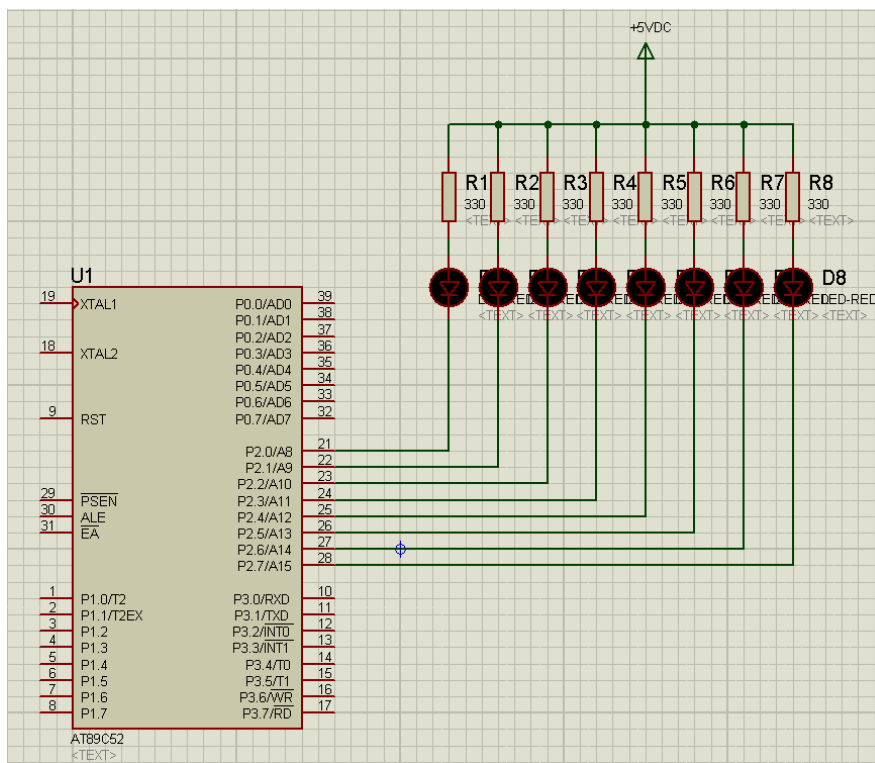


# BÀI THÍ NGHIỆM VI XỬ LÝ 2017

## Bài thí nghiệm 1: Bật tắt led đơn, hàng led

### 1.1. Mô tả mạch phần cứng



Hình 1: Mạch nguyên lý nháy led

Mạch phần cứng được mô tả như trong hình 1, cổng P2 nối với 8 led đơn (ghi các giá trị 0/1 ra các chân tương ứng của cổng 1 để bật/tắt led)

### 1.2. Chương trình mẫu

#### 1.2.1. Chương trình nháy 1 led đơn

```

=====
;
; Nhay mot led don
; start
=====
led bit p2.0                ;led tai chan P2.0
org 00h
main:
    clr led                ;bat led

```

```

        lcall delay_500ms      ;delay 500ms
        setb led              ;tat led
        lcall delay_500ms
        sjmp main
delay_500ms:
        MOV 50H,#200
loop:   MOV 51H,#250
        DJNZ 51H,$
        DJNZ 50H,loop
        RET
end

```

### 1.2.2 Chương trình sáng lần lượt các led

```

;=====
; Sang lan luot cac led
; start
;=====
led_data equ p2
        org 00h
        main :
        mov a,#0x00
        setb c
        loop :
        rlc a
        cpl a
        mov led_data,a
        cpl a
        lcall delay_500ms
        sjmp loop

        delay_500ms :
        MOV 50H,#200
ll: MOV 51H,#250
        DJNZ 51H,$
        DJNZ 50H,ll

        RET
end

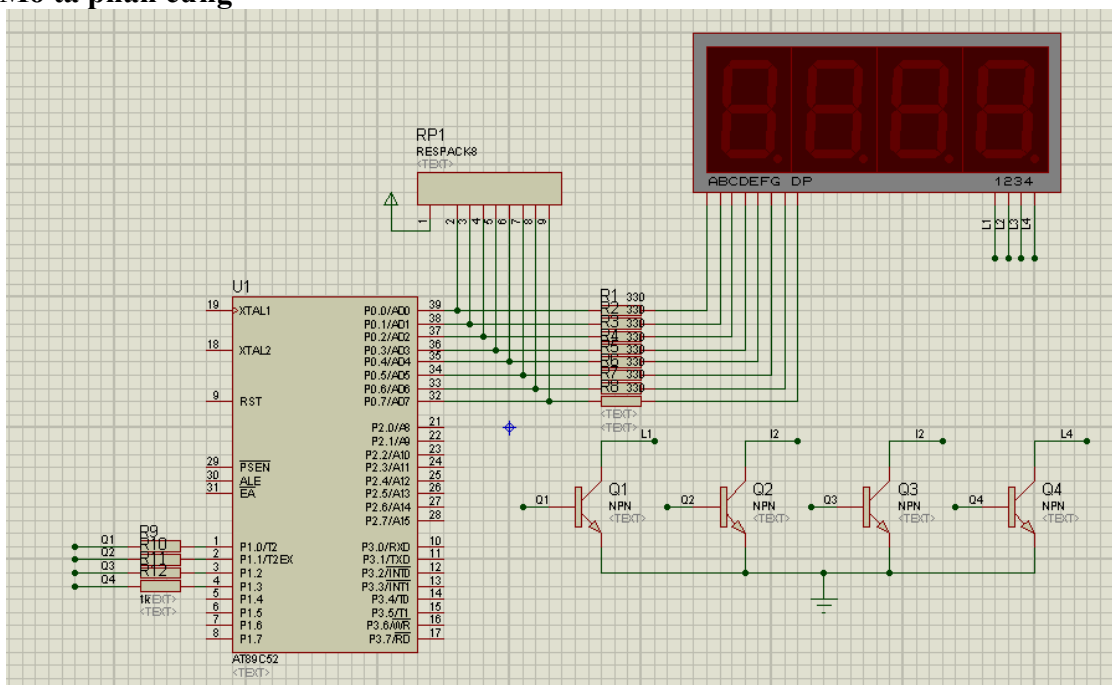
```

### 1.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus
- Nạp chương trình (dạng .hex file) và chạy được trên kit

### Bài thí nghiệm 2: Led 7 thanh

## 2.1. Mô tả phần cứng



Hình 2: Mạch nguyên lý nối led 7 thanh

Các chân A-DP của led 7 thanh tương ứng được nối với P0, 4 chân chọn led từ 1 đến 4 tương ứng nối với P1.0 đến P1.3

## 2.2 Chương trình mẫu

### 2.2.1. Hiện 1 số

```

=====
;
; Hien mot so
; start
=====
data_7seg equ p1
led1 bit p2.2
led2 bit p2.3
org 00h
main:
    mov dptr,#ma7seg ;so 0
    inc dptr ;+1
    inc dptr
    inc dptr
    inc dptr ;so 4
    clr led2 ;disable 7seg no2
    clr a
    movc a,@a+dptr
    mov data_7seg,a

```

```

    sjmp $    ;stop here
ma7seg:
    db 0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
end

```

### 2.2.2. Hiện 2 số (quét led)

```

=====
; hien hai so
; start
=====
data_7seg equ p1
led1 bit p2.2
led2 bit p2.3
org 00h
main:                                ;hien so 57 theo pp quet led
    mov dptr,#ma7seg
    setb led1                        ;enable 7seg no1
    mov a,#5
    movc a,@a+dptr
    mov data_7seg,a                  ;so 5
    lcall delay                      ;delay 250us
    clr led1                         ;disable 7seg no1

    setb led2
    mov a,#7
    movc a,@a+dptr
    mov data_7seg,a                  ;so 7
    lcall delay
    clr led2
    sjmp main

delay:
    mov r7,#250
    djnz r7,$
    ret

```

```

ma7seg:
    db 0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
end

