

LAB 1-1

GIAO TIẾP I/O VÀ CÁC LỆNH TÍNH TOÁN

MỤC TIÊU:

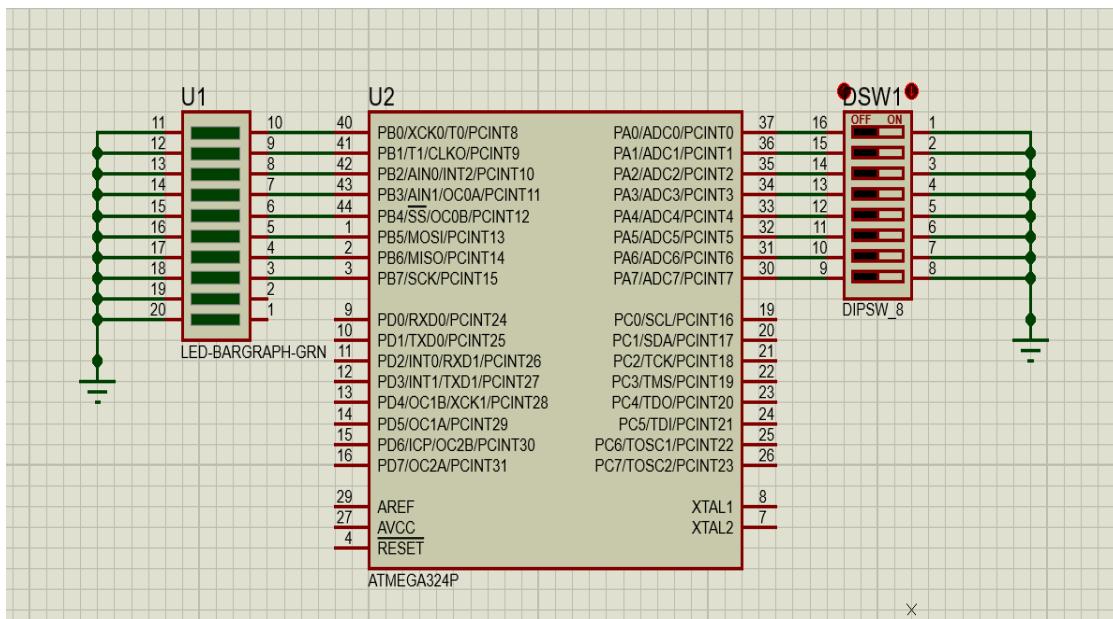
- Thực hiện các giao tiếp I/O Port, các lệnh tính toán

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 1, 2

BÀI 1

- Kết nối 1 port của AVR (VD PORT A) vào dip switch. Kết nối 1 port khác vào bar LED (Ví dụ PORT B)



- Viết chương trình đọc liên tục trạng thái của DIP Switch và gửi ra LED. Nếu Swich ở trạng thái OFF, LED tương ứng sẽ tắt.

BÀI 2

- Viết chương trình đọc giá trị của Port đang nối với Dip Switch, cộng thêm 5 và gửi ra Port đang nối với Bar LED.
- Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

BÀI 3

LAB 1-1

GIAO TIẾP I/O VÀ CÁC LỆNH TÍNH TOÁN

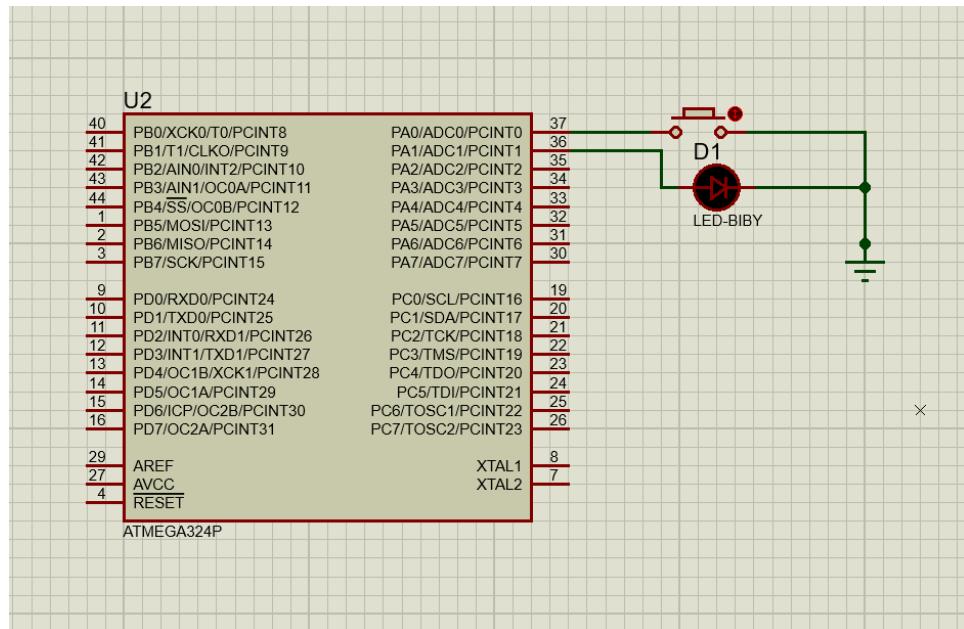
- a) Kết nối và thực hiện chương trình tính tích của 2 nibble cao và thấp của PORTA và gửi ra PORT B. Coi như 2 nibble này là 2 số không dấu
VD: PORTA = 0b0111_1111, thì PORTB = 3*15.
- b) Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

BÀI 4

- a) Kết nối và thực hiện chương trình tính tích của 2 nibble cao và thấp của PORTA và gửi ra PORT B. Coi như 2 nibble này là 2 số có dấu
VD: PORTA = 0b0111_1111, thì PORTB = 3* (-1).
- b) Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

BÀI 5

- a) Kết nối PA0 vào 1 Switch đơn và PA1 vào 1 LED đơn trên khối LED (lưu ý là cùng 1 Port)



- b) Viết chương trình bật LED nếu SW nhấn, tắt LED nếu SW nhả.

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

BÀI 1

- Trả lời các câu hỏi

- Lấy giá trị từ 2 nibble của PORTA như thế nào

Để lấy 4byte thấp:

IN R18, PINA

MOV R19, R18

ANDI R18, \$0F

Để lấy 4byte cao:

ANDI R19, \$F0

SWAP R19

- Enable điện trở pullup như thế nào?

PORTA là port nhập nên để xét điện trở kéo lên cho nó ta chỉ cần out giá trị \$FF ra thanh ghi PORTA.

- Khi Switch ở trạng thái ON/OFF, giá trị chân Port bao nhiêu?

Switch ON: chân PORTA có giá trị 0, chân PORTB có giá trị 1

Switch OFF: chân PORTA có giá trị 1, chân PORTB có giá trị 0

- Khi chân port ở trạng thái 1, BAR LED sáng hay tắt?

Port ở trạng thái 1 thì BAR LED sáng.

- Mã nguồn với chú thích

a) Lấy giá trị từ 2 nibble bằng cách sử dụng thanh ghi làm MASK với các bit tại vị trí của nibble cần lấy có giá trị 1 và thực hiện toán tử AND giữa thanh ghi MASK với PORTA.

VD: Lấy nibble cao của port A lưu vào r17

ldi r16, 0xFF ; Set DDR A to input

out DDRA, r16

ldi r17, 0xF0

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

ldi r16, PINA

and r17, portA

b) Enable điện trở pullup bằng cách cho xuất giá trị 1 ra các chân của PORTA.

c) Do sử dụng điện trở kéo lên, khi Switch ở trạng thái ON/OFF, giá trị chân Port bằng 1/0.

d) Khi chân port ở trạng thái 1, BARLED sáng.

e) Mã nguồn chương trình:

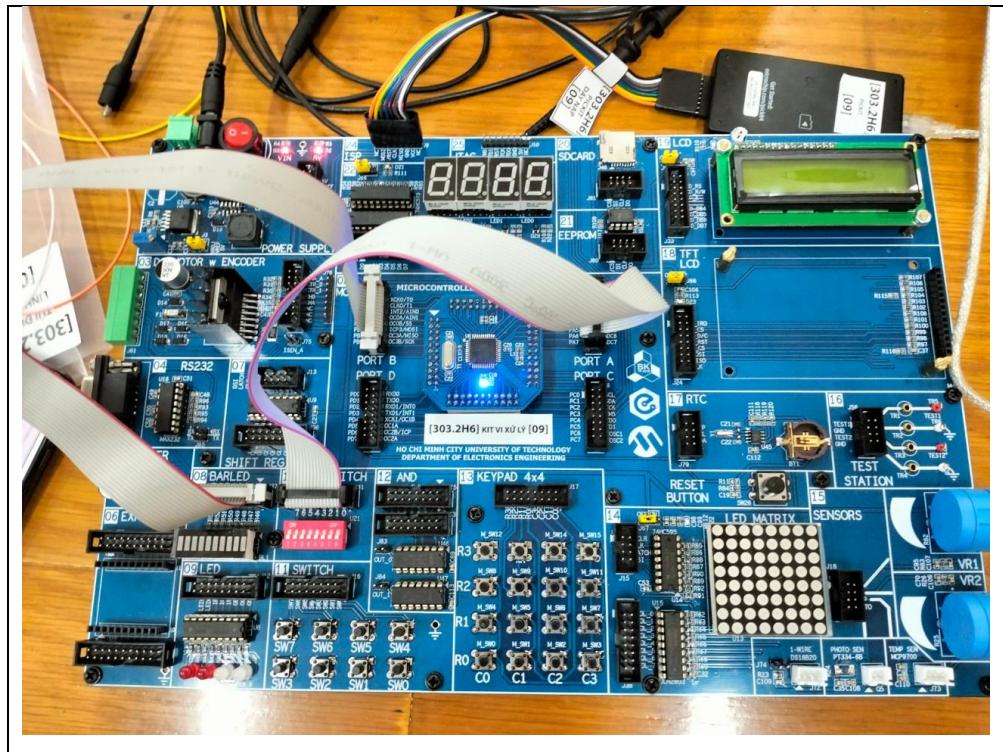
```
ldi    R16, 0xFF
out   DDRB, R16 ; port B xuất
ldi   R16, 0x00 ;
out  DDRA, R16 ; port A nhập
ldi   R16, 0xFF
out  PORTA, R16; port A có điện trở kéo lên
loop: in  r16, PINA ; Đọc bit từ portA ;
com  r16
out  PORTB, r16 ; Xuất ra port B
rjmp loop ; Quay lại vòng lặp
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 2

1. Trả lời các câu hỏi

a. Mã nguồn với chú thích

Giả sử port A nối với dip_switch, port B nối với BARLED.

```
LDI R16, 0X00
OUT DDRA, R16
LDI R16, 0xFF
OUT PORTA, R16
OUT DDRB, R16
LOOP:
IN R16, PINA
LDI R17, 0xFF
```

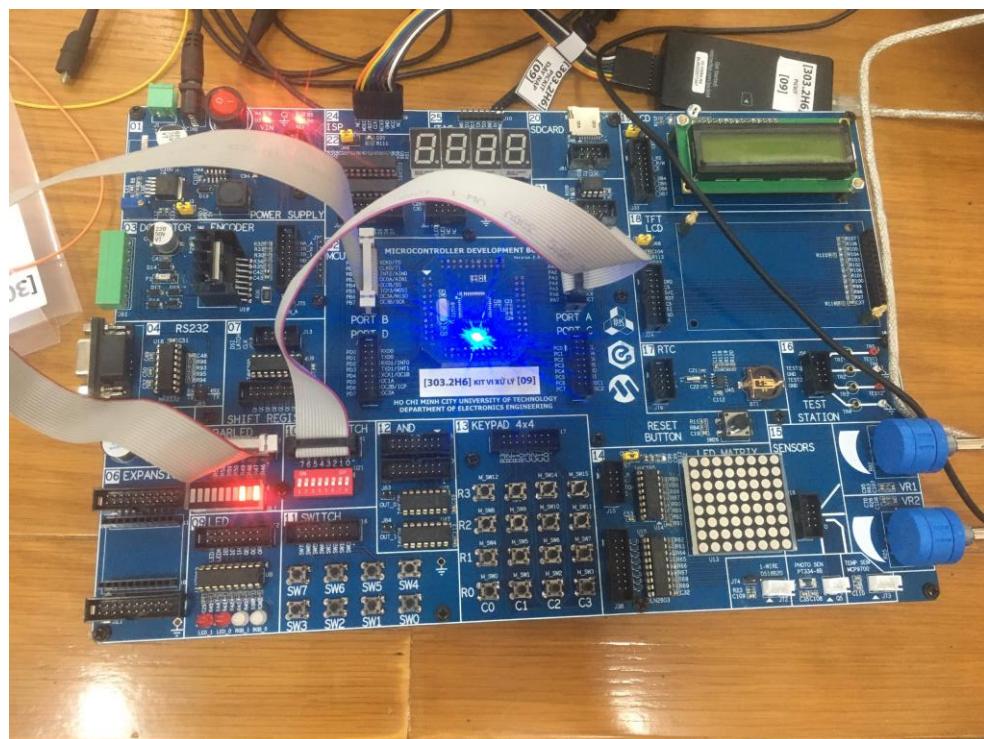
BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
EOR R16,R17  
LDI R17, 0X05  
ADD R16,R17  
OUT PORTB, R16  
RJMP LOOP
```



BÀI 3

- Trả lời các câu hỏi

Làm thế nào lấy giá trị từ 2 nibble của PORT A

Để lấy 4byte thấp:

IN R18, PINA

MOV R19, R18

ANDI R18, \$0F

Để lấy 4byte cao:

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

ANDI R19, \$F0

SWAP R19

a. Mã nguồn với chú thích

a) Lấy giá trị từ 2 nibble bằng cách sử dụng thanh ghi làm MASK với các bit tại vị trí của nibble cần lấy có giá trị 1 và thực hiện toán tử AND giữa thanh ghi MASK với PORTA.

b) Mã nguồn:

