```
;* FILENAME:     AT1307-D.ASM
;* DISPLAY:      6-DIGIT LED DISPLAY
;* MICRO:        ATMEL AT89C4051 FLASH BASED MICRO
;* PROJECT:      CLOCK
;* DATE:         11/2003
;* TIMEKEEPER:   DS1307 SERIAL TIMEKEEPER I.C.
;
$MOD51
SCL        BIT    P1.0
SDA        BIT    P1.1
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
DRAPEAU    DATA   21H
SETHOURS   BIT    DRAPEAU.1
SETMINS    BIT    DRAPEAU.2
SETSECS    BIT    DRAPEAU.3
SETCHOURS  BIT    DRAPEAU.4
SETCMINS   BIT    DRAPEAU.5
SETCSECS   BIT    DRAPEAU.6
SETDOFF    BIT    DRAPEAU.7    ; ALL DISPLAY OFF

BYTECNT    DATA   22H

SECS       DATA   24H      ;   '   SECONDS STORAGE RAM
MINS       DATA   25H      ;   '   MINUTES   '     '
HRS        DATA   26H      ;   '   HOURS     '     '
YEAR       DATA   2AH      ;   '   YEAR      '     '
BITCNT     DATA   27H
COUNTH     DATA   28H      ; COMPTEUR SUR DIGIT HEURES
COUNTS     DATA   29H      ; COMPTEUR SUR DIGIT SECONDES

SECSC_BUF  DATA   2BH      ;BUFFER CHRONO SECONDS
MINSC_BUF  DATA   2CH      ;BUFFER CHRONO MINUTES
HRSC_BUF   DATA   2DH      ;BUFFER CHRONO HOURS
DISBUF     DATA   2FH      ; DISPLAY BUFFER
DISBUF1    DATA   30H
DISBUF2    DATA   31H
DISBUF3    DATA   32H
DISBUF4    DATA   33H
DISBUF5    DATA   34H
COUNT      DATA   35H
MAINE      DATA   36H
SCORE      DATA   37H
SECSC      DATA   38H      ;   '   CHRONO       '     '
MINSC      DATA   39H      ;   '   CHRONO    '     '
HRSC       DATA   40H      ;   '   CHRONO    '     '

;
; ***MAX7219 CONTROL***
MACLK      EQU    P3.4     ; P3.4 MAX7219 CLOCK
MALDB      EQU    P1.3     ; P1.3 MAX7219 LOAD
MADAT      EQU    P3.7     ; P3.7 MAX7219 DATA

; ***BUZZER PIN***
BEEPER     EQU    P3.5     ;P3.5  BUZZER PIN

; ***USER INPUT KEYS***

UP_BTTN    BIT    0B0H     ; P3.0  UP
DOWN_BTTN  BIT    0B1H     ; P3.1  DOWN
ENTER      BIT    0B2H     ; P3.2  ENTER
BP4        BIT    092H     ; P1.2  ALARM TIME SET SELECT
SWMODE     BIT    094H     ; P1.4  JUMPER MODE SELECT
SWMODE2    BIT    095H     ; P1.5  JUMPER MODE SELECT2
; ***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
          MOV          SP,#42H    ; 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
          CLR          BEEPER
          CLR          MACLK
          CLR          MALDB
          MOV          DRAPEAU,#0
          ACALL        MXSET
          JNB          SWMODE,COUNT_MODES
          JNB          SWMODE2,COUNT_MODE3
          JMP          OSC_CONTROL
COUNT_MODES:
          JNB          SWMODE2,COUNTMODE2
;
          JMP          COUNTMODE
COUNT_MODE3:
          JMP          COUNTMODE3

COUNTMODE2:
          MOV          A,#7FH
          MOV          DISBUF,A
          MOV          DISBUF1,A
          MOV          DISBUF2,A
          MOV          DISBUF3,A
          MOV          DISBUF4,A
          MOV          DISBUF5,A
          MOV          COUNTH,#0
          MOV          COUNTS,#0
          ACALL        MXLOAD
          ACALL        DISP_COUNTS
          ACALL        DISP_COUNTH


WAIT_COUNTHS:                       ; WAIT HERE UNTIL BUTTON PRESS
          JNB          UP_BTTN,UP_COUNTH
          JNB          DOWN_BTTN,DOWN_COUNTH
          JNB          ENTER,UP_COUNTS
          JNB          BP4,DOWN_COUNTS
          SJMP         WAIT_COUNTHS

UP_COUNTH:
          JNB          UP_BTTN,UP_COUNTH
          ACALL        MDELAY
          MOV          A,COUNTH
          ADD          A,#1
          MOV          COUNTH,A
          SJMP         RAZ1H
RELEASE_UPH:
          JNB          DOWN_BTTN,RAZ_COUNT_UPH
          ACALL        MDELAY
RAZ1H:    JNB          UP_BTTN,RELEASE_UPH; HOLDS HERE UNTIL BUTTON IS RELEASED
          ACALL        MDELAY
```

```
         ACALL       DISP_COUNTH
         SJMP        WAIT_COUNTHS

DOWN_COUNTH:
         JNB         DOWN_BTTN,DOWN_COUNTH
         ACALL       MDELAY
         MOV         A,COUNTH
         JNZ         ZEROMH
         MOV         A,#64H
ZEROMH:  CLR         C
         SUBB        A,#1
         MOV         COUNTH,A
         SJMP        RAZ2H
RELEASE_DOWNH:
         JNB         UP_BTTN,RAZ_COUNT_DOWNH
         ACALL       MDELAY
RAZ2H:   JNB         DOWN_BTTN,RELEASE_DOWNH; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_COUNTH
         SJMP        WAIT_COUNTHS

RAZ_COUNT_DOWNH:
         MOV         COUNTH,#0
         ACALL       DISP_COUNTH
         SJMP        RELEASE_DOWNH

RAZ_COUNT_UPH:
         MOV         COUNTH,#0
         ACALL       DISP_COUNTH
         SJMP        RELEASE_UPH


UP_COUNTS:
         JNB         ENTER,UP_COUNTS
         ACALL       MDELAY
         MOV         A,COUNTS
         ADD         A,#1
         MOV         COUNTS,A
         SJMP        RAZ1S
RELEASE_UPS:
         JNB         BP4,RAZ_COUNT_UPS
         ACALL       MDELAY
RAZ1S:   JNB         ENTER,RELEASE_UPS; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY
         ACALL       DISP_COUNTS