```

### 2.2.3. Đếm từ 01-99 (quét led)

```

=====
; dem tu 1 den 99
=====
data_7seg equ p1
led1 bit p2.2

```

```

led2    bit p2.3

org 00h

main:
    mov r5,#100                ;1 so quet 100 lan
reset_r6:
    mov R6,#0
loop:
    mov a,r6
    cjne a,#100,skip           ;dem len 100 thi quay ve 0
    sjmp reset_r6
skip:
    mov b,#10
    div ab                     ;tach chu so hang chuc luu trong a, don vi luu trong b

    mov dptr,#ma7seg
    setb led1                  ;enable 7seg no1
    movc a,@a+dptr
    mov data_7seg,a           ;chu so hang chuc
    lcall delay_250us         ;delay 250us
    clr led1                  ;disable 7seg no1

    setb led2
    mov a,b
    movc a,@a+dptr
    mov data_7seg,a           ; chu so hang don vi
    lcall delay_250us
    clr led2

    djnz r5,loop               ; r5=0 tang r6 len 1
    inc r6
    mov r5,#100
    sjmp loop

;=====
delay_250us:
    mov r7,#250
    djnz r7,$
    ret
;=====
ma7seg:
    db 0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90
end

```

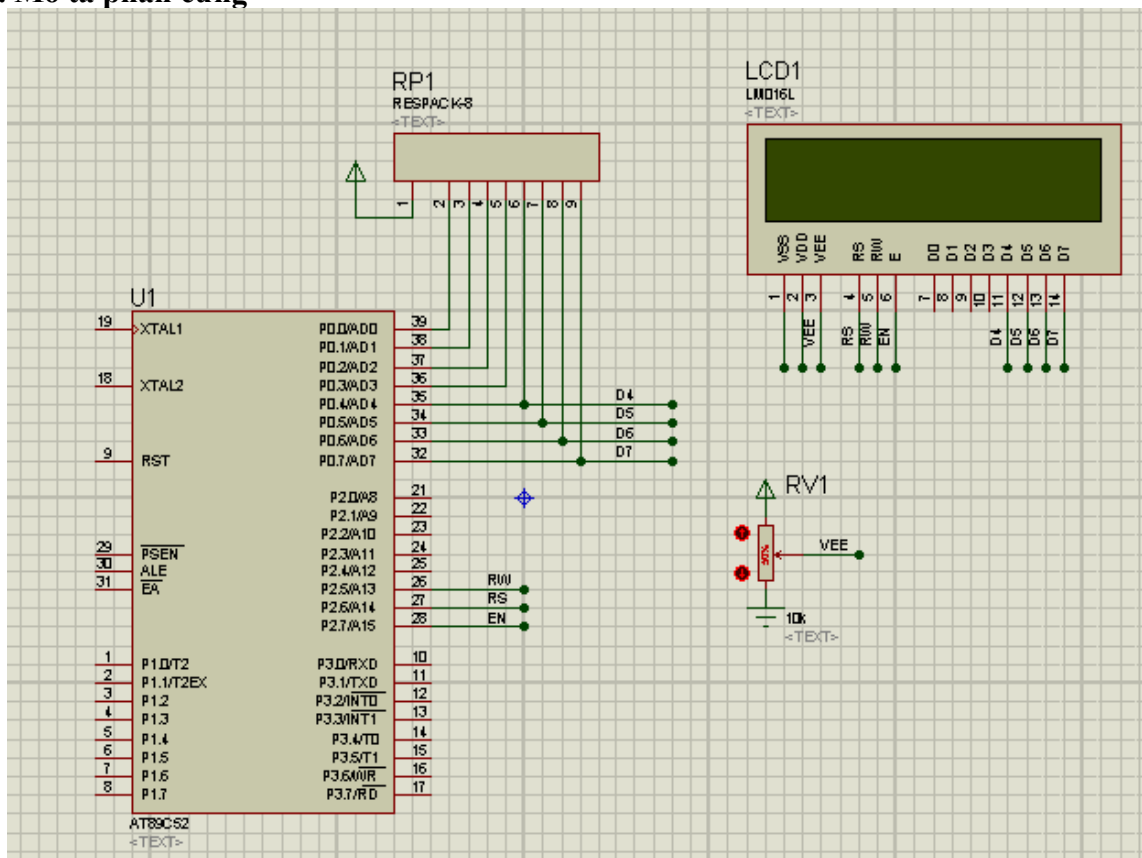
## 2.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả

- Mô phỏng trên phần mềm proteus
- Nạp chương trình (dạng .hex file) và chạy được trên kit

### Bài thí nghiệm 3: Hiện thị LCD

#### 3.1. Mô tả phần cứng



*Hình 3: Mạch nguyên lý nối LCD 16x2*

Sử dụng LCD 16x2, 4 chân tín hiệu D4 đến D7 của LCD được nối với 4 chân từ P0.4 đến P0.7. Tín hiệu điều khiển từ cổng P2.6 nối chân RS, cổng P2.5 nối với chân RW và tín hiệu từ chân P2.7 được nối đến chân E của LCD.

#### 3.2. Chương trình mẫu

```
=====
;
; Hiện thị LCD
;
=====
$mod51
```

```
U equ 31      ; memory location to hold upper nibble
L equ 21      ; memory location to hold lower nibble
Port equ P0   ; data port to connect LCD
RS equ P2.6   ; RS pin connection
RW equ P2.5   ; RW pin connection
EN equ P2.7   ; EN pin connection
```

```

;=====
; Connection of Port
; Port.4 = DB4
; Port.5 = DB5
; Port.6 = DB6
; Port.7 = DB7
;=====

```

```

SS MACRO L1
MOV R1,#0
MOV DPTR,L1
LCALL lcd_puts
ENDM

;=====
ORG 0000h
CLR RW
ACALL init
SS #STRING1
MOV A, #0c0H ; switch to 2nd line of LCD
ACALL lcd_cmd
SS #STRING2
SJMP $ ; INFINITE LONG LOOP
;=====
; Separator
;=====
separator:
MOV U,A ; save A at temp location U
ANL U,#0F0H ; mask it with 0Fh (28h & F0h = 20h)
SWAP A ; swap nibble (28h => 82H)
ANL A,#0F0H ; mask it with 0fh (82h & f0h = 80h)
MOV L,A ; save it at temp location L
RET ; return ;=====
; Move To Port
; MOV port,A
; put content of A to port
; ANL port,#0x0FH
; ORL port,A
;=====
move_to_Port:
MOV C,Acc.4
MOV port.4,C
MOV C,Acc.5
MOV port.5,C

```



```
MOV C,Acc.6
MOV port.6,C
MOV C,Acc.7
MOV port.7,C
SETB EN    ; make EN high
ACALL DELAY ; call a short delay routine
CLR EN     ; clear EN
ACALL DELAY ; short delay
RET
; return
;=====
```

```
;LCD command
;=====
lcd_cmd:
CLR RS    ; clear rs, going to send command
ACALL separator ; separate the command and save to U and L
MOV A, U   ; copy U to A
ACALL move_to_port ; move content of a to port
MOV A, L   ; copy L to A
ACALL move_to_port ; move content of a to port
RET
; return
;=====
; LCD data
;=====
lcd_data:
SETB RS    ; RS=1, going to send DATA
ACALL separator ; separate the data and save to U & L
MOV A, U   ; copy U to A
ACALL move_to_port ; send it to LCD
MOV A, L   ; copy L to A
ACALL move_to_port ; send it to LCD
RET        ; return
;=====
; Initilization
;=====
init:
ACALL delay ; some delay to lcd after power on
ACALL delay
CLR port.4
SETB port.5
CLR port.6
```



```
CLR port.7 ; send 20h to LCD to set 4 bit mode
CLR RS ; after that we can use lcd_cmd
SETB EN ; make EN switching
ACALL delay
CLR EN
MOV A, #28H
ACALL lcd_cmd
MOV A, #0CH
ACALL lcd_cmd
MOV A, #06H
ACALL lcd_cmd
MOV A, #01H
ACALL lcd_cmd
RET
```

```
;=====
lcd_puts:
```

```
MOV A,R1
MOVC A,@A+DPTR
LCALL lcd_data
INC R1
CJNE R1,#15,lcd_puts
RET
```

```
;=====
delay:
MOV R6, #5FH
L2: MOV R7,#3FH
L1: DJNZ R7, L1
DJNZ R6, L2
RET ;=====
STRING1: DB ' VI XU LY '
STRING2: DB '==TEST PROGRAM=='
```

