```
;
; Test program for CMOS Clock on S100Computers/N8VEM board (Load with CPM).
;
;      V0.1                ;Original version 11/26/2011
;
;      John Monahan  S100Computers.com
;
; This is a simple test program to work with the MS-DOS Support Board. It is written so
; the code can be easily "spliced" ino a Z80 Monitor or another program.
; The only other hardware use is the Consol I/O port.


;      PORT ASSIGNMENTS

CMOS_PORT     EQU    70H          ;Base Port for CMOS Chip on MSDOS Support Board
PIC_MASK_PORT EQU    0A1H         ;Port to block 8259A interrupts

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

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

       ORG    100H
START:
       LD     SP,STACK

       LD     HL,SIGNON           ; Signon
       CALL   PRINT_STRING

LOOP:  LD     HL,TIMEMENU
       CALL   PRINT_STRING

       CALL   ZCI                 ;Get a char
       LD     C,A
       PUSH   AF
       CALL   ZCO
       CALL   ZCRLF
       POP    AF

       CP     ESC
       JP     Z,DONE
       CP     32H                 ;Show Time
       JP     Z,READ_TIME
       CP     33H                 ;Set Time information
       JP     Z,SET_TIME
       CP     34H
       JP     Z,READ_DATE         ;Read date
       CP     35H
       JP     Z,SET_DATE
       CP     36H
       JP     Z,SET_ALARM
       CP     37H
       JP     Z,RESET_ALARM
       CP     38H
       JP     Z,DUMP_RAM
       CP     39H
       JP     Z,SET_RAM

ERROR: LD     HL,ABORTMSG
       CALL   PRINT_STRING
       JP     LOOP

DONE:  JP     0H
```

```
;-----------------------------------------------------------------------------------
READ_TIME:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE IN PROCESS
       JP     NC,RTC_2A           ;GO AROUND IF OK
       JP     ERROR               ;IF ERROR

RTC_2A:       LD     E,-2               ;-2 goes to 0 for PORT_INC_2
       CALL   PORT_INC_2          ;SET ADDRESS OF SECONDS
       IN     A,(CMOS_PORT+1)           ;Get BCD value returned
       LD     D,A                 ;SAVE IN D
       CALL   PORT_INC_2          ;SET ADDRESS OF MINUTES
       IN     A,(CMOS_PORT+1)           ;Get BCD value returned
       LD     C,A                 ;SAVE IN C
       CALL   PORT_INC_2          ;SET ADDRESS OF HOURS
       IN     A,(CMOS_PORT+1)           ;Get BCD value returned
       LD     B,A                 ;SAVE
       LD     E,0                 ;SET E TO ZERO
       CALL   DisplayTime
       CALL   ZCRLF
       JP     LOOP                ;BACK TO START


;Display time
;      Arrive with B = HOURS IN BCD
;                   C = Minutes in BCD
;                   D = Seconds in BCD
DisplayTime:
       PUSH   HL
       PUSH   DE
       PUSH   BC
       PUSH   BC
       LD     HL,Time_Msg
       CALL   PRINT_STRING

       POP    BC
       LD     A,B
       CALL   PRINT_BCD           ;Hours.  Convert BCD to ASCII
       LD     C,':'
       CALL   ZCO
       POP    BC
       LD     A,C
       CALL   PRINT_BCD           ;Minutes.  Convert BCD to ASCII
       LD     C,':'
       CALL   ZCO
       POP    DE
       LD     A,D
       CALL   PRINT_BCD           ;Seconds.  Convert BCD to ASCII
       LD     HL,Time1_Msg
       CALL   PRINT_STRING
       POP    HL
       RET


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

RTC_3: CALL   UPD_IN_PR           ;CHECK FOR UPDATE IN PROCESS
       JP     NC,RTC_3A           ;GO AROUND IF CLOCK OPERATING
       CALL   INITIALIZE_STATUS
RTC_3A:
       PUSH   DE                  ;SAVE for below
       LD     E,-2                ;-2 goes to 0 for PORT_INC_2
       CALL   PORT_INC_2          ;UPDATE ADDRESS
       LD     A,D                 ;GET TIME BYTE - SECONDS
       OUT    (CMOS_PORT+1),A           ;STORE TIME BYTE
       CALL   PORT_INC_2          ;UPDATE ADDRESS
       LD     A,C                 ;GET TIME BYTE - MINUTES
