SOFTWARE PROGRAMMING

ASSEMBLER AUTOMATIC COLLEGE BELL

```
RB0     EQU     000H    ; Select Register Bank 0
RB1     EQU     008H    ; Select Register Bank 1  ...poke to PSW to use


;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;                       PORT DECLERATION
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


SDA     EQU     P1.1    ;SDA=PIN5
SCL     EQU     P1.0    ;SCL=PIN6


DS1307W   EQU    0D0H     ; SLAVE ADDRESS 1101 000 + 0 TO WRITE
DS1307R   EQU    0D1H     ; SLAVE ADDRESS 1101 000 + 1 TO READ
KEYS    EQU     P3
ROW1    EQU     P3.1
ROW2    EQU     P3.2
ROW3    EQU     P3.3
ROW4    EQU     P3.4
COL1    EQU     P3.5
COL2    EQU     P3.6
```

```
COL3  EQU    P3.7

DIS_A EQU    P0.2
DIS_B EQU    P0.3
DIS_C EQU    P0.4
DIS_D EQU    P0.6
DIS_E EQU    P0.5
DIS_F EQU    P0.1
DIS_G EQU    P0.0

DIS1    EQU    P0.7
DIS2    EQU    P2.7
DIS3    EQU    P2.6
DIS4    EQU    P2.5

RELAY         EQU    P2.4

WMCON   DATA    96h                 ; watchdog and memory control register
EEMEN   EQU     00001000b           ; EEPROM access enable bit
EEMWE   EQU     00010000b           ; EEPROM write enable bit
WDTRST  EQU     00000010b           ; EEPROM RDY/BSY bit
DPS     EQU     00000100b     ; data pointer select bit
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
DSEG          ; This is internal data memory
ORG     20H     ; Bit adressable memory
FLAGS     DATA   20H
LASTREAD  BIT    FLAGS.0
SQW       BIT    FLAGS.4
ACK       BIT    FLAGS.5
BUS_FLT   BIT    FLAGS.6
_2W_BUSY  BIT    FLAGS.7
CANCEL    BIT    FLAGS.1
CANCEL1   BIT    FLAGS.2
ALARM           BIT    FLAGS.3
BITCNT    DATA   21H
BYTECNT   DATA   22H
SECS      DATA   24H     ;  '  SECONDS STORAGE RAM
MINS      DATA   25H     ;  '  MINUTES  '    '
HRS       DATA   26H     ;  '  HOURS    '    '
DAY       DATA   27H     ;  '  DAY      '    '
DATE1     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.
```

```
COUNT        DATA 2FH
SPEED        DATA 30H
VALUE_1      DATA 31H
VALUE_2      DATA 32H
VALUE_3      DATA 33H
VALUE_4      DATA 34H

NUMBER1      DATA 35H     ;temp to store dialled number
KBELL        DATA 36H
NUMB1        DATA 37H     ;Temp Reg to store pressed Keys
NUMB2        DATA 38H     ;Temp Reg to store pressed Keys
NUMB3        DATA 39H     ;Temp Reg to store pressed Keys
NUMB4        DATA 3AH     ;Temp Reg to store pressed Keys
KEY    DATA 3BH
TIM   DATA  3CH

STACK        DATA 3FH
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; ***MACRO'S***
SCL_HIGH  MACRO
     SETB   SCL      ; SET SCL HIGH
     JNB    SCL,$    ; LOOP UNTIL STRONG 1 ON SCL
     ENDM
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
CSEG    AT      0       ; RESET VECTOR
;---------==========---------==========---------=========---------
;            PROCESSOR INTERRUPT AND RESET VECTORS
;---------==========---------==========---------=========---------
        ORG     00H                     ; Reset
        JMP     MAIN

        ORG    000BH           ;Timer Interrupt0
        JMP     REFRESH
        ORG 001BH   ;Timer Interrupt1
  JMP RELAY_TIMER
;---------==========---------==========---------=========---------
;  Main routine. Program execution starts here.