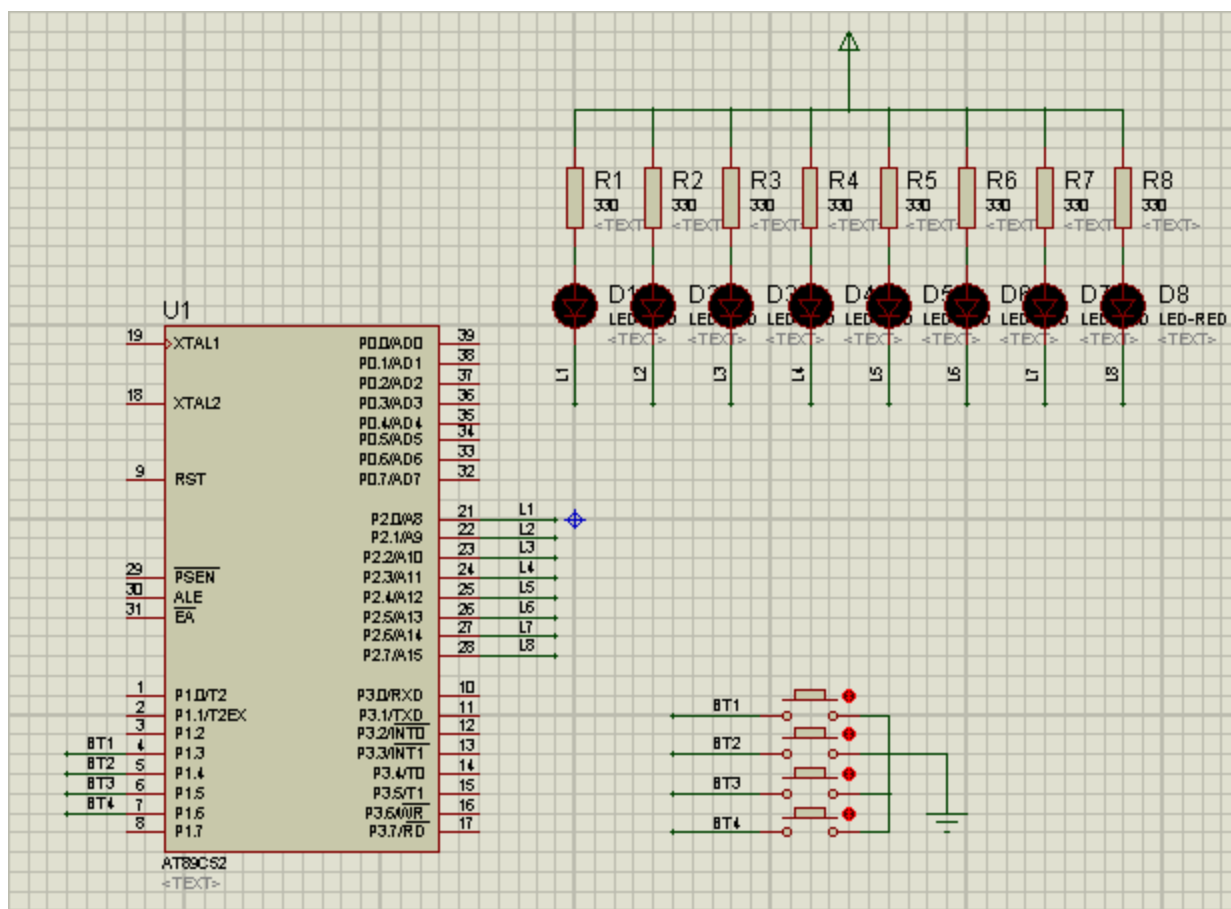
end

### 3.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus
- Nạp chương trình (dạng .hex file) và chạy được trên kit

## Bài thí nghiệm 4: Giải mã phím

### 4.1. Mô tả phần cứng



Hình 4: Mạch nguyên lý giải mã mã phím

Phím BT1 đến BT4 được nối với 4 chân P1.3, P1.4, P1.5, P1.6, 8 led từ L1 đến L8 được nối đến 8 chân của P2 theo thứ tự từ P2.0 đến P2.7. Khi nhấn phím BT1 thì led L1 sẽ sáng và nhà nút thì led L1 sẽ tắt.

## 4.2. Chương trình

### 4.2.1. 1 phím bật tắt led đơn

```
=====
;
; Mot phim bat tat mot led
;
=====
ORG 00H
MAIN:

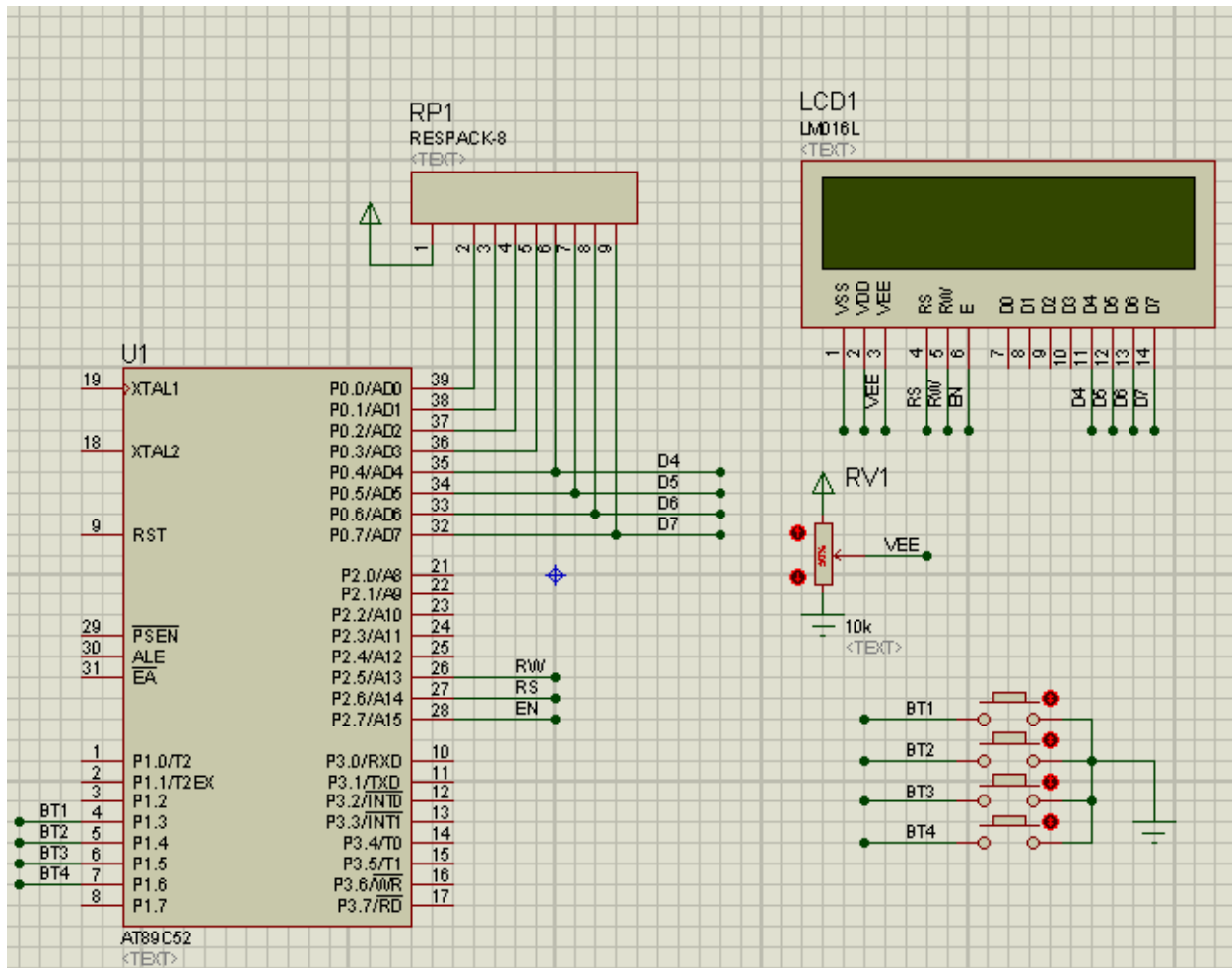
JNB P1.4,PHIM1
JMP MAIN
PHIM1:
CLR P2.0
acall delay_500ms
SETB P2.0
RET
```

```

delay_500ms:
MOV 50H,#200
I1: MOV 51H,#250
DJNZ 51H,$
DJNZ 50H,I1
RET
END

