```
;* FILENAME:     AT89C51.ASM
;* DISPLAY:      16X2 LCD 4-BIT OPERATION
;* MICRO:        ATMEL AT89C51 FLASH BASED MICRO
;* PROJECT:      ALARM CLOCK
;* DATE:         17/02/24
;* TIMEKEEPER:   DS3231 SERIAL TIMEKEEPER I.C.
;
;
;
$MOD51
SCL        BIT    P1.7
SDA        BIT    P1.6
DS1307W    EQU    0D0H     ; SLAVE ADDRESS 1101 000 + 0 TO WRITE
DS1307R    EQU    0D1H     ; SLAVE ADDRESS 1101 000 + 1 TO READ
FLAGS      DATA   20H
LASTREAD   BIT    FLAGS.0
ACK        BIT    FLAGS.5
BUS_FLT    BIT    FLAGS.6
_2W_BUSY   BIT    FLAGS.7
BITCNT     DATA   21H
BYTECNT    DATA   22H
SECS       DATA   24H      ;   '   SECONDS STORAGE RAM
MINS       DATA   25H      ;   '   MINUTES   '     '
HRS        DATA   26H      ;   '   HOURS     '     '
DAY        DATA   27H      ;   '   DAY       '     '
DATE       DATA   28H      ;   '   DATE      '     '
MONTH      DATA   29H      ;   '   MONTH     '     '
YEAR       DATA   2AH      ;   '   YEAR      '     '
CONTROL    DATA   2BH      ; FOR STORAGE OF CONTROL REGISTER WHEN READ.
ALM_HOUR   DATA   2CH      ; INTERNAL (ALARM HOURS) STORAGE.
ALM_MIN    DATA   2DH      ; INTERNAL (ALARM MINUTES) STORAGE.
ALM_CNTRL  DATA   2EH      ; INTERNAL STORAGE FOR ALARM (ON) TIME.
ALM_STORE  EQU    08H      ; DS1307 RAM, ALARM STORAGE, BEGINNING.

; ***LCD CONTROL***

LCD_DATA   EQU    080H
LCD_DB4    EQU    084H     ; P0.4 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB5    EQU    085H     ; P0.5 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB6    EQU    086H     ; P0.6 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_DB7    EQU    087H     ; P0.7 HIGH NIBBLE OF PORT 1 IS USED FOR DATA
LCD_RS     EQU    0B3H     ; P3.3 LCD REGISTER SELECT LINE
LCD_RW     EQU    0B4H     ; P3.4 LCD READ / WRITE LINE
LCD_E      EQU    0B5H     ; P3.5 LCD ENABLE LINE

; ***USER INPUT KEYS***

UP_BTTN    BIT    090H     ; P1.0  UP
DOWN_BTTN  BIT    091H     ; P1.1  DOWN
ENTER      BIT    092H     ; P1.2  ENTER
ALARM_SET  BIT    093H     ; P1.3  ALARM TIME SET SELECT

; ***SYSTEM INSTRUCTIONS***

CONFIG     EQU    28H      ; 4-BIT DATA,2 LINES,5X7 MATRIX LCD
ENTRYMODE  EQU    6        ; INCREMENT CURSOR DON'T SHIFT DISPLAY
```

```
; ***CURSOR CONTROL INSTRUCTIONS***

OFFCUR    EQU    0CH
BLINKCUR  EQU    0DH

; ***DISPLAY CONTROL INSTRUCTIONS***

CLRDSP    EQU    01H
ONDSP     EQU    0EH

; ***MACRO'S***

SCL_HIGH  MACRO
          SETB   SCL      ; SET SCL HIGH
          JNB    SCL,$    ; LOOP UNTIL STRONG 1 ON SCL
          ENDM

CSEG      AT     0        ; RESET VECTOR
          JNB    ENTER,INIT_YEAR
          AJMP   START

INIT_YEAR:
          JNB    ENTER,INIT_YEAR
          MOV    YEAR,#97H; INITIALIZE YEAR IF NEW 1307 TO 1997
          ACALL  SET_CLOCK; ELSE GO ON.

; ********************************************************
; RESET GOES HERE TO START PROGRAM
; ********************************************************

START:
          MOV         YEAR,#97H  ; SET YEAR TO 1997 ON BOOT
          ACALL       OFF        ; ALARM OFF ON BOOT
          MOV         SP,#2FH    ; POSITION STACK ABOVE BUFFER
          MOV         IE,#0      ; DISABLE INTERUPTS
          SETB        SDA        ; ENSURE SDA HIGH
          SCL_HIGH               ; ENSURE SCL HIGH
          CLR         ACK        ; CLEAR STATUS FLAGS
          CLR         BUS_FLT
          CLR         _2W_BUSY

; ********************************************************
; SUB SETS THE DS1307 OSCILLATOR
; ********************************************************

OSC_CONTROL:
          ACALL       SEND_START ; GENERATE START CONDITION
          MOV         A,#DS1307W ; 1101 0000 ADDRESS + WRITE-BIT
          ACALL       SEND_BYTE  ; SEND BYTE TO 1307
          MOV         A,#00H     ; ADDRESS BYTE TO REGISTER 00H
          ACALL       SEND_BYTE  ; SECONDS REGISTER, ALWAYS LEAVE
          SETB        LASTREAD   ; REG 00H-BIT #7 = 0 (LOW)
          ACALL       SEND_STOP  ; IF REG 00H-BIT #7 = 1 CLOCK
          ACALL       SEND_START ; OSCILLATOR IS OFF.
          MOV         A,#DS1307R ; 1101 0001 ADDRESS + READ-BIT
          ACALL       SEND_BYTE  ;
          ACALL       READ_BYTE  ; READ A BYTE FROM THE 1307
          CLR         ACC.7      ; CLEAR REG 00H-BIT #7 TO ENABLE
OSC_SET:                         ; OSCILLATOR.