       OUT    (CMOS_PORT+1),A     ;STORE TIME BYTE
```

```
       CALL   PORT_INC_2          ;UPDATE ADDRESS
       LD     A,B                 ;GET TIME BYTE - HOURS
       OUT    (CMOS_PORT+1),A     ;STORE TIME BYTE
       LD     E,0AH
       CALL   PORT_INC
       POP    DE                  ;RESTORE

       IN     A,(CMOS_PORT+1)            ;GET CURRENT VALUE
       AND    A,23H               ;MASK FOR VALID BIT POSITIONS
       OR     A,E                 ;GET DST BIT
       OR     A,02H               ;TURN ON 24 HR MODE (For compatability with AT)
       PUSH   AF                  ;
       LD     E,0AH               ;
       CALL   PORT_INC            ;
       POP    AF                  ;
       OUT    (CMOS_PORT+1),A
       JP     LOOP                ;DONE


;Input time
;      Return B = HOURS IN BCD
;             C = Minutes in BCD
;             D = Seconds in BCD
InputTime:
       PUSH   HL
       LD     HL,Input_Hours_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       LD     B,A
       PUSH   BC

       LD     HL,Input_Minutes_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       POP    BC
       LD     C,A
       PUSH   BC

       LD     HL,Input_Seconds_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD                    ;Return with 2 BCD digits in AL
       LD     D,A

       POP    BC
       POP    HL
       RET

;-----------------------------------------------------------------------------------------------
READ_DATE:
       CALL   UPD_IN_PR
       JP     NC,RTC_4A
       JP     ERROR               ;ON ERROR

RTC_4A:       LD     E,6
       CALL   PORT_INC            ;POINT TO DAY
       IN     A,(CMOS_PORT+1)
       LD     B,A                 ;SAVE IN A
       CALL   PORT_INC            ;POINT TO MONTH
       IN     A,(CMOS_PORT+1)
       LD     D,A                 ;SAVE IN D
       CALL   PORT_INC            ;POINT TO YEAR
       IN     A,(CMOS_PORT+1)
       LD     C,A                 ;SAVE IN C
       LD     E,31H               ;POINT TO CENTURY BYTE SAVE AREA
       CALL   PORT_INC            ;
       IN     A,(CMOS_PORT+1)     ;GET VALUE
       LD     E,B                 ;GET DAY BACK
       LD     B,A
       CALL   DisplayDate
       CALL   ZCRLF
       JP     LOOP                ;FINISHED
```

```
;Display date
;       Return B = CENTURY IN BCD
;              C = Year in BCD
;              D = Month in BCD
;              E = Day in BCD
DisplayDate:
       PUSH   HL
       PUSH   DE
       PUSH   DE
       PUSH   BC
       PUSH   BC
       LD     HL,Date_Msg
       CALL   PRINT_STRING

       POP    BC
       LD     A,B
       CALL   PRINT_BCD           ;Century (19/20).  Convert BCD to ASCII
       POP    BC
       LD     A,C
       CALL   PRINT_BCD           ;Year.  Convert BCD to ASCII
       LD     C,'/'
       CALL   ZCO
       POP    DE
       LD     A,D
       CALL   PRINT_BCD           ;Month.  Convert BCD to ASCII
       LD     C,'/'
       CALL   ZCO
       POP    DE
       LD     A,E
       CALL   PRINT_BCD           ;Day.  Convert BCD to ASCII
       LD     HL,Date1_Msg
       CALL   PRINT_STRING
       POP    HL
       RET

;-----------------------------------------------------------------------------------------
SET_DATE:
       CALL   InputDate           ;Return      CH = HOURS IN BCD, CL = Minutes in BCD, DH = Seconds in BCD
       CALL   ZCRLF               ;No registers changed
RTC_5:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE  IN PROCESS
       JP     NC,RTC_5A           ;GO AROUND  IF CLOCK UPDATING
       CALL   INITIALIZE_STATUS
RTC_5A:
       PUSH   BC                  ;SAVE
       LD     B,E                 ;SAVE DAY OF MONTH
       LD     E,5                 ;ADDRESS OF DAY OF WEEK REGISTER
       CALL   PORT_INC
       LD     A,0H
       OUT    (CMOS_PORT+1),A     ;LOAD ZEROS TO 'DAY OF WEEK' BYTE
