EE2401 微算機系統Fall 2019

HW#5 (8051 applications on MCU8051IDE) (10/14/2019)

Due date: 11/14/2019. Severe penalty will be given to late homework.

1070611112 王昊文

Note:

(a) The homework will be graded based on your **documentation** and **demonstration**.

(b) For all **(Software Design)** problems, you are required to us **MCU8051IDE**simulators to simulate and verify your programs.

(c) You are required to **type** your homework (first the problem then your solution) by using a **word processor** and submit in .doc (or .docx) format under a filename **EE2401f19-hw5-student_no-vn.doc (or .docx)**, where **student_no** is your student number, e.g., **107061xxx** and **vn** is your version number, e.g., **v3**. You should **upload your .doc file** in **iLMS** by the specified deadline whenever you have a newer version. Follow the iLMS upload homework process to upload your file.

(d) The homework will be graded based on your **latest version**. Old version(s) will be discarded.

(e) Each homework assignment will have full score of 100 points. 5 points will be deducted if you do not comply with the naming convention. Severe grade penalty will be given to late homework. 20 points will be taken off per day after deadline till zero point.

(f) Please treat the above requirements as a kind of training in writing a decent homework report. If you have any problem regarding this homework, please feel free to consult with TA or me. If you think the time is too short to accomplish this homework, please let me know in class.

1. (50%)

MCU8051IDE 中有提供一個 LCD 顯示器(HD44780)以及 Matrix keypad 的 Virtual Hardware，請設計一個程式來讀入 Matrix keypad 的按鍵並將之顯示到 2x16 的虛擬 LCD 上。程式一開始沒有任何按鍵時先不顯示 LCD，當開始有按鍵時，把該按鍵對應的文數字依序顯示在 LCD 上，從第一列左上角(第一行)開始，之後的按鍵就依序顯示在第二行、第三行、…直到第 16 行，之後則跳到第二列的第一行，其餘依此類推，直到第二列的第 16 行。之後再有按鍵則清除 LCD 畫面從心開始從第一列第一行開始顯示。你最少必須提供流程圖(或 pseudo code)，解釋你的做法，你的程式碼，Virtual Hardware 儲存檔。可以讓助教驗證你的程式。

For debugging conveniences, Debounce is commented out from this code since it is not practical under the scale of time.

Also the keypad part is similar to HW4. The different part is changing the output ssd to ASCII code. The Pseudo code will focus on the LCD part and the whole structure.

```
[INIT()]                // initialize LCD
[NEW()]                 // refresh LCD
FOR(COUNT = 32; COUNT > 0; COUNT++) {    // LCD number count
NEXT:[GET_KEY()]                        // This will use the carry bit to determine
    IF (C == 0)
        GOTO NEXT
    ELSE {
        [OUTCHR()]
        [DISP()]
        IF (A != 16) {
            IF ( A == 0) {
                [COUNT = 32]       // if last, reinitialize char count
                [NEW()]            // refresh LCD
            }
            ELSE {
                [GOTO NEXT]
            }
        ELSE {
            [SEC_LINE()]
            [GOTO NEXT]
        }
INIT() {
    [INPUT command 38H]     // 2 lines, 5 x 7 matrix
    [WAIT()]
    [RS = 0]                // output a command
    [OUT()]                 // send it out
    [INPUT command 0EH]     // LCD on cursor on
    [WAIT()]
    [RS = 0]                // output a command
    [OUT()]                 // send it out
    [RET]
}
DISP() {
    [WAIT()]
    [RS = 0]                // output a command
    [OUT()]                 // send it out
    [RET]
}
OUT() {
    [A = DBUS]              // move the data to the data bus
    [RW = 0]                // write
    [E = 1]
    [E = 0]
    RET
}
WAIT() {
      Wait for the LCD to be ready (DBUS.7)
}
SEC_LINE() {
    [A = #0C0H]         // force LCD start at second line[WAIT()]
```

```
        [RS = 0]                    // output a command
        [OUT()]                     // send it out
        [RET]
}
```