         SJMP        WAIT_COUNTHS

DOWN_COUNTS:
         JNB         BP4,DOWN_COUNTS
         ACALL       MDELAY
         MOV         A,COUNTS
         JNZ         ZEROMS
         MOV         A,#64H
ZEROMS:  CLR         C
         SUBB        A,#1
         MOV         COUNTS,A
         SJMP        RAZ2S
RELEASE_DOWNS:
         JNB         ENTER,RAZ_COUNT_DOWNS
         ACALL       MDELAY
RAZ2S:   JNB         BP4,RELEASE_DOWNS; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_COUNTS
         JMP         WAIT_COUNTHS

RAZ_COUNT_DOWNS:
         MOV         COUNTS,#0
         ACALL       DISP_COUNTS
         SJMP        RELEASE_DOWNS

RAZ_COUNT_UPS:
         MOV         COUNTS,#0
```

```
         ACALL       DISP_COUNTS
         SJMP        RELEASE_UPS

COUNTMODE3:
         MOV         A,#7FH
         MOV         DISBUF,A
         MOV         DISBUF1,A
         MOV         DISBUF2,A
         MOV         DISBUF3,A
         MOV         DISBUF4,A
         MOV         DISBUF5,A
         MOV         COUNTH,#1
         MOV         COUNTS,#1
         ACALL       MXLOAD
         ACALL       DISP_COUNTS
         ACALL       DISP_COUNTH


WAIT_COUNTHS3:                    ; WAIT HERE UNTIL BUTTON PRESS
         JNB         UP_BTTN,UP_COUNTH3
         JNB         DOWN_BTTN,DOWN_COUNTH3
         JNB         ENTER,UP_COUNTS3
         JNB         BP4,DOWN_COUNTS3
         SJMP        WAIT_COUNTHS3

UP_COUNTH3:
         JNB         UP_BTTN,UP_COUNTH3
         ACALL       MDELAY
         MOV         A,COUNTH
         CJNE        A,#8,RAZ83
         MOV         COUNTH,#1
         SJMP        RAZ1H3
RAZ83:   ADD         A,#1
         MOV         COUNTH,A
         SJMP        RAZ1H3
RELEASE_UPH3:
         JNB         DOWN_BTTN,RAZ_COUNT_UPH3
         ACALL       MDELAY
RAZ1H3:   JNB         UP_BTTN,RELEASE_UPH3; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY
         ACALL       DISP_COUNTH
         SJMP        WAIT_COUNTHS3

DOWN_COUNTH3:
         JNB         DOWN_BTTN,DOWN_COUNTH3
         ACALL       MDELAY
         MOV         A,COUNTH
         CJNE        A,#1,ZEROMH3
         MOV         A,#9H
ZEROMH3:  CLR         C
         SUBB        A,#1
         MOV         COUNTH,A
         SJMP        RAZ2H3
RELEASE_DOWNH3:
         JNB         UP_BTTN,RAZ_COUNT_DOWNH3
         ACALL       MDELAY
RAZ2H3:   JNB         DOWN_BTTN,RELEASE_DOWNH3; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_COUNTH
         SJMP        WAIT_COUNTHS3

RAZ_COUNT_DOWNH3:
         MOV         COUNTH,#1
         ACALL       DISP_COUNTH
         SJMP        RELEASE_DOWNH3

RAZ_COUNT_UPH3:
         MOV         COUNTH,#1
         ACALL       DISP_COUNTH
         SJMP        RELEASE_UPH3


UP_COUNTS3:
```

```
         JNB         ENTER,UP_COUNTS3
         ACALL       MDELAY
         MOV         A,COUNTS
         CJNE        A,#8,RAZ8S3
         MOV         COUNTS,#1
         SJMP        RAZ1S3
RAZ8S3:  ADD         A,#1
         MOV         COUNTS,A
         SJMP        RAZ1S3
RELEASE_UPS3:
         JNB         BP4,RAZ_COUNT_UPS3
         ACALL       MDELAY
RAZ1S3:  JNB         ENTER,RELEASE_UPS3; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY
         ACALL       DISP_COUNTS
         JMP         WAIT_COUNTHS3

DOWN_COUNTS3:
         JNB         BP4,DOWN_COUNTS3
         ACALL       MDELAY
         MOV         A,COUNTS
         CJNE        A,#1,ZEROMS3
         MOV         A,#9H
ZEROMS3: CLR         C
         SUBB        A,#1
         MOV         COUNTS,A
         SJMP        RAZ2S3
RELEASE_DOWNS3:
         JNB         ENTER,RAZ_COUNT_DOWNS3
         ACALL       MDELAY
RAZ2S3:   JNB         BP4,RELEASE_DOWNS3; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_COUNTS
         JMP         WAIT_COUNTHS3

RAZ_COUNT_DOWNS3:
         MOV         COUNTS,#1
         ACALL       DISP_COUNTS
         SJMP        RELEASE_DOWNS3

RAZ_COUNT_UPS3:
         MOV         COUNTS,#1
         ACALL       DISP_COUNTS
         SJMP        RELEASE_UPS3


;



COUNTMODE:
         MOV         A,#7FH
         MOV         DISBUF,A
         MOV         DISBUF1,A
         MOV         DISBUF2,A
         MOV         DISBUF3,A
         MOV         DISBUF4,A
         MOV         DISBUF5,A
         MOV         COUNT,#0
         ACALL       MXLOAD
         ACALL       DISP_COUNT



WAIT_COUNT:                      ; WAIT HERE UNTIL BUTTON PRESS
         JNB         UP_BTTN,UP_COUNT
         JNB         DOWN_BTTN,DOWN_COUNT
         JNB         ENTER,MAINE_COUNT
         SJMP        WAIT_COUNT

UP_COUNT:
         JNB         UP_BTTN,UP_COUNT
         ACALL       MDELAY
```

```
         MOV         A,COUNT
         ADD         A,#1
         MOV         COUNT,A
         SJMP        RAZ1
RELEASE_UP:
         JNB         DOWN_BTTN,RAZ_COUNT_UP
         ACALL       MDELAY
RAZ1:    JNB         UP_BTTN,RELEASE_UP; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY
