```
;*************************************************************************
; Pll control program
; plla.asm
;-------------------------------------------------------------------------
;   Version: 8.0  By Dillian Wong
;-------------------------------------------------------------------------
; Reference frequency = 12MHz, Sample cycle = 1
;


;*************************************************************************
;                               CONST
;*************************************************************************
;
; SPI data transfer for MC145170 VHF frequency synthesiser
; 8052 is Master; MC145170 is Slave.
; The following SPI-defined pins are used. (write only, so only 3 wires are needed.)
; CLK: Serial Data Clock Input (pin 7 of MC145170)
; Din: Serial Data input (pin 5 of MC145170)
; ENB: Active-Low Enable Input (pin 6 of MC145170)
;
; LCD Connections:
; ---------------
; (DI) LCD_RS = P2.7     D0 = P0.0      D4 = P0.4
; (RW) LCD_RW = P2.6     D1 = P0.1      D5 = P0.5
; (EN) LCD_EN = P2.5     D2 = P0.2      D6 = P0.6
;                        D3 = P0.3      D7 = P0.7

CLK             EQU             P3.0
ENB             EQU             P3.4
DIN             EQU             P3.2


LCD_RS          EQU             P2.7
LCD_RW          EQU             P2.6
LCD_EN          EQU             P2.5
LCD_BUS         EQU             080H            ;PORT 0

KEYPORT         EQU             090H            ;PORT 1

LAMP_1          EQU             P2.1
LAMP_2          EQU             P2.2



;*************************************************************************
;                               VARIABLES
;*************************************************************************
; Now we need to use some of the internal RAM in the 8052 as data storage.
; For speed of execution, two of these variables must be located in the RAM area
; that allows bit addressing within the byte. In 8052, the RAM space corresponds
; to addresses 20H to 2FH. Addresses below 20H or above 2FH cannot be bit-addressed!
;
C_REG           EQU             20H   ; C register
R_REG_H         EQU             21H   ; High byte of R register
R_REG_L         EQU             22H   ; Low byte of R register
N_REG_H         EQU             23H   ; High byte of N register
N_REG_L         EQU             24H   ; Low byte of N register
                                      ; we need a loop counter to keep track of
LOOP            EQU             30H   ; number of bits that are required to send.

KEYINDEX        EQU             31H
KEYDATA         EQU             32H
KEY_TMR         EQU             33H

KEY             EQU             34H   ;KEY.1 = check key enable
                                      ;KEY.2 = KEY PRESS

NDIGIT          EQU             35H   ;decimal of N register 2 bytes
```

```
;*****************************************************************************
;                           DEFINED VERTOR ADDRESS
;*****************************************************************************

;//VERTOR ADDRESS
                ORG             0H
                JMP             INITIAL
                ORG             0003H       ;(INT0)
                AJMP            INT0
                ORG             000BH       ;TIMER0 OVERFLOW (TF0)
                AJMP            TF_0
                ORG             0013H       ;(INT1)
                AJMP            INT1
                ORG             001BH       ;TIMER1 OVERFLOW (TF1)
                AJMP            TF_1
                ORG             0023H       ;UART
                AJMP            UART

                ORG             100H


;//Program code for special function
INT0:           RETI

INT1:           RETI

;//TIMER0 OVERFLOW INTERRUPT
;//check key will be implemented, after many time of timer0 overflow
TF_0:
                PUSH            Acc

                MOV             TL0,#00H    ;set timer0 with 2^16 machine period,2^16 x 12/12M=0.065s
                MOV             TH0,#00H
                SETB            TR0

                INC             KEY_TMR     ;KEY_TMR contain a value specificed in CHECK_KEY subroutine
                MOV             A,KEY_TMR   ;KEY_TMR must be overflow for impletment CHECK_KEY
                CJNE            A,#1,Q_TF_0 ;vaule added for overflow * timer0 period = check key period

                SETB            KEY.1
                MOV             KEY_TMR,#0  ;this cannot be omitted, KEY.1 = 1, until a key is
                                            ;pressed, as each check key will clear KEY.1
  Q_TF_0:       POP             Acc
                RETI

TF_1:           RETI

UART:           RETI



;*****************************************************************************