GET_KEY() : same as homework 4, construct a table to convert the HEX code to ASCII table at the OUTCHR part.

```
;****************************************************************
; LCD interface
;
; This program continuously displays on the LCD
; the ASCII characters are stored in internal RAM
; locations 30H-70H
;****************************************************************
                RS      EQU     P3.0            ; RS = 0 command, RS = 1 data
                RW      EQU     P3.1            ; RW = 1 read, RW = 0 write
                E       EQU     P3.2            ; E = 1-to-0 enable LCD
                DBUS    EQU     P1              ; D7 LCD busy status
                PTR     EQU     R0              ; memory data pointer
                COUNT           EQU     R1

                ORG     0000H
                CALL    INIT                    ; initialize LCD
                CALL    NEW                     ; refresh LCD
                MOV     COUNT,          #32             ; initialize char count
NEXT:           ACALL           GET_KEY                         ; read char from keypad
                JNC     NEXT                    ; C=0->no key pressed->read again
                ACALL           OUTCHR
                ACALL           DISP                    ; display on LCD
                DEC     COUNT                           ; count - 1
                MOV     A,      COUNT
                CJNE    A,#16,TEST1             ; end of 1st line?
                ACALL           SEC_LINE                        ; yes, go to 2nd line
                SJMP    NEXT                    ; and go back to NEXT
TEST1:                  CJNE    A,#0,NEXT       ; end of 2nd line?
                MOV     COUNT,          #32     ; yes, reinitialize char count
                ACALL           NEW             ; and refresh LCD
                SJMP    NEXT            ; go back to NEXT
;************************************************************
; Initialize the LCD
;************************************************************
INIT:           MOV     A,      #38H            ; 2lines, 5x7 dot matrix, command
                ACALL           WAIT                    ; wait for LCD to be free
                CLR     RS                      ; output a command
                ACALL           OUT                     ; send it out
                MOV     A,      #0EH            ; LCD on, cursor on, command
                ACALL           WAIT                    ; wait for LCD to be free
                CLR     RS                      ; output a command
                ACALL           OUT                     ; send it out
                RET
;************************************************************
```