         ACALL       DISP_COUNT
         SJMP        WAIT_COUNT

DOWN_COUNT:
         JNB         DOWN_BTTN,DOWN_COUNT
         ACALL       MDELAY
         MOV         A,COUNT
         JNZ         ZEROM
         MOV         A,#64H
ZEROM:   CLR         C
         SUBB        A,#1
         MOV         COUNT,A
         SJMP        RAZ2
RELEASE_DOWN:
         JNB         UP_BTTN,RAZ_COUNT_DOWN
         ACALL       MDELAY
RAZ2:    JNB         DOWN_BTTN,RELEASE_DOWN; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_COUNT
         SJMP        WAIT_COUNT

RAZ_COUNT_DOWN:
         MOV         COUNT,#0
         ACALL       DISP_COUNT
         SJMP        RELEASE_DOWN

RAZ_COUNT_UP:
         MOV         COUNT,#0
         ACALL       DISP_COUNT
         SJMP        RELEASE_UP
RETOUR_MAINE:
         JNB         BP4,RETOUR_MAINE
         ACALL       MDELAY
         SJMP        MAINE_SUITE

MAINE_COUNT:
         JNB         ENTER,MAINE_COUNT
         ACALL       MDELAY

         MOV         MAINE,#1
MAINE_SUITE:
         ACALL       DISP_MAINE


WAIT_MAINE:
         JNB         ENTER,WAIT_MAINE
         ACALL       MDELAY
         JNB         UP_BTTN,UP_MAINE
         JNB         DOWN_BTTN,DOWN_MAINE
         JNB         ENTER,SCORE_COUNT
         SJMP        WAIT_MAINE

UP_MAINE:
         JNB         UP_BTTN,UP_MAINE
         ACALL       MDELAY
         MOV         A,MAINE
         CJNE        A,#8,RAZ8
         MOV         MAINE,#1
         SJMP        RAZ1MAINE
RAZ8:    ADD         A,#1
         MOV         MAINE,A
         SJMP        RAZ1MAINE

RELEASE_UPM:
```

```
        JNB         DOWN_BTTN,RAZ_COUNT_UPM
        ACALL       MDELAY
RAZ1MAINE:
        JNB         UP_BTTN,RELEASE_UPM; HOLDS HERE UNTIL BUTTON IS RELEASED
        ACALL       MDELAY
        ACALL       DISP_MAINE
        SJMP        WAIT_MAINE

DOWN_MAINE:
        JNB         DOWN_BTTN,DOWN_MAINE
        ACALL       MDELAY
        MOV         A,MAINE
        CJNE        A,#1,ZEROM2
        MOV         A,#9H
ZEROM2:   CLR        C
        SUBB        A,#1
        MOV         MAINE,A
        SJMP        RAZ2M
RELEASE_DOWNM:
        JNB         UP_BTTN,RAZ_COUNT_DOWNM
        ACALL       MDELAY
RAZ2M:  JNB         DOWN_BTTN,RELEASE_DOWNM; HOLDS HERE UNTIL BUTTON IS RELEASED
        ACALL       MDELAY

        ACALL       DISP_MAINE
        SJMP        WAIT_MAINE

RAZ_COUNT_DOWNM:
        MOV         MAINE,#0
        ACALL       DISP_MAINE
        SJMP        RELEASE_DOWNM

RAZ_COUNT_UPM:
        MOV         MAINE,#0
        ACALL       DISP_MAINE
        SJMP        RELEASE_UPM

SCORE_COUNT:
        JNB         ENTER,SCORE_COUNT
        ACALL       MDELAY
        MOV         SCORE,#0
        ACALL       DISP_SCORE

WAIT_SCORE:
        JNB         ENTER,WAIT_SCORE
        ACALL       MDELAY
        JNB         UP_BTTN,UP_SCORE
        JNB         DOWN_BTTN,DOWN_SCORE
        JNB         ENTER,RESULTAT
        SJMP        WAIT_SCORE

RESULTAT:
        JNB         ENTER,RESULTAT
        ACALL       MDELAY
RETOUR_RESULTAT:
        ACALL       DISP_SCORE
        MOV         R0,#25
RES1:   ACALL       DELAY_1SRES
        DJNZ        R0,RES1
        ACALL       DISP_MAINE
        MOV         R0,#5
RES2:   ACALL       DELAY_1SRES
        DJNZ        R0,RES2

        SJMP        RETOUR_RESULTAT

DISP_MAINE_ONLY:
        ACALL       DISP_MAINE
        JNB         BP4,RETOUR_MAINE1
        SJMP        DISP_MAINE_ONLY

RETOUR_MAINE1:      JMP RETOUR_MAINE
```

```
UP_SCORE:
         JNB         UP_BTTN,UP_SCORE
         ACALL       MDELAY
         MOV         A,SCORE
         ADD         A,#1
         MOV         SCORE,A
         SJMP        RAZ1US
RELEASE_UPUS:
         JNB         DOWN_BTTN,RAZ_SCORE_UP
         ACALL       MDELAY
RAZ1US:  JNB         UP_BTTN,RELEASE_UPUS; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY
         ACALL       DISP_SCORE
         JMP         WAIT_SCORE

DOWN_SCORE:
         JNB         DOWN_BTTN,DOWN_SCORE
         ACALL       MDELAY
         MOV         A,SCORE
         JNZ         ZEROM3
         MOV         A,#64H
ZEROM3:   CLR         C
         SUBB        A,#1
         MOV         SCORE,A
         JMP         RAZ2US
RELEASE_DOWNUS:
         JNB         UP_BTTN,RAZ_SCORE_DOWN
         ACALL       MDELAY
RAZ2US:  JNB         DOWN_BTTN,RELEASE_DOWNUS; HOLDS HERE UNTIL BUTTON IS RELEASED
         ACALL       MDELAY

         ACALL       DISP_SCORE
         JMP        WAIT_SCORE

RAZ_SCORE_DOWN:
         MOV         SCORE,#0
         ACALL       DISP_SCORE
         SJMP        RELEASE_DOWNUS

RAZ_SCORE_UP:
         MOV         SCORE,#0
         ACALL       DISP_SCORE
         SJMP        RELEASE_UPUS


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

OSC_CONTROL:
         MOV         A,#00000000B
         MOV         SECSC,A
         MOV         HRSC,A
         MOV         MINSC,A