;---------==========---------==========---------=========---------
MAIN:
        MOV PSW,#RB0       ; Select register bank 0
  MOV SP,STACK
  CLR RELAY                                 ;Switch OFF relay


        MOV SPEED,#00H
        MOV COUNT,#00H
        MOV KBELL,#00H

        CLR ALARM
```

```
        MOV VALUE_1,#15H
        MOV VALUE_2,#15H
        MOV VALUE_3,#15H
        MOV VALUE_4,#15H
        CLR DIS1
        CLR DIS2
        CLR DIS3
        CLR DIS4
        MOV TMOD,#01H           ;enable timer0 for scanning
        MOV TL0,#00H
        MOV TH0,#0FDH
        SETB ET0
        SETB EA
        SETB TR0            ;Start the Timer
; ************************************************************
;               INITILIZE RTC
; ************************************************************
                    SETB SDA            ; ENSURE SDA HIGH
    SCL_HIGH            ; ENSURE SCL HIGH
    CLR ACK             ; CLEAR STATUS FLAGS
    CLR BUS_FLT
    CLR _2W_BUSY
      CLR SQW
      CALL OSC_CONTROL              ;Initilize the RTC
      ACALL SQW_CONTROL_1HZ
```

```
; *********************************************************
;                CHECK FOR ENTER THE TIME
; *********************************************************
        LCALL SEND_START            ; SEND 2WIRE START CONDITION
        MOV A,#DS1307W              ; SEND DS1307 WRITE COMMAND
        LCALL SEND_BYTE
        MOV A,#08H                  ; SET POINTER TO REG 08H ON DS1307
        LCALL SEND_BYTE
        LCALL SEND_STOP             ; SEND STOP CONDITION
        LCALL SEND_START            ; SEND START CONDITION
        MOV A,#DS1307R              ; SEND DS1307 READ COMMAND
        LCALL SEND_BYTE
        LCALL READ_BYTE             ; READ A BYTE OF DATA
        MOV R1,A
        LCALL SEND_STOP             ; SEND 2WIRE STOP CONDITION

        MOV NUMBER1,#01H
        CJNE A,#0AAH,KEYBOARD1
        AJMP START_PROGRAM

;*********************************************************
;                KEYBOARD ROUTINE
;*********************************************************
KEYBOARD1:
        MOV KBELL,#0FFH
```

```
KEYBOARD:
        MOV KEY,#00H
        SETB COL1
        SETB COL2
        SETB COL3
K11:    CLR ROW1
        CLR ROW2
        CLR ROW3
        CLR ROW4
        MOV A,KEYS
        ANL A,#11100000B
        CJNE A,#11100000B,K11       ;check till all keys released
K2:     ACALL DEALAY                ;call 20 msec delay
        MOV A,KEYS                  ;see if any key is pressed
        ANL A,#11100000B            ;mask unused bits
        CJNE A,#11100000B,OVER      ;key pressed, await closure
        SJMP K2
OVER:       ACALL DEALAY
        MOV A,KEYS
        ANL A,#11100000B
        CJNE A,#11100000B,OVER1
        SJMP K2
OVER1:      MOV A,KEYS
        ORL A,#11111110B
        MOV KEYS,A
```

```
CLR ROW1
MOV A,KEYS
ANL A,#11100000B
CJNE A,#11100000B,ROW_1
MOV A,KEYS
ORL A,#11111110B
MOV KEYS,A
CLR ROW2
MOV A,KEYS
ANL A,#11100000B
CJNE A,#11100000B,ROW_2
MOV A,KEYS
ORL A,#11111110B
MOV KEYS,A
CLR ROW3
MOV A,KEYS
ANL A,#11100000B
CJNE A,#11100000B,ROW_3
MOV A,KEYS
ORL A,#11111110B
MOV KEYS,A
CLR ROW4
MOV A,KEYS
ANL A,#11100000B
CJNE A,#11100000B,ROW_4
```

```
        LJMP K2


ROW_1:  RLC A
        JC MAT1
        MOV KEY,#01H
        AJMP K1
MAT1:   RLC A
        JC MAT2
        MOV KEY,#02H
        AJMP K1
MAT2:   RLC A
        JC K1
        MOV KEY,#03H