       CALL   PORT_INC            ;ADDRESS OF DAY OF MONTH REGISTER
       LD     A,B                 ;GET DAY OF MONTH BYTE
       OUT    (CMOS_PORT+1),A           ;STORE IT
       CALL   PORT_INC            ;ADDRESS MONTH REGISTER
       LD     A,D                 ;GET MONTH BYTE
       OUT    (CMOS_PORT+1),A           ;STORE IT
       CALL   PORT_INC            ;ADDRESS OF YEAR REGISTER
       LD     A,C                 ;GET YEAR BYTE
       OUT    (CMOS_PORT+1),A     ;STORE IT
       LD     E,0AH
       CALL   PORT_INC
       IN     A,(CMOS_PORT+1)     ;GET CURRENT SET ING
       AND    A,07FH              ;CLEAR  'SET BIT'
       OUT    (CMOS_PORT+1),A     ;AND START CLOCK UPDATING
       POP    BC                  ;GET BACK
       LD     E,31H               ;POINT TO SAVE AREA
       CALL   PORT_INC            ;
       LD     A,B                 ;GET CENTURY BYTE
       OUT    (CMOS_PORT+1),A     ;SAVE IT
```

```
       JP     LOOP


;Input Date
;      Return B = CENTURY IN BCD
;             C = Year in BCD
;             D = Month in BCD
;             E = Day in BCD
InputDate:
       PUSH   HL
       LD     HL,Input_Year_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD             ;Return with 2 BCD digits in AL
       LD     C,A
       LD     B,20H               ;Assume 20 for century
;      LD     B,19H               ;Assume 19 for century

       PUSH   BC

       LD     HL,Input_Month_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD             ;Return with 2 BCD digits in AL
       LD     D,A
       PUSH   DE

       LD     HL,Input_Day_Msg
       CALL   PRINT_STRING
       CALL   GET2BCD             ;Return with 2 BCD digits in AL
       POP    DE
       LD     E,A

       POP    BC
       POP    HL
       RET

;-----------------------------------------------------------------------------------------------
DUMP_RAM:
       LD     HL,DumpRamMsg       ;Print Dump msg
       CALL   PRINT_STRING
       LD     E,-1                ;-1 goes to 0 for PORT_INC
       LD     HL,LookupTable
       LD     B,39H               ;Count of registers
       PUSH   BC                  ;Save it

RAM2:  CALL   PORT_INC             ;SET ADDRESS OF Register/Ram area
       IN     A,(CMOS_PORT+1)           ;Get BCD value returned
       CALL   A_HEXOUT            ;Show Hex data
       CALL   PRINT_STRING        ;Note HL will increase on each call
       POP    BC
       DEC    B
       JP     Z,DONE_RAM
       LD     A,B
       CP     1AH
       JP     NZ,RAM3
       PUSH   HL
       LD     HL,EnterCR
       CALL   PRINT_STRING
       POP    HL
       CALL   ZCI                 ;Get a char before next screen full
       CP     ESC                 ;Was an abort requested
       JP     Z,DONE_RAM
       PUSH   BC
       CALL   ZCRLF
       CALL   ZCRLF
       JP     RAM2
RAM3:  PUSH   BC
       JP     RAM2
DONE_RAM:
       CALL   ZCRLF
       JP     LOOP
```

```
SET_RAM:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE  IN PROCESS
       JP     NC,SET_RAM2         ;GO AROUND  IF CLOCK UPDATING
       CALL   INITIALIZE_STATUS
SET_RAM2:
       PUSH   HL
       LD     HL,Input_ADDRESS_Msg
       CALL   PRINT_STRING
       CALL   GETHEX              ;Return with 2 digits in AL
       JP     C,SET_RAM1
       LD     E,A                 ;temp store here

       LD     HL,Input_RAM_Msg
       CALL   PRINT_STRING
       CALL   GETHEX              ;Return with 2 digits in AL
       JP     C,SET_RAM1
       PUSH   AF
       LD     A,E
       OUT    (CMOS_PORT),A       ;Select RAM area (Note data must be sent immediatly)
       POP    AF
       OUT    (CMOS_PORT+1),A     ;send data
SET_RAM1:
       CALL   ZCRLF
       POP    HL
       JP     LOOP

;-------------------------------------------------------------------------
SET_ALARM:                        ;This routine will not cause an interrupt on a Z80 correctly.