```

#### 4.2.2. Quét phím hiển thị LCD



Hình 5: Mạch nguyên lý giải mã phím hiển thị LCD

```

=====
;
; Quet phim hien thi len LCD
;
=====
$mod51
U equ 31      ; memory location to hold upper nibble
L equ 21      ; memory location to hold lower nibble

```



Port equ P0 ; data port to connect LCD

RS equ P2.6 ; RS pin connection

RW equ P2.5 ; RW pin connection

EN equ P2.7 ; EN pin connection

=====

; Connection of Port

; Port.4 = DB4

; Port.5 = DB5

; Port.6 = DB6

; Port.7 = DB7

=====

SS MACRO L1

MOV R1,#0

MOV DPTR,L1

LCALL lcd\_puts

ENDM

=====

ORG 0000h

CLR RW

ACALL init

main:

SS #STRING1

MOV A, #0c0H ; switch to 2nd line of LCD

ACALL lcd\_cmd

SS #STRING2

quetphim:

JNB P1.4,PHIM1

JNB P1.5,PHIM2

===== Q=====

JNB P1.6,PHIM3

JNB P1.7,PHIM4

```

sjmp main
;=====
PHIM1:
MOV A, #0c0H      ; switch to 2nd line of LCD
ACALL lcd_cmd
SS #KEY1
JNB P1.4,$
sjmp main
;=====
PHIM2:
MOV A, #0c0H      ; switch to 2nd line of LCD
ACALL lcd_cmd
SS #KEY2
JNB P1.5,$
sjmp main
;=====
PHIM3:
MOV A, #0c0H      ; switch to 2nd line of LCD
ACALL lcd_cmd
SS #KEY3
JNB P1.6,$
sjmp main
;=====
PHIM4:
MOV A, #0c0H      ; switch to 2nd line of LCD
ACALL lcd_cmd
SS #KEY4
JNB P1.7,$
sjmp main
//SJMP $      ; INFINITE LONG LOOP
;=====
; Separator
;=====

```

separator:

```
MOV U,A      ; save A at temp location U
ANL U,#0F0H  ; mask it with 0Fh (28h & F0h = 20h)
SWAP A       ; swap nibble (28h => 82H)
ANL A,#0F0H  ; mask it with 0fh (82h & f0h = 80h)
MOV L,A      ; save it at temp location L
RET          ; return ;=====
```

; Move To Port

; MOV port,A

; put content of A to port

; ANL port,#0x0FH

; ORL port,A

;=====

move\_to\_Port:

MOV C,Acc.4

MOV port.4,C

MOV C,Acc.5

MOV port.5,C

MOV C,Acc.6

MOV port.6,C

MOV C,Acc.7

MOV port.7,C

SETB EN ; make EN high

ACALL DELAY ; call a short delay routine

CLR EN ; clear EN

ACALL DELAY ; short delay

RET

; return

;=====

;LCD command

;=====

lcd\_cmd:

```

CLR RS      ; clear rs, going to send command
ACALL separator ; separate the command and save to U and L
MOV A, U     ; copy U to A
ACALL move_to_port ; move content of a to port
MOV A, L     ; copy L to A
ACALL move_to_port ; move content of a to port
RET
; return
;=====
; LCD data
;=====
lcd_data:
SETB RS      ; RS=1, going to send DATA
ACALL separator ; separate the data and save to U & L
MOV A, U     ; copy U to A
ACALL move_to_port ; send it to LCD
MOV A, L     ; copy L to A
ACALL move_to_port ; send it to LCD
RET          ; return
;=====
; Initilization
;=====
init:
ACALL delay   ; some delay to lcd after power on
ACALL delay
CLR port.4
SETB port.5
CLR port.6
CLR port.7    ; send 20h to LCD to set 4 bit mode
CLR RS       ; after that we can use lcd_cmd
SETB EN      ; make EN switching
ACALL delay
CLR EN

```

```
MOV A, #28H
ACALL lcd_cmd
MOV A, #0CH
ACALL lcd_cmd
MOV A, #06H
ACALL lcd_cmd
MOV A, #01H
ACALL lcd_cmd
RET
;=====
lcd_puts:
MOV A,R1
MOVC A,@A+DPTR
LCALL lcd_data
INC R1
CJNE R1,#15,lcd_puts
RET
;=====
delay:
MOV R6, #5FH
L2: MOV R7,#3FH
L1: DJNZ R7, L1
DJNZ R6, L2
RET ;=====
STRING1: DB ' VI XU LY '
        STRING2: DB ' Press one key '
        KEY1:  DB ' KEY 1   '
        KEY2:  DB ' KEY 2   '
        KEY3:  DB ' KEY 3   '
        KEY4:  DB ' KEY 4   '

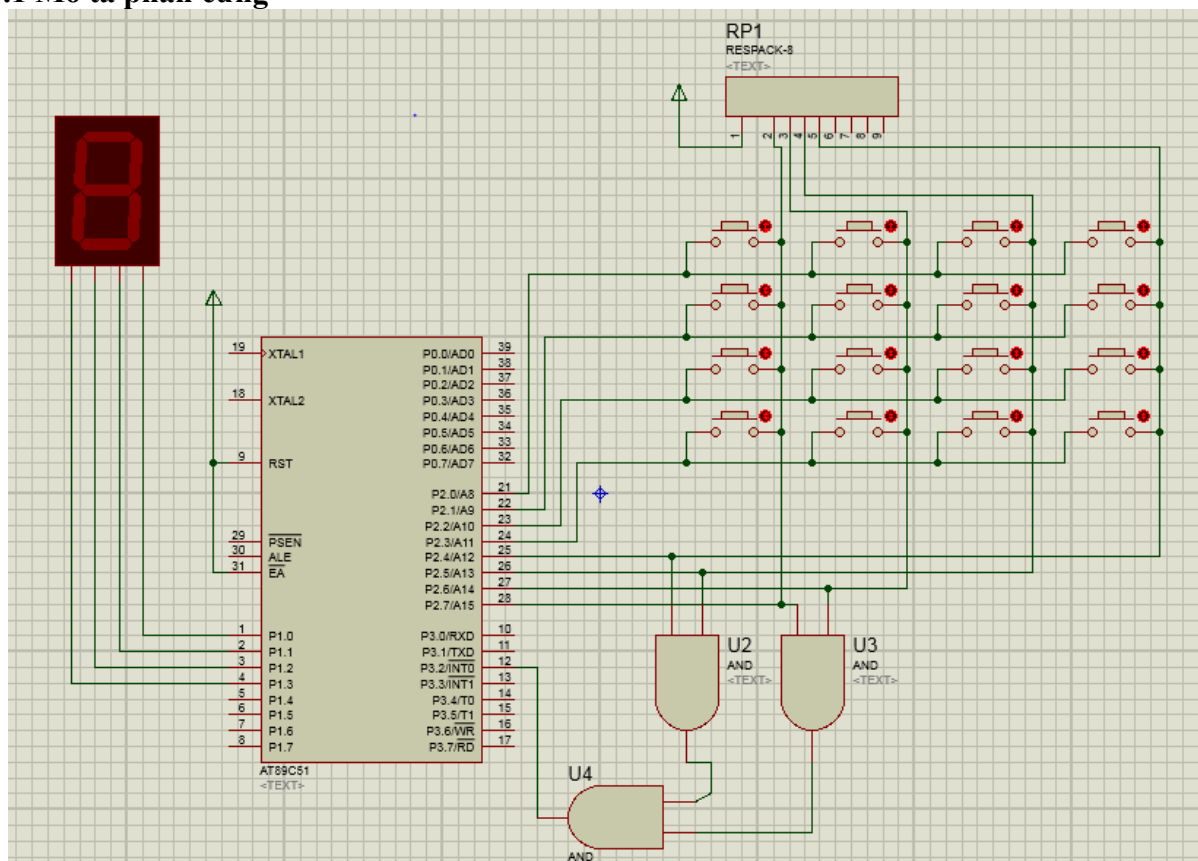
End
```