        AJMP K1

ROW_2:  RLC A
        JC MAT3
        MOV KEY,#04H
        AJMP K1
MAT3:   RLC A
        JC MAT4
        MOV KEY,#05H
        AJMP K1
MAT4:   RLC A
```

```
        JC K1
        MOV KEY,#06H
        AJMP K1
ROW_3:  RLC A
        JC MAT5
        MOV KEY,#07H
        AJMP K1
MAT5:   RLC A
        JC MAT6
        MOV KEY,#08H
        AJMP K1
MAT6:   RLC A
        JC K1
        MOV KEY,#09H
        AJMP K1

ROW_4:  RLC A
        JC MAT7
        MOV KEY,#10H            ;for *
        AJMP K1
MAT7:   RLC A
        JC MAT8
        MOV KEY,#00H            ;for 0
        AJMP K1
MAT8:   RLC A
```

```
        JC K1
        MOV KEY,#12H                    ;for =
K1:

        MOV A,KBELL
        CJNE A,#0FFH,KB_RET1

        MOV A,KEY
        CJNE A,#10H,CXCX0       ;Key to Erase last dislled NUMBER1
        MOV KEY,#00H
        MOV NUMBER1,#01H
        MOV VALUE_1,#15H
        MOV VALUE_2,#15H
        MOV VALUE_3,#15H
        MOV VALUE_4,#15H
        AJMP KEYBOARD
KB_RET1: JMP KB_RET
CXCX0:      MOV A,NUMBER1
        CJNE A,#01H,CXCX1
        MOV A,KEY
        CLR C
        SUBB A,#03H                     ; Chk Key Pressed 0,1
        JNC CXCX5
        MOV A,KEY
        INC NUMBER1
```

```
        MOV NUMB1,KEY
        MOV VALUE_1,KEY
        AJMP KEYBOARD
CXCX1:  CJNE A,#02H,CXCX2

        MOV A,NUMB1
        CJNE A,#02,JKJL
        MOV A,KEY
        CLR C
        SUBB A,#04H             ; Chk Key Pressed 0,1,2,3
        JNC CXCX5
JKJL:   MOV A,KEY
        CLR C
        SUBB A,#10H             ; Chk Key Pressed 0,1...8,9
        JNC CXCX5
        INC NUMBER1
        MOV NUMB2,KEY
        MOV VALUE_2,KEY
        AJMP KEYBOARD
CXCX2:  CJNE A,#03H,CXCX3
        MOV A,KEY
        CLR C
        SUBB A,#06H             ; Chk Key Pressed 0,1...,5
        JNC CXCX5
        INC NUMBER1
```

```
        MOV NUMB3,KEY
        MOV VALUE_3,KEY
        AJMP KEYBOARD
CXCX3:  CJNE A,#04H,CXCX4
        MOV A,KEY
        CLR C
        SUBB A,#10H                 ; Chk Key Pressed 0,1,....,8,9
        JNC CXCX5
        INC NUMBER1
        MOV NUMB4,KEY
        MOV VALUE_4,KEY
CXCX5:  AJMP KEYBOARD
CXCX4:  CJNE A,#05H,CXCX5
        MOV A,KEY
        CJNE A,#12H,CXCX5           ;Key to OK TIME

        CALL FLASHING

        MOV KBELL,#00H
        MOV A,NUMB1
        SWAP A
        ORL A,NUMB2
        MOV NUMB2,A
        MOV A,NUMB3
        SWAP A
```

```
        ORL A,NUMB4
        MOV NUMB4,A


;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
;                   STORE THE TIME TO RTC CHIP
;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
        LCALL SEND_START            ; SEND 2WIRE START CONDITION
        MOV A,#DS1307W              ; LOAD DS1307 WRITE COMMAND
        LCALL SEND_BYTE             ; SEND WRITE COMMAND
        MOV A,#08H                  ; SET DS1307 DATA POINTER TO BEGINNING
        LCALL SEND_BYTE             ; OF USER RAM 08H
        MOV A,#0AAH                 ; WRITE BYTE TO ENTIRE RAM SPACE
        LCALL SEND_BYTE
        LCALL SEND_STOP             ; SEND 2WIRE STOP CONTION



        LCALL SEND_START            ; SEND 2WIRE START CONDITION
        MOV A,#DS1307W              ; LOAD DS1307 WRITE COMMAND
        LCALL SEND_BYTE             ; SEND WRITE COMMAND
        MOV A,#01H                  ; SET DS1307 DATA POINTER TO BEGINNING
        LCALL SEND_BYTE             ; OF 00H
        MOV A,NUMB4                 ; SET DS1307 DATA POINTER TO BEGINNING