;                           INITALIZATION
;*****************************************************************************

;//LCD
INITIAL:
                MOV             SP,#6FH         ; set stack pointer
                CALL            INIT_LCD


;//RAM
INIT_RAM:
                MOV             @R0,#0

                INC             R0
                CJNE            R0,#7FH,INIT_RAM
```

```
;//TIMER0 WITH 0.065S OVERFLOW TIME
INIT_TMR:
                MOV             TMOD,#00100001B     ;timer1,mode2,8 bit auto reload
                                                    ;timer0,mode1,16 bit timer
                MOV             TL0,#0              ;set timer0 with 2^16 machine period,2^16 x 12/12M=0.065s
                MOV             TH0,#0
                SETB            TR0                 ;turn on timer0

                MOV             TL1,#(256-24)       ;set timer1 with 24 machine period,24 x 12/12M=24us
                MOV             TH1,#(256-24)       ;TL1,TH1 must be the same for auto reload
                SETB            TR1                 ;turn on timer1

INIT_UART:
                MOV             SCON,#01100000B     ;serial port control register
                                                    ;mode1,8bit UART,variable BaudRate
                                                    ;RI activated when receive a valid stop bit
INIT_INTERRUPT:
                MOV             IE,#10010010B       ;IE.7=1 each interrupt source is individually enable or disable
                                                    ;IE.6=0 reserved
                                                    ;IE.5=0 disable timer2 overflow
                                                    ;IE.4=1 enable serial port interrupt
                                                    ;IE.3=0 disable timer1 overflow interrupt
                                                    ;IE.2=0 disable external interrupt1
                                                    ;IE.1=1 enable the timer0 overflow interrupt
                                                    ;IE.0=0 disable external interrupt0



;*************************************************************************
;                           MAIN PROGRAM
;*************************************************************************
MAIN:

                        MOV         DPTR,#STRING1
                        CALL        WRITE_STR

                        CALL        LCD_NEXTLINE
                        MOV         A,#'N'
                        CALL        WRITE_CHAR
                        MOV         A,#'='
                        CALL        WRITE_CHAR
                        MOV         NDIGIT,#0



WAIT_KEY:
                        ACALL       CHECK_KEY
                        JNB         KEY.2,WAIT_KEY      ;if key not press, loop
                        CLR         KEY.2               ;if key press, display in lcd

                        MOV         A,KEYDATA
                        CALL        WRITE_CHAR

                        ;//store 4 digit number to r0,r1,r2 and r3
                        MOV         A,KEYINDEX          ;if 0 is pressed, keyindex = 0
                        CJNE        A,#10,DEC_3
                        SUBB        A,#10
                        MOV         KEYINDEX,A

      DEC_3:            MOV         A,NDIGIT
                        CJNE        A,#03,DEC_2
                        MOV         R3,KEYINDEX         ;units
                        SJMP        HEX_NEXT

      DEC_2:            MOV         A,NDIGIT
                        CJNE        A,#02,DEC_1
                        MOV         R2,KEYINDEX         ;tens
                        SJMP        HEX_NEXT

      DEC_1:            MOV         A,NDIGIT
```

```
                        CJNE        A,#01,DEC_0
                        MOV         R1,KEYINDEX          ;hunreds
                        SJMP        HEX_NEXT

        DEC_0:          MOV         A,NDIGIT
                        CJNE        A,#00,HEX_NEXT
                        MOV         R0,KEYINDEX          ;thousands

        HEX_NEXT:       INC         NDIGIT

                        CJNE        A,#03,WAIT_KEY         ;4 digits is pressed

;//DISPLAY HEXDECIMAL FROM ENTERED DIGIT
DISP_HEX:                                                ;hex displayment
                        CALL        DEC_HEX
                        MOV         N_REG_H,R4
                        MOV         N_REG_L,R5

                        MOV         A,#'D'
                        CALL        WRITE_CHAR
                        MOV         A,#' '
                        CALL        WRITE_CHAR
                        MOV         A,#'('
                        CALL        WRITE_CHAR