### 4.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả



## 5.1 Mô tả phần cứng



**Hình 6: Mạch nguyên lý giải mã phím 4x4**

Bốn hàng được nối với P2.0-P2.3, bốn cột được nối với P2.4-P2.7. Bấm từng phím từ trái sang phải, từ trên xuống dưới sẽ hiện số tuần tự 0 đến 9 trên led 7 thanh. . Mô phỏng trên proteus bằng cách mở file “Keypad\_Scan\_Interrupt” theo đường dẫn sau: C:\TNVXL\students\Bai 5 - Quet ban phim dung ngat.

## 5.2 Chương trình mẫu

```

;=====
; giai ma ban phim 4x4
;=====
$NOMOD51
$INCLUDE (8051.MCU)
;=====

```

```

; Reset Vector
org 0000h
jmp Start

ORG 0003H
LJMPINT0_ISR

;=====
; CODE SEGMENT
;=====

org 0100h

Start:
    SETB EA                ; Enable Interrupt
    SETB EX0               ; Enable INT0

;=====
;    SETB IT0              ; INT0 NGAT THEO SUON XUONG
;=====

    MOV    TMOD, #20H
    MOV    TH1, #0B0H

    MOV    P2, #0FH
    MOV    P1, #0
    MOV    R1, #0          ; COUNT
    CLR    A
    SJMP $

;=====
INT0_ISR:
    CALL DELAY
    JB     P3.2, EXIT
    MOV    P2, #11111110B
    MOV    A, P2
    ANL    A, #0F0H
    CJNE   A, #0F0H, ROW0
    MOV    P2, #11111101B

```

```
MOV      A, P2
MOV      R1, #4
ANL      A, #0F0H
CJNE     A, #0F0H, ROW1
MOV      P2, #11111011B
MOV      A, P2
MOV      R1, #8
ANL      A, #0F0H
CJNE     A, #0F0H, ROW2
MOV      P2, #11110111B
MOV      A, P2
MOV      R1, #12
ANL      A, #0F0H
CJNE     A, #0F0H, ROW3
```

ROW0:

```
RLC      A
JNC      MATCH
INC      R1
SJMP     ROW0
```

ROW1:

```
RLC      A
JNC      MATCH
INC      R1
SJMP     ROW1
```

ROW2:

```
RLC      A
JNC      MATCH
INC      R1
SJMP     ROW2
```

ROW3:

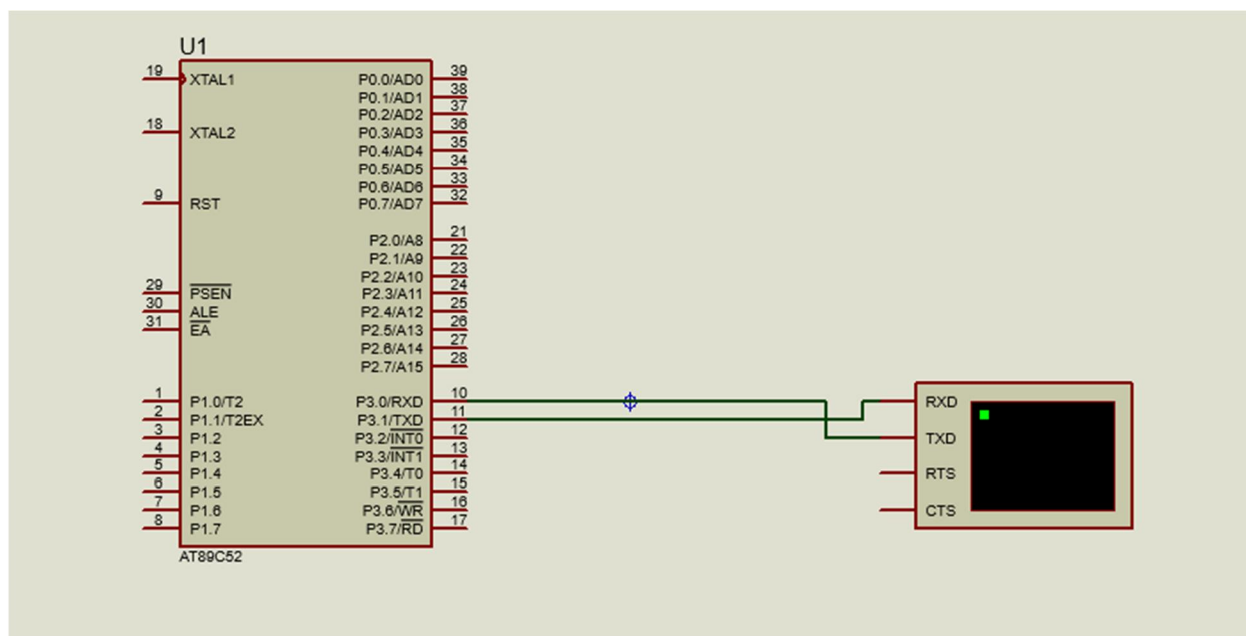
```
RLC      A
JNC      MATCH
INC      R1
```

```
                SJMP ROW3
MATCH:
                MOV     A, #0
                MOV     A, R1
                MOV     P1, A
EXIT:
                MOV     R1, #0
                MOV     P2, #0F0H
                RETI
;=====
DELAY:
                SETB    TR1
                JNB     TF1, $
                CLR     TF1
                CLR     TR1
                RET
;=====
END
```

### 5.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus

### Bài thí nghiệm 6: Truyền tin UART



*Hình 7: Mạch nguyên lý truyền thông UART*

## 6.1. Chương trình mẫu

```

;=====
; Gửi 1 byte từ máy tính xuống vi điều khiển,vi điều khiển rồi đọc rồi gửi lại byte tương tự lên máy
; tính
;=====
org 000h
ljmp begin
org 23h
ljmp serial_IT
;=====
;/**
; * FUNCTION_PURPOSE: This file set up uart in mode 1 (8 bits uart) with
; * timer 1 in mode 2 (8 bits auto reload timer).
; * FUNCTION_INPUTS: void
; * FUNCTION_OUTPUTS: void
; */
;=====
org 0100h
begin:
MOV SCON, #50h ; /* uart in mode 1 (8 bit), REN=1 */
ORL TMOD, #20h ; /* Timer 1 in mode 2 */
MOV TH1, #0F9h ; /* 9600 Bds at 12MHz */
MOV TL1, #0F9h ; /* 9600 Bds at 12MHz */
MOV A,PCON
SETB ACC.7
MOV PCON,A
SETB ES ; /* Enable serial interrupt*/
SETB EA ; /* Enable global interrupt */

```

```
SETB TR1 ; /* Timer 1 run */
JMP $ ; /* endless */
;/**
; * FUNCTION_PURPOSE: serial interrupt, echo received data.
; * FUNCTION_INPUTS: P3.0(RXD) serial input
; * FUNCTION_OUTPUTS: P3.1(TXD) serial output
; */
serial_IT:
JNB RI,EMIT_IT ; test if it is a reception
CLR RI ; clear reception flag for next reception
MOV A,SBUF ; read data from uart
MOV SBUF,A ; write same data to uart
LJMP END_IT
EMIT_IT:
CLR TI ; clear transmittion flag for next transmittion
END_IT:
RETI
end
```