```

```
        PUSH    ACC         ; SAVE ON STACK
        ACALL   SEND_STOP   ;
        ACALL   SEND_START  ;
        MOV     A,#DS1307W  ; SETUP TO WRITE
        ACALL   SEND_BYTE   ;
        MOV     A,#00H      ; REGISTER 00H ADDRESS
        ACALL   SEND_BYTE   ;
        POP     ACC         ; GET DATA TO START OSCILLATOR
        ACALL   SEND_BYTE   ; SEND IT
        ACALL   SEND_STOP
        ACALL   BEGIN

; ********************************************************
; TIME AND DATE ROUTINE, GET & DISPLAY MONTH/DATE/YEAR
; ********************************************************

GET_TIME:
        ACALL   READ_CLOCK              ; GET TIME,DATE,ALARM DATA
        JNB     ENTER,SET_TIME          ; GOTO SET TIME IF PUSHED
        JNB     ALARM_SET,SET_ALARMS    ; GOTO SET ALARMS ELSE SHOW TIME
        MOV     A,#1        ; LINE #1
        MOV     B,#1        ; COLUMN #1
        ACALL   PLACECUR4   ; PLACE CURSOR FOR DAY
        MOV     A,DAY       ; GET DAY
        ACALL   GET_DAY     ; SHOW DAY OF WEEK
        MOV     A,#1
        MOV     B,#7
        ACALL   PLACECUR4   ; PLACE CURSOR FOR MONTH
        MOV     A,MONTH     ; LOAD MONTH DATA
        ACALL   WRITE_BCD
        MOV     A,#1
        MOV     B,#10       ; PLACE CURSOR FOR DATE
        ACALL   PLACECUR4
        MOV     A,DATE
        ACALL   WRITE_BCD
        MOV     A,#1
        MOV     B,#13
        ACALL   PLACECUR4
        MOV     A,YEAR
        ACALL   WRITE_BCD

; ********************************************************
; NOW GET & DISPLAY TIME
; ********************************************************

        MOV     A,#2
        MOV     B,#4
        ACALL   PLACECUR4
        MOV     A,HRS
        ACALL   WRITE_BCD
        MOV     A,#2
        MOV     B,#7
        ACALL   PLACECUR4
        MOV     A,MINS
        ACALL   WRITE_BCD
        MOV     A,#2
        MOV     B,#10
        ACALL   PLACECUR4
        MOV     A,SECS
```

```
          ACALL       WRITE_BCD
          ACALL       CHECK_ALARM_TIME
          SJMP        GET_TIME

SET_ALARMS:
          JNB         ALARM_SET,SET_ALARMS
          ACALL       MDELAY
          AJMP        ALARMS

SET_TIME:
          JNB         ENTER,SET_TIME; HOLDS HERE UNTIL BUTTON IS RELEASED
          ACALL       MDELAY

SET_DAY:
          MOV         A,#1
          MOV         B,#0
          ACALL       PLACECUR4
          ACALL       PRTLCD4
          DB          '*',0       ; PRINT '*' BY DAY

WAIT_DAY:                         ; WAIT HERE FOR BUTTON PRESS
          JNB         UP_BTTN,UP_DAY
          JNB         DOWN_BTTN,DOWN_DAY
          JNB         ENTER,ENTER_DAY
          ACALL       UPDATE
          SJMP        WAIT_DAY

REDO_DAY:                         ; RESETS DAY TO SUN
          MOV         A,#1
          MOV         DAY,A
          AJMP        WAIT_DAY

REDO_DAY_DOWN:                    ; RESETS DAY TO SAT
          MOV         A,#7H
          MOV         DAY,A
          AJMP        WAIT_DAY

OK_DAY:                           ; PLACES NEW DAY DATA INTO REGISTER
          MOV         DAY,A
          AJMP        WAIT_DAY

UP_DAY:                           ; CHANGES DAY DATA UP BY 1
          JNB         UP_BTTN,UP_DAY
          ACALL       MDELAY
          MOV         R1,#DAY
          MOV         A,@R1
          INC         A
          CJNE        A,#8H,OK_DAY
          AJMP        REDO_DAY

DOWN_DAY:                         ; CHANGES MONTH DATA DOWN BY 1
          JNB         DOWN_BTTN,DOWN_DAY
          ACALL       MDELAY
          MOV         R1,#DAY
          MOV         A,@R1
          DEC         A
          CJNE        A,#0,OK_DAY
          AJMP        REDO_DAY_DOWN
```

```
ENTER_DAY:                      ; DONE WITH DAY GO ON TO MONTH
        JNB         ENTER,ENTER_DAY
        ACALL       MDELAY

SET_MONTH:
        MOV         A,#1
        MOV         B,#0
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          ' ',0       ; PRINT '*' FOR MONTH
        MOV         A,#1
        MOV         B,#6
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          '*',0

WAIT_MONTH:                     ; WAIT HERE FOR BUTTON PRESS
        JNB         UP_BTTN,UP_MONTH
        JNB         DOWN_BTTN,DOWN_MONTH
        JNB         ENTER,ENTER_MONTH
        ACALL       UPDATE
        SJMP        WAIT_MONTH

REDO_MONTH:                     ; RESETS MONTH TO 1
        MOV         A,#1
        MOV         MONTH,A
        AJMP        WAIT_MONTH

REDO_MONTH_DOWN:                ; RESETS MONTH TO 12
        MOV         A,#12H
        MOV         MONTH,A
        AJMP        WAIT_MONTH

OK_MONTH:                       ; PLACES NEW MONTH DATA INTO REGISTER
        MOV         MONTH,A
        AJMP        WAIT_MONTH

UP_MONTH:                       ; CHANGES MONTH DATA UP BY 1
        JNB         UP_BTTN,UP_MONTH
        ACALL       MDELAY
        MOV         R1,#MONTH
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD
        CJNE        A,#13H,OK_MONTH
        AJMP        REDO_MONTH

DOWN_MONTH:                     ; CHANGES MONTH DATA DOWN BY 1
        JNB         DOWN_BTTN,DOWN_MONTH
        ACALL       MDELAY
        MOV         R1,#MONTH
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD_DOWN
        DEC         A
        CJNE        A,#0,OK_MONTH
        AJMP        REDO_MONTH_DOWN

ENTER_MONTH:                    ; DONE WITH MONTH GO ON TO DATE
```

```
        JNB     ENTER,ENTER_MONTH
        ACALL   MDELAY

SET_DATE:
        MOV     A,#1
        MOV     B,#6
        ACALL   PLACECUR4
        ACALL   PRTLCD4
        DB      ' ',0       ; ERASE '*' BY MONTH
        MOV     A,#1
        MOV     B,#9
        ACALL   PLACECUR4
        ACALL   PRTLCD4
        DB      '*',0       ; PRINT '*' BY DATE

WAIT_DATE:                  ; WAIT HERE UNTIL BUTTON PRESS
        JNB     UP_BTTN,UP_DATE
        JNB     DOWN_BTTN,DOWN_DATE
        JNB     ENTER,ENTER_DATE
        ACALL   UPDATE
        SJMP    WAIT_DATE

REDO_DATE:                  ; RESETS DATE TO 1
        MOV     A,#1
        MOV     DATE,A
        AJMP    WAIT_DATE

REDO_DATE_DOWN:             ; RESETS DATE TO 31
        MOV     A,#31H
        MOV     DATE,A
        AJMP    WAIT_DATE

OK_DATE:                    ; PLACES NEW DATE DATA INTO REGISTER
        MOV     DATE,A
        AJMP    WAIT_DATE

UP_DATE:                    ; CHANGES DATE DATA UP BY 1