                        MOV         A,N_REG_H
                        ACALL       HEX_TO_ASCII
                        MOV         A,R1
                        ACALL       WRITE_CHAR
                        MOV         A,R0
                        ACALL       WRITE_CHAR

                        MOV         A,N_REG_L
                        ACALL       HEX_TO_ASCII
                        MOV         A,R1
                        ACALL       WRITE_CHAR
                        MOV         A,R0
                        ACALL       WRITE_CHAR

                        MOV         A,#'H'
                        CALL        WRITE_CHAR
                        MOV         A,#')'
                        CALL        WRITE_CHAR



;;-------------------------------------------------------------------------
;//MC_CONTROL
; C7=1 changed polarity  C6=0 Enable detector B  C5=0 enable Lock detector
; C4=0 C3=0 C2=0 OSC/16  C1=0 enable Fv output    C0=0 enable Fr output
;-------------------------------------------------------------------------
MC_CONTROL:
                         MOV        C_REG,#080H     ;
                         CALL       CDATA
                         CALL       NDATA           ;N register is defined
                         MOV        R_REG_L,#078H
                         MOV        R_REG_H,#000H   ; R=120
                         CALL       RDATA
OK:                      JMP        OK




                ;01234567890123456789
STRING1: DB     '88M-108MHz ',00h
STRING2: DB     'R = 120',00h
STRING3: DB     'press = N + 5',00h
```

```
;*************************************************************************
;                       DECIMAL TO HEXDECIMAL CODE
;-------------------------------------------------------------------------
;Convert 4 ditial decimal number into 2 bytes hexdecimal number
;DEC_HEX
;               INPUT  : R0,R1,R2,R3  (Thousands,Hundreds,Tens,Units)
;               OUTPUT : R4,R5 (H byte,L byte)

DEC_HEX:
                MOV             R4,#00
                MOV             R5,#00

                MOV             A,R3            ;Take the units directly as it is to the LSB
                MOV             R5,A

                MOV             A,R2            ;Take the tens of multiplying and adding to
                MOV             B,#10           ;LSB
                MUL             AB
                ADD             A,R5
                MOV             R5,A

CAL_HUNDREDS:

                CJNE            R1,#00,HEX_LOOP;Skip if the digital is zero
                SJMP            CAL_THOUSANDS

HEX_LOOP:
                MOV             A,#100          ;add constant 100D to LSB as many times
                CLR             C
                ADD             A,R5
                MOV             R5,A
                JNC             NO_EXCEED

                INC             R4              ;Increase the next higher bytes when the
                MOV             A,R4            ;LSB overflows

NO_EXCEED:
                DJNZ            R1,HEX_LOOP

CAL_THOUSANDS:

                MOV             A,R0            ;quit, if the thousands ditital is zero
                CJNE            A,#00H,R0_CON
                SJMP            Q_DEC_HEX

R0_CON:
                DEC             R0              ;else, add hundreds ditials ten time
                MOV             R1,#10
                SJMP            CAL_HUNDREDS

Q_DEC_HEX:
                RET


;*************************************************************************
;                       HEXDECIMAL TO ASCII CODE
;-------------------------------------------------------------------------
;HEX_TO_ASCII
;               INPUT  : A (00-FF)
;               OUTPUT : R1,R0(ASCII CODE)
;TRAN
;               INPIT  : A (00-0F)
;               OUTPUT : A (ASCII CODE)

;//8 BITS
HEX_TO_ASCII:
                MOV             R1,A
                ACALL           TRAN
```

```
                MOV             R0,A
                MOV             A,R1
                SWAP            A
                ACALL           TRAN
                MOV             R1,A
                RET

;//4 BITS
TRAN:
                ANL             A,#0FH
                ADD             A,#90H
                DA              A
                ADDC            A,#40H
                DA              A
                RET


;*************************************************************************
;                       CHECK KEY
;-------------------------------------------------------------------------*
;Subroutine will be implemented:
;                               When KEY.1 = 1
;Subroutine will be terminated:
;                              1) When KEY.2 = 1
;                                 then KEYDATA contain the key value (1,2,3......)