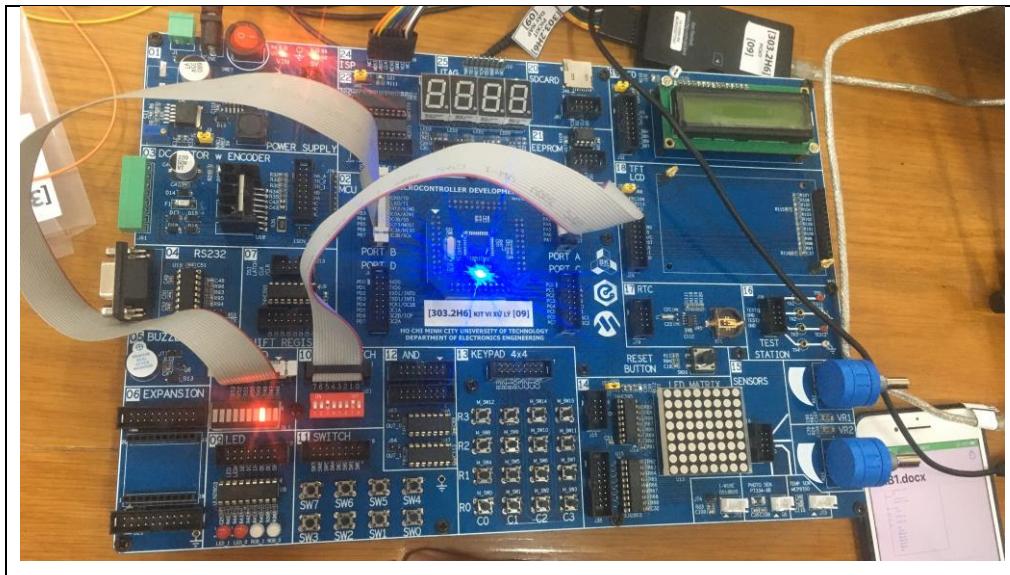
```
ldi    R16, 0x00
out   DDRA, R16 ; port A nhập
ldi    R16, 0xFF
out   DDRB, R16 ; port B xuất
out   PORTA, R16 ; port A có điện trở lên
loop: in     R17, PINA      ; đọc dữ liệu 8 bit từ port A
      COM      R17
      mov   R18, R17      ; sao chép R17 sang R18
      ldi   R16, 0xF0      ; khởi tạo thanh ghi MASK
      and   R17, R16      ; lấy nibble cao lưu ở R17
      swap  R17
      swap  R16          ; đảo nibble R16
      and   R18, R16      ; lấy nibble thấp lưu ở R18
      mul   R17, R18      ; nhân hai nibble, kết quả lưu là
số 8bit lưu ở R0
      mov   R16, R0          ; Lấy R16 lưu kết quả
      out   PORTB, R16 ; Xuất kết quả ra port B
      rjmp  loop
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 4

1. Trả lời các câu hỏi

a. Mã nguồn với chú thích

```
ldi    R16, 0x00
out   DDRA, R16      ; port A nhập
ldi    R16, 0xFF
out   DDRB, R16      ; port B xuất
out   PORTA, R16     ; port A có điện trở kéo lên
loop: in    R17, PINA      ; đọc dữ liệu 8 bit từ port A
      com   R17
      mov   R18, R17      ; sao chép R17 sang R18
      ldi   R16, 0xF0      ; khởi tạo thanh ghi MASK
      and   R17, R16      ; lấy nibble cao lưu ở R17
      swap  R16           ; đảo nibble R16
      and   R18, R16      ; lấy nibble thấp lưu ở R18
```

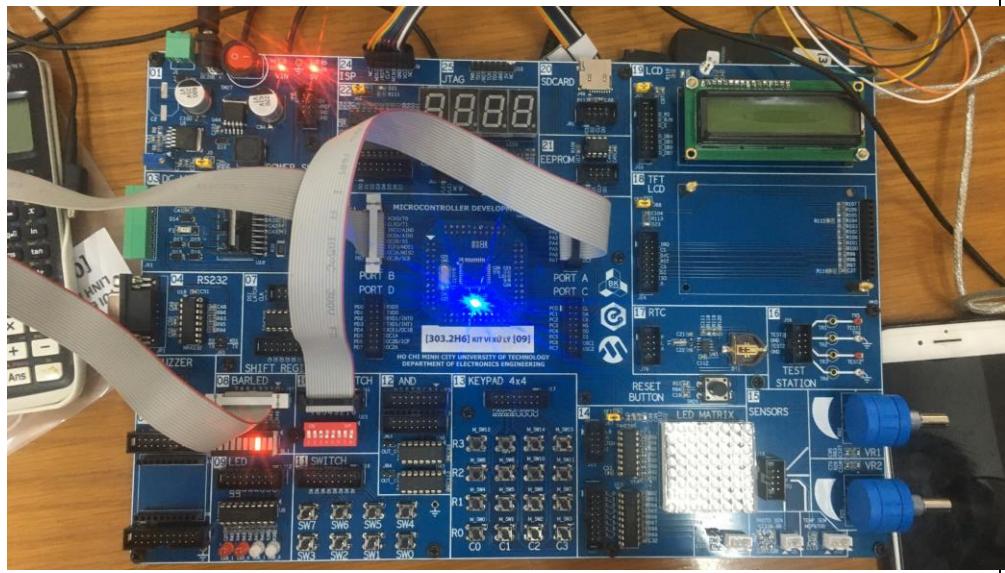
BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
muls    R17, R18          ; nhân hai nibble, kết quả lưu là số  
8bit lưu ở R0  
  
mov     R16, R0          ; Lấy R16 lưu kết quả  
  
swap    R16  
  
out    PORTB, R16        ; Xuất kết quả ra port B  
  
rjmp   loop
```



BÀI 5

- Trả lời các câu hỏi

- Khi Switch ở trạng thái nhấn/nhả, giá trị chân Port bằng bao nhiêu?

Nhấn: PORTA0 = 0

Nhả: PORTA0 = 1

- Để LED sáng, chân port xuất ra mức logic gì?

LED sáng chân PORTA1 xuất mức 1

- Mã nguồn với chủ thích

a) Khi Switch nhấn/nhả, giá trị chân Port bằng 0/1.

b) Để LED sáng, chân port xuất ra mức logic 1.

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

c) Mã nguồn:

LDI R16, 0X02

OUT DDRA, R16

LDI R16, 0X01

OUT PORTA, R16

LOOP:

SBIC PINA, 0

RJMP OFF

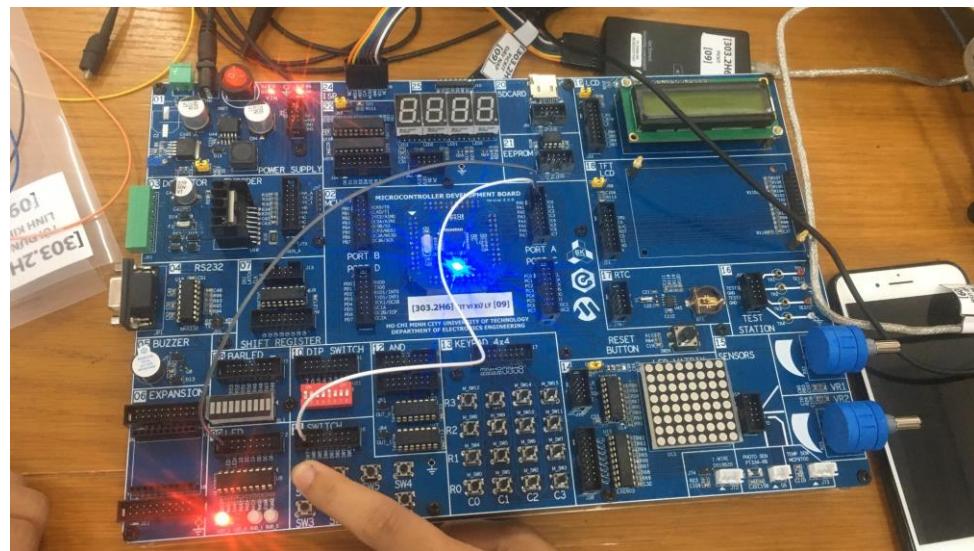
SBI PORTA, 1

RJMP LOOP

OFF:

CBI PORTA, 1

RJMP LOOP



LAB 1-2

DELAY DÙNG LỆNH

MỤC TIÊU:

- Thực hiện các lệnh tạo trễ dùng câu lệnh
- Thực hiện giao tiếp với thanh ghi dịch

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 1, 2

BÀI 1

- c) Cho chương trình như sau:

```
.include "m324PAdef.inc"
.org 00
    ldi r16,0x01
    out DDRA, r16
start:
    sbi PORTA,PINA0
    cbi PORTA, PINA0
    rjmp start
```

Kết nối PA0 vào một kênh đo trên khối TEST STATION và đo dạng xung trên oscilloscope

BÀI 2

- a) Viết chương trình con Delay1ms và dùng nó để viết chương trình tạo xung vuông tần số 1Khz trên PA0.

- Chương trình con Delay 1ms:

DELAY_1MS:

LDI R16, 8

LOOP1:

LDI R17, 250

LOOP2:

DEC R17

NOP

LAB 1-2

DELAY DÙNG LỆNH

```
BRNE LOOP2
```

```
DEC R16
```

```
BRNE LOOP1
```

```
RET
```

- Chương trình tạo xung vuông tần số 1kHz:

```
.cseg
.org 0X00
SBI DDRA, 0
START_LOOP:
CBI PORTA, 0
CALL DELAY_1S
SBI PORTA, 0
CALL DELAY_1S
RJMP START_LOOP
```

```
DELAY_1MS:
```

```
LDT R16, 8
```

```
LOOP1:
```

```
LDI R17, 250
```

```
LOOP2:
```

```
DEC R17
```

```
NOP
```

```
BRNE LOOP2
```

```
DEC R16
```

```
BRNE LOOP1
```

```
RET
```

- b) Dùng chương trình con này viết các chương trình con Delay10ms, Delay100ms, Delay1s.
- Chương trình con Delay10ms:

```
DELAY_10MS:
```

<https://doe.dee.hcmut.edu.vn/>

LAB 1-2

DELAY DÙNG LỆNH

```
LDI R16, 80
LOOP1:
LDI R17, 250
LOOP2:
DEC R17
NOP
BRNE LOOP2
DEC R16
BRNE LOOP1
RET
```

- Chương trình con Delay100ms:

```
DELAY_100MS:
LDI R16, 10
LOOP3:
LDI R17, 80
LOOP1:
LDI R18, 250
LOOP2:
DEC R18
NOP
BRNE LOOP2
DEC R17
BRNE LOOP1
DEC R16
BRNE LOOP3
RET
```

- Chương trình con Delay1s:

```
DELAY_1S:
```

LAB 1-2

DELAY DÙNG LỆNH

```
LDI R16, 10
```

```
LOOP4:
```

```
LDI R17, 10
```

```
LOOP3:
```

```
LDI R18, 80
```

```
LOOP1:
```

```
LDI R19, 250
```

```
LOOP2:
```

```
DEC R19
```

```
NOP
```

```
BRNE LOOP2
```

```
DEC R18
```

```
BRNE LOOP1
```

```
DEC R17
```

```
BRNE LOOP3
```

```
DEC R16
```

```
BRNE LOOP4
```

```
RET
```

- c) Dùng chương trình con Delay1s viết chương trình chớp/tắt 1 LED gắn vào PA0.
- Chương trình chớp/tắt 1 LED gắn vào PA0:

```
.cseg
.org 0X00
SBI DDRA, 0
START_LOOP:
CBI PORTA, 0
CALL DELAY_1S
SBI PORTA, 0
CALL DELAY_1S
RJMP START_LOOP
```

LAB 1-2

DELAY DÙNG LỆNH

DELAY_1S:

LDI R16, 10

LOOP4:

LDI R17, 10

LOOP3:

LDI R18, 80

LOOP1:

LDI R19, 250

LOOP2:

DEC R19

NOP

BRNE LOOP2

DEC R18

BRNE LOOP1

DEC R17

BRNE LOOP3

DEC R16

BRNE LOOP4

RET

BÀI 3

- d) Kết nối các tín hiệu cần thiết từ 1 port của AVR đến các tín hiệu điều khiển thanh ghi dịch trên header J13. Kết nối ngõ ra của thanh ghi dịch đến Bar LED.
- e) Dùng các chương trình trong ví dụ mẫu trong tài liệu hướng dẫn thí nghiệm, viết chương trình tạo hiệu ứng LED sáng dần từ trái qua phải, sau đó tắt dần từ trái qua phải sau mỗi khoảng thời gian 500ms.