        LCALL SEND_BYTE             ; OF 00H
```

```
        MOV A,NUMB2
        CLR ACC.6
        LCALL SEND_BYTE
        LCALL SEND_STOP                 ; SEND 2WIRE STOP CONTION

;*************************************************************************
**
;                   MAIN PROGRAM
;*************************************************************************
**
START_PROGRAM:
        CALL READ_CLOCK
        MOV R1,#25H                     ;GET MIN AND DISPLAY
        MOV A,@R1
        ANL A,#0FH
        MOV VALUE_4,A
        MOV R1,#25H
        MOV A,@R1
        ANL A,#0F0H
        SWAP A
        MOV VALUE_3,A
        MOV R1,#26H                     ;GET HOUR AND DISPLAY
        MOV A,@R1

        CLR C
        SUBB A,#12H
```

```
        JNC CCX
        MOV A,@R1
CCX:
        CJNE A,#00H,HHGH
        MOV A,#12H
HHGH:
        ANL A,#0FH
        MOV VALUE_2,A
        MOV R1,#26H
        MOV A,@R1

        CLR C
        SUBB A,#12H
        JNC CCX1
        MOV A,@R1
CCX1:
        CJNE A,#00H,HHGH1
        MOV A,#12H
HHGH1:
        ANL A,#0F0H
        SWAP A
        MOV VALUE_1,A
        CALL LOAD_ALRM

        CLR ROW4
```

```
        SETB COL2
        JB COL2,NEXT1
;(((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
;                   EMERGENCY BELL
;(((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
        SETB RELAY
        JNB COL2,$
        CLR RELAY
        AJMP START_PROGRAM
;(((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
;(((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
;
;(((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
NEXT1:      CLR ROW4
        SETB COL3
        JB COL3,START_PROGRAM
        CALL SQW_CONTROL_32KHZ
        MOV NUMBER1,#01H
        SETB CANCEL
        SETB CANCEL1
        MOV DPTR,#0001H

START_PROG:
        ORL  WMCON, #EEMEN              ; enable EEPROM accesses
        MOVX A,@DPTR
```

```
        CJNE A,#0FFH,TFT1
        MOV VALUE_1,#16H
        MOV VALUE_2,#16H
        AJMP TFT3
TFT1:   MOV R1,A                ;GET MIN AND DISPLAY
        ANL A,#0FH
        MOV VALUE_2,A
        MOV A,R1
        ANL A,#0F0H
        SWAP A
        MOV VALUE_1,A
TFT3:   INC DPTR
        MOVX A,@DPTR
        CJNE A,#0FFH,TFT2
        XRL WMCON, #EEMEN           ; disable EEPROM accesses
        MOV VALUE_3,#16H
        MOV VALUE_4,#16H
        JMP KEYBOARD
TFT2:
        MOV R1,A
        ANL A,#0FH
        MOV VALUE_4,A
        MOV A,R1
        ANL A,#0F0H
        SWAP A
```

```
        MOV VALUE_3,A
        JMP KEYBOARD

START_PM:
        CLR ROW4
        SETB COL3
        JNB COL3,$
        CALL DEC_DPTR                   ;store the count of timings
        CALL DEC_DPTR
        MOV A,DPL
        MOV DPTR,#0100H
        MOV WMCON,#18H
        MOVX @DPTR,A
CZTHD:
        MOV A,WMCON             ;Check for eeprom finished or not
        JNB ACC.1,CZTHD
        MOV WMCON,#08H
        CALL SQW_CONTROL_1HZ
        AJMP START_PROGRAM
;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
;               CHECK FOR TIME IS EQUAL
;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((

LOAD_ALRM:
        MOV DPTR,#0100H
```

```
        ORL  WMCON, #EEMEN              ; enable EEPROM accesses
        MOVX A,@DPTR
        MOV B,#02H
        DIV AB
        MOV R5,A
        MOV DPTR,#0001H

REPEAT:
        MOVX A,@DPTR
        MOV ALM_HOUR,A
        INC DPTR
        MOVX A,@DPTR