         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

; *********************************************************
; 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         BP4,GET_CHRONO

; *********************************************************
; NOW GET & DISPLAY TIME
; *********************************************************
         ACALL       DISPMAX
         SJMP        GET_TIME
GET_CHRONO:
         JNB         BP4,GET_CHRONO
         ACALL       MDELAY
         JMP         SET_CHRONO



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


;SET_HOUR:
         SETB        SETHOURS

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
         CLR         SETHOURS
         SETB        SETMINS

;SET_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
         CLR         SETMINS

;SET_SECONDS:
         ACALL       MDELAY
         MOV         A,#00000000B ; SECONDS ARE SET TO 00
         MOV         SECS,A
         ACALL       SET_CLOCK
         AJMP        GET_TIME    ; RETURN TO MAIN LOOP

UPDATE:

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

         ACALL       DISPMAX
; *********************************************************
; 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


; *********************************************************
; 1 MILLISECOND c ROUTINE
; *********************************************************

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



MXSET:   CLR         MACLK       ; FALLING CLOCK EDGE
         CLR         MALDB
         MOV         R0,#0FH     ; DISPLAY TEST -NORMAL (XXXXXXX0)
         MOV         R1,#00H
         ACALL       MXBYTE

         MOV         R0,#0CH     ; SHUTDOWN -NORMAL (XXXXXXX1)
         MOV         R1,#01H
         ACALL       MXBYTE

         MOV         R0,#0BH     ; SCAN LIMIT - 6 DIGIT (05H)
         MOV         R1,#05H
         ACALL       MXBYTE

         MOV         R0,#09H     ; DECODE MODE - W/  DECODE
         MOV         R1,#0FFH
         ACALL       MXBYTE

         MOV         R0,#05H     ; ALL SEGMENTS OFF
CLEAR:   MOV         R1,#' '
         ACALL       MXBYTE
         DJNZ        R0,CLEAR

         MOV         R0,#0AH     ; INTENSITY - MIN-MAX (00-0FH)
         MOV         R1,#0FH
         ACALL       MXBYTE
         RET

;************** MXBYTE SUB *****************
; SEND ADDRESS,DATA TO MAX7219
; IN = R0 ADDRESS (B0-B3)
;    = R1 DATA
; REG = A,R0,R1,R2

MXBYTE:  MOV         R2,#8       ; SEND ADDRESS
         MOV         A,R0
MXBYTE1: RLC         A
         MOV         MADAT,C
         SETB        MACLK
         CLR         MACLK
         DJNZ        R2,MXBYTE1

         MOV         R2,#8       ; SEND DATA
         MOV         A,R1
MXBYTE2: RLC         A
         MOV         MADAT,C
         SETB        MACLK
```

```
        CLR         MACLK
        DJNZ        R2,MXBYTE2

        SETB        MALDB       ; LOAD BIT
        CLR         MALDB
        RET

;*********** MXLOAD SUB ***************
; LOAD DISBUF TO DISPLAY (MAX7219)
; IN= DISBUF
; REG A,R0,R1,R2

MXLOAD: MOV         R0,#1       ; SEND 0
        MOV         R1,DISBUF
        ACALL       MXBYTE
        MOV         R0,#2       ; SEND 1
        MOV         R1,DISBUF1
        ACALL       MXBYTE
        MOV         R0,#3       ; SEND 2
        MOV         R1,DISBUF2
        ACALL       MXBYTE
        MOV         R0,#4       ; SEND 3
        MOV         R1,DISBUF3
        ACALL       MXBYTE
        MOV         R0,#5       ; SEND 4
        MOV         R1,DISBUF4
        ACALL       MXBYTE
        MOV         R0,#6       ; SEND 5
        MOV         R1,DISBUF5
        ACALL       MXBYTE
        RET

HEXASC: ANL         A,#0FH
        ADD         A,#90H
        DA          A
        ADDC        A,#40H
        DA          A
        RET

DISPMAX:
        MOV         A,HRS
        PUSH        ACC
        SWAP        A
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF,A
        POP         ACC
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF1,A
        MOV         A,MINS
        PUSH        ACC
        SWAP        A
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF2,A
        POP         ACC
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF3,A
        MOV         A,SECS
        PUSH        ACC
        SWAP        A
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF4,A
        POP         ACC
        ANL         A,#0FH
        ACALL       HEXASC
        MOV         DISBUF5,A
        JB          SETHOURS,HOURS_ONLY
        JB          SETMINS,MINS_ONLY
        SJMP        NORMAL
HOURS_ONLY:
```

```
        MOV     A,#7FH
        MOV     DISBUF2,A
        MOV     DISBUF3,A
        MOV     DISBUF4,A
        MOV     DISBUF5,A
        SJMP    NORMAL
MINS_ONLY:
        MOV     A,#7FH
        MOV     DISBUF,A
        MOV     DISBUF1,A
        MOV     DISBUF4,A
        MOV     DISBUF5,A