;                              2) complete the subroutine
;
;KEY_TMR can adjust the durtion for key press
;

KEYTBL: DB      '!1234567890CE',00h

CHECK_KEY:
                JNB             KEY.1,Q_CHECK_KEY
                CLR             KEY.1

                MOV             KEYINDEX,#0

                ;scan row1 for 1,2,3
                MOV             P1,#10111111B   ;ROW1
                NOP
                MOV             A,P1
                ACALL           SCAN_ROW
                JB              KEY.2,Q_CHECK_KEY

                ;scan row2 for 4,5,6
                MOV             P1,#11011111B   ;ROW2
                NOP
                MOV             A,P1
                ACALL           SCAN_ROW
                JB              KEY.2,Q_CHECK_KEY

                ;scan row3 for 7,8,9
                MOV             P1,#11101111B    ;ROW3
                NOP
                MOV             A,P1
                ACALL           SCAN_ROW
                JB              KEY.2,Q_CHECK_KEY

                ;scan row4 for c,0,e
                MOV             P1,#11110111B    ;ROW4
                NOP
                MOV             A,P1
                ACALL           SCAN_ROW
                JB              KEY.2,Q_CHECK_KEY

   Q_CHECK_KEY:
                RET

SCAN_ROW:
                INC             KEYINDEX
```

```
                JNB             ACC.0,SCAN_OK    ;COL1
                INC             KEYINDEX
                JNB             ACC.1,SCAN_OK    ;COL2
                INC             KEYINDEX
                JNB             ACC.2,SCAN_OK    ;COL3
                AJMP            Q_SCAN_ROW       ;if not scanned, quit
   SCAN_OK:
                SETB            KEY.2            ;If scanned, KEY.2 = 1,and return key value in keydata
                MOV             KEY_TMR,#254     ;adjust the durtion for key press

                MOV             DPTR,#KEYTBL
                MOV             A,KEYINDEX
                MOVC            A,@A+DPTR
                MOV             KEYDATA,A

   Q_SCAN_ROW:
                RET



;*************************************************************************
;//0.5SEC_DELAY
;-------------------------------------------------------------------------
Wait50ms:
                        mov     R2,     #50
                        mov     R3,     #6
WL_01:                  nop
                        nop
                        djnz    R3,     WL_01
                        djnz    R2,     WL_01
                        ret

Wait200ms:
                        CALL    Wait50ms
                        CALL    Wait50ms
                        CALL    Wait50ms
                        CALL    Wait50ms
                        RET
;*************************************************************************
;//TRY_LAMP
;-------------------------------------------------------------------------
TRY_LAMP:
                        MOV     R1,#05H
REPEAT_LAMP:            CALL    Wait50ms
                        SETB    LAMP_1
                        SETB    LAMP_2
                        CALL    Wait50ms
                        CLR     LAMP_1
                        CLR     LAMP_2
                        DJNZ    R1,REPEAT_LAMP
                        RET

;*************************************************************************
; Input/Output a byte from/to the LCD panel
;-------------------------------------------------------------------------
;LCD CONNECTIONS REQUIRED:
;       LCD_RS  REG     P?.?    (1 Bit)
;       LCD_RW  REG     P?.?    (1 Bit)
;       LCD_EN  REG     P?.?    (1 Bit)
;       LCD_BUS REG     P?      (8 Bit)
;
;FUNCTION PROVIDED
;       WAIT_LCD                :CHECKING THE BUSY STATUS OF THE LCD
;       READ_LCD_COMMAND        :READ A DATA BYTE FROM LCD
;       WRITE_LCD_COMMAND       :WRITE AN INSTRUCTION BYTE TO LCD
;       INIT_LCD                :INITIALIZING THE LCD
;       CLEAR_LCD               :CLEARING THE DISPLAY
;       LCD_LOCATION            :POSTION CURSOR
;       LCD_NEXTLINE            :POSITION CURSOR AT SECONDS LINE
;       WRITE_CHAR              :WRITING CHAR TO THE LCD
;       WRITE_STR               :WRITING STRING TO THE LCD
```

```
 ;//CHECKING THE BUSY STATUS OF THE LCD
WAIT_LCD:
                        PUSH    Acc
        LOPWAIT:
                        ACALL   READ_LCD_COMMAND
                        JB      ACC.7,LOPWAIT
                        POP     Acc
                        RET

;//READ A DATA BYTE FROM LCD
READ_LCD_COMMAND:
                        CLR     LCD_RS                  ;Instruction I/O
                        SETB    LCD_RW                  ;Read command
