
```

```
;      Return CH = CENTURY IN BCD
;             CL = Year in BCD
;             DH = Month in BCD
;             DL = Day in BCD
InputDate:
       PUSH   BX
       MOV    BX,Input_Year_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       MOV    CL,AL
       MOV    CH,20H              ;Assume 20 for century
       PUSH   CX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       MOV    BX,Input_Month_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       MOV    DH,AL
       PUSH   DX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       MOV    BX,Input_Day_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       POP    DX
       MOV    DL,AL
       PUSH   DX
   %if   MSDOS
   %else
       CALL   PRINT_REG           ;Hours.  Convert BCD to ASCII
   %endif
       POP    DX
       POP    CX
       POP    BX
       RET


;------------------------- SUPPORT ROUTINES ---------------------------------------------

CO:                               ;Character in CL
    %if      MSDOS
       PUSH   DX
       MOV    DL,CL
       MOV    AH,02H
       INT    21H
       POP    DX
       RET
    %else
       IN     AL,KEYSTAT          ;PROPELLER CONSOLE (or SD SYSTEMS) VIDIO BOARD PORT
       AND    AL,4H
       JZ     CO
       MOV    AL,CL
       OUT    KEYOUT,AL
       MOV    AL,CL               ;MAKE SURE TO RETURN WITH [AL] CONTAINING CHAR
       RET
    %endif


CI:                               ;Return with character in AL
    %if      MSDOS                       ;Note character is echoed in MSDOS
       MOV    AH,01H
       INT    21H
       RET
    %else
       CALL   CSTS                ;Wait until something is there
       JZ     CI
       IN     AL,KEYIN
       AND    AL,7FH
       RET
CSTS:  IN     AL,KEYSTAT
       TEST   AL,02H
       JZ     NONE
       XOR    AL,AL
       DEC    AL
       RET                         ;RETURN WITH 0FFH IN [A] IF SOMETHING
NONE:  XOR    AL,AL
       RET
    %endif


GET2BCD:                           ;Return with 2 BCD digits in AL
       CALL   CI
       SUB    AL,'@'
       SHL    AL,1
       SHL    AL,1
       SHL    AL,1
       SHL    AL,1
       PUSH   AX
       CALL   CI
       SUB    AL,'@'
       AND    AL,0FH
       MOV    CL,AL
       POP    AX
       OR     AL,CL
       RET



CRLF:  PUSH   AX                   ;Send CR/LF to console. No registers changed
       PUSH   BX
       PUSH   CX
       PUSH   DX
       MOV    CL,CR
       CALL   CO
       MOV    CL,LF
       CALL   CO
       POP    DX
       POP    CX
       POP    BX
       POP    AX
       RET



PRINT_STRING:                      ;Use CS over-ride so it will splice into 8086 BIOS easily
       push   cx
print1:       mov    al,[CS:bx]           ;Note this routine does NOT assume DS = CS here.
       inc    bx                   ;By using the CS over-ride we will always have
       cmp    al,'$'               ;a valid pointer to messages at the end of this monitor
       jz     print2
       cmp    AL,0                 ;Also terminate with 0's
       JZ     print2
       mov    cl,al
       call   CO
       jmp    print1
print2:       pop    cx
       ret


;      AL_HEXOUT                   ;output the 2 hex digits in [AL]
AL_HEXOUT:                         ;No registers altered (except AL)
       push   cx
       push   ax
       mov    cl,4                 ;first isolate low nibble
       shr    al,cl
       call   hexdigout
       pop    ax
       call   hexdigout            ;get upper nibble
```

```
      pop    cx
      ret

hexdigout:
      and    al,0fh              ;convert nibble to ascii
      add    al,90h
      daa
      adc    al,40h
      daa
      mov    cl,al
      call   CO
      ret