                                  ;The 8259A requires an 8080,8085 or 8086 for sINTA processing.
                                  ;However on a Z80 you could look at the 8259A status port to
                                  ;see if an alarm was triggered.
       LD     HL,SetAlarmMsg      ;Print set alarm msg
       CALL   PRINT_STRING

       CALL   InputTime           ;Return      CH = HOURS IN BCD, CL = Minutes in BCD, DH = Seconds in BCD
       CALL   ZCRLF               ;No registers changed
RTC_6:
       LD     E,0AH               ;CHECK FOR ALARM ALREADY ENABLED
       CALL   PORT_INC            ;
       IN     A,(CMOS_PORT+1)     ;GET CURRENT SETTING OF ALARM ENABLE
       AND    A,20H
       JP     Z,RTC_6A            ;ALARM NOT SET - GO PROCESS
       LD     HL,AlarmBusyMsg     ;Print set alarm msg
       CALL   PRINT_STRING
       XOR    A,A                 ;
       JP     ERROR               ;RETURN   IF ERROR
RTC_6A:
       CALL   UPD_IN_PR           ;CHECK FOR UPDATE  IN PROCESS
       JP     NC,RTC_6B           ;
       CALL   INITIALIZE_STATUS
RTC_6B:
       LD     E,-1
       CALL   PORT_INC_2
       LD     A,D                 ;GET SECONDS BYTE
       OUT    (CMOS_PORT+1),A     ;LOAD ALARM BYTE - SECONDS

       CALL   PORT_INC_2
       LD     A,C                 ;GET MINUTES PARAMETER
       OUT    (CMOS_PORT+1),A           ;LOAD ALARM BYTE - MINUTES
       CALL   PORT_INC_2
       LD     A,B                 ;GET HOURS PARAMETER
       OUT    (CMOS_PORT+1),A     ;LOAD ALARM BYTE - HOURS

       IN     A,(PIC_MASK_PORT)   ;ENSURE INTERRUPT UNMASKED
       AND    A,0FEH              ;Note the 8259A will only work in 8086 Mode (choose the appropiate mask)
       OUT    (PIC_MASK_PORT),A   ;for this board.

       LD     E,0AH
       CALL   PORT_INC
       IN     A,(CMOS_PORT+1)     ;GET CURRENT VALUE
```

```
       AND    A,07FH              ;ENSURE SET BIT TURNED OFF
       OR     A,20H               ;TURN ON ALARM ENABLE
       PUSH   AF                  ;
       LD     E,0AH               ;
       CALL   PORT_INC            ;
       POP    AF                  ;
       OUT    (CMOS_PORT+1),A     ;ENABLE ALARM

       LD     HL,AlarmSetMsg      ;Print set alarm msg
       CALL   PRINT_STRING
       JP     LOOP

;-------------------------------------------------------------------------------------
RESET_ALARM:
       LD     E,0AH
       CALL   PORT_INC
       IN     A,(CMOS_PORT+1)     ;GET STATUS BYTE
       AND    A,57H               ;TURN OFF ALARM ENABLE
       PUSH   AF                  ;SAVE
       LD     E,0AH
       CALL   PORT_INC            ;
       POP    AF                  ;
       OUT    (CMOS_PORT+1),A     ;RESTORE
       LD     HL,AlarmResetMsg    ;Print set alarm msg
       CALL   PRINT_STRING
        JP    LOOP

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

UPD_IN_PR:                        ;Check we are ready to read clock
       PUSH   BC
       LD     BC,600              ;SET LOOP COUNT
UPDATE:       LD     A,0AH        ;ADDRESS OF [A] REGISTER
       OUT    (CMOS_PORT),A
       NOP
       NOP
       NOP
       IN     A,(CMOS_PORT+1)     ;READ  IN REGISTER [A]
       AND    A,80H               ;IF 8XH--> UIP BIT  IS ON (CANNOT READ TIME)
       JP     Z,UPD_IN_PREND      ;Are we ready/done
       DEC    BC
       LD     A,C
       OR     A,B
       JP     NZ,UPDATE           ;Try again
       XOR    A,A                 ;
       SCF                        ;SET CARRY FOR ERROR
       POP    BC
       RET
UPD_IN_PREND:
       XOR    A,A                 ;Clear Carry
       POP    BC
       RET                        ;RETURN


PORT_INC:
       LD     A,E
       INC    A                   ;INCREMENT ADDRESS
       LD     E,A
       OUT    (CMOS_PORT),A
       RET


PORT_INC_2:
       LD     A,E
       ADD    2                   ;INCREMENT ADDRESS
       LD     E,A
       OUT    (CMOS_PORT),A
       RET
```

```