        JNB     UP_BTTN,UP_DATE
        ACALL   MDELAY
        MOV     R1,#DATE
        MOV     A,@R1
        MOV     R4,A
        ACALL   CHECK_BCD
        CJNE    A,#32H,OK_DATE
        AJMP    REDO_DATE

DOWN_DATE:                  ; CHANGES DATE DATA DOWN BY 1
        JNB     DOWN_BTTN,DOWN_DATE
        ACALL   MDELAY
        MOV     R1,#DATE
        MOV     A,@R1
        MOV     R4,A
        ACALL   CHECK_BCD_DOWN
        DEC     A
        CJNE    A,#0,OK_DATE
        AJMP    REDO_DATE_DOWN

ENTER_DATE:                 ; DONE WITH DATE GO ON TO YEAR
        JNB     ENTER,ENTER_DATE
```

```
        ACALL       MDELAY

SET_YEAR:
        MOV     A,#1
        MOV     B,#9
        ACALL   PLACECUR4
        ACALL   PRTLCD4
        DB      '-',0           ; PRINT '-' BESIDE DATE
        MOV     A,#1
        MOV     B,#12
        ACALL   PLACECUR4
        ACALL   PRTLCD4
        DB      '*',0           ; PRINT '*' BESIDE YEAR

WAIT_YEAR:                      ; WAIT HERE FOR BUTTON PRESS
        JNB     UP_BTTN,UP_YEAR
        JNB     DOWN_BTTN,DOWN_YEAR
        JNB     ENTER,ENTER_YEAR
        ACALL   UPDATE
        SJMP    WAIT_YEAR

REDO_YEAR:                      ; RESETS YEAR TO 1
        MOV     A,#97H
        MOV     YEAR,A
        SJMP    WAIT_YEAR

REDO_YEAR_DOWN:                 ; RESETS YEAR TO 99
        MOV     A,#99H
        MOV     YEAR,A
        AJMP    WAIT_YEAR

OK_YEAR:                        ; PLACES NEW YEAR DATA INTO REGISTER
        MOV     YEAR,A
        AJMP    WAIT_YEAR

UP_YEAR:                        ; CHANGES YEAR DATA UP BY 1
        JNB     UP_BTTN,UP_YEAR
        ACALL   MDELAY
        MOV     R1,#YEAR
        MOV     A,@R1
        INC     A
        CJNE    A,#9AH,OK_YEAR
        AJMP    REDO_YEAR

DOWN_YEAR:                      ; CHANGES YEAR DATA DOWN BY 1
        JNB     DOWN_BTTN,DOWN_YEAR
        ACALL   MDELAY
        MOV     R1,#YEAR
        MOV     A,@R1
        DEC     A
        CJNE    A,#96H,OK_YEAR
        AJMP    REDO_YEAR_DOWN

ENTER_YEAR:                     ; DONE WITH YEAR GO ON TO HOUR
        JNB     ENTER,ENTER_YEAR
        ACALL   MDELAY

SET_HOUR:
        MOV         A,#1
```

```
        MOV         B,#12
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          '-',0       ; PRINT '-' BESIDE YEAR
        MOV         A,#2
        MOV         B,#3
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          '*',0       ; PRINT '*' BESIDE HOUR
        AJMP        WAIT_HOUR

WAIT_HOUR:                      ; WAIT HERE UNTIL BUTTON PRESS
        JNB         UP_BTTN,UP_HOUR
        JNB         DOWN_BTTN,DOWN_HOUR
        JNB         ENTER,ENTER_HOUR
        ACALL       UPDATE
        SJMP        WAIT_HOUR

REDO_HOUR:                      ; RESET HOUR TO 00
        MOV         A,#0
        MOV         HRS,A
        AJMP        WAIT_HOUR

REDO_HOUR_DOWN:                 ; RESET HOUR TO 23
        MOV         A,#23H
        MOV         HRS,A
        AJMP        WAIT_HOUR

OK_HOUR:                        ; PLACE NEW HOUR DATA INTO REGISTER
        MOV         HRS,A
        AJMP        WAIT_HOUR

UP_HOUR:                        ; CHANGES HOUR DATA UP BY 1
        JNB         UP_BTTN,UP_HOUR
        ACALL       MDELAY
        MOV         R1,#HRS
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD
        CJNE        A,#24H,OK_HOUR
        AJMP        REDO_HOUR

DOWN_HOUR:                      ; CHANGES HOUR DATA DOWN BY ONE
        JNB         DOWN_BTTN,DOWN_HOUR
        ACALL       MDELAY
        MOV         R1,#HRS
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD_DOWN
        DEC         A
        CJNE        A,#0F9H,OK_HOUR
        AJMP        REDO_HOUR_DOWN

ENTER_HOUR:                     ; DONE WITH HOUR GO ON TO MINUTES
        JNB         ENTER,ENTER_HOUR
        ACALL       MDELAY

SET_MIN:
        MOV         A,#2
```

```
        MOV         B,#3
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          ' ',0       ; ERASE '*' BESIDE HOUR
        MOV         A,#2
        MOV         B,#6
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          '*',0       ; PRINT '*' BESIDE MINUTES

WAIT_MIN:                       ; WAIT HERE UNTIL BUTTON PRESS
        JNB         UP_BTTN,UP_MIN
        JNB         DOWN_BTTN,DOWN_MIN
        JNB         ENTER,ENTER_MIN
        ACALL       UPDATE
        SJMP        WAIT_MIN

REDO_MIN:                       ; RESET MINUTES TO 00
        MOV         A,#0
        MOV         MINS,A
        AJMP        WAIT_MIN

REDO_MIN_DOWN:                  ; RESETS MINUTES TO 59
        MOV         A,#59H
        MOV         MINS,A
        AJMP        WAIT_MIN

OK_MIN:                         ; PLACE NEW MINUTE DATA INTO REGISTER
        MOV         MINS,A
        AJMP        WAIT_MIN

UP_MIN:                         ; CHANGES MINUTES DATA UP BY 1
        JNB         UP_BTTN,UP_MIN
        ACALL       MDELAY
        MOV         R1,#MINS
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD
        CJNE        A,#60H,OK_MIN
        AJMP        REDO_MIN

DOWN_MIN:                       ; CHANGES MINUTES DATA DOWN BY 1
        JNB         DOWN_BTTN,DOWN_MIN
        ACALL       MDELAY
        MOV         R1,#MINS
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD_DOWN
        DEC         A
        CJNE        A,#0F9H,OK_MIN
        AJMP        REDO_MIN_DOWN

ENTER_MIN:                      ; DONE WITH MINUTES GO ON TO SECONDS
        JNB         ENTER,ENTER_MIN
        ACALL       MDELAY

SET_SECONDS:
        MOV         A,#2
        MOV         B,#6
```

```
          ACALL       PLACECUR4
          ACALL       PRTLCD4
          DB          ':',0       ; PRINT ':' BESIDE MINUTES
          ACALL       MDELAY
          MOV         A,#00000000B ; SECONDS ARE SET TO 00
          MOV         SECS,A
          ACALL       SET_CLOCK
          AJMP        GET_TIME    ; RETURN TO MAIN LOOP

ALARMS:
          ACALL       RESETLCD4
          ACALL       INITLCD4
          MOV         A,#1
          MOV         B,#0
          ACALL       PLACECUR4
          ACALL       PRTLCD4
          DB          '-SET ALARM TIME-',0
          ACALL       BEGIN3      ; PRINT HOUR & MINUTE SEPERATORS
          ACALL       UPDATE3     ; DISPLAY ALARM TIME
          AJMP        ALM_A       ; GO TO SET HOURS & MINUTES ROUTINE
                                  ; FOR ALARM TIME.