                        SETB    LCD_EN                  ;Start LCD command
                        MOV     LCD_BUS,11111111B
                        MOV     A,LCD_BUS
                        CLR     LCD_EN                  ;Finsih the command
                        SETB    LCD_RW                  ;Read command
                        SETB    LCD_RS                  ;Data I/O
                        RET

;//WRITE AN INSTRUCTION BYTE TO LCD
WRITE_LCD_COMMAND:
                        ACALL WAIT_LCD
                        MOV   LCD_BUS,A                 ; Post Accumulator at LCD data inpu
                        CLR   LCD_RS                    ; Set RS low (Instruction)
                        CLR   LCD_RW                    ; Set RW low (Write)
                        NOP                             ; Wait circuit settling
                        SETB  LCD_EN                    ; Tells the LCD to work (receive instruction byte)
                        NOP                             ; Wait a little bit
                        NOP                             ;
                        CLR   LCD_EN                    ; Drop the selection line
                        RET

;//INITIALIZING THE LCD
INIT_LCD:
                        MOV     A,#38H                  ;Indicate that the display is a two-line display
                        ACALL   WRITE_LCD_COMMAND
                        MOV     A,#0EH                  ;Turn the cursor on
                        ACALL   WRITE_LCD_COMMAND
                        MOV     A,#06H                  ;The cursor position automatically moves to the right
                        ACALL   WRITE_LCD_COMMAND
                        MOV     A,#0CH                  ;Open LCD display and hidden cursor
                        ACALL   WRITE_LCD_COMMAND
                        ACALL   CLEAR_LCD
                        ACALL   Wait50ms
                        RET

;//CLEARING THE DISPLAY
CLEAR_LCD:
                        MOV     A,#01H                  ;Clear LCD
                        ACALL   WRITE_LCD_COMMAND
                        RET

;//POSITION CURSOR
LCD_LOCATION:
                        ANL     A,#11111B
                        JNB     B.0,DOSETBIT6
                        SETB    ACC.6
        DOSETBIT6:
                        SETB    ACC.7
                        ACALL   WRITE_LCD_COMMAND
                        RET

;//POSITION CURSOR AT SECONDS LINE
LCD_NEXTLINE:
                        MOV     A,#40H
                        MOV     B,#1H
                        ACALL    LCD_LOCATION
```

```
                        RET

;//WRITING CHAR TO THE LCD
WRITE_CHAR:
                        ACALL WAIT_LCD
                        MOV   LCD_BUS,A             ; Post Accumulator at LCD data input
                        SETB  LCD_RS                ; Set RS high (Data)
                        CLR   LCD_RW                ; Set RW low (Write)
                        NOP                         ; Wait circuit settling
                        SETB  LCD_EN                ; Tells the LCD to work (receive data byte)
                        NOP                         ; Wait a little bit
                        NOP                         ;
                        CLR   LCD_EN                ; Drop the selection line
                        RET

;//WRITING STRING TO THE LCD
WRITE_STR:
                        PUSH    Acc
        LOPWRITE_STR:
                        CLR     A                   ; Clear Index
                        MOVC    A,@A+DPTR           ; Get byte pointed by Dptr
                        INC     DPTR                ; Point to the next byte
                        JZ      LCD_ENDWRITE_STR    ; Return if found the zero (end of stringz)
                        ACALL   WRITE_CHAR          ; It was data, write it to Lcd
                        JMP     LOPWRITE_STR        ; Go get next byte from stringz
        LCD_ENDWRITE_STR:
                        POP     Acc
                        RET


;*************************************************************************
; Input/Output a byte from/to the PLL panel
;-------------------------------------------------------------------------
; (CLK)CLK = P1.0   (Din)DIN = P1.1   (ENb)ENB = P1.2
;
; CDATA: Subroutine for Register C data transfer
; NDATA: Subroutine for Register N data transfer
; RDATA: Subroutine for Register R data transfer
;

; Subroutine for Register C data transfer
CDATA:          SETB            ENB             ; To make sure that ENB is deactivate
                CLR             ClK             ; Serial Data Clock starts off low.