;Return with 2 digits in [A]. If abort, Carry flag set + ESC in [A]
;
GETHEX:      CALL   GETCMD       ;Get a character from keyboard & ECHO
       CP     ESC
       JP     Z,HEXABORT
       CP     '/'           ;check 0-9, A-F
       JP     C,HEXABORT
       CP     'F'+1
       JP     NC,HEXABORT
       CALL   ASBIN         ;Convert to binary
       rlca                 ;Shift to high nibble
       rlca
       rlca
       rlca
       LD     B,A           ;Store it
       CALL   GETCMD        ;Get 2nd character from keyboard & ECHO
       CP     ESC
       JP     Z,HEXABORT
       CP     '/'           ;check 0-9, A-F
       JP     C,HEXABORT
       CP     'F'+1
       JP     NC,HEXABORT
       CALL   ASBIN         ;Convert to binary
       OR     B             ;add in the first digit
       OR     A             ;To return NC
       ret
HEXABORT:
       scf                  ;Set Carry flag
       ret


GETCMD:      CALL   ZCI    ;GET A CHARACTER, convert to UC, ECHO it
       CALL   UPPER
       CP     ESC
       ret    z             ;Don't echo an ESC
       push   af            ;Save it
       push   bc
       LD     C,A
       CALL   ZCO           ;Echo it
       pop    bc
       pop    af            ;get it back
       ret

                            ;Convert LC to UC
UPPER: CP     'a'           ;must be >= lowercase a
       ret    c             ; else go back...
       CP     'z'+1         ;must be <= lowercase z
       ret    nc            ; else go back...
       SUB    'a'-'A'       ;subtract lowercase bias
       ret

                            ;ASCII TO BINARY CONVERSION ROUTINE
ASBIN: SUB    30H
       CP     0AH
       ret    m
       SUB    07H
       ret


INITIALIZE_STATUS:
       PUSH   DE                   ;SAVE
       LD     E,09H
       CALL   PORT_INC
       LD     A,26H
       OUT    (CMOS_PORT+1),A      ;INITIALIZE 'A' REGISTER
       CALL   PORT_INC
       LD     A,82H                ;SET 'SET BIT' FOR CLOCK INITIALIZATION
                                   ;AND 24 HOUR MODE
       OUT    (CMOS_PORT+1),A      ;INITIALIZE 'B' REGISTER
       CALL   PORT_INC
       IN     A,(CMOS_PORT+1)      ;READ REGISTER 'C' TO INITIALIZE
```

```
       CALL   PORT_INC
       IN     A,(CMOS_PORT+1)     ;READ REGISTER  'D'  TO INITIALIZE
       POP    DE                  ;RESTORE
       RET


PRINT_BCD:                        ;Print BCD in [A]
       PUSH   AF
       PUSH   AF
       RRA
       RRA
       RRA
       RRA
       AND    A,0FH
       ADD    A,30H
       LD     C,A                 ;Write high byte mins to CRT
       CALL   ZCO
       POP    AF
       AND    A,0FH
       ADD    A,30H
       LD     C,A
       CALL   ZCO
       POP    AF
       RET


; Main consol I/O routines
;
ZCO:   IN     A,(KEYSTAT)
       AND    04H
       JP     Z,ZCO
       LD     A,C
       OUT    (KEYOUT),A
       RET

ZCI:   IN     A,(KEYSTAT)
       AND    02H
       JP     Z,ZCI
       IN     A,(KEYIN)
       RET

;
; Send CR/LF to Consol
;
ZCRLF: PUSH   AF
       PUSH   BC
       LD     C,CR
       CALL   ZCO
       LD     C,LF
       CALL   ZCO
       POP    BC
       POP    AF
       RET


PRINT_STRING:
       push   BC
print1:       LD     a,(HL)       ;Point to start of string
       inc    HL                  ;By using the CS over-ride we will always have
       cp     A,'$'               ;a valid pointer to messages at the end of this monitor
       JP     z,print2
       cp     A,0                 ;Also terminate with 0's
       JP     Z,print2
       LD     C,A
       call   ZCO
       jp     print1
print2:       pop    BC
       ret


GET2BCD:                          ;Return with 2 BCD digits in A
```

```
       CALL   ZCI
       PUSH   AF
       LD     C,A
       CALL   ZCO
       POP    AF
       SUB    A,'@'
       SLA    A
       SLA    A
       SLA    A
       SLA    A
       PUSH   AF
       CALL   ZCI
       PUSH   AF
       LD     C,A
       CALL   ZCO