UPDATE:
          MOV         A,#1        ; LINE #1
          MOV         B,#1        ; COLUMN #1
          ACALL       PLACECUR4   ; PLACE CURSOR FOR DAY
          MOV         A,DAY       ; GET DAY DATA
          ACALL       GET_DAY     ; CONVERT BCD TO ASCII FOR LCD
          MOV         A,#1
          MOV         B,#7
          ACALL       PLACECUR4   ; PLACE CURSOR FOR MONTH
          MOV         A,MONTH     ; LOAD DATE MONTH
          ACALL       WRITE_BCD
          MOV         A,#1
          MOV         B,#10       ; PLACE CURSOR FOR DATE
          ACALL       PLACECUR4
          MOV         A,DATE      ; LOAD DATE DATA
          ACALL       WRITE_BCD
          MOV         A,#1
          MOV         B,#13
          ACALL       PLACECUR4
          MOV         A,YEAR
          ACALL       WRITE_BCD

; ********************************************************
; NOW GET & DISPLAY TIME
; ********************************************************

UPDATE2:
          MOV         A,#2
          MOV         B,#4
          ACALL       PLACECUR4
          MOV         A,HRS
          ACALL       WRITE_BCD
          MOV         A,#2
          MOV         B,#7
          ACALL       PLACECUR4
          MOV         A,MINS
          ACALL       WRITE_BCD
          MOV         A,#2
```

```
        MOV         B,#10
        ACALL       PLACECUR4
        MOV         A,SECS
        ACALL       WRITE_BCD
        MOV         R4,#OFFCUR
        ACALL       WRLCDCOM4  ; TURN OFF CURSOR
        ACALL       BIG_DELAY
        RET
UPDATE3:
        MOV         A,#2
        MOV         B,#6
        ACALL       PLACECUR4
        MOV         A,ALM_HOUR
        ACALL       WRITE_BCD
        MOV         A,#2
        MOV         B,#9
        ACALL       PLACECUR4
        MOV         A,ALM_MIN
        ACALL       WRITE_BCD
        MOV         R4,#OFFCUR
        ACALL       WRLCDCOM4
        ACALL       BIG_DELAY
        RET

CHECK_ALARM_TIME:
        MOV         A,MINS
        CJNE        A,ALM_MIN,OFF
        MOV         A,HRS
        CJNE        A,ALM_HOUR,OFF
        MOV         ALM_CNTRL,#10H ; LOAD VALUE TO TURN ON SQW OUT & ALARM
        ACALL       ALARM_CONTROL
        RET

OFF:    MOV         ALM_CNTRL,#00H
        ACALL       ALARM_CONTROL  ; LOAD VALUE TO TURN OFF ALARM & SQW OUT.
        RET
ALM_A:
        MOV         A,#2
        MOV         B,#5
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          '*',0      ; PRINT '*' BESIDE HOUR

WAIT_HOUR1:                    ; WAIT HERE UNTIL BUTTON PRESS
        JNB         UP_BTTN,UP_HOUR1
        JNB         DOWN_BTTN,DOWN_HOUR1
        JNB         ENTER,ENTER_HOUR1
        ACALL       UPDATE3
        SJMP        WAIT_HOUR1

REDO_HOUR1:                    ; RESET HOUR TO 00
        MOV         A,#0
        MOV         ALM_HOUR,A
        AJMP        WAIT_HOUR1

REDO_HOUR_DOWN1:               ; RESET HOUR TO 23
        MOV         A,#23H
        MOV         ALM_HOUR,A
        AJMP        WAIT_HOUR1
```

```
OK_HOUR1:                       ; PLACE NEW HOUR DATA INTO REGISTER
         MOV        ALM_HOUR,A
         AJMP       WAIT_HOUR1

UP_HOUR1:                       ; CHANGES HOUR DATA UP BY 1
         JNB        UP_BTTN,UP_HOUR1
         ACALL      MDELAY
         MOV        R1,#ALM_HOUR
         MOV        A,@R1
         MOV        R4,A
         ACALL      CHECK_BCD
         CJNE       A,#24H,OK_HOUR1
         AJMP       REDO_HOUR1

DOWN_HOUR1:                     ; CHANGES HOUR DATA DOWN BY ONE
         JNB        DOWN_BTTN,DOWN_HOUR1
         ACALL      MDELAY
         MOV        R1,#ALM_HOUR
         MOV        A,@R1
         MOV        R4,A
         ACALL      CHECK_BCD_DOWN
         DEC        A
         CJNE       A,#0F9H,OK_HOUR1
         AJMP       REDO_HOUR_DOWN1

ENTER_HOUR1:                    ; DONE WITH HOUR GO ON TO MINUTES
         JNB        ENTER,ENTER_HOUR1
         ACALL      MDELAY

SET_MIN1:
         MOV        A,#2
         MOV        B,#5
         ACALL      PLACECUR4
         ACALL      PRTLCD4
         DB         ' ',0       ; ERASE '*' BESIDE HOUR
         MOV        A,#2
         MOV        B,#8
         ACALL      PLACECUR4
         ACALL      PRTLCD4
         DB         '*',0       ; PRINT '*' BESIDE MINUTES

WAIT_MIN1:                      ; WAIT HERE UNTIL BUTTON PRESS
         JNB        UP_BTTN,UP_MIN1
         JNB        DOWN_BTTN,DOWN_MIN1
         JNB        ENTER,ENTER_MIN1