## 6.2 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus

## Bài thí nghiệm 7: Đo nhiệt độ bằng cảm biến LM35

### 7.1 Mô tả phần cứng

Sử dụng CLOCK ngoài cấp cho ADC0808. Đọc nhiệt độ vào chân IN0 của ADC. Chân dữ liệu của 8051 gửi lên led 7 thanh là: P0.0 – P0.6 nối lần lượt với các chân A – G của led 7 thanh . Mô phỏng trên proteus bằng cách mở file “do nhiet do\_89c51” theo đường dẫn sau: C:\TNVXL\students\Bai 7 Do nhiet do\mo phong.

do nhiet do

```

led1 bit p1.0
led2 bit p1.1
led3 bit p1.2
led4 bit p1.3
org 000h
main: lcall cdoi
lcall hex_bcd
lcall bcd_7doan
lcall hienthi
jmp main
cdoi:
setb ale
clr ale
setb start

```

```

jb eoc,$
clr start
mov r7,#150
de: lcall hienthi
djnz r7,de
mov a,p3
ret
hex_bcd:
mov b,#10
div ab
mov 10h,b
mov 11h,a
ret
bcd_7doan:
mov dptr,#900h
mov a,10h
movc a,@a + dptr
mov 20h,a
mov a,11h
movc a,@a + dptr
mov 21h,a
ret
hienthi: mov p0,21h
setb led4
lcall delay
anl p1,#0f0h    ; p1=----1111

mov p0,20h
setb led3
lcall delay
anl p1,#0f0h

mov p0,#063h
setb led2
lcall delay
anl p1,#0f0h

mov p0,#039h
setb led1
lcall delay
anl p1,#0f0h

ret
delay: mov 7fh,#100
djnz 7fh,$
ret

```



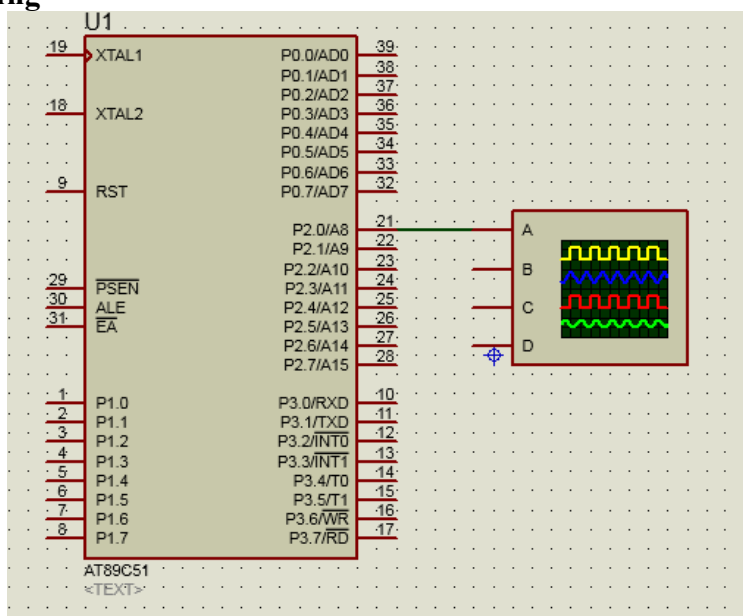
```
org 900h
db 03fh,006h,05bh,04fh,066h,06dh,07dh,007h,07fh,06fh
END
```

### 7.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus

## Bài thí nghiệm 8: Tạo xung vuông tần số 5 KHz

### 8.1 Mô tả phần cứng



Hình 9: Mạch tạo xung vuông

Tạo xung vuông tần số 5 kHz trên chân P2.0, sử dụng Oscilloscope để đo tần số của xung được tạo ra. Mô phỏng trên proteus bằng cách mở file “tạo xung” theo đường dẫn sau: C:\TNVXL\students\Bai9 PWM\mo phong.

### 8.2 Chương trình

```
;=====
;
; tạo xung vuông
;=====
;tan so thach anh 11.0592MHz
;=====
;
; su dung timer0 mode 2 de tạo tan so xung vuông f=5kHz
;=====
org 0
ljmp main
org 0bh
```

```

ljmp    interrupt_timer0
org     800h
main:
    mov     tmod,#2
    mov     tl0,#0a3h
    mov     th0,#0a3h
    setb    tr0
    setb    et0
    setb    ea
    sjmp    $

interrupt_timer0:
    cpl     p2.0    ;dao chan p2.0
    reti
end

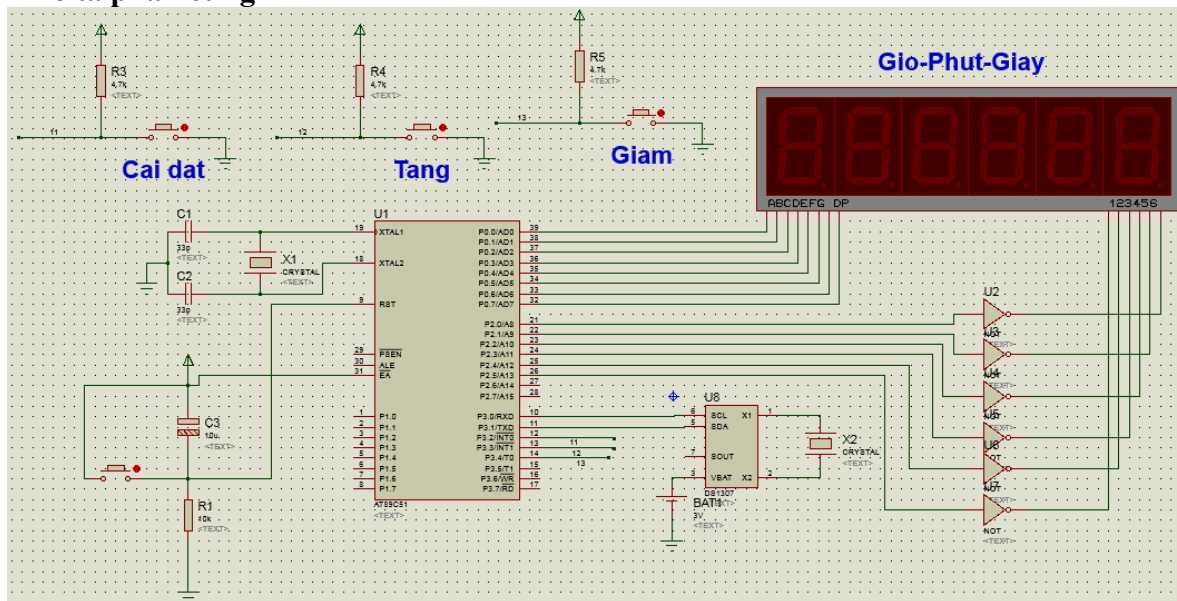
```

### 8.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả, thay đổi tần số khác
- Mô phỏng trên phần mềm proteus
- Sinh viên viết chương trình tạo xung không dùng ngắt, chạy thử và so sánh
- Nạp chương trình (dạng .hex file) và chạy được trên kit

## Bài thí nghiệm 9: Giao tiếp I2C, thiết kế mạch đồng hồ.

### 9.1 Mô tả phần cứng



Hình 10: Mạch nguyên lý mạch đồng hồ

Sử dụng IC thời gian thực DS1307 để hiển thị thời gian trên led 7 thanh. Mạch sử dụng các nút bấm để cài đặt thời gian cho đúng. Chân dữ liệu của 8051 gửi lên led 7 thanh là: P0.0 – P0.7 nối

lần lượt với các chân A – G - DP của led 7 thanh. Mô phỏng trên proteus bằng cách mở file “Dong ho ds1307 Dn proteus” theo đường dẫn sau: C:\TNVXL\students\Bai 8 Mach dong ho\mo phong