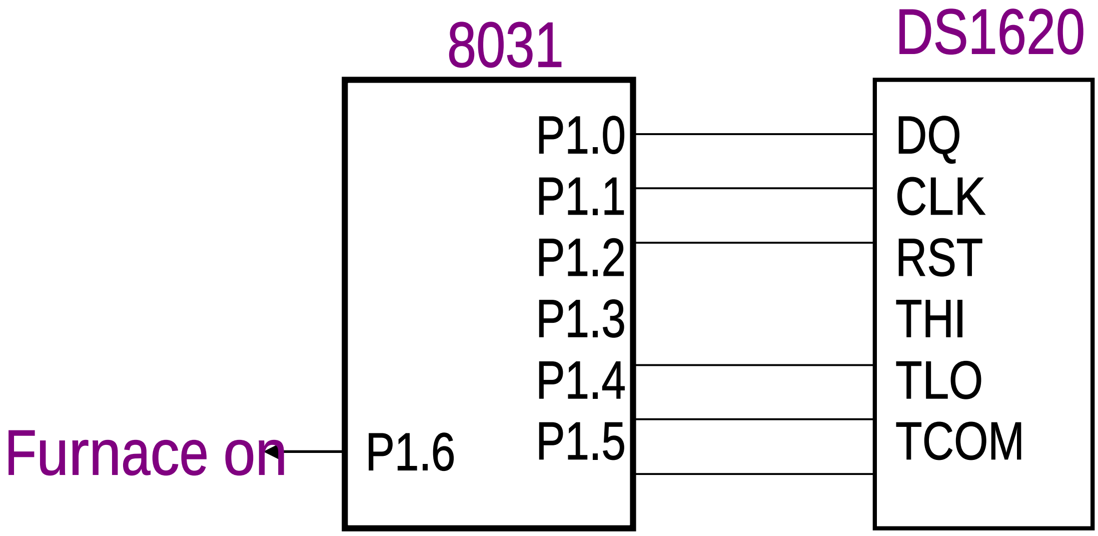
```
; DISP
; This function will display data on LCD
;**************************************************************
DISP:           ACALL           WAIT                    ; wait for LCD to be free
                SETB    RS                      ; output data mode
                ACALL           OUT                     ; send it out
                RET
;***************************************************************
; OUT
; This function will output command or data to LCD
;**************************************************************
OUT:            MOV     DBUS,           A               ; A will store the command / data
                CLR     RW                      ; write mode
                SETB    E                       ; send!
                CLR     E                       ; send finish
                RET
;**************************************************************
; NEW
; This function will refresh the LCD when all displayed
;**************************************************************
NEW:            MOV     A,      #01H            ; clear LCD, command
                ACALL           WAIT                    ; wait for LCD to be free
                CLR     RS                      ; output command mode
                ACALL           OUT                     ; send it out
                MOV     A,      #80H            ; cursor off, line1, pos1, command
                ACALL           WAIT                    ; and refresh LCD
                CLR     RS                      ; output command mode
                ACALL           OUT                     ; send it out
;***************************************************************
; WAIT
; Wait for LCD to be free
;***************************************************************
WAIT:           CLR     RS              ; command
                SETB    RW              ; read
                SETB    DBUS.7                  ; DB7 = input mode
                SETB    E               ; get data from led
                CLR     E               ; close LCD read
                JB      DBUS.7,         WAIT
                RET
;*******************************************************************
; SEC_LINE
; start cusor at 2nd line
;*******************************************************************
SEC_LINE:
                MOV     A,      #0C0H; force cursor beginning at 2nd line
                ACALL           WAIT            ; wait for LCD to be free
                CLR     RS              ; output a command
                ACALL           OUT             ; send it out
                RET
```

```
;*************************************************************************
*****
; IN_HEX - input hex code from keypad with debouncing
; for key press and key release (50 repeat
; operations for each)
;*************************************************************************
*****
;IN_HEX:                MOV     R3,     #50     ; debounce count
;BACK:                  CALL    GET_KEY                 ; key pressed? C = 1 yes, C = 0
No
;               JNC     IN_HEX                  ; no, check again
;               DJNZ    R3,     BACK    ; yes, repeat 50 times for debouncing
;               PUSH    ACC             ; save hex code
;
;BACK2:                 MOV     R3,     #50     ; wait for key up
;BACK3:                 CALL    GET_KEY                 ; key still pressed?
;               JC      BACK2                   ; yes: keep checking
;               DJNZ    R3,     BACK3           ; no, key released, repeat 50 times
debouncing
;               POP     ACC             ; recover hex code and
;               RET                     ; return


;************************************************************************
; GET_KEY - get keypad status
; - return with C = 0 if no key pressed
; - return with C = 1 and hex code in ACC if