         ACALL      UPDATE3
         SJMP       WAIT_MIN1

REDO_MIN1:                      ; RESET MINUTES TO 00
         MOV        A,#0
         MOV        ALM_MIN,A
         AJMP       WAIT_MIN1

REDO_MIN_DOWN1:                 ; RESETS MINUTES TO 59
         MOV        A,#59H
         MOV        ALM_MIN,A
         AJMP       WAIT_MIN1
```

```
OK_MIN1:                        ; PLACE NEW MINUTE DATA INTO REGISTER
        MOV         ALM_MIN,A
        AJMP        WAIT_MIN1

UP_MIN1:                        ; CHANGES MINUTES DATA UP BY 1
        JNB         UP_BTTN,UP_MIN1
        ACALL       MDELAY
        MOV         R1,#ALM_MIN
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD
        CJNE        A,#60H,OK_MIN1
        AJMP        REDO_MIN1

DOWN_MIN1:                      ; CHANGES MINUTES DATA DOWN BY 1
        JNB         DOWN_BTTN,DOWN_MIN1
        ACALL       MDELAY
        MOV         R1,#ALM_MIN
        MOV         A,@R1
        MOV         R4,A
        ACALL       CHECK_BCD_DOWN
        DEC         A
        CJNE        A,#0F9H,OK_MIN1
        AJMP        REDO_MIN_DOWN1

ENTER_MIN1:                     ; DONE WITH MINUTES GO ON TO SECONDS
        JNB         ENTER,ENTER_MIN1
        ACALL       MDELAY

SET_SECONDS1:
        MOV         A,#2
        MOV         B,#8
        ACALL       PLACECUR4
        ACALL       PRTLCD4
        DB          ':',0       ; PRINT ':' BESIDE MINUTES
        ACALL       MDELAY

STORE_ALARMS:
        ACALL       SEND_START ; SEND 2-WIRE START CONDITION
        MOV         A,#DS1307W ; SEND 1307 WRITE COMMAND
        ACALL       SEND_BYTE
        MOV         A,#ALM_STORE; POINT TO REGISTER
                            ; 08H ON THE DS1307
        ACALL       SEND_BYTE
        MOV         R1,#ALM_HOUR; START WITH HOURS REGISTER
                            ; IN INTERNAL RAM
SEND_LOOP2:
        MOV         A,@R1       ; LOAD HOURS DATA FOR ALARM
        ACALL       SEND_BYTE   ; SET HOURS
        MOV         R1,#ALM_MIN
        MOV         A,@R1       ; LOAD MINUTES DATA FOR ALARM
        ACALL       SEND_BYTE   ; SET MINUTES
        ACALL       SEND_STOP   ; SEND 2-WIRE STOP CONDITION
        ACALL       BEGIN       ; SETUP DISPLAY AGAIN BEFORE RETURNING.
        AJMP        GET_TIME    ; GO BACK TO SHOWING TIME AFTER
                            ; SETTING ALARM

; *********************************************************
; BCD CONVERSION ROUTINE
```

```
; ACC CONTAINS TIME DATA FROM RAM WHEN ROUTINE IS CALLED.
; ********************************************************

WRITE_BCD:
         PUSH        ACC         ; DECODE HIGH DIGIT
         SWAP        A
         ANL         A,#0FH
         ACALL       FIND        ; LOOK IT UP & THEN PRINT IT.
         POP         ACC         ; DECODE LOW DIGIT
         ANL         A,#0FH
         ACALL       FIND
         RET

; ********************************************************
; SUB CONTROLS THE ALARM OUTPUT
; THE DS1307 SQW OUTPUT DRIVES THE ALARM BUZZER @ 1Hz
; ********************************************************

ALARM_CONTROL:
         ACALL       SEND_START
         MOV         A,#DS1307W
         ACALL       SEND_BYTE
         MOV         A,#07H
         ACALL       SEND_BYTE
         MOV         R1,#ALM_CNTRL
         MOV         A,@R1       ; 10H = ALARM ON AT 1HZ
         ACALL       SEND_BYTE   ; 00H = ALARM OFF & SQW OUT = GND
         ACALL       SEND_STOP
         RET

; ********************************************************
; SET UP DISPLAY ON BOOT, AND REDO DISPLAY AFTER CHANGE
; ********************************************************

BEGIN:
         ACALL       RESETLCD4
         ACALL       INITLCD4
         MOV         A,#1
         MOV         B,#7
         ACALL       PLACECUR4
         ACALL       PRTLCD4
         DB          '  -  -  ',0
         MOV         A,#2
         MOV         B,#0
         ACALL       PLACECUR4
         ACALL       PRTLCD4
         DB          '*     :  :     *',0
         MOV         R4,#OFFCUR ; TURN OFF CURSOR
         ACALL       WRLCDCOM4
         RET

BEGIN3:
         MOV         A,#2
         MOV         B,#0
         ACALL       PLACECUR4
         ACALL       PRTLCD4
         DB          '*       :      *',0
         MOV         R4,#OFFCUR ; TURN OFF CURSOR
         ACALL       WRLCDCOM4
```

```
        RET

; ********************************************************
; LOOKUP TABLE USED BY WRITE_BCD.