        MOV ALM_MIN,A
        INC DPTR
        MOV A,HRS
        CJNE A,ALM_HOUR,CHKK
        MOV A,MINS
        CJNE A,ALM_MIN,CHKK
        MOV A,SECS
        ANL A,#01111111B
        MOV SECS,A
        MOV A,#00H
        CJNE A,SECS,CHKK
                                        ;Time Is Equal
        JB ALARM,CHKK
```

```
        ORL TMOD,#10H           ;ENABLE TIMER 0
        MOV TL1,#08H
        MOV TH1,#01H
        SETB ET1
        MOV TIM,#100
        SETB TR1
        SETB RELAY
        SETB ALARM
CHKK:   DJNZ R5,REPEAT
        XRL WMCON, #EEMEN           ; disable EEPROM accesses
        RET



;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
KB_RET:
        MOV A,KEY
        CJNE A,#10H,CAXCX0      ;Key to Erase last dislled NUMBER1
        JB CANCEL,START_PM1
        MOV KEY,#00H
        MOV NUMBER1,#01H
        MOV VALUE_1,#15H
        MOV VALUE_2,#15H
        MOV VALUE_3,#15H
        MOV VALUE_4,#15H
```

```
        SETB CANCEL
        CLR CANCEL1
        AJMP KEYBOARD
START_PM1:
        AJMP START_PM
CAXCX0:
        CJNE A,#12H,CAXX5
        CLR CANCEL
        AJMP CAXCX5
CAXX5:
        MOV A,NUMBER1
        CJNE A,#01H,CAXCX1
        MOV A,KEY
        CLR C
        SUBB A,#03H                    ; Chk Key Pressed 0,1,2
        JNC CAXCX5
        MOV A,KEY
        INC NUMBER1
        MOV NUMB1,KEY
        MOV VALUE_1,KEY
        CLR CANCEL
        AJMP KEYBOARD
CAXCX1:     CJNE A,#02H,CAXCX2

        MOV A,NUMB1
```

```
        CJNE A,#02,JAKJL
        MOV A,KEY
        CLR C
        SUBB A,#04H                ; Chk Key Pressed 0,1,2,3
        JNC CAXCX5
JAKJL:  MOV A,KEY
        CLR C
        SUBB A,#10H                ; Chk Key Pressed 0,1...8,9
        JNC CAXCX5
        INC NUMBER1
        MOV NUMB2,KEY
        MOV VALUE_2,KEY
        CLR CANCEL
        AJMP KEYBOARD
CAXCX2: CJNE A,#03H,CAXCX3
        MOV A,KEY
        CLR C
        SUBB A,#06H                ; Chk Key Pressed 0,1...,5
        JNC CAXCX5
        INC NUMBER1
        MOV NUMB3,KEY
        MOV VALUE_3,KEY
        CLR CANCEL
        AJMP KEYBOARD
CAXCX3: CJNE A,#04H,CAXCX4
```

```
        MOV A,KEY
        CLR C
        SUBB A,#10H                     ; Chk Key Pressed 0,1,....,8,9
        JNC CAXCX5
        INC NUMBER1
        MOV NUMB4,KEY
        MOV VALUE_4,KEY
        CLR CANCEL
        SETB CANCEL1
CAXCX4:
        AJMP KEYBOARD
CAXCX5:
        JNB CANCEL1,CAXCX4

        CALL DEC_DPTR
        MOV A,VALUE_1
        SWAP A
        ORL A,VALUE_2
        MOV NUMB2,A
        MOV A,VALUE_3
        SWAP A
        ORL A,VALUE_4
        MOV NUMB4,A
```

```
        MOV WMCON,#18H
        MOV A,NUMB2
        MOVX @DPTR,A
CTHD:   MOV A,WMCON             ;Check for eeprom finished or not
        JNB ACC.1,CTHD
        INC DPTR
        MOV A,NUMB4
        MOVX @DPTR,A
CTTHD:  MOV A,WMCON             ;Check for eeprom finished or not
        JNB ACC.1,CTTHD
        INC DPTR
        MOV WMCON,#08H           ; DISable EEPROM WRITE



        AJMP START_PROG

;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
DEALAY:
        PUSH ACC
        MOV R1,#20
REPP2:  MOV     A,#0A6H
MD_OLP:
      INC     A
      NOP
```

```
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        JNZ        MD_OLP
        NOP
          DJNZ R1,REPP2
          POP ACC
          RET
;((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((
DEC_DPTR:
   XCH    A,DPL             ;Exchange A for DPL