; a key is pressed
;*************************************************************************
GET_KEY:                MOV     A,      #0FEH; start with column 0 "1111 1110"
                MOV     R6,     #4      ; use R6 as column counter

TEST:           MOV     P0,     A       ; activate column line
                MOV     R7,     A       ; save column info in ACC
                MOV     A,      P0      ; read back Port 0
                ANL     A,      #0F0H ; isolate row lines (column turn into 0)
                CJNE    A, #0F0H, KEY_HIT ; row line active?
                MOV     A,      R7      ; no: move to next
                RL      A
                DJNZ    R6,     TEST    ; keep checking until all columns check

                CLR     C               ; no key pressed
                SJMP    EXIT            ; return with C = 0
; if found a key down
KEY_HIT:                MOV     R7,     A       ; save scan code in R7
                MOV     A,      #4      ; prepare to convert to hex code
                CLR     C               ; column weighting
                SUBB    A,      R6      ; 4 - R6 = column number 0-3
                MOV     R6,     A       ; save in R6
                MOV     A,      R7      ; restore scan code
                SWAP A                  ; put in low nibble
                MOV  R5,    #4          ; use R5 as counter
AGAIN:                  RRC   A                 ; rotate for row num until 0
                JNC  DONE               ; done when C = 0
                INC  R6                 ; add 4 to key code to go to next
                INC  R6                 ; row until active low found
                INC  R6
                INC  R6
                DJNZ R5,    AGAIN
DONE:                   SETB  C                 ; C = 1 (key passed)
                MOV  A,     R6          ; hex code in A
EXIT:           RET
;*******************************************************************
***
; SSD
; This function will convert the lcd pad signal to seven segment display
;*******************************************************************
*
OUTCHR:
; First get the DIP input
start:          JB   A.0,   bxxx1
                JB   A.1,   bxx10
                JB   A.2,   bx100
                JB   A.3,   HEX7                ; this indicates the input is 8, and so on…
                SJMP HEX1
bxxx1:          JB   A.1,   bxx11
                JB   A.2,   bx101
                JB   A.3,   HEX8
                SJMP HEX2
bxx10:          JB   A.2,   bx110
                JB   A.3,   HEX9
                SJMP HEX3
bxx11:          JB   A.2,   bx111
                JB   A.3,   HEXC
                SJMP HEXA
bx100:          JB   A.3,   HEXE
                SJMP HEX4
bx101:          JB   A.3,   HEX0
                SJMP HEX5
bx110:          JB   A.3,   HEXF
                SJMP HEX6
bx111:          JB   A.3,   HEXD
                SJMP HEXB
; Lookups
; These are the output of the ASCII code
HEX0:           MOV  A,     #30H
                RET
HEX1:           MOV  A,     #31H
                RET
HEX2:           MOV  A,     #32H
```

```
                RET
HEX3:           MOV   A,      #33H
                RET
HEX4:           MOV   A,      #34H
                RET
HEX5:           MOV   A,      #35H
                RET
HEX6:           MOV   A,      #36H
                RET
HEX7:           MOV   A,      #37H
                RET
HEX8:           MOV   A,      #38H
                RET
HEX9:           MOV   A,      #39H
                RET
HEXA:                 MOV     A,      #41H
                RET
HEXB:                 MOV     A,      #42H
                RET
HEXC:                 MOV     A,      #43H
                RET
HEXD:                 MOV     A,      #44H
                RET
HEXE:           MOV   A,      #45H
                RET
HEXF:           MOV   A,      #46H
                RET
                end
```