; ********************************************************

FIND:
        SJMP        LOOK_UP
ST:
        RET
LOOK_UP:
        CJNE        A,#0000000B,m1
        LCALL       PRTLCD4
        DB          '0',0
        LJMP        ST
m1:
        CJNE        A,#00000001B,m2
        LCALL       PRTLCD4
        DB          '1',0
        LJMP        ST
m2:
        CJNE        A,#00000010B,m3
        LCALL       PRTLCD4
        DB          '2',0
        LJMP        ST
m3:
        CJNE        A,#00000011B,m4
        LCALL       PRTLCD4
        DB          '3',0
        LJMP           ST
m4:
        CJNE        A,#00000100B,m5
        LCALL       PRTLCD4
        DB          '4',0
        LJMP        ST
m5:
        CJNE        A,#00000101B,m6
        LCALL       PRTLCD4
        DB          '5',0
        LJMP        ST
m6:
        CJNE        A,#00000110B,m7
        LCALL       PRTLCD4
        DB          '6',0
        LJMP        ST
m7:
        CJNE        A,#00000111B,m8
        LCALL       PRTLCD4
        DB          '7',0
        LJMP        ST
m8:
        CJNE        A,#00001000B,m9
        LCALL       PRTLCD4
        DB          '8',0
        LJMP        ST
m9:
        CJNE        A,#00001001B,m10
        LCALL       PRTLCD4
        DB          '9',0
```

```
        LJMP        ST
m10:
        LJMP        ST

; ********************************************************
; LOOKUP TABLE FOR DAY
; ********************************************************

GET_DAY:
        CJNE        A,#00000001B,MON
        LCALL       PRTLCD4
        DB          'SUN',0
        RET
MON:
        CJNE        A,#00000010B,TUE
        LCALL       PRTLCD4
        DB          'MON',0
        RET
TUE:
        CJNE        A,#00000011B,WED
        LCALL       PRTLCD4
        DB          'TUE',0
        RET
WED:
        CJNE        A,#00000100B,THU
        LCALL       PRTLCD4
        DB          'WED',0
        RET
THU:
        CJNE        A,#00000101B,FRI
        LCALL       PRTLCD4
        DB          'THU',0
        RET
FRI:
        CJNE        A,#00000110B,SAT
        LCALL       PRTLCD4
        DB          'FRI',0
        RET
SAT:
        CJNE        A,#00000111B,WHAT
        LCALL       PRTLCD4
        DB          'SAT',0
        RET
WHAT:
        RET

; ********************************************************
; SUB SETS THE CLOCK
; ********************************************************

SET_CLOCK:

        ACALL       SEND_START ; SEND 2-WIRE START CONDITION
        MOV         A,#DS1307W ; SEND 1307 WRITE COMMAND
        ACALL       SEND_BYTE
        MOV         A,#00H     ; SET DATA POINTER TO REGISTER
                               ; 00H ON THE DS1307
        ACALL       SEND_BYTE
        MOV         R1,#24H    ; START WITH SECONDS REGISTER
```

```
                                ; IN INTERNAL RAM
SEND_LOOP:
        MOV         A,@R1       ; MOVE FIRST BYTE OF DATA TO ACC
        ACALL       SEND_BYTE   ; SEND DATA ON 2-WIRE BUT
        INC         R1          ; LOOP UNTIL CLOCK DATA SENT
        CJNE        R1,#2BH,SEND_LOOP
        ACALL       SEND_STOP   ; SEND 2-WIRE STOP CONDITION
        RET

; *********************************************************
; INITIALIZE THE LCD 4-BIT MODE
; *********************************************************

INITLCD4:
        CLR         LCD_RS      ; LCD REGISTER SELECT LINE
        CLR         LCD_RW      ; READ / WRITE LINE
        CLR         LCD_E       ; ENABLE LINE
        MOV         R4, #CONFIG; FUNCTION SET - DATA BITS,
                                ; LINES, FONTS
        ACALL       WRLCDCOM4
        MOV         R4, #ONDSP ; DISPLAY ON
        ACALL       WRLCDCOM4
        MOV         R4, #ENTRYMODE ; SET ENTRY MODE
        ACALL       WRLCDCOM4   ; INCREMENT CURSOR RIGHT, NO SHIFT
        MOV         R4, #CLRDSP; CLEAR DISPLAY, HOME CURSOR
        ACALL       WRLCDCOM4
        RET

; *********************************************************
; SOFTWARE VERSION OF THE POWER ON RESET
; *********************************************************

RESETLCD4:
        CLR         LCD_RS      ; LCD REGISTER SELECT LINE
        CLR         LCD_RW      ; READ / WRITE LINE
        CLR         LCD_E       ; ENABLE LINE
        CLR         LCD_DB7     ; SET BIT PATTERN FOR...
        CLR         LCD_DB6     ; ... POWER-ON-RESET
        SETB        LCD_DB5
        SETB        LCD_DB4
        SETB        LCD_E       ; START ENABLE PULSE
        CLR         LCD_E       ; END ENABLE PULSE
        MOV         A, #4       ; DELAY 4 MILLISECONDS
        ACALL       MDELAY
        SETB        LCD_E       ; START ENABLE PULSE
        CLR         LCD_E       ; END ENABLE PULSE
        MOV         A, #1       ; DELAY 1 MILLISECOND
        ACALL       MDELAY
        SETB        LCD_E       ; START ENABLE PULSE
        CLR         LCD_E       ; END ENABLE PULSE
        MOV         A, #1       ; DELAY 1 MILLISECOND
        ACALL       MDELAY
        CLR         LCD_DB4     ; SPECIFY 4-BIT OPERATION
        SETB        LCD_E       ; START ENABLE PULSE
        CLR         LCD_E       ; END ENABLE PULSE
        MOV         A, #1       ; DELAY 1 MILLISECOND
        ACALL       MDELAY
        MOV         R4, #CONFIG; FUNCTION SET
        ACALL       WRLCDCOM4
```

```
        MOV         R4, #08H    ; DISPLAY OFF
        ACALL       WRLCDCOM4
        MOV         R4, #1      ; CLEAR DISPLAY, HOME CURSOR
        ACALL       WRLCDCOM4
        MOV         R4, #ENTRYMODE  ; SET ENTRY MODE
        ACALL       WRLCDCOM4
        AJMP        INITLCD4

; ********************************************************
; SUB WRITES A COMMAND WORD TO THE LCD
; COMMAND MUST BE PLACED IN R4 BY CALLING PROGRAM
; ********************************************************

WRLCDCOM4:
        CLR         LCD_E
        CLR         LCD_RS      ; SELECT SEND COMMAND
        CLR         LCD_RW      ; SELECT WRITE OPERATION
        PUSH        ACC         ; SAVE ACCUMULATOR
        MOV         A, R4       ; PUT DATA BYTE IN ACC
        MOV         C, ACC.4    ; LOAD HIGH NIBBLE ON DATA BUS
        MOV         LCD_DB4, C  ; ONE BIT AT A TIME USING...
        MOV         C, ACC.5    ; BIT MOVE OPERATOINS
        MOV         LCD_DB5, C
        MOV         C, ACC.6
        MOV         LCD_DB6, C
        MOV         C, ACC.7
        MOV         LCD_DB7, C
        SETB        LCD_E       ; PULSE THE ENABLE LINE
        CLR         LCD_E
        MOV         C, ACC.0    ; SIMILARLY, LOAD LOW NIBBLE
        MOV         LCD_DB4, C
        MOV         C, ACC.1
        MOV         LCD_DB5, C
        MOV         C, ACC.2
        MOV         LCD_DB6, C
        MOV         C, ACC.3
        MOV         LCD_DB7, C
        ACALL       PULSEEWAIT4; PULSE THE ENABLE LINE...
                               ; AND WAIT FOR BUSY FLAG TO CLEAR
        POP         ACC
        RET

; ********************************************************
; SUB TO WRITE A DATA WORD TO THE LCD
; DATA MUST BE PLACED IN R4 BY CALLING PROGRAM
; ********************************************************

WRLCDDATA:
        CLR         LCD_E
        SETB        LCD_RS      ; SELECT SEND DATA
        CLR         LCD_RW      ; SELECT WRITE OPERATION
        PUSH        ACC         ; SAVE ACCUMULATOR
        MOV         A, R4       ; PUT DATA BYTE IN ACC
        MOV         C, ACC.4    ; LOAD HIGH NIBBLE ON DATA BUS
        MOV         LCD_DB4, C  ; ONE BIT AT A TIME USING...