   DEC    A                 ;Decrement A (which is DPL)
   CJNE   A,#0FFh,_dec_dptr2  ;If A (DPL) is not #0FFh, continue normally
   DEC    DPH               ;If A=FFh, we need to decrement DPH
_dec_dptr2:
   XCH    A,DPL             ;Exchange A for DPL (thus saving DPL and restoring A)
   RET

; ********************************************************
;                  DELAY TIMER FOR BELL
; ********************************************************
```

```
RELAY_TIMER:

        DJNZ TIM,GAHJ
        CLR TR1
        CLR RELAY
        CLR ALARM
        RETI
GAHJ: MOV TL1,#08H
        MOV TH1,#01H
        SETB TR1
        RETI

; ***********************************************************
; 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
        RET
; ************************************************************
; THIS SUB CONTROLS THE SQW OUTPUT 1HZ
; ************************************************************
SQW_CONTROL_1HZ:
          LCALL SEND_START        ; SEND START CONDITION
          MOV A,#DS1307W          ; SET POINTER TO REG 07H ON
                                  ; DS1307
          LCALL SEND_BYTE
          MOV A,#07H
```

```
        LCALL SEND_BYTE
        MOV A,#90H              ; SQW/OUT ON AT 1HZ
        JNB SQW,SQW_SET             ; JUMP IF SQW BIT IS ACTIVE
        MOV A,#80H              ; TURN SQW/OUT OFF – OFF HIGH
SQW_SET:
        LCALL SEND_BYTE
        LCALL SEND_STOP
        RET
;-------------------------------------------------------------------------
;
; THIS SUB CONTROLS THE SQW OUTPUT 32KHZ
;
;-------------------------------------------------------------------------
SQW_CONTROL_32KHZ:
        LCALL SEND_START        ; SEND START CONDITION
        MOV A,#DS1307W          ; SET POINTER TO REG 07H ON DS1307
        LCALL SEND_BYTE
        MOV A,#07H
        LCALL SEND_BYTE
        MOV A,#93H              ; SQW/OUT ON AT 1HZ
        JNB SQW,SQW_SET3        ; JUMP IF SQW BIT IS ACTIVE
        MOV A,#80H              ; TURN SQW/OUT OFF – OFF HIGH
SQW_SET3:
        LCALL SEND_BYTE
        LCALL SEND_STOP
```

```
        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   DEELAY
        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
; ************************************************************
DEELAY:
        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   DEELAY      ; DELAY FOR AN OPERATION
        CLR     SCL         ; TRANSITION SCL HIGH-TO-LOW
        ACALL   DEELAY      ; 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
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
;       7 SEGMENT DISPLAY ROUTINE
;&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
DISP:
        MOV R2,SPEED
        CJNE R2,#00H,AAS1
        CLR DIS_A
        CLR DIS_B
        CLR DIS_C
        CLR DIS_D
        CLR DIS_E
        CLR DIS_F
        SETB DIS_G
        RET
AAS1:   CJNE R2,#01H,AS2
        CLR DIS_B
        CLR DIS_C
        SETB DIS_A
```

```
        SETB DIS_D
        SETB DIS_E
        SETB DIS_F
        SETB DIS_G
        RET
AS2:    CJNE R2,#02H,AS3
        CLR DIS_A
        CLR DIS_B
        CLR DIS_D
        CLR DIS_E
        CLR DIS_G
        SETB DIS_C
        SETB DIS_F
        RET
AS3:    CJNE R2,#03H,AS4
        CLR DIS_A
        CLR DIS_B
        CLR DIS_C
        CLR DIS_D
        CLR DIS_G
        SETB DIS_E
        SETB DIS_F
        RET
AS4:    CJNE R2,#04H,AS5