NORMAL: ACALL   MXLOAD
        RET

DISP_CHRONO:
        MOV     A,HRSC
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF1,A
        MOV     A,HRSC
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF,A
        MOV     A,MINSC
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF3,A
        MOV     A,MINSC
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF2,A
        MOV     A,SECSC
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF5,A
        MOV     A,SECSC
```

```
         MOV         R7,A
         MOV         R6,#0
         MOV         R5,#0
         MOV         R4,#0
         ACALL       BIN2BCD
         MOV         A,R3
         SWAP        A
         ANL         A,#0FH
         MOV         R0,A
         ACALL       HEXASC
         MOV         DISBUF4,A
         JB          SETCHOURS,CHOURS_ONLY
         JB          SETCMINS,CMINS_ONLY
         JB          SETCSECS,CSECS_ONLY
         JB          SETDOFF,SET_DISPLAYS_OFF
         SJMP        NORMALC
CHOURS_ONLY:
         MOV         A,#7FH
         MOV         DISBUF2,A
         MOV         DISBUF3,A
         MOV         DISBUF4,A
         MOV         DISBUF5,A
         SJMP        NORMALC
CMINS_ONLY:
         MOV         A,#7FH
         MOV         DISBUF,A
         MOV         DISBUF1,A
         MOV         DISBUF4,A
         MOV         DISBUF5,A
         SJMP        NORMALC
CSECS_ONLY:
         MOV         A,#7FH
         MOV         DISBUF,A
         MOV         DISBUF1,A
         MOV         DISBUF2,A
         MOV         DISBUF3,A
         SJMP        NORMALC

SET_DISPLAYS_OFF:
         MOV         A,#7FH
         MOV         DISBUF,A
         MOV         DISBUF1,A
         MOV         DISBUF2,A
         MOV         DISBUF3,A
         MOV         DISBUF4,A
         MOV         DISBUF5,A

NORMALC: ACALL       MXLOAD
         RET


BCD_HEX: MOV         A,R0
         SWAP        A
         ANL         A,#0FH
         MOV         B,#10
         MUL         AB
         MOV         A,R0
         ANL         A,#0FH
         ADD         A,B
         RET

BIN2BCD:
        CLR     A               ; R,sultat BCD … 0 :
        MOV     R0,A            ; ACC:R0:R1:R2:R3 = 0
        MOV     R1,A
        MOV     R2,A
        MOV     R3,A
        MOV     B,#32           ; compteur
BIN2B0: ACALL    DBLBCD           ; multiplie la valeur BCD par deux
        ACALL    RLC4567          ; C <- << R4:R5:R6:R7
        JNC     BIN2B1          ; si retenue ...
         ACALL    INCBCD           ; incr,mente la valeur BCD
BIN2B1: DJNZ    B,BIN2B0
        RET
```

```
INCBCD: XCH     A,R3            ; l'accumulateur est sauvegard,
        ADD     A,#1            ; additionne la retenue pr,c,dente
        DA      A               ; ajustement d,cimal sur le digit suivant
        XCH     A,R3            ; l'accu est restaur,
        XCH     A,R2            ; et sauvegard, de nouveau
        ADDC    A,#0            ; propage la retenue jusqu'au MSB
        DA      A
        XCH     A,R2
        XCH     A,R1
        ADDC    A,#0
        DA      A
        XCH     A,R1
        XCH     A,R0
        ADDC    A,#0
        DA      A
        XCH     A,R0            ; restaure enfin l'accumulateur
        ADDC    A,#0            ; et propage la derniŠre retenue
        DA      A
        RET

RLC4567:
        XCH     A,R7            ; sauvegarde A pendant
        RLC     A               ; le d,calage de R7
        XCH     A,R7            ; restaure A
        XCH     A,R6            ; et continue,
        RLC     A
        XCH     A,R6
        XCH     A,R5
        RLC     A
        XCH     A,R5
        XCH     A,R4
        RLC     A
        XCH     A,R4            ; jusqu'… pousser le dernier bit dans C
        RET

DBLBCD: XCH     A,R3
        ADD     A,ACC           ; multiplie par deux
        DA      A               ; ajustement d,cimal sur le digit suivant
        XCH     A,R3
        XCH     A,R2            ; multiplie par deux les deux digits
        ADDC    A,ACC
        DA      A
        XCH     A,R2
        XCH     A,R1
        ADDC    A,ACC
        DA      A
        XCH     A,R1
        XCH     A,R0
        ADDC    A,ACC
        DA      A
        XCH     A,R0
        ADDC    A,ACC           ; jusqu'au deux derniers digits
        DA      A
        RET

DISP_MAINE:
         MOV        A,MAINE
         MOV        R7,A
         MOV        R6,#0
         MOV        R5,#0
         MOV        R4,#0
         ACALL      BIN2BCD
         MOV        A,R3
         ANL        A,#0FH
         MOV        R0,A
         ACALL      HEXASC
         MOV        DISBUF1,A
         MOV        A,MAINE
         MOV        R7,A
         MOV        R6,#0
         MOV        R5,#0
         MOV        R4,#0
```

```
        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF,A
        ACALL   MXLOAD
        RET

DISP_COUNT:
        MOV     A,COUNT