       POP    AF
       SUB    A,'@'
       AND    A,0FH
       LD     C,A
       POP    AF
       OR     A,C
       RET


;      A_HEXOUT                  ;output the 2 hex digits in [A]
A_HEXOUT:                        ;No registers altered (except A)
       push   BC
       push   AF
       srl    a
       srl    a
       srl    a
       srl    a
       call   hexdigout
       pop    AF
       call   hexdigout          ;get upper nibble
       pop    BC
       ret

hexdigout:
       and    a,0fh              ;convert nibble to ascii
       add    a,90h
       daa
       adc    a,40h
       daa
       LD     c,a
       call   ZCO
       ret

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

SIGNON:             DB     CR,LF,LF
                    DB     'Test CMOS_RTC Clock-Calander S100 MSDOS Support Board,(Ports 70H,71H).'
                    DB     CR,LF,'$'
TIMEMENU:           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      9 = Set RAM Value',CR,LF
                    DB     TAB,'ESC = Return to CPM',CR,LF,LF
                    DB     'Please enter menu option > $'
ABORTMSG:           DB     CR,LF,BELL,'Error or unknown command!',CR,LF,'$'

TIME_MSG:           DB     CR,LF,'Time = $'
Time1_Msg           DB     '  $'
Date1_Msg           DB     '  $'

DATE_MSG:           DB     CR,LF,'Date = $'
YEAR_MSG            DB     '/20$'

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
EnterCR                    DB     CR,LF,'Please press any key to continue... $'
Input_ADDRESS_Msg   DB     CR,LF,'Enter CMOS RAM Address (xxH): $'
Input_RAM_Msg       DB     CR,LF,'Enter CMOS RAM Value (xxH): $'

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
                         DB       'H  <--1B  Unused (reserved AT)',CR,LF,0
                         DB       'H  <--1C  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--1D  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--1E  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--1F  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--20  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--21  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--22  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--23  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--24  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--25  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--26  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--27  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--28  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--29  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--2A  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--2B  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--2C  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--2D  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--2E  CMOS checksum of bytes 10h-20h (MSB)',CR,LF,0
                         DB       'H  <--2F  CMOS checksum of bytes 10h-20h (LSB)',CR,LF,0
                         DB       'H  <--30  LSB of extended memory size found above 1 megabyte during
POST',CR,LF,0
                         DB       'H  <--31  MSB of extended memory size found above 1 megabyte during
POST',CR,LF,0
                         DB       'H  <--32  Date century byte in BCD ( BIOS interface to read and
set)',CR,LF,0
                         DB       'H  <--33  Information flags (set during power-on)',CR,LF,0
                         DB       'H  <--34  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--35  Unused (Reserved AT)',CR,LF,0
                         DB       'H  <--36  Used by CPM+ for Date (LSB) (Reserved AT)',CR,LF,0
```

```
                        DB      'H  <--37  Used by CPM+ for Date (MSB) (Reserved AT)',CR,LF,0
                        DB      'H  <--38....255  All further bytes Unused',CR,LF,0,0


       DS      40H
STACK: DB      0H
;
; END
```