RTC_MENU            DB     CR,LF,LF,'CMOS RTC TEST PROGRAM MENU (V0.1)',CR,LF,LF
                    DB     TAB,'2 = READ TIME     3 = SET TIME',CR,LF
                    DB     TAB,'4 = READ DATE     5 = SET DATE',CR,LF
                    DB     TAB,'6 = SET ALARM     7 = RESET ALARM',CR,LF
                    DB     TAB,'8 = Dump RAM      ESC = Return to MS-DOS',CR,LF,LF
                    DB     'Please enter menu option >',0

TIME_ERROR_MSG             DB     CR,LF,'ERROR',0
Time_Msg            DB     CR,LF,LF,'Time=',0
Time1_Msg           DB     CR,LF,'  ',0
Date_Msg            DB     CR,LF,LF,'Date=',0
Date1_Msg           DB     CR,LF,'  ',0
Input_Hours_Msg            DB     CR,LF,'Please Enter Hours (2 digits, 00-24) ',0
Input_Minutes_Msg   DB     CR,LF,'Please Enter Minutes (2 digits, 00-60) ',0
Input_Seconds_Msg   DB     CR,LF,'Please Enter Seconds (2 digits, 00-60) ',0
Input_Year_Msg             DB     CR,LF,'Please Enter Year (2 digits, 20xx) 20',0
Input_Month_Msg            DB     CR,LF,'Please Enter Month (2 digits, 00-12) ',0
Input_Day_Msg       DB     CR,LF,'Please Enter day (2 digits, 01-31) ',0
SetAlarmMsg         DB     CR,LF,LF,'Set CMOS-RTC Alarm.',0
AlarmSetMsg         DB     CR,LF,'Alarm Set',0
AlarmResetMsg       DB     CR,LF,'Alarm Reset',0
AlarmBusyMsg        DB     CR,LF,'Alarm currently active. Please reset alarm first',0
DumpRamMsg          DB     CR,LF,LF,'Data Dump of first 1AH CMOS-RAM Locations',CR,LF,0

LookupTable             DB      'H  <--00  RTC seconds',CR,LF,0
                        DB      'H  <--01  RTC seconds alarm',CR,LF,0
                        DB      'H  <--02  RTC minutes',CR,LF,0
                        DB      'H  <--03  RTC minutes alarm',CR,LF,0
                        DB      'H  <--04  RTC hours',CR,LF,0
                        DB      'H  <--05  RTC hours alarm',CR,LF,0
                        DB      'H  <--06  RTC day of week',CR,LF,0
                        DB      'H  <--07  RTC day of month',CR,LF,0
                        DB      'H  <--08  RTC month',CR,LF,0
                        DB      'H  <--09  RTC year',CR,LF,0
                        DB      'H  <--0A  RTC Status register A:',CR,LF,0
                        DB      'H  <--0B  RTC Status register B:',CR,LF,0
                        DB      'H  <--0C  RTC Status register C (read only):',CR,LF,0
                        DB      'H  <--0D  RTC Status register D (read only):',CR,LF,0
                        DB      'H  <--0E  Diagnostic status byte',CR,LF,0
                        DB      'H  <--0F  Shutdown status byte',CR,LF,0
                        DB      'H  <--10  Diskette drive type for A: and B:',CR,LF,0
                        DB      'H  <--11  Reserved',CR,LF,0
                        DB      'H  <--12  Fixed disk drive type for drive 0 and drive 1',CR,LF,0
                        DB      'H  <--13  Reserved',CR,LF,0
                        DB      'H  <--14  Equipment byte',CR,LF,0
                        DB      'H  <--15  LSB of system base memory in 1k blocks',CR,LF,0
                        DB      'H  <--16  MSB of system base memory in 1k blocks',CR,LF,0
                        DB      'H  <--17  LSB of total extended memory in 1k blocks',CR,LF,0
                        DB      'H  <--18  MSB of total extended memory in 1k blocks',CR,LF,0
                        DB      'H  <--19  Drive C extension byte (reserved AT)',CR,LF,0
                        DB      'H  <--1A  Drive D extension byte (reserved AT)',CR,LF,0,0
```