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF1,A
        MOV     A,COUNT
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0

        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF,A
        ACALL   MXLOAD
        RET

DISP_COUNTH:
        MOV     A,COUNTH
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF1,A
        MOV     A,COUNTH
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0

        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF,A
        ACALL   MXLOAD
        RET


DISP_COUNTS:
        MOV     A,COUNTS
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0
        ACALL   BIN2BCD
```

```
        MOV     A,R3
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF5,A
        MOV     A,COUNTS
        MOV     R7,A
        MOV     R6,#0
        MOV     R5,#0
        MOV     R4,#0

        ACALL   BIN2BCD
        MOV     A,R3
        SWAP    A
        ANL     A,#0FH
        MOV     R0,A
        ACALL   HEXASC
        MOV     DISBUF4,A
        ACALL   MXLOAD
        RET



DELAY_50US:
        MOV     R6,#00CH
DELAY_50US_1:
        NOP
        NOP
        DJNZ    R6,DELAY_50US_1
        RET
DELAY_100US:
        MOV     R6,#017H
DELAY_100US_1:
        NOP
        NOP
        DJNZ    R6,DELAY_100US_1
        RET
DELAY_1MS:
        MOV     R6,#0E6H
DELAY_1MS_1:
        NOP
        NOP
        DJNZ    R6,DELAY_1MS_1
        RET
DELAY_10MS:
        MOV     R7,#10
DELAY_10MS_1:
        MOV     R6,#0E6H
DELAY_10MS_2:
        NOP
        NOP
        DJNZ    R6,DELAY_10MS_2
        DJNZ    R7,DELAY_10MS_1
        RET
DELAY_100MS:
        MOV     R7,#100
DELAY_100MS_1:
        MOV     R6,#0E6H
DELAY_100MS_2:
        NOP
        NOP
        DJNZ    R6,DELAY_100MS_2
        DJNZ    R7,DELAY_100MS_1
        RET
DELAY_1S:
        MOV     R5,#99
DELAY_1S_1:
        ACALL   DELAY_10MS
        DJNZ    R5,DELAY_1S_1
        RET

DELAY_1SRES:
        MOV     R5,#100
```

```
DELAY_1S_1RES:
        ACALL     DELAY_10MS
        JNB       BP4,AFFICHE_MAINE
        DJNZ      R5,DELAY_1S_1RES
        RET

AFFICHE_MAINE:     JNB BP4,AFFICHE_MAINE
                   ACALL MDELAY
                   JMP DISP_MAINE_ONLY
DELAY_500MS:
        MOV       R5,#50
DELAY_500MS_1:
        ACALL     DELAY_10MS
        DJNZ      R5,DELAY_500MS_1
        RET


DISP_SCORE:
        MOV        A,SCORE
        MOV        R7,A
        MOV        R6,#0
        MOV        R5,#0
        MOV        R4,#0
        LCALL      BIN2BCD
        MOV        A,R3
        ANL        A,#0FH
        MOV        R0,A
        LCALL      HEXASC
        MOV        DISBUF1,A
        MOV        A,SCORE
        MOV        R7,A
        MOV        R6,#0
        MOV        R5,#0
        MOV        R4,#0
        LCALL      BIN2BCD
        MOV        A,R3
        SWAP       A
        ANL        A,#0FH
        MOV        R0,A
        LCALL      HEXASC
        MOV        DISBUF,A
        LCALL      MXLOAD
        RET

SET_CHRONO:
        LCALL      DISP_CHRONO
SET_CHRONO2:
        JNB        ENTER,WAIT_CHRONO; HOLDS HERE UNTIL BUTTON IS RELEASED
        JNB        BP4,RETOUR_HORLOGE
        JNB        UP_BTTN,RESTART_CHRONO
        LCALL      MDELAY
        SJMP       SET_CHRONO2
RETOUR_HORLOGE:
        JNB        BP4,RETOUR_HORLOGE
        LCALL      MDELAY
        JMP        GET_TIME

RESTART_CHRONO:
        JMP        CLIGN_CHRONO

WAIT_CHRONO:                    ; WAIT HERE UNTIL BUTTON PRESS
        JNB        ENTER,WAIT_CHRONO
        LCALL      MDELAY
        SETB       SETCHOURS
        LCALL      DISP_CHRONO

        JNB        UP_BTTN,UP_CHRONO
        JNB        DOWN_BTTN,DOWN_CHRONO
        JNB        ENTER,WAIT_CHRONOM
        JNB        BP4,RETOUR1
        LCALL      MDELAY
        SJMP       WAIT_CHRONO
RETOUR1:                        ; WAIT HERE UNTIL BUTTON PRESS
        CLR        SETCHOURS
```

```
         SJMP        SET_CHRONO

UP_CHRONO:
         JNB         UP_BTTN,UP_CHRONO
         LCALL       MDELAY
         MOV         A,HRSC
         CJNE        A,#23,RAZ24
         MOV         A,#0
         SJMP        RAZ24_1
RAZ24:   INC         A
RAZ24_1: MOV         HRSC,A
         SJMP        RAZ1_CHRONO
         MOV         A,#0
         MOV         HRSC,A
         SJMP        RAZ1_CHRONO
RELEASE_UP_CHRONO:
         JNB         DOWN_BTTN,RAZ_CHRONO_UP
         LCALL       MDELAY
RAZ1_CHRONO:
         JNB         UP_BTTN,RELEASE_UP_CHRONO; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY
         LCALL       DISP_CHRONO
         SJMP        WAIT_CHRONO

DOWN_CHRONO:
         JNB         DOWN_BTTN,DOWN_CHRONO
         LCALL       MDELAY
         MOV         A,HRSC
         JNZ         ZEROM_CHRONO
         MOV         A,#24
ZEROM_CHRONO:
         DEC         A
         MOV         HRSC,A
         SJMP        RAZ2_CHRONO
RELEASE_DOWN_CHRONO:
         JNB         UP_BTTN,RAZ_CHRONO_DOWN
         LCALL       MDELAY