        MOV         C, ACC.5    ; BIT MOVE OPERATOINS
        MOV         LCD_DB5, C
        MOV         C, ACC.6
        MOV         LCD_DB6, C
```

```
        MOV         C, ACC.7
        MOV         LCD_DB7, C
        SETB        LCD_E       ; PULSE THE ENABLE LINE
        CLR         LCD_E
        MOV         C, ACC.0    ; SIMILARLY, LOAD LOW NIBBLE
        MOV         LCD_DB4, C
        MOV         C, ACC.1
        MOV         LCD_DB5, C
        MOV         C, ACC.2
        MOV         LCD_DB6, C
        MOV         C, ACC.3
        MOV         LCD_DB7, C
        ACALL       PULSEEWAIT4; PULSE THE ENABLE LINE...
                                ; AND WAIT FOR BUSY FLAG TO CLEAR
        POP         ACC
        RET

; ********************************************************
; GENERATES A POSITIVE PULSE ON THE LCD ENABLE LINE.
; WAITS FOR THE BUSY FLAG TO CLEAR BEFORE RETURNING.
; ********************************************************

PULSEEWAIT4:
        CLR         LCD_E
        SETB        LCD_E       ; PULSE THE ENABLE LINE
        CLR         LCD_E
        MOV         LCD_DATA, #0FFH ; PREPARE PORT FOR INPUT
        SETB        LCD_RW      ; PREPARE R/W FOR READ OPERATION
        PUSH        ACC         ; SAVE ACCUMULATOR CONTENTS
PEW:
        SETB        LCD_E       ; START THE ENABLE PULSE
        MOV         A, LCD_DATA; READ STATUS NIBBLE
        CLR         LCD_E       ; END ENABLE PULSE
        SETB        LCD_E       ; PRETEND READING STATUS LOW NIBBLE
        CLR         LCD_E
        JB          ACC.7, PEW ; LOOP WHILE BUSY FLAG IS SET
        POP         ACC         ; RESTORE ACCUMULATOR
        RET

; ********************************************************
; SUB TAKES THE STRING IMMEDIATELY FOLLOWING THE CALL AND
; DISPLAYS ON THE LCD. STRING MUST BE TERMINATED WITH A
; NULL (0).
; ********************************************************

PRTLCD4:
        POP         DPH         ; POP RETURN ADDRESS INTO DPTR
        POP         DPL
PRTNEXT:
        CLR         A           ; SET OFFSET = 0
        MOVC        A, @A+DPTR ; GET CHR FROM CODE MEMORY
        CJNE        A, #0, CHROK; IF CHR = 0 THEN RETURN
        SJMP        RETPRTLCD
CHROK:
        MOV         R4, A
        ACALL       WRLCDDATA  ; SEND CHARACTER
        INC         DPTR        ; POINT AT NEXT CHARACTER
        AJMP        PRTNEXT     ; LOOP TILL END OF STRING
RETPRTLCD:
```

```
        MOV         A,  #1H     ; POINT TO INSTRUCTION AFTER STRING
        JMP         @A+DPTR     ; RETURN WITH A JUMP INSTRUCTION

; *********************************************************
; SUB SETS THE CURSOR POSITION.
; *********************************************************

PLACECUR4:
        DEC         ACC         ; ACC=0 FOR LINE=1
        JNZ         LINE2       ; IF ACC=0 THEN FIRST LINE
        MOV         A, B
        ADD         A, #080H    ; CONSTRUCT CONTROL WORD FOR LINE 1
        SJMP        SETCUR
LINE2:
        MOV         A, B
        ADD         A, #0C0H    ; CONSTRUCT CONTROL WORD FOR LINE 2
SETCUR:
        MOV         R4, A       ; PLACE CONTROL WORD
        ACALL       WRLCDCOM4   ; ISSUE COMMAND
        RET

SETCGRAM:
        PUSH        B
        MOV         B, #8
        MUL         AB           ; multiply character number by 8
        POP         B            ; b holds row number
        ADD         A,B          ; a holds CGRAM address
        ADD         A,#40h       ; convert to instruction
        MOV         R4,A         ; place instruction
        ACALL       WRLCDCOM4    ; issue command
        RET

; *********************************************************
; 1 MILLISECOND DELAY ROUTINE
; *********************************************************

MDELAY:
        PUSH        ACC
        MOV         A,#0A6H
MD_OLP:
        INC         A
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        JNZ         MD_OLP
        NOP
        POP         ACC
        RET

BIG_DELAY:
        MOV         R3,#1       ; DELAY ROUTINE.