BÁO CÁO

Nhóm:6

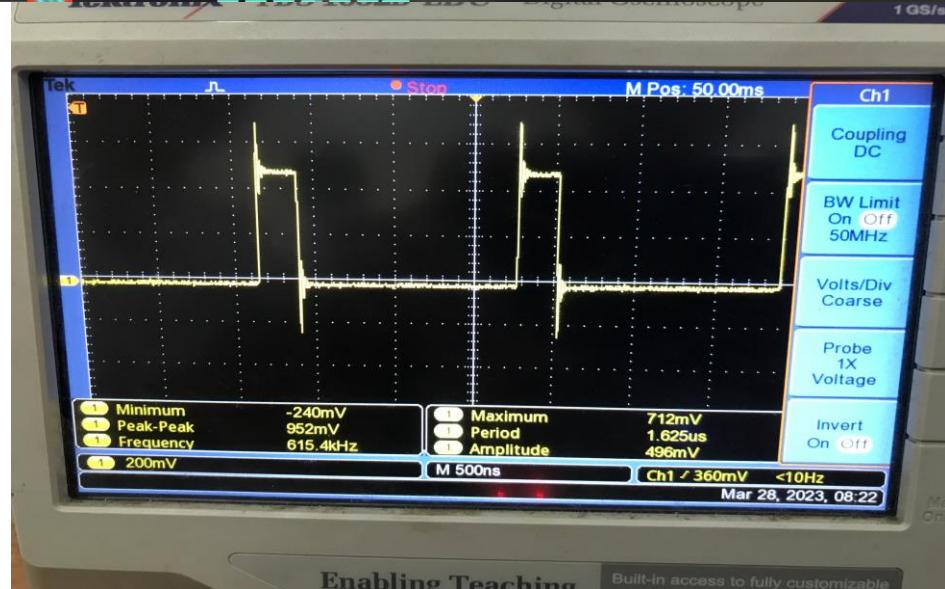
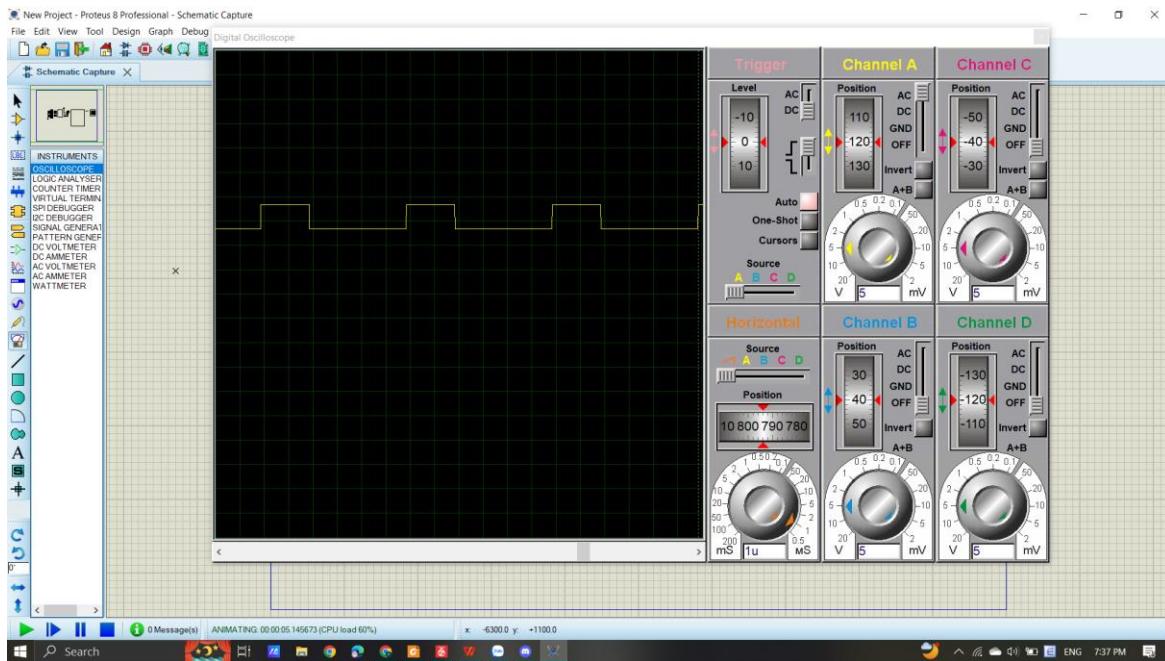
Nhóm môn học: L13

Môn thí nghiệm: Vi xử lý

BÀI 1

2. Trả lời các câu hỏi

Chụp ảnh dạng xung trên PA0



BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

- f. Tần số, thời gian tín hiệu bằng 1, thời gian tín hiệu bằng 0 là bao nhiêu?
- Thời gian tín hiệu bằng 1 là $2 \cdot 10^{-6}$ s
 - Thời gian tín hiệu bằng 0 là $4 \cdot 10^{-6}$ s
- g. Giải thích kết quả đo được.
- Kết quả đo có thời gian tín hiệu bằng 1 ít hơn thời gian tín hiệu bằng 0 do sao lệnh CBI (1MC) là lệnh RJMP (2MC), trong lúc thực hiện lệnh RJMP thì PORTA0 có điện áp vẫn bằng 0.

BÀI 2

2. Trả lời các câu hỏi
 - b. Cách tính số chu kỳ máy để thực hiện chương trình con Delay1ms. Trình bày hình ảnh mô phỏng

DELAY_1MS:

LDI R16, 8 (1 MC)

LOOP1:

LDI R17, 250 (1 MC)

LOOP2:

DEC R17 (1 MC)

NOP (1 MC)

BRNE LOOP2 (1/2 MC)

DEC R16 (1 MC)

BRNE LOOP1 (1/2 MC)

RET (4 MC)

$$t_{\text{delay}} = 1 + (4.250 + 1 + 2).8 - 1 + 4 (\text{MC})$$

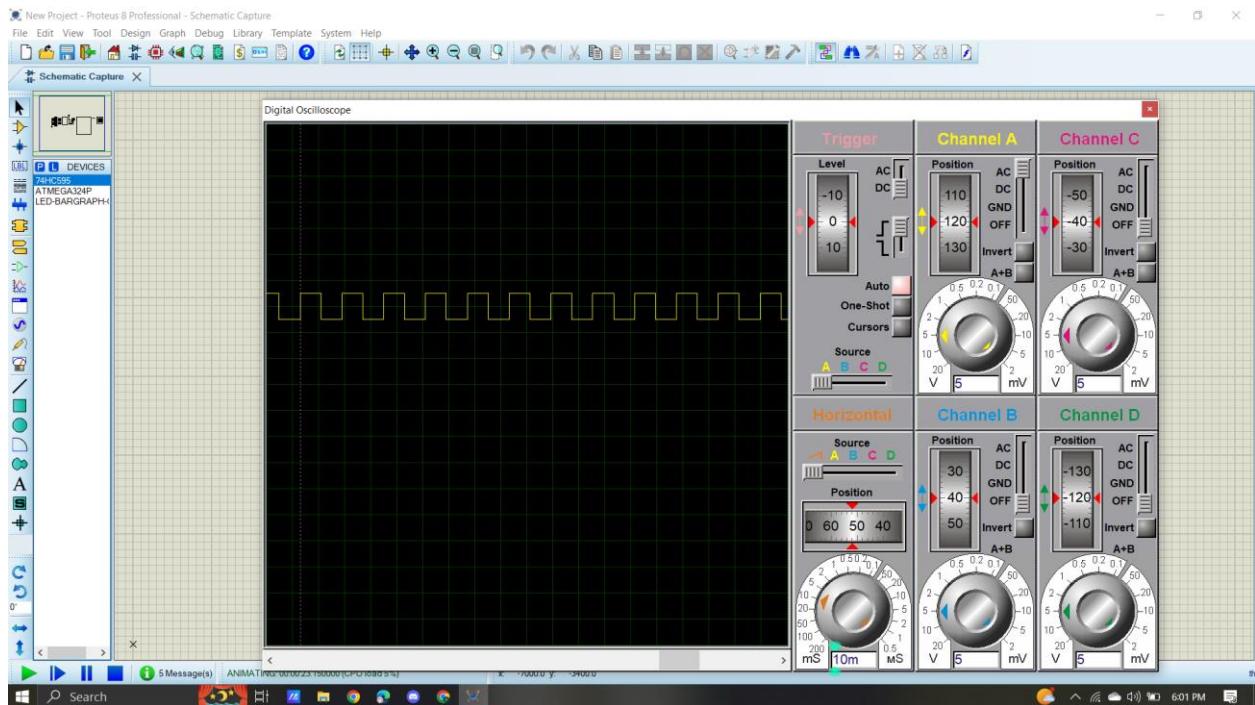
BÁO CÁO

Nhóm:6

Nhóm môn học: L13

Môn thí nghiệm: Vi xử lý

c. Hình ảnh xung 1Khz trên PA0.



d. Sai số là bao nhiêu?

3. Mã nguồn câu 2.c với chú thích

```
.cseg  
.org 0X00  
SBI DDRA, 0  
  
START_LOOP:  
  
CBI PORTA, 0  
  
CALL DELAY_1S  
  
SBI PORTA, 0  
  
CALL DELAY_1S  
  
RJMP START_LOOP:
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

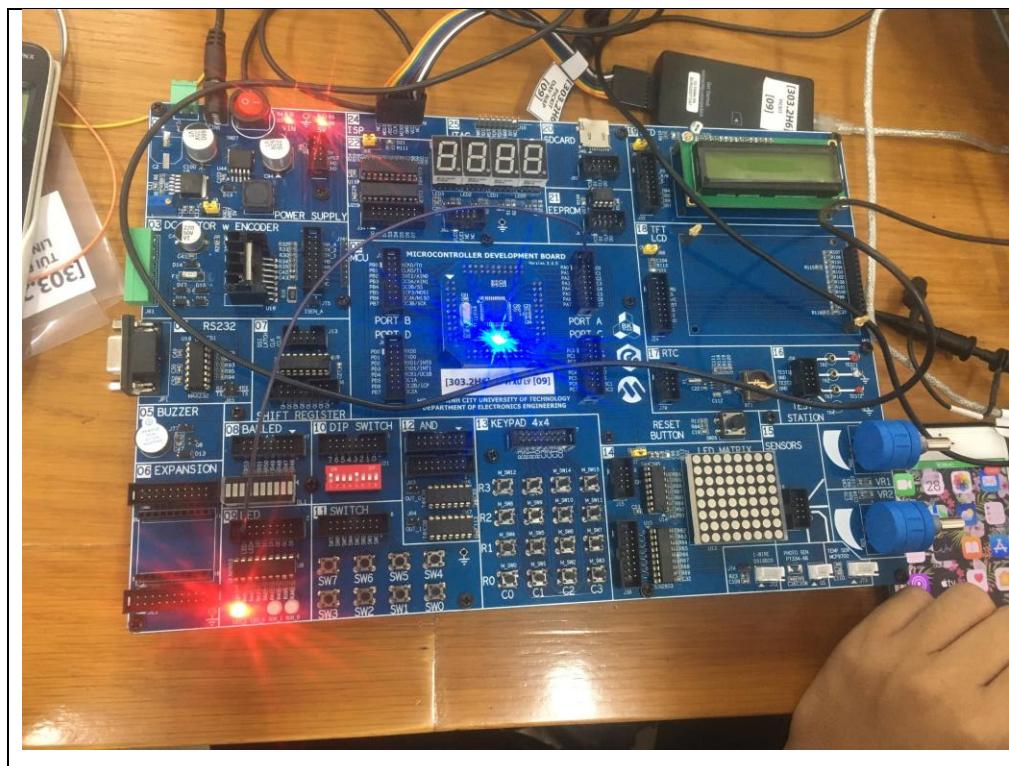
```
DELAY_1S:  
LDI R16, 10  
LOOP4:  
LDI R17, 10  
LOOP3:  
LDI R18, 80  
LOOP1:  
LDI R19, 250  
LOOP2:  
DEC R19  
NOP  
BRNE LOOP2  
DEC R18  
BRNE LOOP1  
DEC R17  
BRNE LOOP3  
DEC R16  
BRNE LOOP4  
RET
```

BÁO CÁO

Nhóm môn học: L13

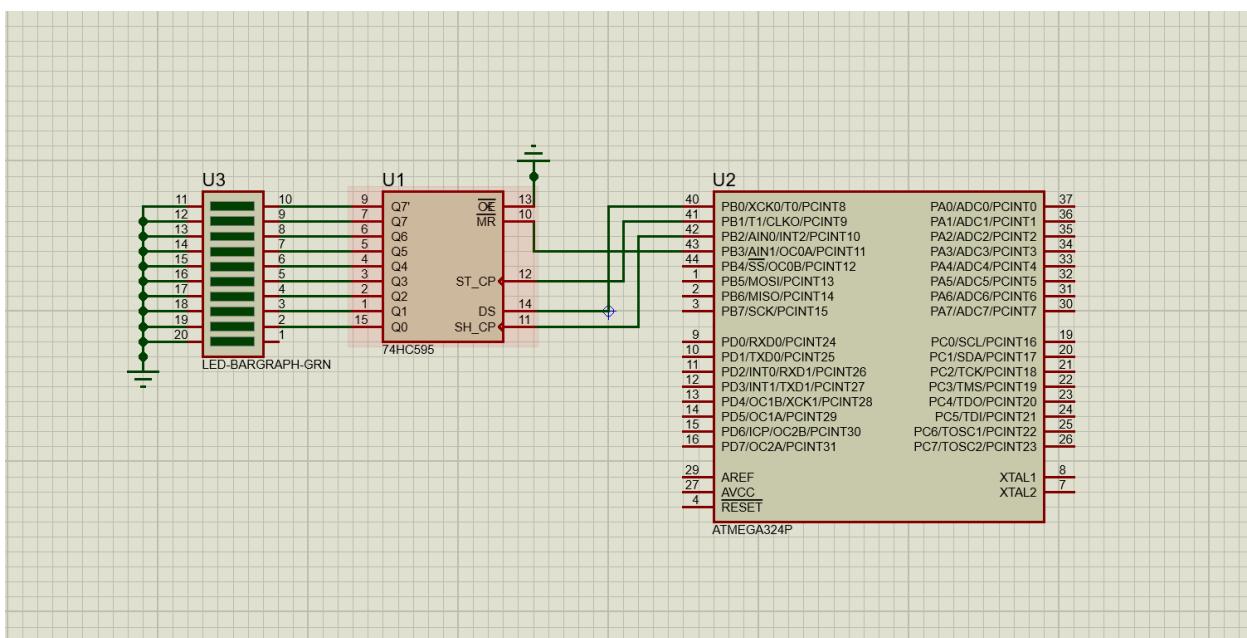
Nhóm:6

Môn thí nghiệm: Vi xử lý



BÀI 3

2. Trả lời các câu hỏi
 - b. Mô tả kết nối trên kit thí nghiệm



BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

- c. Theo như datasheet của 74HC595, tần số clock cao nhất mà nó có thể hoạt động được là bao nhiêu
- Theo datasheet, tần số clock cao nhất mà 74HC595 có thể hoạt động được là 31MHz ở nhiệt độ 25°C khi Vcc = 6V.
- d. Nếu muốn mở rộng hiển thị ra 16 LED thì ta phải làm như thế nào?
- Nếu muốn mở rộng hiển thị ra 16 LED thì ta nối thêm một IC dịch vào, pin DS của IC dịch thứ 2 nối với pin Q7' của IC dịch thứ
- e. Mã nguồn với chú thích