RAZ2_CHRONO:
         JNB         DOWN_BTTN,RELEASE_DOWN_CHRONO; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY

         LCALL       DISP_CHRONO
         JMP         WAIT_CHRONO

RAZ_CHRONO_DOWN:
         MOV         HRSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_DOWN_CHRONO

RAZ_CHRONO_UP:
         MOV         HRSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_UP_CHRONO

WAIT_CHRONOM:                    ; WAIT HERE UNTIL BUTTON PRESS
         JNB         ENTER,WAIT_CHRONOM
         LCALL       MDELAY
         CLR         SETCHOURS
         SETB        SETCMINS
         LCALL       DISP_CHRONO

WAIT_CHRONOM2:
         JNB         ENTER,WAIT_CHRONOM2
         LCALL       MDELAY

         JNB         UP_BTTN,UP_CHRONOM
         JNB         DOWN_BTTN,DOWN_CHRONOM
         JNB         ENTER,WAIT_CHRONOS
         JNB         BP4,RETOUR2
         LCALL       MDELAY
         SJMP        WAIT_CHRONOM2

RETOUR2: JNB         BP4,RETOUR2
         LCALL       MDELAY
```

```
         CLR         SETCMINS
         JMP         WAIT_CHRONO

UP_CHRONOM:
         JNB         UP_BTTN,UP_CHRONOM
         LCALL       MDELAY
         MOV         A,MINSC
         CJNE        A,#59,RAZ59M
         MOV         A,#0
         SJMP        RAZ59M_1
RAZ59M:  INC         A
         MOV         MINSC,A
         SJMP        RAZ1_CHRONOM
         MOV         A,#0
RAZ59M_1:
         MOV         MINSC,A
         SJMP        RAZ1_CHRONOM
RELEASE_UP_CHRONOM:
         JNB         DOWN_BTTN,RAZ_CHRONO_UPM
         LCALL       MDELAY
RAZ1_CHRONOM:
         JNB         UP_BTTN,RELEASE_UP_CHRONOM; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY
         LCALL       DISP_CHRONO
         SJMP        WAIT_CHRONOM

DOWN_CHRONOM:
         JNB         DOWN_BTTN,DOWN_CHRONOM
         LCALL       MDELAY
         MOV         A,MINSC
         JNZ         ZEROM_CHRONOM
         MOV         A,#60
ZEROM_CHRONOM:
         DEC         A
         MOV         MINSC,A
         SJMP        RAZ2_CHRONOM
RELEASE_DOWN_CHRONOM:
         JNB         UP_BTTN,RAZ_CHRONO_DOWNM
         LCALL       MDELAY
RAZ2_CHRONOM:
         JNB         DOWN_BTTN,RELEASE_DOWN_CHRONOM; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY

         LCALL       DISP_CHRONO
         SJMP        WAIT_CHRONOM

RAZ_CHRONO_DOWNM:
         MOV         MINSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_DOWN_CHRONOM

RAZ_CHRONO_UPM:
         MOV         MINSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_UP_CHRONOM


WAIT_CHRONOS:                    ; WAIT HERE UNTIL BUTTON PRESS
         JNB         ENTER,WAIT_CHRONOS
         LCALL       MDELAY
         CLR         SETCMINS
         SETB        SETCSECS
         LCALL       DISP_CHRONO

         JNB         UP_BTTN,UP_CHRONOS
         JNB         DOWN_BTTN,DOWN_CHRONOS
         JNB         ENTER,CLIGN_CHRONO
         JNB         BP4,RETOUR3
         LCALL       MDELAY
         SJMP        WAIT_CHRONOS

RETOUR3: JNB         BP4,RETOUR3   ;WAIT HERE UNTIL BUTTON PRESS
         LCALL       MDELAY
         CLR         SETCSECS
```

```
         JMP         WAIT_CHRONOM

UP_CHRONOS:
         JNB         UP_BTTN,UP_CHRONOS
         LCALL       MDELAY
         MOV         A,SECSC
         CJNE        A,#59,RAZ59S
         MOV         A,#0
         SJMP        RAZ59S_1
RAZ59S:  INC         A
RAZ59S_1:
         MOV         SECSC,A
         SJMP        RAZ1_CHRONOS
         MOV         A,#0
         MOV         SECSC,A
         SJMP        RAZ1_CHRONOS
RELEASE_UP_CHRONOS:
         JNB         DOWN_BTTN,RAZ_CHRONO_UPS
         LCALL       MDELAY
RAZ1_CHRONOS:
         JNB         UP_BTTN,RELEASE_UP_CHRONOS; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY
         LCALL       DISP_CHRONO
         SJMP        WAIT_CHRONOS

DOWN_CHRONOS:
         JNB         DOWN_BTTN,DOWN_CHRONOS
         LCALL       MDELAY
         MOV         A,SECSC
         JNZ         ZEROM_CHRONOS
         MOV         A,#60
ZEROM_CHRONOS:
         DEC         A
         MOV         SECSC,A
         SJMP        RAZ2_CHRONOS
RELEASE_DOWN_CHRONOS:
         JNB         UP_BTTN,RAZ_CHRONO_DOWNS
         LCALL       MDELAY
RAZ2_CHRONOS:
         JNB         DOWN_BTTN,RELEASE_DOWN_CHRONOS; HOLDS HERE UNTIL BUTTON IS RELEASED
         LCALL       MDELAY

         LCALL       DISP_CHRONO
         SJMP        WAIT_CHRONOS

RAZ_CHRONO_DOWNS:
         MOV         SECSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_DOWN_CHRONOS

RAZ_CHRONO_UPS:
         MOV         SECSC,#0
         LCALL       DISP_CHRONO
         SJMP        RELEASE_UP_CHRONOS

RETOUR4: JNB         BP4,RETOUR4    ;WAIT UNTIL BUTTON PRESS
         LCALL       MDELAY
         SETB        SETCSECS
         JMP         WAIT_CHRONOS

CLIGN_CHRONO:
         JNB         ENTER,CLIGN_CHRONO ; WAIT UNTIL BUTTON PRESS
         LCALL       MDELAY