## 9.2 Chương trình

```

=====
;
; Mach dong ho
;
=====
$mod51
TEMP          DATA 37H
XUNG_NHAY     DATA 38H ; XUNG 100ms
BIEN_NHAY     DATA 39H ; 0 = SANG TAT CA DEN , 1 = NHAY led TUONG UNG KHI set
GIAY         DATA 40H
PHUT         DATA 41H
GIO          DATA 42H
DONVI_GIAY    DATA 47H
CHUC_GIAY     DATA 48H
DONVI_PHUT    DATA 49H
CHUC_PHUT     DATA 4AH
DONVI_GIO     DATA 4BH
CHUC_GIO      DATA 4CH
PHAN_TRAM_GIAY DATA 4DH
FLAG_SET DATA 4EH ; 0 = KHONG SET , 1 = SET PHUT , 2 = SET GIO
LED_GIAY BIT P2.0
LED_C_GIAY BIT P2.1
LED_PHUT BIT P2.2
LED_C_PHUT BIT P2.3
LED_GIO BIT P2.4
LED_C_GIO BIT P2.5
;=====I2C=====
SCL BIT P3.0
SDA BIT P3.1
SW_1 BIT P3.2
SW_2 BIT P3.3
SW_3 BIT P3.4
LED_DATA EQU P0
BYTE_W EQU 11010000B
BYTE_R EQU 11010001B
ADD_LOW EQU 62H
DATA_DS EQU 63H
;=====
ORG 00H
LJMP MAIN
;=====
ORG 0BH
LJMP NGAT_TIME
;=====
ORG 030H

```

```

MAIN: ;reset tat ca cac bien
      MOV GIAY,#0
      MOV PHUT,#0
      MOV GIO,#0
      MOV BIEN_NHAY,#0
      MOV XUNG_NHAY,#0
      MOV FLAG_SET,#0
      MOV R0,#0
      MOV IE,#10001010B
      MOV TMOD,#11H
      MOV TL0,#LOW(-9216)
      MOV TH0,#HIGH(-9216)
      SETB TR0
      MOV A,#0FFH
      MOV LED_DATA,A
      MOV DPTR,#BANGSO
      CLR SCL
      CLR SDA
      NOP
      SETB SCL
      SETB SDA
      NOP
      MOV ADD_LOW,#00H
      MOV DATA_DS,#00H
      LCALL WRITE_BYTE
;=====
LOOP_HIEN_THI: ; chuong trinh chinh chay tai day
;=====
      MOV A,FLAG_SET
      CJNE A,#0,L_HT
      CALL INIT_PORT
L_HT:
      LCALL HIEN_THI
      LCALL SCAN_KEY
      SJMP LOOP_HIEN_THI
;=====
INIT_PORT:
;=====
; READS SECONDS
;=====
READ_SEC:
      MOV ADD_LOW,#00h
      LCALL READ_BYTE
      MOV A,DATA_DS
      CALL BCD_HEX
      MOV GIAY,A

```



```
LCALL I2C_STOP
;=====
; READS MINUTES
;=====
MOV ADD_LOW,#01h
LCALL READ_BYTE
MOV A,DATA_DS
CALL BCD_HEX
MOV PHUT,A
LCALL I2C_STOP
;=====
; READS HOURS
;=====
MOV ADD_LOW,#02h
LCALL READ_BYTE
MOV A,DATA_DS
CALL BCD_HEX
MOV GIO,A
LCALL I2C_STOP
RET
;=====
; Stop I2C communication
;=====
I2C_Stop:
CLR SDA
SETB SCL
NOP
SETB SDA
RET
;=====
;*****
; * WRITE DATA_DS TO DS1307 1 BYTE *
; * INPUT : ADD_LOW *
; * : DATA_DS *
;*****
;=====
WRITE_BYTE:
CLR SDA ; start bit
CLR SCL
MOV A,#BYTE_W ; send control byte
LCALL LOOP_BYTE
SETB SDA
SETB SCL
JB SDA,WRITE_BYTE ; loop until busy
CLR SCL
MOV A,ADD_LOW ; send address low
```

```

    LCALL LOOP_BYTE
    SETB SDA
    SETB SCL
    JB SDA,WRITE_BYTE    ; loop until busy
    CLR SCL
    MOV A,DATA_DS        ; send DATA
    LCALL LOOP_BYTE
    SETB SDA
    SETB SCL
    JB SDA,WRITE_BYTE    ; loop until busy
    CLR SDA
    CLR SCL
    SETB SCL              ; stop bit
    SETB SDA
    RET

;=====
;
BCD_HEX:
;=====
;
    MOV B,#10H
    DIV AB
    MOV TEMP,B           ;CAT HANG DON VI
    MOV B,#10
    MUL AB
    ADD A,TEMP
    ret

;=====
;
HEX_BCD:
;=====
;
    MOV B,#10
    DIV AB
    MOV TEMP,B           ;CAT HANG DON VI
    MOV B,#10H
    MUL AB
    ADD A,TEMP
    ret

;=====
;
;*****
;
; * READ DATA FROM DS1307 1 BYTE *
; * INPUT : ADD_HIGH *
; * : ADD_LOW *
; * OUTPUT : DATA_DS *
;*****
;
;=====
;
READ_BYTE:
    CLR SDA              ; start bit
    CLR SCL

```



```
MOV A,#BYTE_W          ; send control byte
LCALL LOOP_BYTE
SETB SDA
SETB SCL
JB SDA,READ_BYTE       ; loop until busy
CLR SCL
MOV A,ADD_LOW          ; send address low
LCALL LOOP_BYTE
SETB SDA
SETB SCL
JB SDA,READ_BYTE       ; loop until busy
CLR SCL
SETB SCL
SETB SDA
CLR SDA                ;start bit
CLR SCL
MOV A,#BYTE_R          ;send control byte
LCALL LOOP_BYTE
SETB SDA
SETB SCL
JB SDA,READ_BYTE       ;loop until busy
CLR SCL
LCALL LOOP_READ
SETB SDA
SETB SCL
CLR SCL
SETB SCL               ;stop bit
SETB SDA
RET

;=====
;
;*****
;
;* WRITE *
;* INPUT: ACC
;*****
;
;=====
LOOP_BYTE:
    PUSH 02H
    MOV R2,#08H
LOOP_SEND:
    RLC A
    MOV SDA,C
    SETB SCL
    CLR SCL
    DJNZ R2,LOOP_SEND
    POP 02H
    RET
```

```

=====
;
;*****
;
;* READ *
;* OUTPUT: ACC *
;*****
;
=====
LOOP_READ:
    PUSH 02H
    MOV R2,#08H
LOOP_READ1:
    SETB SCL
    MOV C,SDA
    CLR SCL
    RLC A
    DJNZ R2,LOOP_READ1
    MOV DATA_DS,A
    POP 02H
    RET
;
=====
TACHSO: ; tach rieng hang chuc va hang don vi bang cach chia cho 10
;
    MOV A,GIAY ;Lan luot chia cac Bien: Giay, Phut, Gio cho 10
    MOV B,#10 ;de tach phan Don Vi va Hang Chuc ra, de cat rieng vao cac Bien tuong ung.
    DIV AB ;PHAN NGUYEN trong A, PHAN DU trong B
    MOV CHUC_GIAY,A ;Luu lai HANG CHUC Giay
    MOV DONVI_GIAY,B ;luu lai DON VI Giay
;
    MOV A,PHUT
    MOV B,#10
    DIV AB
    MOV CHUC_PHUT,A
    MOV DONVI_PHUT,B
;
    MOV A,GIO
    MOV B,#10
    DIV AB
    MOV CHUC_GIO,A
    MOV DONVI_GIO,B
    RET
;
=====
HIEN_THI: ; HIEN THI LED 7 DOAN
;
    MOV A,FLAG_SET
    CJNE A,#0,CHOP_NHAY
    LCALL HIENTHI
    AJMP THOAT_HIENTHI