```
.include "m324padef.inc" ; Include Atmega324pa definitions
.def shiftData = r20 ; Define the shift data register
.equ clearSignalPort = PORTB ; Set clear signal port to PORTB
.equ clearSignalPin = 3 ; Set clear signal pin to pin 0 of PORTB
.equ shiftClockPort = PORTB ; Set shift clock port to PORTB
.equ shiftClockPin = 2 ; Set shift clock pin to pin 1 of PORTB
.equ latchPort = PORTB ; Set latch port to PORTB
.equ latchPin = 1 ; Set latch pin to pin 0 of PORTB
.equ shiftDataPort = PORTB ; Set shift data port to PORTB
.equ shiftDataPin = 0 ; Set shift data pin to pin 3 of PORTB
main:
call initport
ldi shiftData,0x55
call cleardata
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
recall:  
  
    call shiftoutdata  
  
    rjmp recall  
  
; Initialize ports as outputs  
  
initport:  
  
    ldi r24,  
    (1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin)  
  
    out DDRB, r24 ; Set DDRB to output  
  
    ret  
  
    ldi shiftData,0x55  
  
cleardata:  
  
    cbi clearSignalPort, clearSignalPin ; Set clear signal pin to low  
  
; Wait for a short time  
  
    sbi clearSignalPort, clearSignalPin ; Set clear signal pin to high  
  
    ret  
  
; Shift out data  
  
shiftoutdata:  
  
    cbi shiftClockPort, shiftClockPin  
  
    ldi r18, 8  
  
    ldi r17, 8  
  
    sbi shiftDataPort, shiftDataPin ; Set shift data pin to high  
  
shift_on:  
  
    sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high  
    cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low  
  
    dec r18  
  
    breq begin_shift_off  
  
    call DELAY_500MS
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
sbi latchPort, latchPin ; Set latch pin to high
cbi latchPort, latchPin ; Set latch pin to low
rjmp shift_on

begin_shift_off:
    cbi shiftDataPort, shiftDataPin ; Set shift data pin to low
    shift_off:
        sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high
        cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low
        dec r17
        breq Retired
        call DELAY_500MS
        sbi latchPort, latchPin ; Set latch pin to high
        cbi latchPort, latchPin ; Set latch pin to low
        rjmp shift_off

Retired:
    ret

DELAY_500MS:
    ldi R16, 50
    LOOP3:
    ldi R17, 80
    LOOP1:
    ldi R18, 250
    LOOP2:
    dec R18
    nop
    brne LOOP2
```

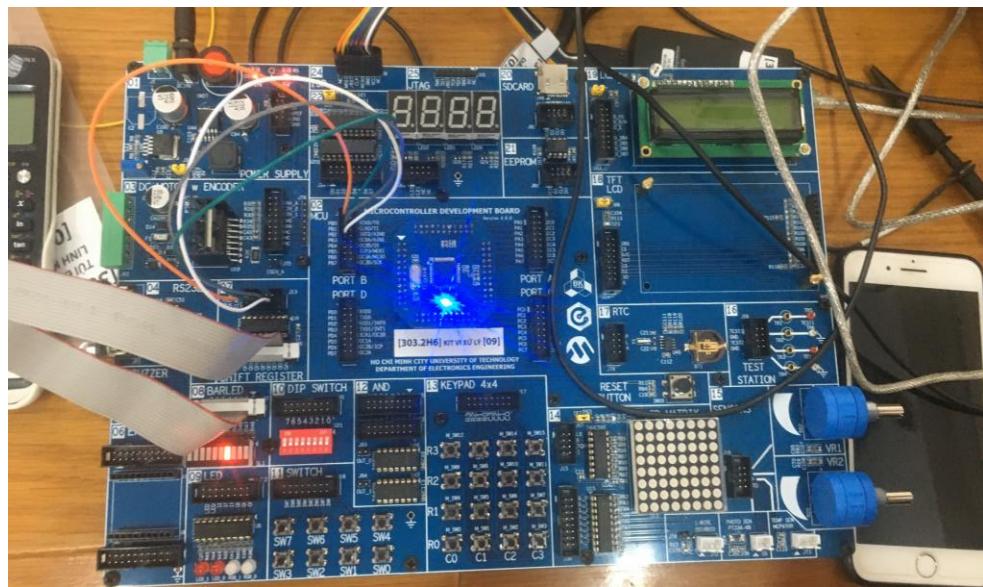
BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
dec R17  
brne LOOP1  
dec R16  
brne LOOP3  
ret
```



LAB 1-3

GIAO TIẾP NÚT NHẤN, BÀN PHÍM MA TRẬN

MỤC TIÊU:

- Hiểu cách chống rung phím
- Hiểu cách giao tiếp LCD
- Hiểu cách giao tiếp phím đơn
- Hiểu cách giao tiếp bàn phím ma trận

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 1, 2 , 3 ,6

BÀI 1

- a) Kết nối một PORT của AVR vào J33 (Header điều khiển LCD) trên kit thí nghiệm.
- b) Dùng các chương trình mẫu trong tài liệu hướng dẫn thí nghiệm, viết chương trình khởi động LCD và xuất lên LCD như sau. (XX là số nhóm)

TN VXL-AVR

Nhom: XX

BÀI 2

- c) Kết nối 1 switch đến 1 chân port của AVR, kết nối module BAR LED đến 1 port của AVR, kết nối LCD đến 1 port của AVR
- d) Viết chương trình đếm số lần nhấn nút và xuất kết quả ra barled, đồng thời xuất ra LCD (không chống rung)
- e) Thêm tính năng chống rung phím vào chương trình
- f) Thực hiện chương trình, nhấn/nhả nút và quan sát kết quả

BÀI 3

- a) Kết nối tín hiệu từ một port của AVR đến module bàn phím ma trận , kết nối module BAR LED và LCD đến 2 port khác của AVR.

LAB 1-3

GIAO TIẾP NÚT NHẤN, BÀN PHÍM MA TRẬN

- b)** Viết chương trình con SCANKEY để quét bàn phím ma trận và trả về giá trị từ 0x0 đến 0xF ứng với mã của phím được nhấn. Nếu không có phím nào được nhấn trả về giá trị 0xFF. Giá trị trả về chứa trong R24
- c)** Dùng chương trình con này, viết chương trình thực hiện việc quét phím và xuất giá trị đọc được lên bar led và LCD.
- d)** Thực hiện chương trình, quan sát kết quả

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

BÀI 1

1. Trả lời các câu hỏi
 - a. LCD phân biệt command và data bằng cách nào?
 - b. Ngoài cách đọc bit BUSY, còn cách nào để đảm bảo là LCD rảnh khi gửi dữ liệu/command?
 - c. Mô tả kết nối trên kit thí nghiệm.
 - d. Mã nguồn chương trình với chú thích
 - a. LCD phân biệt giữa các lệnh và dữ liệu là thông qua việc sử dụng chân điều khiển được gọi là chân RS (Đăng ký Chọn). Khi chân RS được đặt ở mức logic thấp (0), dữ liệu được gửi tới LCD được hiểu là lệnh. Khi chân RS được đặt ở mức logic cao (1), dữ liệu được gửi được hiểu là dữ liệu sẽ được hiển thị trên màn hình.
 - b. Ngoài cách đọc bit BUSY, ta có thể thực hiện như sau: Sau mỗi lệnh ghi/đọc vào CPU, ta làm chương trình chờ khoảng 2 ms trước khi thực hiện lệnh kế tiếp. Tuy nhiên, cách làm này sẽ làm cho chương trình không đạt hiệu năng cao nhất nếu ta cho thời gian trễ quá dài. Nếu ta giảm thời gian trễ này thì có thể làm LCD hoạt động sai.
 - c. Mã nguồn:

```
.INCLUDE "M324PADEF.INC"
.EQU RS = 0
.EQU RW = 1
.EQU EN = 2
.EQU CR = $0D ; Enter
.EQU NULL = $00 ; End string

.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
LDI R16, $FF
OUT DDRA, R16 ;PA7,6,5,4,2,1,0 is output
CBI PORTA, RS ;Recieve command
CBI PORTA, RW ;Write data
CBI PORTA, EN ;Unenable LCD
CALL RESET_LCD
CALL INIT_LCD4

START:
    CBI PORTA, RS
    LDI R17, $01 ;Clear display before
    RCALL OUT_LCD4_2
    LDI R16, 20           ;Delay 20ms after clearing
    RCALL DELAY_US
    CBI PORTA, RS
    LDI R17, $80 ;Pointer start at line1, pos1
    RCALL OUT_LCD4_2
    LDI ZH, HIGH(TAB<<1)
    LDI ZL, LOW(TAB<<1)

LINE1:
    LPM R17, Z+
    CPI R17, CR           ;Check enter code
    BREQ DOWN
    SBI PORTA, RS ;Display on LCD
    RCALL OUT_LCD4_2
    RJMP LINE1

DOWN:
    LDI R16, 1           ;Xuong dong phai doi 100us
    RCALL DELAY_US
    CBI PORTA, RS
    LDI R17, $C0 ;Set pointer to line2 pos1
    RCALL OUT_LCD4_2

LINE2:
    LPM R17, Z+
    CPI R17, NULL
    BREQ DONE
    SBI PORTA, RS
    RCALL OUT_LCD4_2
    RJMP LINE2

DONE:
    RJMP DONE

OUT_LCD4:
    OUT PORTA, R17
    SBI PORTA, EN
    CBI PORTA, EN
    RET

OUT_LCD4_2:
    LDI R16, 1           ;Delay 1us
    RCALL DELAY_US
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
IN R16, PORTA
ANDI R16, (1<<RS)
PUSH R16
PUSH R17
ANDI R17, $F0
OR R17, R16
RCALL OUT_LCD4
LDI R16, 1           ;Delay 1us between first and
second access
RCALL DELAY_US
POP R17
POP R16
SWAP R17
ANDI R17, $F0
OR R17, R16
RCALL OUT_LCD4
RET

INIT_LCD4:
LDI R18, $28 ;Function set
LDI R19, $01 ;Clear display
LDI R20, $0C ;Display on, pointer off
LDI R21, $06
CBI PORTA, RS
MOV R17, R18
RCALL OUT_LCD4_2
MOV R17, R19 ;Clear display
RCALL OUT_LCD4_2
LDI R16, 20           ;Delay 20ms after clearing
display
RCALL DELAY_US
MOV R17, R20
RCALL OUT_LCD4_2
MOV R17, R21
RCALL OUT_LCD4_2
RET

RESET_LCD:
LDI R16, 250           ;Delay 25ms
RCALL DELAY_US
LDI R16, 250           ;Delay 25ms
RCALL DELAY_US
CBI PORTA, RS
LDI R17, $30           ;Lan 1
RCALL OUT_LCD4

LDI R16, 250           ;Delay 25ms
RCALL DELAY_US
LDI R16, 170            ;Delay 17ms
RCALL DELAY_US
```

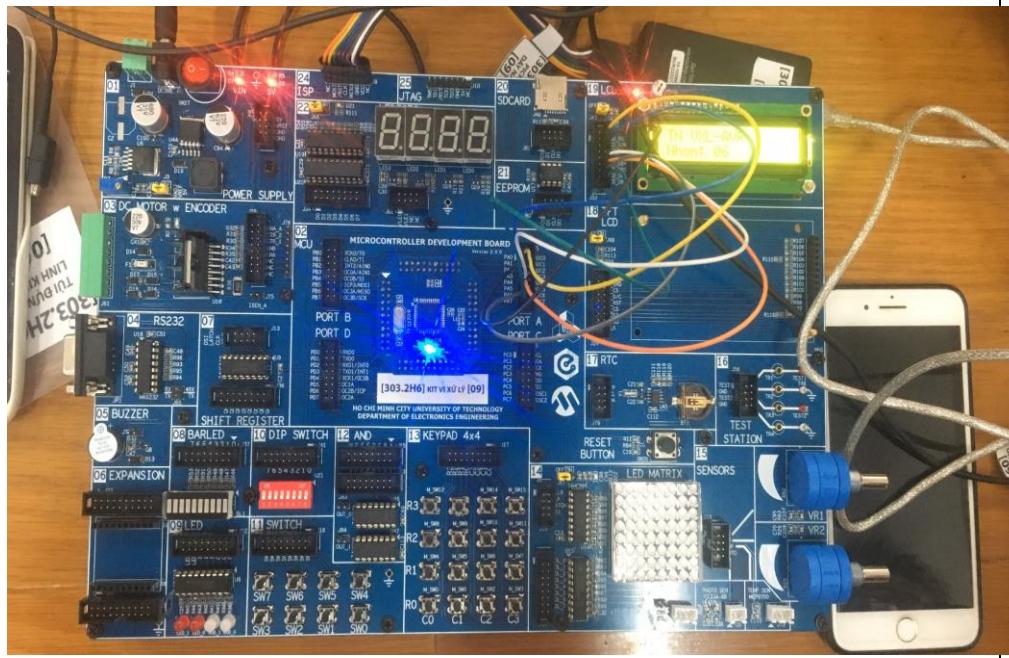
BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
CBI PORTA, RS  
LDI R17, $30 ;Lan 2  
RCALL OUT_LCD4  
  
LDI R16, 2 ;Delay 200us  
RCALL DELAY_US  
CBI PORTA, RS  
LDI R17, $32 ;Lan 3  
RCALL OUT_LCD4_2  
RET  
  
DELAY_US:  
    MOV R15, R16  
    LDI R16, 200  
LP1:  
    MOV R14, R16  
LP2:  
    NOP  
    DEC R14  
    BRNE LP2  
    DEC R15  
    BRNE LP1  
    RET  
  
TAB:  
.DB "TN VXL-AVR", $0D, "Nhóm: ", "06", $00
```



BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

BÀI 2

1. Trả lời các câu hỏi

- Hiện tượng gì xảy ra khi không chống rung phím

Với phím nhấn cơ khí, tiếp điểm cơ khí sau khi được nhấn hay nhả sẽ bị rung động. Do đó, khi nhấn hay nhả phím sẽ tạo ra một chuỗi các xung thay vì một xung đơn. Khi đó, nếu ta dùng tín hiệu này để xác định số lần phím nhấn thì kết quả sẽ bị sai lệch.

-
-
- Mô tả cách kết nối trên kit thí nghiệm
- Mã nguồn chương trình không chống rung phím và chú thích

```
.include "m328pdef.inc" ; Define Atmega file
.def temp = r18 ; Define a temporary register
.def count = r16 ; Define count variable
.org 0x0000 ;

;Define stack pointer
ldi temp, LOW(RAMEND) ; Initialize the stack pointer
out SPL, temp
ldi temp, HIGH(RAMEND)
out SPH, temp

;Define LCD
ldi temp, 0x38 ; Function set: 8-bit interface, 2 lines, 5x7 dots
call lcd_send_command
ldi temp, 0x0C ; Display on, cursor off, blink off
call lcd_send_command
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
ldi temp, 0x01 ; Clear display
call lcd_send_command