                CLR             ENB             ; Lowering ENB begins the data transfer.
                MOV             LOOP,#8         ; Eight bits to transfer.
XFER_C:         MOV             A,C_REG
                MOV             C,ACC.7         ; Move bit 7 into Carry (SPI is MSB first).
                MOV             DIN,C           ; Carry bit is sent as the Data bit.
                SETB            CLK             ; Generate Serial Data Clock rising edge.
                CLR             CLK             ; Generate Serial Data Clock falling edge.
                MOV             A,C_REG         ; Accumulator is temp holder for shift operation.
                RL              A               ; Rotate left (but not through Carry!).
                MOV             C_REG,A         ; Prepare bit 7 for next transfer to MC145170.
                DJNZ            LOOP,XFER_C     ; Decrement LOOP. Jump if not zero to XFER.
; Transfer done.
                SETB            ENB             ; Inactivate ENB.
                RET

; Subroutine for Register N data transfer
NDATA:          SETB            ENB             ; To make sure that ENB is deactivate
                CLR             ClK             ; Serial Data Clock starts off low.
                CLR             ENB             ; Lowering ENB begins the data transfer.
                MOV             LOOP,#16        ; Sixteen bits to transfer.
XFER_N:         MOV             A,N_REG_H
                MOV             C,ACC.7         ; Move bit 7 into Carry (SPI is MSB first).
                MOV             DIN,C           ; Carry bit is sent as the Data bit.
                SETB            CLK             ; Generate Serial Data Clock rising edge.
                CLR             CLK             ; Generate Serial Data Clock falling edge.
                MOV             A,N_REG_L       ;Accumulator is temp holder for shift operation.
                MOV             C,ACC.7         ; Get the bit 7 of N_REG_L
                MOV             A,N_REG_H       ;
```

```
                RLC             A               ; Shift high byte one bit to left through Carry.
                MOV             N_REG_H,A
                MOV             A,N_REG_L
                RLC             A               ; Shift low byte one bit to left through Carry.
                MOV             N_REG_L,A
                DJNZ            LOOP,XFER_N     ; Decrement LOOP. Jump if not zero to XFER.
; Transfer done.
                SETB            ENB             ; Inactivate ENB.
                RET

; Subroutine for Register R data transfer
RDATA:          SETB            ENB             ; To make sure that ENB is deactivate
                CLR             ClK             ; Serial Data Clock starts off low.
                CALL            SH2RL           ;
                CLR             ENB             ; Lowering ENB begins the data transfer.
                MOV             LOOP,#15        ; Fifteen bits to transfer.
XFER_R:         MOV             A,R_REG_H
                MOV             C,ACC.7         ; Move bit 7 into Carry (SPI is MSB first).
                MOV             DIN,C           ; Carry bit is sent as the Data bit.
                SETB            CLK             ; Generate Serial Data Clock rising edge.
                CLR             CLK             ; Generate Serial Data Clock falling edge.
                CALL            SH2RL;
                DJNZ            LOOP,XFER_R     ; Decrement LOOP. Jump if not zero to XFER.
; Transfer done.
                SETB            ENB             ; Inactivate ENB.
                RET
;
;
SH2RL:          MOV             A,R_REG_L       ; Accumulator is temp holder for shift operation.
                MOV             C,ACC.7         ;
                MOV             A,R_REG_H       ; Get the bit 7 of R_REG_L
                RLC             A               ; Shift high byte one bit to left through Carry.
                MOV             R_REG_H,A
                MOV             A,R_REG_L
                RLC             A               ; Shift low byte one bit to left through Carry.
                MOV             R_REG_L,A
                RET
                END                             ; Tell assembler code completed.
```