2. (50%)

MCU8051IDE 中有提供一個 DS1620 的 Virtual Hardware，他是一顆溫度偵測與控制用的 IC，它透過 3 條線(DQ, CLK, RST)與 microcontroller 連接。請上網找該 IC 的 datasheet，了解它的功能以及如何使用它。假定我們打算做加熱器控制如下圖所示，如果溫度低於 17 度 C 就打開加熱器加熱，溫度如果高於 23 度 C 就把加熱器關閉，停止加熱。同時，必須持續的從 DS1620Virtual Hardware 讀入目前的溫度值存到內部資料記憶體，它可以用 MCU8051IDE 直接觀察。請設計一個程式利用 MCU8051IDE 的 DS1620 Virtual Hardware 來完成這項工作。你最少必須提供流程圖(或 pseudo code)，解釋你的做法，你的程式碼，Virtual Hardware 儲存檔。可以讓助教驗證你的程式。



Note: There should be 10ms delay time after RST is set high, due to the temperature sensor’s design. But for debugging convenience, the delay part will be ignored under this time scale. If executed under an actual hardware, delay should be added every time after delay.

```
[Turn off FURN]
CONF() {                        // IC configuration
        [RST = 1]
        [A = #0CH]              // write configuration
        [SEND()]
        [A == #0AH]             // CPU = 1, 1-SHOT = 0
        [SEND()]
        [RST = 0]               // stop transfer

        [RST = 1]
        [A = #01H]              // write TH
        [SEND()]
        [A == #48]              // set TH = 24
        [SEND_TEMP()]           // different from sending command, 9 bits
        [RST = 0]               // stop transfer

        [RST = 1]
        [A = #02H]              // write TL
        [SEND()]
        [A == #32]              // set TL = 16
        [SEND_TEMP()]           // different from sending command, 9 bits
        [RST = 0]               // stop transfer
}
CONV() {
        [RST = 1]
        [A = #0EEH]                     // start temperature sensing
        [SEND()]
        [RST = 0]
}

SENS() {
        [RST = 1]
        [A = #0AAH]                             // read temperature to 8051
        [SEND()]
        [READ()]
        IF (THI == 1)                   // T >= 24, off
                OFF()
        IF(TL0 == 1)                    // T < 16, on
                ON()
}
ON() {
        FURN = ON
}
OFF {
        FURN = OFF
}
SEND() {                                        // send command 8 bits
        FOR (R0 = 8; R0 > 0; R0--) {
                [CLK = 0]                       // activate clock
                [RRC A]                         // rotate A into C
                [DQ = C]                        // send to DQ
                [CLK = 1]                       // rising edge, high impedance
        }
}
READ() {                                        // read 9 bits
        FOR (R1 = 9; R1 > 0; R1--) {
                [CLK = 0]                       // activate clock
                [C = DQ]
                [RRC A]                         // rotate A into C
                [DQ = C]                        // send to C
                [CLK = 1]                       // rising edge, high impedance
        }
SEND_TEMP() {                                   // temp has 9 bits
        FOR (R2= 9; R2 > 0; R2--) {
                [CLK = 0]                       // activate clock
                [RRC A]                         // rotate A into C
                [DQ = C]                        // send to C
                [CLK = 1]                       // rising edge, high inpedence
        }
}
```

```
                DQ      EQU     P1.0
                CLK     EQU     P1.1
                RST     EQU     P1.2
                THI     EQU     P1.3
                TLO     EQU     P1.4
                TCOM            EQU     P1.5
                FURN            EQU     P1.6

                ORG     0000H
                CLR     FURN

CONF:           SETB            RST             ; initiate transfer
                MOV A,          #0CH ; write config
                ACALL           SEND            ; send to DS1620
                MOV A,          #0AH ; CPU = 1, 1-SHOT = 0
                ACALL           SEND            ; send to DS1620
                CLR     RST             ; stop transfer

                SETB            RST             ; initiate transfer
                MOV A,          #01H            ; write TH
                ACALL           SEND            ; send to DS1620
                MOV A,          #48             ; 48 * 0.5 = 24 deg.c
                ACALL           SEND_TEMP               ; send to DS 1620
                CLR     RST                             ; stop transfer

                SETB            RST
                MOV A,          #02H                    ;write TL
                ACALL           SEND
                MOV A,          #32     ;32 *0.5 = 16 deg.c
                ACALL           SEND_TEMP
                CLR     RST             ; stop transfer

CONV:                   SETB            RST             ; initiate transfer
                MOV A,          #0EEH           ; start temperature sensing
                ACALL           SEND            ; send
                CLR     RST             ; stop transfer

SENS:                   SETB RST
                MOV A,          #0AAH           ; read temperature
                ACALL           SEND
                ACALL           READ
                JB      THI,    OFF     ; if T >= 24 degrees, off
                JB      TLO,    ON      ; if T <= 16 degrees, on
```

```
CONTINUE:               CLR     RST
                SJMP    SENS            ; loop

OFF:            CLR     FURN                    ; turn furnace off
                SJMP    CONTINUE                ; keep sensing

ON:             SETB            FURN                    ; turn furnace on
                SJMP    CONTINUE                ; keep sensing
;*******************************************************
; This subroutine sends a byte of command or data to the
; DS1620
;*******************************************************

SEND:                   MOV     R0,     #08H    ; use R0 as counter

NEXT:                   CLR     CLK             ; start clock cycle
                RRC     A               ; rotate A into C, LSB first
                MOV     DQ,     C       ; send out bit to DQ
                SETB            CLK             ; complete the clock cycle
                DJNZ            R0,     NEXT; process next bit
                RET

READ:                   MOV     R1,     #09H

NEXT1:                  CLR     CLK
                MOV     C,      DQ
                RRC     A
                SETB            CLK
                DJNZ            R1,     NEXT1
                MOV     30H,    A
                CLR     CLK
                MOV     C,      DQ
                RRC     A
                SETB            CLK
                MOV     31H,    A
                RET

SEND_TEMP:              MOV     R2,     #09H

NEXT2:                  CLR     CLK
                RRC     A
                MOV     DQ,     C
                SETB    CLK
                DJNZ    R2,     NEXT2
```

```
RET
END
```