; Define input-output
cbi DDRA, 0 ; Set Swith input is PA0
sbi PORTA, 0; Pull-up resistor for PA0
ldi temp, 0xFF
out DDRB, temp : Port B Barleds (Output)
out DDRC, temp : Port C LCD (Output)

loop:
sbic PORTA, 0      ; Check if PA0 = 0 (Switch pressed)
rjmp loop          ; if not, come back to loop
inc count          ; If yes, increment count
out PORTB, count ; Output count to bar LEDs
call lcd_send_data ; sent count to LCD
rjmp loop

; module lcd_send_command (command code in r18)
lcd_send_command:
push r17
call LCD_wait_busy ; check if LCD is busy
mov r17,r18 ;save the command
; Set RS low to select command register
; Set RW low to write to LCD
andi r17,0xF0 ; Send command to LCD
out LCDPORT, r17
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
nop  
nop  
; Pulse enable pin  
sbi LCDPORT, LCD_EN  
nop  
nop  
cbi LCDPORT, LCD_EN  
swap r18 andi r18,0xF0 ; Send command to LCD  
out LCDPORT, r18  
; Pulse enable pin  
sbi LCDPORT, LCD_EN  
nop  
nop  
cbi LCDPORT, LCD_EN  
pop r17  
ret  
  
LCD_wait_busy:  
LCD_wait_busy:  
push r18  
ldi r18, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  
out LCDPORTDIR, r18  
ldi r18,0b11110010  
; set RS=0, RW=1 for read the busy flag  
out LCDPORT, r18  
nop
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
LCD_wait_busy_loop:  
  
    sbi LCDPORT, LCD_EN  
  
    nop  
  
    nop  
  
    in r18, LCDPORTPIN  
  
    cbi LCDPORT, LCD_EN  
  
    nop  
  
    sbi LCDPORT, LCD_EN  
  
    nop  
  
    nop cbi LCDPORT, LCD_EN  
  
    nop  
  
    andi r18,0x80  
  
    cpi r16=8,0x80  
  
    breq LCD_wait_busy_loop  
  
    ldi r18, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output out  
LCDPORTDIR, r18  
  
    ldi r18,0b00000000  
  
; set RS=0, RW=1 for read the busy flag  
  
    out LCDPORT, r16  
  
    pop r16  
  
    ret  
  
  
;module lcd_send_data (data is r16-count)  
  
lcd_send_data:  
  
    push r17  
  
    call LCD_wait_busy ;check if LCD is busy
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
mov r17,r16 ;save the command

; Set RS high to select data register
; Set RW low to write to LCD

andi r17,0xF0
ori r17,0x01

; Send data to LCD
out LCDPORT, r17
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
cbi LCDPORT, LCD_EN
; Delay for command execution ;send the lower nibble
nop
swap r16
andi r16,0xF0
; Set RS high to select data register
; Set RW low to write to LCD
andi r16,0xF0
ori r16,0x01
; Send command to LCD
out LCDPORT, r16
nop
; Pulse enable pin
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
sbi LCDPORT, LCD_EN  
nop  
cbi LCDPORT, LCD_EN  
pop r17  
ret
```

- e. Mã nguồn chương trình có chống rung và chú thích

```
include "m328pdef.inc" ; Define Atmega file  
.def temp = r18 ; Define a temporary register  
.def count = r16 ; Define count variable  
.org 0x0000 ;  
  
;Define stack pointer  
ldi temp, LOW(RAMEND) ; Initialize the stack pointer
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
out SPL, temp
ldi temp, HIGH(RAMEND)
out SPH, temp

;Define LCD
ldi temp, 0x38 ; Function set: 8-bit interface, 2 lines, 5x7 dots
call lcd_send_command
ldi temp, 0x0C ; Display on, cursor off, blink off
call lcd_send_command
ldi temp, 0x01 ; Clear display
call lcd_send_command

; Define input-output
cbi DDRA, 0 ; Set Swith input is PA0
sbi PORTA, 0; Pull-up resistor for PA0
ldi temp, 0xFF
out DDRB, temp : Port B Barleds (Output)
out DDRC, temp : Port C LCD (Output)

loop:
sbic PORTA, 0      ; Check if PA0 = 0 (Switch pressed)
rjmp loop          ; if not, come back to loop
rcall DELAY_10ms
sbic PORTA, 0      ; Check if PA0 = 0 again
rjmp loop          ; if PA0=1, come back loop
inc count          ; If yes, increment count
out PORTB, count ; Output count to bar LEDs
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
call lcd_send_data ; sent count to LCD
rjmp loop
; module DELAY_10ms
DELAY_10ms:
LDI    R21, 80
LOOP1: LDI    R22, 250
LOOP2: DEC    R22
        NOP
        BRNE   LOOP2
        DEC    R21
        BRNE   LOOP1
        RET
; module lcd_send_command (command code in r18)
lcd_send_command:
push r17
call LCD_wait_busy ; check if LCD is busy
mov r17,r18 ;save the command
; Set RS low to select command register
; Set RW low to write to LCD
andi r17,0xF0 ; Send command to LCD
out LCDPORT, r17
nop
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
nop  
cbi LCDPORT, LCD_EN  
swap r18 andi r18,0xF0 ; Send command to LCD  
out LCDPORT, r18  
; Pulse enable pin  
sbi LCDPORT, LCD_EN  
nop  
nop  
cbi LCDPORT, LCD_EN  
pop r17  
ret  
  
LCD_wait_busy:  
LCD_wait_busy:  
push r18  
ldi r18, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  
out LCDPORTDIR, r18  
ldi r18,0b11110010  
; set RS=0, RW=1 for read the busy flag  
out LCDPORT, r18  
nop  
  
LCD_wait_busy_loop:  
sbi LCDPORT, LCD_EN  
nop  
nop
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
in r18, LCDPORTPIN
cbi LCDPORT, LCD_EN
nop
sbi LCDPORT, LCD_EN
nop
nop cbi LCDPORT, LCD_EN
nop
andi r18,0x80
cpi r16=8,0x80
breq LCD_wait_busy_loop
ldi r18, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output out
LCDPORTDIR, r18
ldi r18,0b00000000
; set RS=0, RW=1 for read the busy flag
out LCDPORT, r16
pop r16
ret

;module lcd_send_data (data is r16-count)
lcd_send_data:
push r17 call LCD_wait_busy ;check if LCD is busy
mov r17,r16 ;save the command

; Set RS high to select data register
; Set RW low to write to LCD
andi r17,0xF0
ori r17,0x01
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
; Send data to LCD
out LCDPORT, r17
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
cbi LCDPORT, LCD_EN
; Delay for command execution ;send the lower nibble
nop
swap r16
andi r16,0xF0
; Set RS high to select data register
; Set RW low to write to LCD
andi r16,0xF0
ori r16,0x01
; Send command to LCD
out LCDPORT, r16
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
cbi LCDPORT, LCD_EN
pop r17
ret
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

BÀI 3

4. Trả lời các câu hỏi
 - e. Cách kết nối các module trên bài thí nghiệm
 - f. Có hiện tượng rung phím đối với bàn phím ma trận hay không? Nếu có thì xử lý bằng cách nào?
 - g. Trình bày mã nguồn chương trình và chú thích.
 - f. Có hiện tượng rung phím đối với bàn phím ma trận. Một trong những cách xử lý là dùng các chương trình tạo trễ một khoảng thời gian sau khi nhận tín hiệu nhấn/nhả.
 - g. Mã nguồn của chương trình (có chống rung nhấn/nhả)

```
.include "m324padef.inc" ; Include Atmega324pa definitions
.org 0x0000 ; interrupt vector table
rjmp reset_handler ; reset

.equ LCDPORT = PORTA ; Set signal port reg to PORTA
.equ LCDPORTDIR = DDRA ; Set signal port dir reg to PORTA
.equ LCDPORTPIN = PINA ; Set clear signal port pin reg to PORTA
.equ LCD_RS = PINA0
.equ LCD_RW = PINA1
.equ LCD_EN = PINA2
.equ LCD_D7 = PINA7
.equ LCD_D6 = PINA6
.equ LCD_D5 = PINA5
.equ LCD_D4 = PINA4
.def LCDData = r16
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
;***** Program ID
*****  
  
;PORTD      -> CONTROL KEYPAD  
;PORTC      -> BAR LED  
  
;*****MAIN*****  
  
reset_handler:  
    CALL LCD_Init  
    SER R16  
    OUT DDRC, R16  
  
    LDI ZL, 0  
    LDI ZH, 7  
  
    LDI R16, $30  
    MOV R10, R16 ;DIGIT -> ASCII  
    LDI R16, $37  
    MOV R11, R16 ;ALPHA -> ASCII  
  
    CLR R15  
  
; display the first line of information  
start:  
  
    CALL KEY_PAD_SCAN  
    MOV R24, R23
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
OUT PORTC, R24

CPI R24, 0xFF
BREQ CLEAR
CPI R24, 10
BRCC ALPHA

ADD R24, R10 ;ASCII -> DIGIT
ST Z+, R24      ;DATA
ST Z, R15       ;END LINE

LDI ZL, 0
LDI ZH, 7

CLR R16
CLR R17
CALL LCD_Move_Cursor
CALL LCD_Send_String
RJMP start

ALPHA:
ADD R24, R11 ;ASCII -> ALPHA
ST Z+, R24      ;DATA
ST Z, R15       ;END LINE

LDI ZL, 0
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
LDI ZH, 7

CLR R16
CLR R17
CALL LCD_Move_Cursor
CALL LCD_Send_String
rjmp start

CLEAR:
ldi r16, 0x01      ; Clear Display
call LCD_Send_Command
rjmp start

;*****FUNCTION*****
*****;

;*****INIT*****;
LCD_Init:
    ; Set up data direction register for Port A
    ldi r16, 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output
    out LCDPORTDIR, r16
    ; Wait for LCD to power up
    call     DELAY_10MS
    call     DELAY_10MS

    ; Send initialization sequence
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
ldi r16, 0x02      ; Function Set: 4-bit interface
call LCD_Send_Command

ldi r16, 0x28      ; Function Set: enable 5x7 mode for chars
call LCD_Send_Command

ldi r16, 0x0C      ; Display Control: Display OFF, Cursor OFF
call LCD_Send_Command

ldi r16, 0x01      ; Clear Display
call LCD_Send_Command

ldi r16, 0x80      ; Clear Display
call LCD_Send_Command

ret

;*****SEND CMD*****
LCD_Send_Command:
    push r17
    call LCD_wait_busy ; check if LCD is busy
    mov r17,r16          ;save the command

    ; Set RS low to select command register
    ; Set RW low to write to LCD
    andi r17,0xF0
    ; Send command to LCD
    out LCDPORT, r17
    nop
    nop
    ; Pulse enable pin
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
sbi LCDPORT, LCD_EN
nop
nop
cbi LCDPORT, LCD_EN
swap    r16
andi    r16,0xF0
; Send command to LCD
out    LCDPORT, r16
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
nop
cbi LCDPORT, LCD_EN
pop    r17
ret

;*****SEND DATA*****
LCD_Send_Data:
push   r17
call    LCD_wait_busy ;check if LCD is busy
mov     r17,r16           ;save the command
; Set RS high to select data register
; Set RW low to write to LCD
andi    r17,0xF0
ori     r17,0x01
; Send data to LCD
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6
Môn thí nghiệm: Vi xử lý

```
out LCDPORT, r17
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
cbi LCDPORT, LCD_EN
; Delay for command execution
;send the lower nibble
nop
swap    r16
andi    r16,0xF0
; Set RS high to select data register
; Set RW low to write to LCD
andi    r16,0xF0
ori     r16,0x01
; Send command to LCD
out LCDPORT, r16
nop
; Pulse enable pin
sbi LCDPORT, LCD_EN
nop
cbi LCDPORT, LCD_EN
pop    r17
ret
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
;*****SET_CURSOR*****;

LCD_Move_Cursor:

    cpi      r16,0 ;check if first row
    brne   LCD_Move_Cursor_Second
    andi    r17, 0x0F
    ori     r17,0x80
    mov     r16,r17
; Send command to LCD
    call   LCD_Send_Command
    ret

LCD_Move_Cursor_Second:

    cpi      r16,1 ;check if second row
    brne   LCD_Move_Cursor_Exit      ;else exit
    andi    r17, 0x0F
    ori     r17,0xC0
    mov     r16,r17
; Send command to LCD
    call   LCD_Send_Command

LCD_Move_Cursor_Exit:

    ; Return from function
    ret

;*****SEND STRING*****;

LCD_Send_String:
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
push    ZH                                ; preserve pointer
registers

push    ZL

push    LCDData

; fix up the pointers for use with the 'lpm' instruction

// lsl    ZL                                ; shift the pointer one
bit left for the lpm instruction

// rol    ZH

; write the string of characters

LCD_Send_String_01:

LD     LCDData, Z+                      ; get a character
cpi    LCDData,  0                       ; check for end of
string

breq   LCD_Send_String_02                ; done

; arrive here if this is a valid character

call   LCD_Send_Data                   ; display the character
rjmp   LCD_Send_String_01                ; not done, send another
character

; arrive here when all characters in the message have been sent to the
LCD module

LCD_Send_String_02:

pop    LCDData
pop    ZL                                ; restore pointer
registers
pop    ZH
ret
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
;*****LCD_WAIT_BUSY*****;

LCD_wait_busy:
    push    r16
    ldi     r16, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output
    out    LCDPORTDIR, r16
    ldi     r16, 0b11110010      ; set RS=0, RW=1 for read the busy
flag
    out    LCDPORT, r16
    nop
LCD_wait_busy_loop:
    sbi   LCDPORT, LCD_EN
    nop
    nop
    in    r16, LCDPORTPIN
    cbi   LCDPORT, LCD_EN
    nop
    sbi   LCDPORT, LCD_EN
    nop
    nop
    cbi   LCDPORT, LCD_EN
    nop
    andi  r16, 0x80
    cpi    r16, 0x80
    breq  LCD_wait_busy_loop
    ldi   r16, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
    out LCDPORTDIR, r16
    ldi    r16,0b00000000      ; set RS=0, RW=1 for read the busy
flag

    out    LCDPORT, r16
    pop    r16
    ret

;*****DELAY10MS*****
DELAY_10MS:
    LDI R16,10
LOOP2:
    LDI R17,250
LOOP1:
    NOP
    DEC R17
    BRNE LOOP1
    DEC R16
    BRNE LOOP2
    RET