        CLR DIS_B
```

```
        CLR DIS_C
        CLR DIS_F
        CLR DIS_G
        SETB DIS_A
        SETB DIS_D
        SETB DIS_E
        RET
AS5:    CJNE R2,#05H,AS6
        CLR DIS_A
        CLR DIS_C
        CLR DIS_D
        CLR DIS_F
        CLR DIS_G
        SETB  DIS_B
        SETB DIS_E
        RET
AS6:    CJNE R2,#06H,AS7
        CLR DIS_A
        CLR DIS_C
        CLR DIS_D
        CLR DIS_E
        CLR DIS_F
        CLR DIS_G
        SETB DIS_B
        RET
```

```
AS7:  CJNE R2,#07H,AS8
      CLR DIS_A
      CLR DIS_B
      CLR DIS_C
      SETB DIS_D
      SETB DIS_E
      SETB DIS_F
      SETB DIS_G
      RET
AS8:  CJNE R2,#08H,AS9
      CLR DIS_A
      CLR DIS_B
      CLR DIS_C
      CLR DIS_D
      CLR DIS_E
      CLR DIS_F
      CLR DIS_G
      RET
AS9:  CJNE R2,#09H,AS10
      CLR DIS_A
      CLR DIS_B
      CLR DIS_C
      CLR DIS_D
      CLR DIS_F
      CLR DIS_G
```

```
        SETB DIS_E
        RET
AS10:   CJNE R2,#15H,AS11       ;symbol for -
        SETB DIS_A
        SETB DIS_B
        SETB DIS_C
        SETB DIS_D
        SETB DIS_E
        SETB DIS_F
        CLR DIS_G
        RET
AS11:   CJNE R2,#16H,AS12       ;switch off all disp
        SETB DIS_A
        SETB DIS_B
        SETB DIS_C
        SETB DIS_D
        SETB DIS_E
        SETB DIS_F
        SETB DIS_G
        RET
AS12:   MOV SPEED,#00H
        AJMP DISP
;*************************************************************
;               INTRRUPT ROUTINE TO REFRESH THE DISPLAY
;*************************************************************
```

```
REFRESH:
        PUSH   PSW          ; save current registerset
      MOV    PSW,#RB1
      PUSH   ACC
        INC COUNT
        MOV R4,COUNT
QA1:  CJNE R4,#01H,QA2
        MOV SPEED,VALUE_1
        SETB DIS1
        CLR DIS2
        CLR DIS3
        CLR DIS4
        CALL DISP
        AJMP DOWN
QA2:  CJNE R4,#02H,QA3
        MOV SPEED,VALUE_2
        CLR DIS1
        SETB DIS2
        CLR DIS3
        CLR DIS4
        CALL DISP
        AJMP DOWN
QA3:  CJNE R4,#03H,QA4
        MOV SPEED,VALUE_3
        CLR DIS1
```

```
        CLR DIS2
        SETB DIS3
        CLR DIS4
        CALL DISP
        AJMP DOWN
QA4:    CJNE R4,#04H,QA5
        MOV SPEED,VALUE_4
        CLR DIS1
        CLR DIS2
        CLR DIS3
        SETB DIS4
        CALL DISP
        AJMP DOWN
QA5:    MOV COUNT,#01H
        MOV R4,COUNT
        AJMP QA1
DOWN:       MOV TL0,#0FFH
        MOV TH0,#0F0H
        POP    ACC
      POP    PSW
        RETI
;*************************************************************
FLASHING:
        CALL DELAY                      ;Display on/off for 2 times
        CALL DELAY
```

```
        MOV VALUE_1,#16H
        MOV VALUE_2,#16H
        MOV VALUE_3,#16H
        MOV VALUE_4,#16H
        CALL DELAY
        CALL DELAY
        MOV VALUE_1,NUMB1
        MOV VALUE_2,NUMB2
        MOV VALUE_3,NUMB3
        MOV VALUE_4,NUMB4
        CALL DELAY                  ;Display on-off for 2 times
        CALL DELAY
        MOV VALUE_1,#16H
        MOV VALUE_2,#16H
        MOV VALUE_3,#16H
        MOV VALUE_4,#16H
        CALL DELAY
        CALL DELAY
        MOV VALUE_1,NUMB1
        MOV VALUE_2,NUMB2
        MOV VALUE_3,NUMB3
        MOV VALUE_4,NUMB4
;*************************************************************
DELAY:
        MOV R1,#0CCH
```

```
REP2:  MOV R2,#0FFH
REP1:  NOP
       DJNZ R2,REP1
       DJNZ R1,REP2
       RET
END
```