```



```

;=====
;
CHOP_NHAY: ; KIEM TRA BIEN NHAY VA FLAG_SET DE TAO HIEU UNG NHAY LED
DANG SETING
;=====
;
    MOV A,BIEN_NHAY
    CJNE A,#0,CHOP_NHAY1
    LCALL HIENTHI
    AJMP THOAT_HIENTHI
CHOP_NHAY1:
    LCALL NHAY
    JMP CHOP_NHAY
THOAT_HIENTHI:
    RET
;=====
;
HIENTHI:
;=====
;
    LCALL HIENTHI_S
    LCALL HIENTHI_P
    LCALL HIENTHI_G
    RET
;=====
;
NHAY:
;=====
;
    MOV A,FLAG_SET
    CJNE A,#1,KT1
    LCALL HIENTHI_S
    LCALL HIENTHI_G
KT1:
    MOV A,FLAG_SET
    CJNE A,#2,THOAT_N
    LCALL HIENTHI_S
    LCALL HIENTHI_P
THOAT_N:
    RET
;=====
;
HIENTHI_S:
;=====
;
;hien thi hang don vi cua Giay
    MOV A,DONVI_GIAY
    MOVC A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_GIAY
    LCALL DL
    SETB LED_GIAY
;=====
;
    MOV A,CHUC_GIAY ;hien thi hang chuc cua Giay

```

```

    MOV A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_C_GIAY
    LCALL DL
    SETB LED_C_GIAY
    RET
;=====
;
HIENTHI_P:
;=====
;
    MOV A,DONVI_PHUT ;hien thi hang don vi cua Phut
    MOV A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_PHUT
    LCALL DL
    SETB LED_PHUT
;=====
;
    MOV A,CHUC_PHUT ;hien thi hang chuc cua Phut
    MOV A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_C_PHUT
    LCALL DL
    SETB LED_C_PHUT
    RET
;=====
;
HIENTHI_G:
;=====
;
    MOV A,DONVI_GIO ;hien thi hang don vi cua gio
    MOV A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_GIO
    LCALL DL
    SETB LED_GIO
;=====
;
    MOV A,CHUC_GIO ;hien thi hang chuc cua Gio
    MOV A,@A+DPTR
    MOV LED_DATA,A
    CLR LED_C_GIO
    LCALL DL
    SETB LED_C_GIO
    RET
;=====
;
NGAT_TIME:
;=====
;
    INC XUNG_NHAY
    INC PHAN_TRAM_GIAY          ;DAT TIMER CHAY 1/100 GIAY
    MOV TL0,#LOW(-9216)

```

```

MOV TH0,#HIGH(-9216)
SETB TR0
;=====
;
PUSH ACC
PUSH PSW                                ;Thanh ghi trang th?i chuong tr?nh
;=====
;
MOV A,XUNG_NHAY ;TAO XUNG NHAP NHAY = 1/4 GIAY
CJNE A,#25,TIME1
MOV XUNG_NHAY,#0
INC BIEN_NHAY
MOV A,BIEN_NHAY
CJNE A,#3,TIME1
MOV BIEN_NHAY,#0
;=====
;
TIME1:
MOV A,PHAN_TRAM_GIAY ;Kiem tra bien PHAN_TRAM_GIAY - Thoat khoi ngat
Time0 neu khong =
CJNE A,#100,THOAT_NGAT_TIME
MOV PHAN_TRAM_GIAY,#0 ;Neu = 100 th? set bien nay = 0
;=====
;
THOAT_NGAT_TIME:
LCALL TACHSO
POP PSW
POP ACC
RETI
;=====
;
SCAN_KEY: ;KIEM TRA PHIM NHAN
;=====
;
SW1: ;SET TIME
JB SW_1,SW2
INC FLAG_SET
MOV A,FLAG_SET
CJNE A,#3,L_SW1
;=====
;
MOV A,PHUT
CALL HEX_BCD
MOV DATA_DS,A
MOV ADD_LOW,#01H
LCALL WRITE_BYTE
;=====
;
MOV A,GIO
CALL HEX_BCD
MOV DATA_DS,A
MOV ADD_LOW,#02H
LCALL WRITE_BYTE
MOV FLAG_SET,#0

```

```

L_SW1:
    LCALL DL1
    LCALL DL1
    LCALL DL1
    LCALL DL1
    LJMP NOKEY
;=====
SW2: ;SET_MIN
    JB SW_2,SW3
    MOV A,FLAG_SET
    CJNE A,#0,SW20
    LJMP NOKEY
SW20:
    MOV A,FLAG_SET
    CJNE A,#1,TANG_GIO ;
    JB SW_2,SW3
;=====
TANG_PHUT:
    INC PHUT ;Roi tang Bien phut them 1
    MOV A,PHUT
    CJNE A,#60,L_SW2 ;
    MOV PHUT,#0 ;Neu = 60 th? set bien nay = 0
L_SW2:
    LCALL DL1
    LCALL DL1
    LJMP SW2
;=====
TANG_GIO: ;SET HOUR
    JB SW_2,SW3
    MOV A,FLAG_SET
    CJNE A,#2,SW3
    JB SW_2,SW3
    INC GIO ;Roi tang Bien Gio them 1
    MOV A,GIO
    CJNE A,#24,L_TANG_GIO ;Bien gio = 60? - Thoat khoi ngat Time0 neu khong =
    MOV GIO,#0
L_TANG_GIO:
    LCALL DL1
    LCALL DL1
    LJMP TANG_GIO
;=====
SW3: ;DANG NHAN SW3?
    JB SW_3,NOKEY ;KHONG NHAN SW2? KIEM TRA SW3
    MOV A,FLAG_SET
    CJNE A,#0,SW30
    LJMP NOKEY

```

SW30:

MOV A,FLAG\_SET ;DANG NHAN SW2. KIEM TRA CHE DO CHINH GIO HAY CHINH PHUT.

CJNE A,#1,GIAM\_GIO ;

JB SW\_3,NOKEY

=====

GIAM\_PHUT:

DEC PHUT ;Roi tang Bien phut them 1

MOV A,PHUT

CJNE A,#-1,L\_SW3 ;Bien Phut = -1? - Thoat khoi ngat Time0 neu khong =

MOV PHUT,#59 ;Neu = -1 th? set bien nay = 60

L\_SW3:

LCALL DL1

LCALL DL1

LJMP SW3

=====

GIAM\_GIO:

=====

JB SW\_3,NOKEY

MOV A,FLAG\_SET

CJNE A,#2,NOKEY

LCALL HIENTHI

JB SW\_3,NOKEY

DEC GIO ;Roi Giam Bien Gio them 1

MOV A,GIO

CJNE A,#-1,L\_GIAM\_GIO ;Bien gio = -1? - Thoat khoi ngat Time0 neu khong =

MOV GIO,#23

L\_GIAM\_GIO:

LCALL DL1

LCALL DL1

LJMP GIAM\_GIO

=====

NOKEY:

RET

=====

DL:

=====

MOV R7,#200

DJNZ R7,\$

RET

=====

DL1:

=====

PUSH 00H

PUSH 01H

MOV R1,#200

```

DEL:
    LCALL HIEN_THI
    LCALL HIEN_THI
    LCALL HIEN_THI
    MOV R0,#250
    DJNZ R0,$
    DJNZ R1,DEL
    POP 01H
    POP 00H
    RET
;=====
;
DELAY_1MS:
;=====
;
    MOV R7,#200
DL_1MS_1:
    MOV R6,#200
    DJNZ R6,$
    DJNZ R7,DL_1MS_1
    RET
;=====
;
DELAY:
;=====
;
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    RET
;=====
;
BANGSO:
;=====
;
    DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H
    RET
    END

```

### 9.3 Yêu cầu

- Soạn thảo, dịch, chạy, ghi lại kết quả
- Mô phỏng trên phần mềm proteus, cài đặt múi giờ khác bằng các nút bấm.