KEY_PAD_SCAN:
;PD_0 -> PD_3: OUTPUT, COL
;PD_4 -> PD_7: INPUT, ROW

    LDI R16, $0F
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
OUT DDRD, R16

LDI R16, $F0
OUT PORTD, R16
CALL BUTTON

LDI R22, 0B11110111 ;INITIAL COLUMN MASK
LDI R23, 0           ;INITIAL PRESSED ROW VALUE
LDI R24, 3           ;SCANNING COLUMN INDEX

KEYPAD_SCAN_LOOP:
    OUT PORTD, R22
    SBIC PIND, 4      ;CHECK ROW 0
    RJMP KEYPAD_SCAN_CHECK_COL2
    RJMP KEYPAD_SCAN_FOUND          ;ROW 0 IS PRESSED

KEYPAD_SCAN_CHECK_COL2:
    SBIC PIND, 5      ;CHECK ROW 1
    RJMP KEYPAD_SCAN_CHECK_COL3
    LDI R23, 1         ;ROW 1 IS PRESSED
    RJMP KEYPAD_SCAN_FOUND

KEYPAD_SCAN_CHECK_COL3:
    SBIC PIND, 6      ;CHECK ROW 2
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6

Môn thí nghiệm: Vi xử lý

```
RJMP KEYPAD_SCAN_CHECK_COL4
LDI R23, 2 ;ROW 2 IS PRESSED
RJMP KEYPAD_SCAN_FOUND

KEYPAD_SCAN_CHECK_COL4:
SBIC PIND, 7 ;CHECK ROW 3
RJMP KEYPAD_SCAN_NEXT_ROW
LDI R23, 3 ;ROW 3 IS PRESSED
RJMP KEYPAD_SCAN_FOUND

KEYPAD_SCAN_NEXT_ROW:
CPI R24, 0
BREQ KEYPAD_SCAN_NOT_FOUND
ROR R22
DEC R24
RJMP KEYPAD_SCAN_LOOP

KEYPAD_SCAN_FOUND:
; combine row and column to get key value (0-15)
;key code = row*4 + col
LSL R23 ; shift row value 4 bits to the left
LSL R23
ADD R23, R24 ; add row value to column value
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6
Môn thí nghiệm: Vi xử lý

```
RET

KEYPAD_SCAN_NOT_FOUND:
    LDI R23, 0xFF ;NO KEY PRESSED
    RET

BUTTON:
    LDI R17, 50
DEBOUNCING_1:
    IN R16, PIND
    CPI R16, $FF ;DETECTE STATUS OF BUTTON
    BREQ BUTTON
    DEC R17
    BRNE DEBOUNCING_1

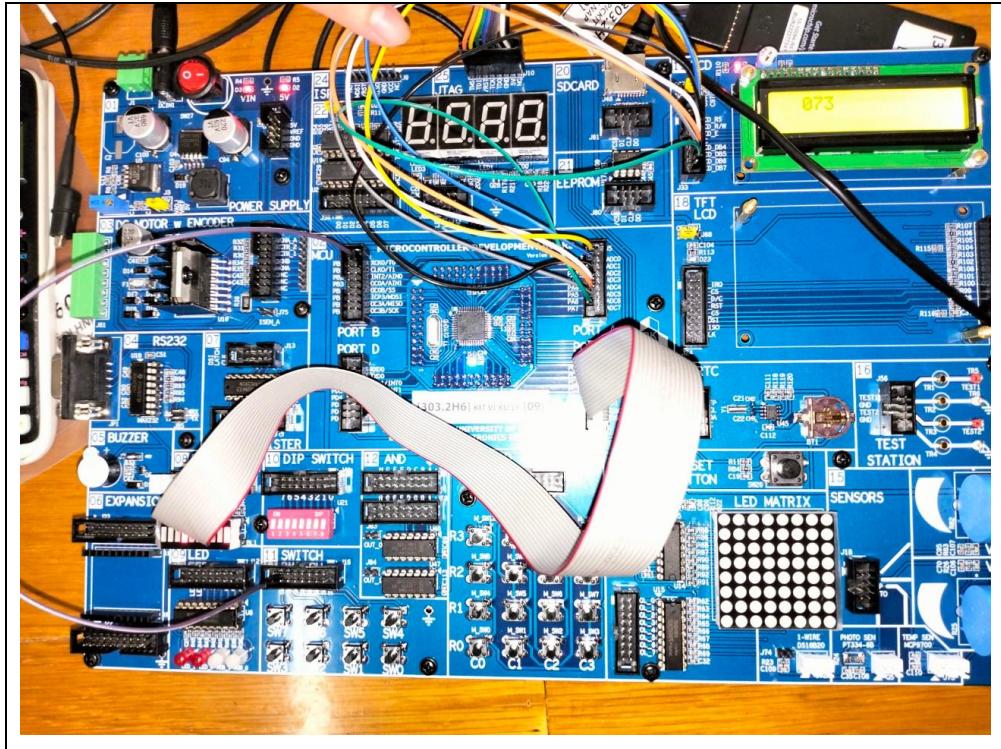
    RET

; module lcd_send_data, module lcd_command is similar to previous
sections.
```

BÁO CÁO

Nhóm môn học: L13

Nhóm:6
Môn thí nghiệm: Vi xử lý



LAB 2-1

DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

MỤC TIÊU:

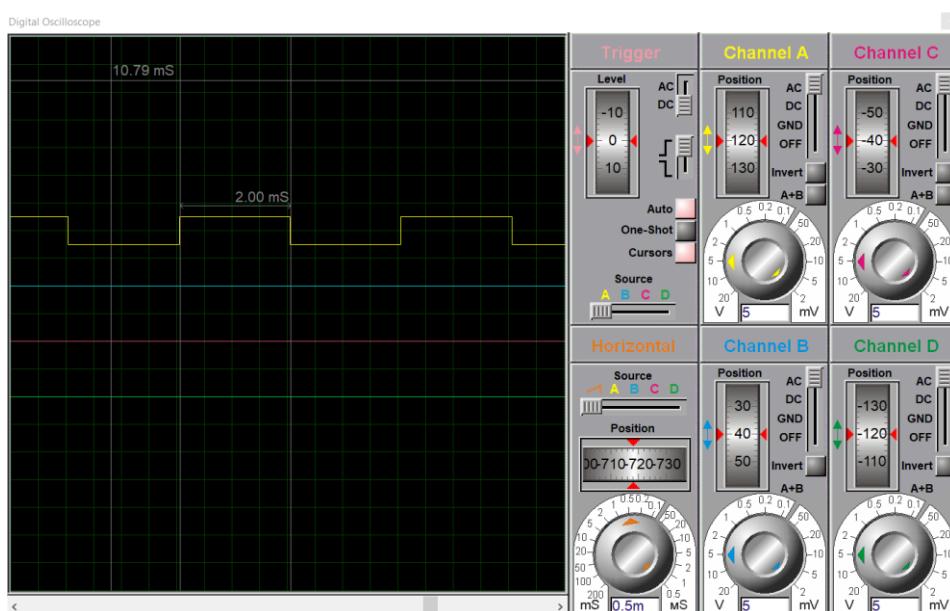
- Hiểu các mode hoạt động của timer
- Hiểu cách sử dụng timer để tạo delay và tạo xung

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 4, 5
- Atmel-2505-Setup-and-Use-of-AVR-Timers_ApplicationNote_AVR130.pdf

BÀI 1

- a) Viết chương trình con delay 1 ms sử dụng timer 0. Sử dụng chương trình con này để tạo xung 500hz trên chân PA0.
- b) Mô phỏng, chỉnh sửa chương trình để tạo ra xung chính xác.
- c) Kết nối chân PA0 vào oscilloscope để kiểm tra

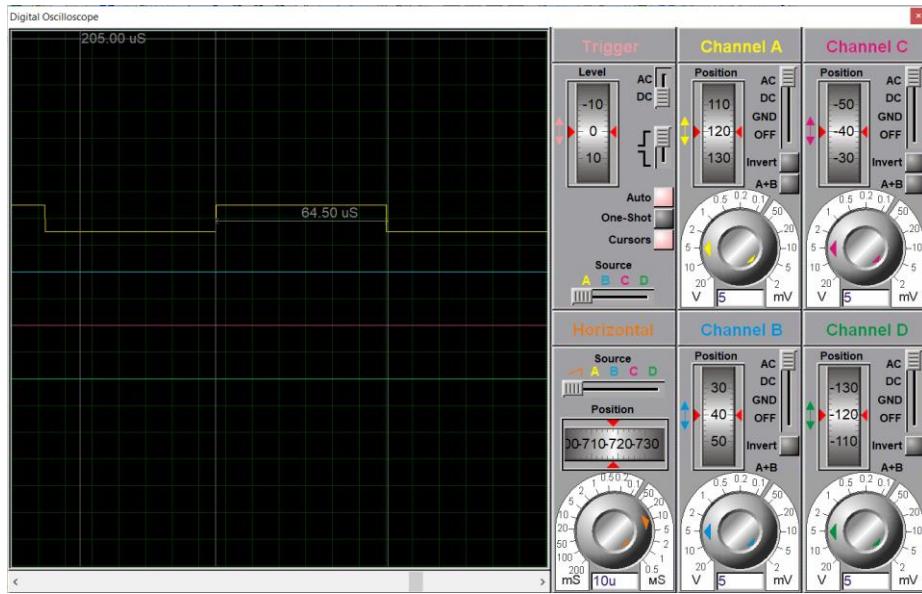


LAB 2-1

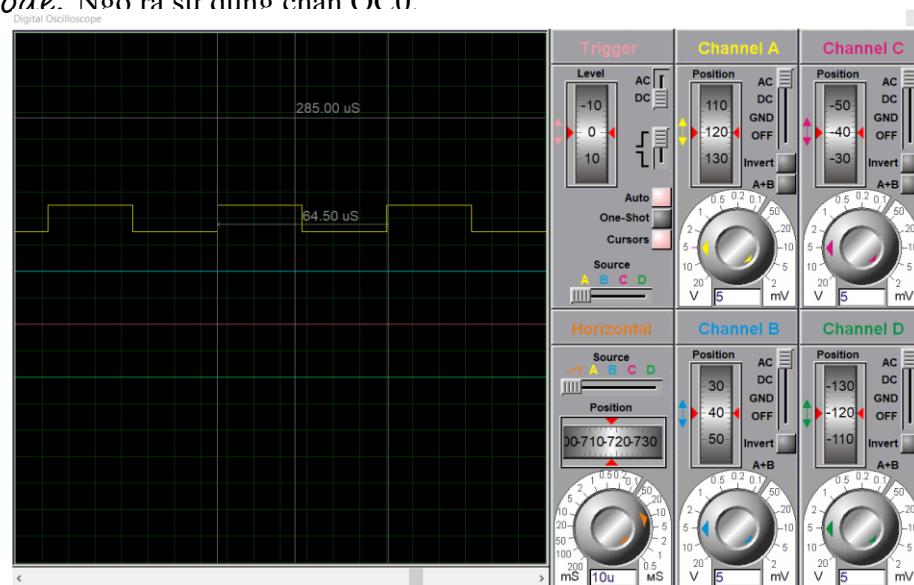
DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

BÀI 2

- a) Viết chương trình tạo 1 xung vuông 64 us sử dụng timer 0 ở chế độ *Normal mode*. Ngõ ra sử dụng chân OC0.



- b) Viết chương trình thực hiện tạo xung vuông có chu kỳ 64 us sử dụng timer 1 ở chế độ *CTC mode*. Ngõ ra sử dụng chân OC0.



LAB 2-1

DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

- c) Kết nối chân OC0 ra oscilloscope và quan sát

BÀI 2

- a) Viết chương trình tạo 1 xung vuông 64 us sử dụng timer 0 ở chế độ *Normal mode*. Ngõ ra sử dụng chân OC0.
- b) Viết chương trình thực hiện tạo xung vuông có chu kỳ 64 us sử dụng timer 1 ở chế độ *CTC mode*. Ngõ ra sử dụng chân OC0.
- c) Kết nối chân OC0 ra oscilloscope và quan sát

BÀI 3

- a) Cho chương trình sau tạo 2 xung PWM trên OC0A và OC0B

```
.org 00
call initTimer0

start:
    rjmp start

initTimer0:
    // Set OC0A (PB3) and OC0B (PB4) pins as outputs
    ldi r16, (1 << PB3) | (1 << PB4);
    out DDRB,r16
    ldi r16, (1 << COM0B1)|(1 << COM0A1) | (1 << WGM00)|(1 << WGM01)
    out TCCR0A,r16          // setup TCCR0A
    ldi r16, (1 << CS01)
    out TCCR0B,r16          // setup TCCR0B
    ldi r16, 100
```

LAB 2-1

DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