         CLR         SETCSECS
         CLR         SETCHOURS
         CLR         SETCMINS
         LCALL       DISP_CHRONO
         MOV         A,SECSC
         MOV         SECSC_BUF,A
         MOV         A,MINSC
         MOV         MINSC_BUF,A
         MOV         A,HRSC
         MOV         HRSC_BUF,A
```

```
CLIGN_RETOUR:
         JNB         UP_BTTN,START_CHRONO ;BP1 START DU DECOMPTAGE
         LCALL       MDELAY
         JNB         BP4,RETOUR4
         LCALL       MDELAY
;         SETB        SETDOFF
;         LCALL       DISP_CHRONO
;         LCALL       DELAY_500MS
;         CLR         SETDOFF
;         LCALL       DISP_CHRONO
;         LCALL       DELAY_500MS
         SJMP        CLIGN_RETOUR
PAUSE_CHRONO:
         JNB         DOWN_BTTN,PAUSE_CHRONO ; WAIT UNTIL PRESS
         LCALL       MDELAY
         CLR         SETCSECS
         CLR         SETCHOURS
         CLR         SETCMINS
PAUSE_RETOUR:
         JNB         UP_BTTN,START_CHRONO ;BP1 START DU DECOMPTAGE
         LCALL       MDELAY
;         SETB        SETDOFF
;         LCALL       DISP_CHRONO
;         LCALL       DELAY_500MS
;         CLR         SETDOFF
;         LCALL       DISP_CHRONO
;         LCALL       DELAY_500MS
         SJMP        PAUSE_RETOUR


WAIT_DECOMPT:
         JNB         ENTER,WAIT_DECOMPT
         LCALL       MDELAY

WAIT_DECOMPT1:
TEST0:
         MOV        A,HRSC
         JNZ        TEST0_FIN
         MOV        A,MINSC
         JNZ        TEST0_FIN
         MOV        A,SECSC
         JNZ        TEST0_FIN
         LCALL      DISP_CHRONO
         SJMP       BEEPER_ON
TEST0_FIN:
         LCALL       DISP_CHRONO
;         LCALL      DELAYS
         LCALL       DELAY_1S
         SJMP        CONT_DECOMPT
START_CHRONO:
         JNB         UP_BTTN,START_CHRONO ; WAIT UNTIL BUTTON PRESS
;         LCALL       MDELAY
CONT_DECOMPT:
         JNB         DOWN_BTTN,PAUSE_CHRONO
;        LCALL       MDELAY

         CLR         SETDOFF
         CLR         SETCSECS
         CLR         SETCHOURS
         CLR         SETCMINS
         ACALL       DECOMPT              ; COUNT DOWN
;        LCALL       DISP_CHRONO          ; DISPLAY CHRONO
         SJMP        WAIT_DECOMPT1        ; LOOP

; DECREMENTE LE CHRONO
DECOMPT:
TEST_0:
         MOV        A,HRSC
         JNZ        TEST_0_FIN
         MOV        A,MINSC
         JNZ        TEST_0_FIN
         MOV        A,SECSC
         JNZ        TEST_0_FIN
         LCALL      DISP_CHRONO
```

```
        SJMP        BEEPER_ON
TEST_0_FIN:


                                        ; IS CHRONO AT 00:00:00 ?
                                        ; IF YES, ALARM

        MOV         A,SECSC
        CJNE        A,#0,DECOMPTS       ; 00 SECONDS?
        MOV         SECSC,#59           ; YES => SECSC = #59
        MOV         A,MINSC             ;        AND MINSC=MINSC-1

        CJNE        A,#0,DECOMPTMIN
        MOV         MINSC,#59
        MOV         A,HRSC
        DEC         A
        MOV         HRSC,A
        RET
DECOMPTS:
        MOV         A,SECSC
        DEC         A
        MOV         SECSC,A
        RET
DECOMPTMIN:
        MOV         A,MINSC
        DEC         A
        MOV         MINSC,A
        RET


BEEPER_ON:
        SETB        BEEPER
        LCALL       DELAY_1S
        CLR         BEEPER
        MOV         A,SECSC_BUF
        MOV         SECSC,A
        MOV         A,MINSC_BUF
        MOV         MINSC,A
        MOV         A,HRSC_BUF
        MOV         HRSC,A
        LCALL       DISP_CHRONO
        JMP         SET_CHRONO

;************************************************************************
DELAYMS:                        ;millisecond delay routine
;                               ;
   MOV R7,#00H                  ;put value of 0 in register R7
LOOPA:
   INC R7                       ;increase R7 by one (R7 = R7 +1)
   MOV A,R7                     ;move value in R7 to Accumlator (also known as A)
   CJNE A,#0FFH,LOOPA           ;compare A to FF hex (256). If not equal go to LOOPA
   RET                          ;return to the point that this routine was called from
;
;************************************************************************
DELAYS:                         ;One second delay routine
   MOV R6, #00H                 ;put 0 in register R6 (R6 = 0)
   MOV R5, #004H                ;put 5 in register R5 (R5 = 4)
LOOPB:
   INC R6                       ;increase R6 by one (R6 = R6 +1)
   ACALL DELAYMS                ;call the routine above. It will run and return to here.
   MOV A, R6                    ;move value in R6 to A
   JNZ LOOPB                    ;if A is not 0, go to LOOPB
   DEC R5                       ;decrease R5 by one. (R5 = R5 -1)
   MOV A, R5                    ;move value in R5 to A
   JNZ LOOPB                    ;if A is not 0 then go to LOOPB.
   RET
;


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