X0B:    MOV         R2,#0FFH
X1B:    MOV         R5,#0FFH
X2B:    DJNZ        R5,X2B
```

```
        DJNZ        R2,X1B
        DJNZ        R3,X0B
        RET

; ********************************************************
; THIS SUB READS ONE BYTE OF DATA FROM THE DS1307
; ********************************************************

READ_BYTE:
        MOV         BITCNT,#08H; SET COUNTER FOR 8-BITS DATA
        MOV         A,#00H
        SETB        SDA         ; SET SDA HIGH TO ENSURE LINE
                                ; FREE
READ_BITS:
        SCL_HIGH                ; TRANSITION SCL LOW-TO-HIGH
        MOV         C,SDA       ; MOVE DATA BIT INTO CARRY
        RLC         A           ; ROTATE CARRY-BIT INTO ACC.0
        CLR         SCL         ; TRANSITION SCL HIGH-TO-LOW
        DJNZ        BITCNT,READ_BITS
                                ; LOOP FOR 8-BITS
        JB          LASTREAD,ACKN
                                ; CHECK TO SEE IF THIS IS
                                ; THE LAST READ
        CLR         SDA         ; IF NOT LAST READ SEND ACK-BIT

ACKN:
        SCL_HIGH                ; PULSE SCL TO TRANSMIT ACKNOWLEDGE
        CLR         SCL         ; OR NOT ACKNOWLEDGE BIT
        RET

; ********************************************************
; SUB SENDS START CONDITION
; ********************************************************

SEND_START:
        SETB        _2W_BUSY    ; INDICATE THAT 2-WIRE
        CLR         ACK         ; OPERATION IS IN PROGRESS
        CLR         BUS_FLT     ; CLEAR STATUS FLAGS
        JNB         SCL,FAULT
        JNB         SDA,FAULT
        SETB        SDA         ; BEGIN START CODITION
        SCL_HIGH
        CLR         SDA
        ACALL       DELAY
        CLR         SCL
        RET
FAULT:
        SETB        BUS_FLT
        RET

; ********************************************************
; SUB SENDS STOP CONDITION
; ********************************************************

SEND_STOP:
        CLR         SDA
        SCL_HIGH
        SETB        SDA
        CLR         _2W_BUSY
```

```
        RET

; ********************************************************
; SUB DELAYS THE BUS
; ********************************************************

DELAY:
        NOP                     ; DELAY FOR BUS TIMING
        RET

; ********************************************************
; THIS SUB SENDS 1 BYTE OF DATA TO THE DS1307
; CALL THIS FOR EACH REGISTER SECONDS TO YEAR
; ACC MUST CONTAIN DATA TO BE SENT TO CLOCK
; ********************************************************

SEND_BYTE:
        MOV         BITCNT,#08H; SET COUNTER FOR 8-BITS
SB_LOOP:
        JNB         ACC.7,NOTONE; CHECK TO SEE IF BIT-7 OF
        SETB        SDA         ; ACC IS A 1, AND SET SDA HIGH
        JMP         ONE
NOTONE:
        CLR         SDA         ; CLR SDA LOW
ONE:
        SCL_HIGH                ; TRANSITION SCL LOW-TO-HIGH
        RL          A           ; ROTATE ACC LEFT 1-BIT
        CLR         SCL         ; TRANSITION SCL LOW-TO-HIGH
        DJNZ        BITCNT,SB_LOOP; LOOP FOR 8-BITS
        SETB        SDA         ; SET SDA HIGH TO LOOK FOR
        SCL_HIGH                ; ACKNOWLEDGE PULSE
        CLR         ACK
        JNB         SDA,SB_EX   ; CHECK FOR ACK OR NOT ACK
        SETB        ACK         ; SET ACKNOWLEDGE FLAG FOR
                                ; NOT ACK
SB_EX:
        ACALL       DELAY       ; DELAY FOR AN OPERATION
        CLR         SCL         ; TRANSITION SCL HIGH-TO-LOW
        ACALL       DELAY       ; DELAY FOR AN OPERATION
        RET

; ********************************************************
; SUB READS THE CLOCK AND WRITES IT TO THE SCRATCHPAD MEMORY
; ON RETURN FROM HERE DATE & TIME DATA WILL BE STORED IN THE
; DATE & TIME REGISTERS FROM 24H (SECS) TO 2AH (YEAR)
; ALARM SETTINGS IN REGISTERS 2CH(HRS) AND 2DH(MINUTES).
; ********************************************************

READ_CLOCK:

        MOV         R1,#24H     ; SECONDS STORAGE LOCATION
        MOV         BYTECNT,#00H
        CLR         LASTREAD
        ACALL       SEND_START
        MOV         A,#DS1307W
        ACALL       SEND_BYTE
        MOV         A,#00H
        ACALL       SEND_BYTE
        ACALL       SEND_STOP
```

```
        ACALL   SEND_START
        MOV     A,#DS1307R
        ACALL   SEND_BYTE

READ_LOOP:
        MOV     A,BYTECNT
        CJNE    A,#09H,NOT_LAST
        SETB    LASTREAD

NOT_LAST:
        ACALL   READ_BYTE
        MOV     @R1,A
        MOV     A,BYTECNT
        CJNE    A,#00H,NOT_FIRST
        MOV     A,@R1
        CLR     ACC.7       ; ENSURE OSC BIT=0 (ENABLED)
        MOV     @R1,A
NOT_FIRST:
        INC     R1
        INC     BYTECNT
        MOV     A,BYTECNT
        CJNE    A,#0AH,READ_LOOP
        ACALL   SEND_STOP
        RET

; *********************************************************
; MOST OF THE REAL WORK IS DONE IN THE ADJUST ROUTINE
; THIS ROUTINE SIMPLY CHECKS TO SEE IF THE NUMBER IS
; ZERO OR ENDS IN NINE.
; *********************************************************

CHECK_BCD:
        MOV     A,R4        ; SAVES THE NUMBER TO R4
        CLR     C           ; CLEARS CARRY BIT
        ADD     A,#67H      ; CHECKS TO SEE IF EQUAL 99
        JC      ZERO        ; IF EQUAL THEN BECOMES 0
        MOV     A,R4        ; GETS ORIGINAL NUMBER
        ANL     A,#0FH      ; GETS RID OF UPPER 4 BITS
        CLR     C           ; CLEARS CARRY BIT
        ADD     A,#0F7H     ; CHECKS TO SEE IF NUMBER ENDS IN 9
        JC      ADJUST      ; IF LARGER ADJUST IT
        MOV     A,R4        ; GET ORIGINAL NUMBER
        INC     A           ; ITS UNDER 9 AND NOT 0 SO INCREASE
        RET                 ; BY 1 AND GO BACK.

; *********************************************************
; BCD CONVERSION ROUTINE.
; *********************************************************
ADJUST:
        MOV     A,R4        ; GET ORIGINAL NUMBER
        SWAP    A           ; CHANGE UPPER BITS TO LOWER BITS
        ANL     A,#0FH      ; GET RID OF WHAT WAS LOWER BITS
        INC     A           ; INCREASE BY ONE
        SWAP    A           ; SWITCH THE UPPER AND LOWER BITS BACK
        RET                 ; GO BACK WITH NEW NUMBER

ZERO:                       ; IF NUMBER IS ZERO SEND BACK A ZERO
        MOV     A,#00000000B
        RET
```

```
; ********************************************************
; THIS ROUTINE CHECKS TO SEE IF THE LOWER 4 BITS ARE
; EQUAL TO ZERO. IF THEY ARE THEN A STRAIGHT DECREASE
; CAN BE DONE. IF NOT THEN CONVERT.
; ********************************************************

CHECK_BCD_DOWN:
         MOV         A,R4        ; SAVE ORIGINAL NUMBER
         ANL         A,#0FH      ; GET RID OF UPPER FOUR BITS
         CJNE        A,#00000000B,GO_ON_BACK ;DO LOWER BITS =0 ?
         MOV         A,R4        ; GET ORIGINAL NUMBER
         SUBB        A,#06H      ; CONVERT IT TO A BINARY NUMBER
         RET

; ********************************************************
; IF THE NUMBER ENDS IN NUMBER LESS THAN 9 THEN
; GO BACK AND, DO A NORMAL DECREMENT.
; ********************************************************

GO_ON_BACK:
         MOV         A,R4        ; GET ORIGINAL NUMBER
         RET                     ; GO BACK

; ********************************************************
; END OF PROGRAM
; ********************************************************
  END
```