```
out  OCR0A,r16          //OCR0A = 100  
ldi   r16, 75  
out  OCR0B,r16          //OCR0B = 75  
ret
```

- b) Kết nối chân OC0A và chân OC0B ra 2 kênh đo của oscilloscope, đo và ghi nhận, giải thích dạng sóng thu được

BÀI 4

- a) Sửa chương trình trên ứng với các trường hợp khác nhau của TCCR0A và TCCR0B

TCCR0A								TCCR0B								
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	COM0A 1	COM0A 0	COM0B 1	COM0B 0			WGM0 1	WGM0 0	FOC0 A	FOC0 B			WGM0 2	CS0 2	CS0 1	CS0 0
2	1	0	1	0			1	1					0	0	1	0
3	1	0	1	0			1	1					1	0	1	0
4	1	0	1	0			0	1					0	0	1	0

- b) Kết nối chân OC0A và chân OC0B ra 2 kênh đo của oscilloscope, đo và ghi nhận, giải thích dạng sóng thu được

BÀI 5

- a) Viết chương trình tạo ra 1 xung tần số 1Khz, duty cycle 25% trên chân OC0B
- b) Kết nối vào Oscilloscope và đo dạng sóng ngõ ra
- c) Kết nối OC0B vào kênh R của 1 LED RGB. Viết chương trình để tăng duty cycle trên OC0B từ 0% lên 100% rồi lại giảm xuống 0, sau 10 ms duty cycle tăng/giảm 1%.

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

BÀI 1

1. Trả lời các câu hỏi

- a. Khoảng thời gian trễ lớn nhất có thể tạo ra bởi timer 0 với tần số 8Mhz là bao nhiêu? Trình bày cách tính toán

Với $F_{osc} = 8\text{Mhz}$ $\rightarrow T_{osc} = 0,125\text{us}$

Ta có: $T_{DL} = n \times T_{osc} \times N$.

Muốn T_{DL} max thì $n \times N$ max $\rightarrow N = 1024$ và $n = 256$.

Khi đó $T_{DL} = 256 \times 0.125\mu\text{s} \times 1024 = 0.032768 \text{ s}$

- b. Khoảng thời gian trễ lớn nhất có thể tạo ra bởi timer 1 với tần số 8Mhz là bao nhiêu? Trình bày cách tính toán

Với $F_{osc} = 8\text{Mhz}$ $\rightarrow T_{osc} = 0,125\text{us}$

Ta có: $T_{DL} = n \times T_{osc} \times N$.

Muốn T_{DL} max thì $n \times N$ max $\rightarrow N = 1024$ và $n = 2^{15}$.

Khi đó $T_{DL} = 2^{15} \times 0.125\mu\text{s} \times 1024 \approx 4,19 \text{ s}$

- c. Trình bày cách tính prescaler và các giá trị nạp vào các thanh ghi của timer0 trong bài thí nghiệm

Ở mode NOR các bit WGM03:WGM02:WGM01 = 000

Suy ra bit 1 và bit 0 của thanh ghi TCCR0A = 00 và bit 3
của thanh ghi TCCR0B = 0

Để chọn prescaler cho $T_{DL} = 1\text{ms}$ ta dùng công thức:

$T_{DL} = n \times T_{osc} \times prescal.$ Với $T_{osc} = 0.125\text{us}$

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

Prescaler của timer0 có các giá trị 1, 8, 64, 256, 1024. Ta chọn lần lượt các giá trị này và tính n. Thấy rằng prescaler = 64 ta được kết quả n là 1 số nguyên 125 (điều này giúp ta dễ nạp giá trị cho thanh ghi vì số không lẻ).

Với n = 125 đồng nghĩa ta nạp giá trị -125 cho thanh ghi TCNT0

Với prescaler = 64 đồng nghĩa 3 bit CS02:CS01:CS00 của thanh ghi TCCR0B = 011

Vậy các giá trị của các thanh ghi trong timer 0 lần lượt là TCNT0 = -125, TCCR0A = \$00, TCCR0B = \$03.

d. Mã nguồn chương trình với chú thích

```
.ORG 0
    RJMP MAIN
    .ORG 0X40
MAIN:
    SBI DDRA, 0           ;A0 is output
    SBI PORTA, 0
    RCALL DELAY_1MS
    CBI PORTA, 0
    RCALL DELAY_1MS
    RJMP MAIN

DELAY_1MS:
    LDI R16, -125 ;Nap gia tri can dem
    OUT TCNT0, R16
    LDI R16, $00 ;Mode NOR
    OUT TCCR0A, R16
    LDI R16, $03 ;Mode NOR, bo chia 64
    OUT TCCR0B, R16

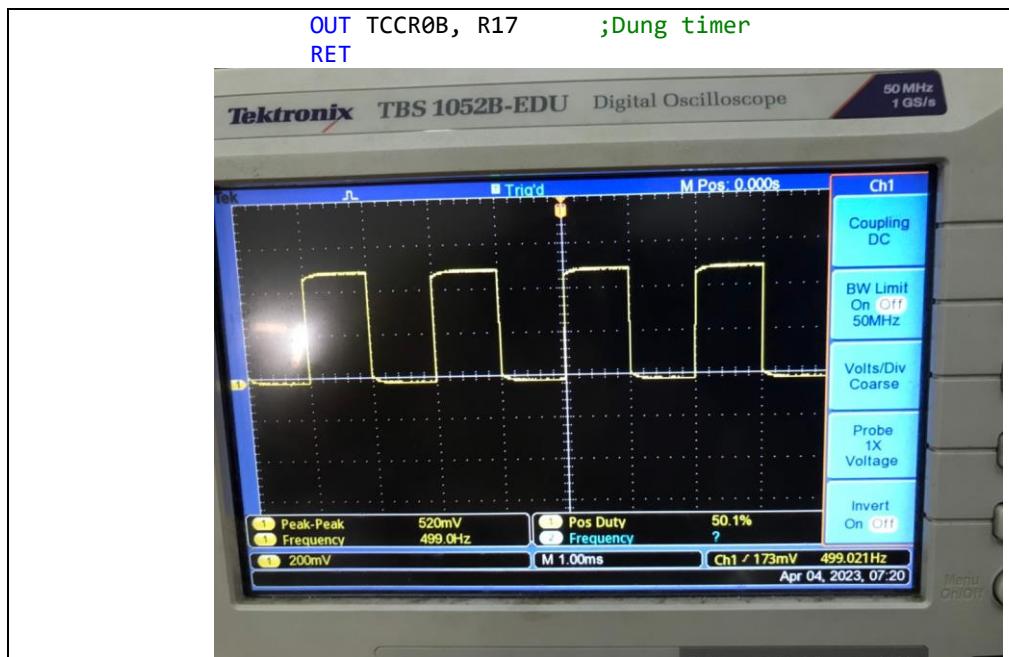
WAIT:
    IN R17, TIFR0 ;Doc thanh ghi TIFR0
    SBRS R17, TOV0      ;Kiem tra bit TOV0 da len 1
    chua
    RJMP WAIT
    OUT TIFR0, R17      ;Xoa bit TOV0 ve 0
    CLR R17
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 2

1. Trả lời các câu hỏi
 - a. Ở mode Normal, khi nào bit TOVx được set lên 1?

Khi timer tràn từ \$FF → \$00 hoặc \$FFFF → \$0000 thì cờ
TOVn set lên 1

- b. Ở mode CTC, khi nào bit OCFx được set lên 1?

Khi timer đạt đến giá trị trong thanh ghi (OCRxA + 1) thì cờ
OCFxA lên 1

Khi timer đạt đến giá trị trong thanh ghi (OCRxB + 1) thì cờ
OCFxB lên 1

- c. Giá trị các thanh ghi cấu hình cho timer 0 cho 2 trường hợp

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

Mode NOR TCNT0 = -125, TCCR0A = \$00, TCCR0B = \$03

Mode CTC OCR0A = 255, TCCR0A = \$02, TCCR0B = \$01

d. Mã nguồn chương trình cho hai trường hợp

```
.ORG 0

START:
    SBI DDRB, 3 ;1MC
//*****TIMER1_MODE_CTC*****
    RCALL TIMER1_MODE_CTC ;3MC
    SBI PORTB, 3
    RCALL DELAY_32US_CTC ;3MC
    CBI PORTB, 3
    RCALL DELAY_32US_CTC
    RJMP START ;2MC
//*****TIMER1_MODE_CTC*****

//*****TIMER0_MODE_NOR*****
    RCALL TIMER0_MODE_NOR ;3MC
    RCALL DELAY_32US_NOR
    RJMP START ;2MC
//*****TIMER1_MODE_NOR*****
TIMER1_MODE_CTC:
    LDI R17, 255 ;1MC
    STS OCR1AL, R17 ;1MC
    LDI R17, $40 ;1MC
    STS TCCR1A, R17 ;1MC Mode CTC
timer1
    CLR R17 ;1MC
    STS TCCR1B, R17 ;1MC Timer
dung mode CTC
    RET

DELAY_32US_CTC:
    LDI R17, $09 ;1MC
    STS TCCR1B, R17 ;1MC
    Mode CTC, N = 1
WAIT_CTC:
    IN R17, TIFR1 ;1MC
    SBRS R17, OCF1A ;2/1MC
    RJMP WAIT_CTC ;2MC
    OUT TIFR1, R17 ;1MC
    CLR R17 ;1MC
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
        STS TCCR1B, R17          ;1MC  Timer
dung
        RET                      ;4MC

        TIMER0_MODE_NOR:
        LDI R17, $40            ;1MC  Enable OC0A
        OUT TCCR0A, R17         ;1MC  Mode NOR
        CLR R17                ;1MC
        OUT TCCR0B, R17         ;1MC
        Timer dung mode NOR
        RET                      ;4MC

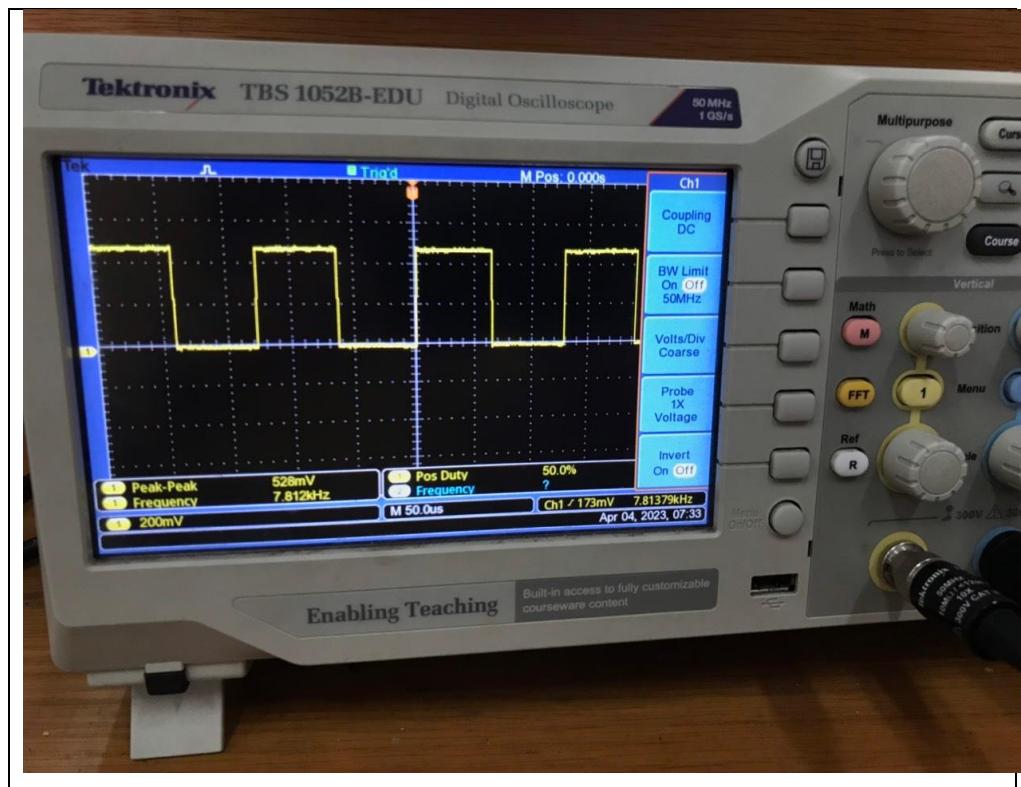
        DELAY_32US_NOR:
        LDI R16, -240           ;1MC
Nap gia tri dem -256 sau khi tinh sai so giam so luong
xung con lai 240
        OUT TCNT0, R16          ;1MC
        LDI R16, $01            ;1MC
        OUT TCCR0B, R16          ;1MC  Mode NOR, N = 1
WAIT:
        IN R17, TIFR0           ;1MC
Lay gia tri thanh ghi bao tran ve
        SBRS R17, TOV0          ;2/1MC
Kiem tra co TOV0 = 1 chua
        RJMP WAIT              ;2MC
        OUT TIFR0, R17          ;1MC
        CLR R17                ;1MC
        OUT TCCR0B, R17          ;1MC  Stop timer0
        RET                      ;4MC
```

BÁO CÁO

Nhóm: 6

Nhóm môn học: L13

Môn thí nghiệm: Vi xử lý

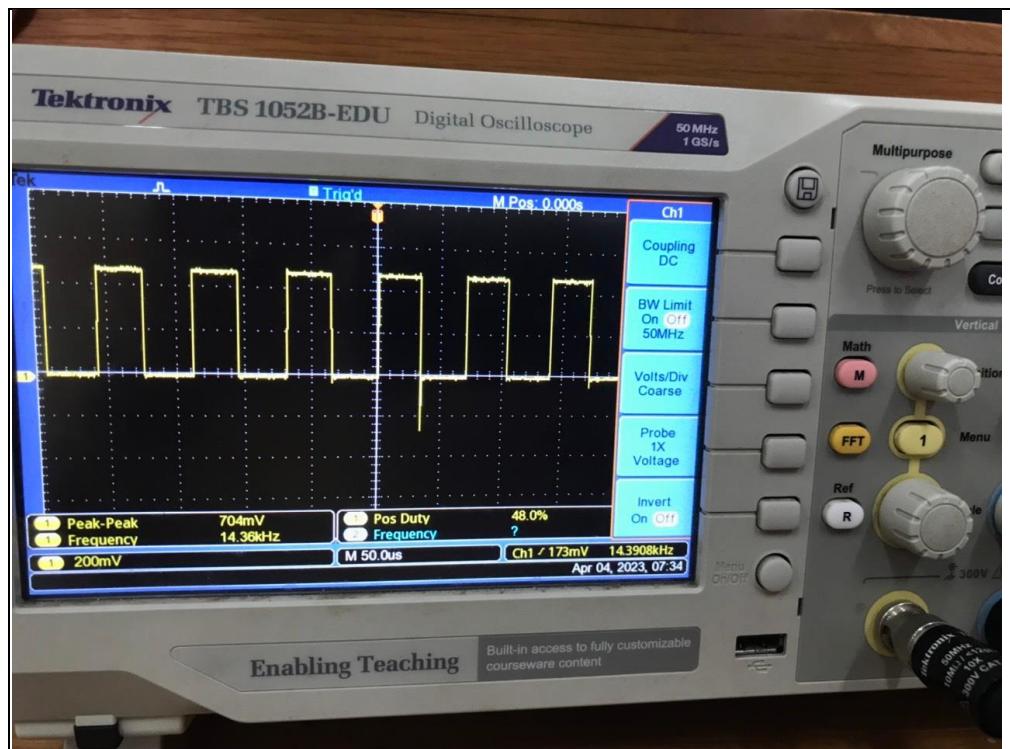


BÁO CÁO

Nhóm: 6

Nhóm môn học: L13

Môn thí nghiệm: Vi xử lý



BÀI 3

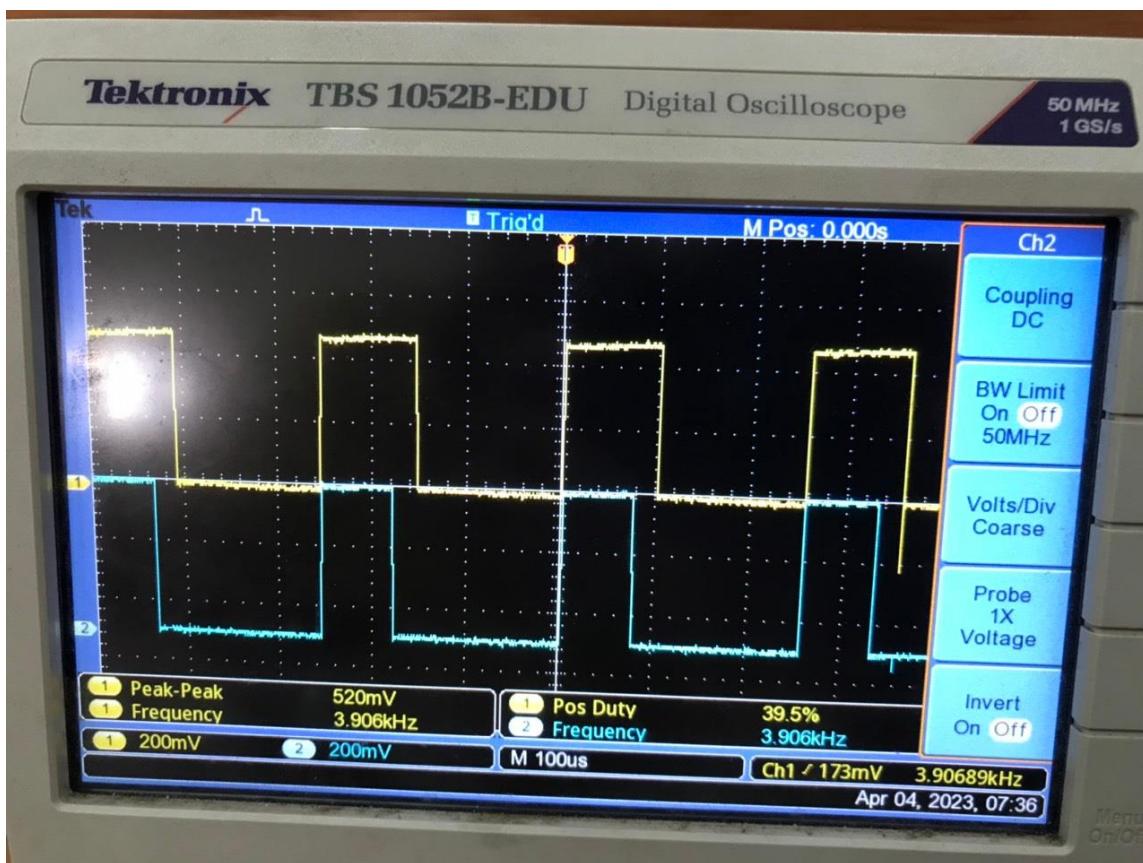
1. Trả lời các câu hỏi
 - a. Dạng sóng trên oscilloscope (chụp và chèn vào)

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

- b. Giải thích lý do có dạng sóng (tần số, chu kỳ làm việc, phase) như kết quả
-Ở thanh ghi TCCR0A, ta có COM0A1 và COM0B1 bằng 1 còn COM0A0 và
COM0B0 bằng 0 nên trạng thái ngõ ra của OCnA/OCnB bằng 0 khi đạt kết quả so
sánh. Timer 0 ở mode CTC. Giá trị của OCR0A lớn hơn OCR0B nên duty cycle
của A lớn hơn B.

BÀI 4

1. Trả lời các câu hỏi
 - a. Các mode làm việc của timer 0 ứng với các giá trị trên bảng

TH_1: Mode FPWM (TOP: \$FF)
TH_2: Mode FPWM (TOP: OCR0A)
TH_3: Mode PCPWM (TOP: \$FF)

2. Chụp ảnh các dạng sóng ứng với các mode làm việc và giải thích.

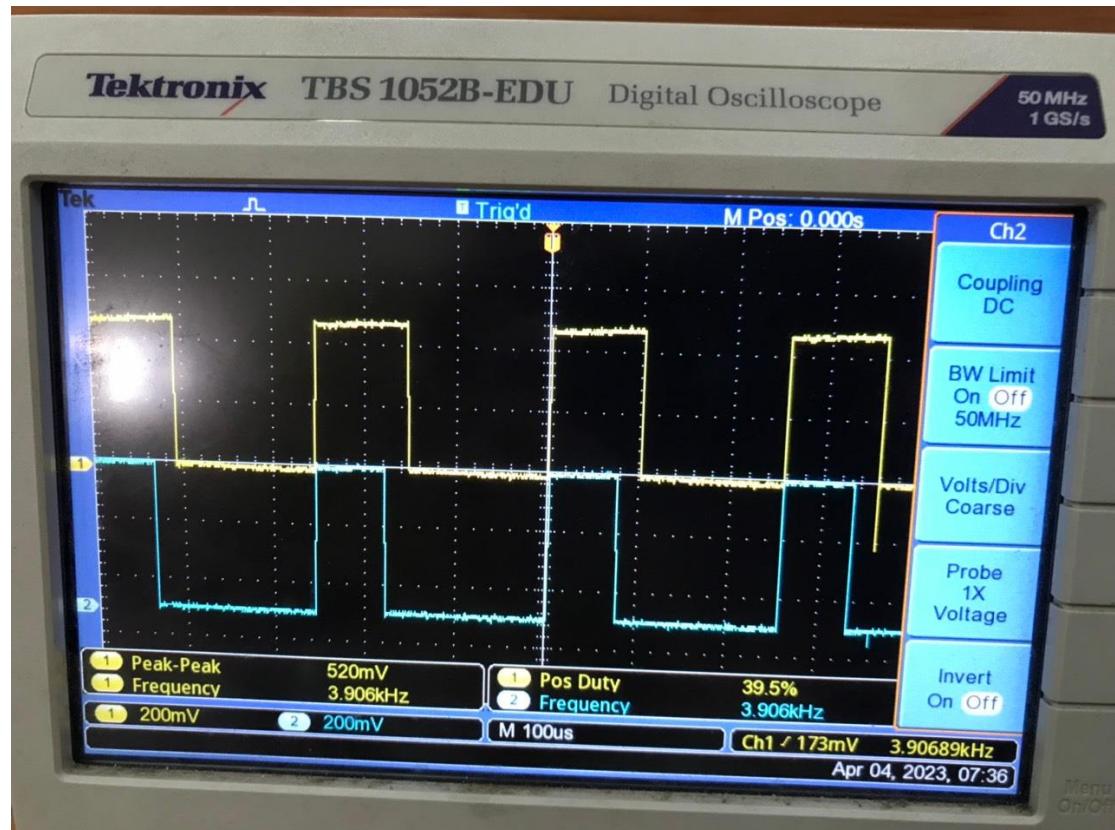
BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

TH_1:



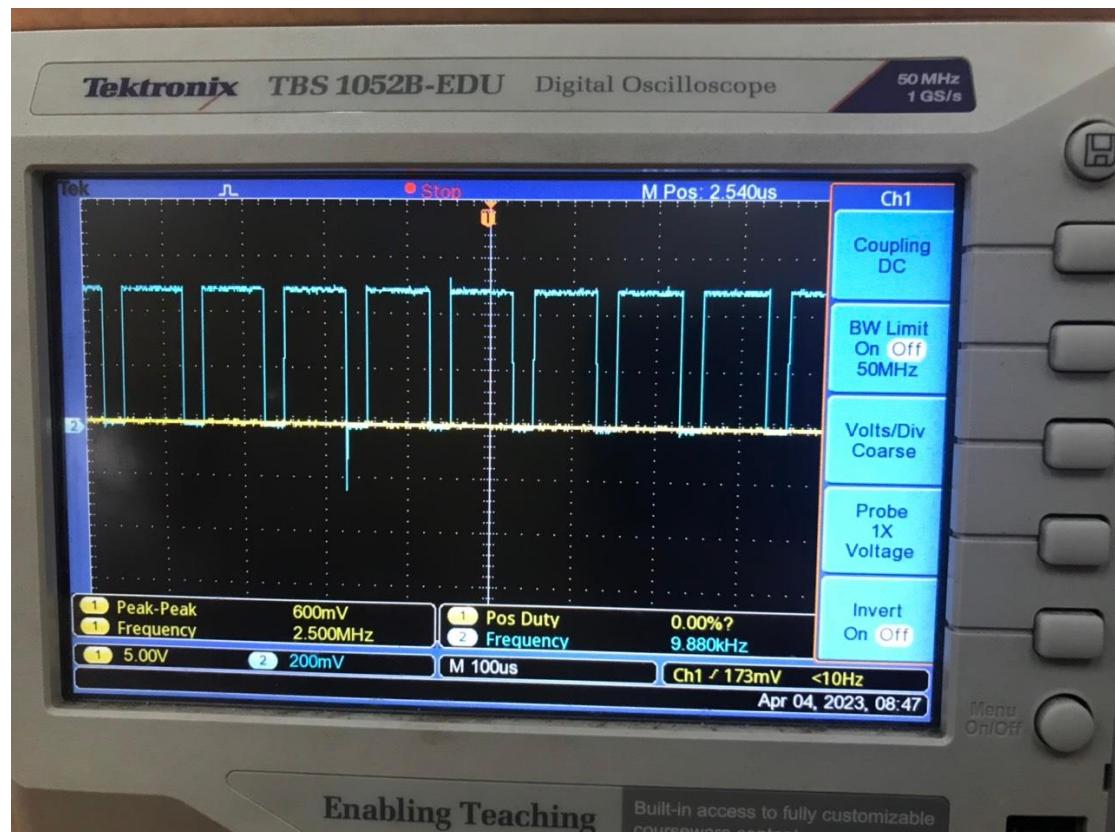
TH_2:

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



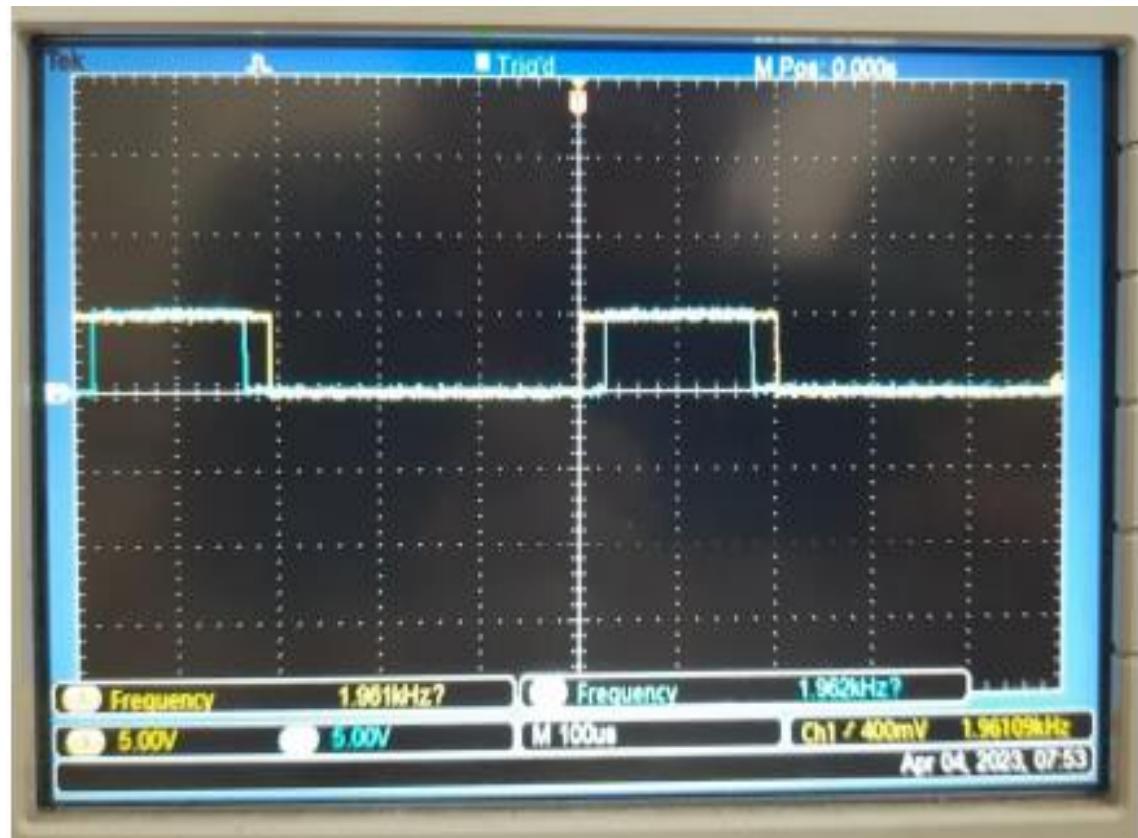
TH_3:

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 5

1. Trả lời các câu hỏi
 - a. Timer 0 làm việc ở mode nào?
- Mode: CTC
 - b. Giá trị đưa vào các thanh ghi của timer 0 là bao nhiêu? Giải thích
 - c. - TCCR0A: 0b00010010, TCCR0B: \$03, OCR0A: các giá trị tính toán khi code
2. Trình bày mã nguồn với chú thích.

```
.DEF SIGN_H = R19  
.DEF SIGN_L = R20  
.ORG 0
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
RJMP MAIN
.ORG 0x40
MAIN:
    LDI R22, 2                      ;So lan lap cau a
    LDI R23, 20                     ;10ms
    LDI R24, 0                      ;Tang 1%
    CLR R25                         ;Change rule
    LDI SIGN_H, 0
    LDI SIGN_L, 125
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, HIGH(RAMEND)
    OUT SPL, R16
    SBI DDRB, 4                     ;OC0B is output
    LDI R16, (1<<WGM01) | (1<<COM0B0) ;Mode CTC dao bit OC0B khi dat
kqua so sanh
    OUT TCCR0A, R16
START:
    RCALL CAU_C                   ;This line to change between a and c
CAU_C:
    MOV R16, SIGN_H               ;Gia tri khai dong
WAVE_C:
    OUT OCR0A, R16                ;Nap gia tri cho bo dem
    LDI R16, (1<<CS01)|(1<<CS00);Mode CTC va Prescaler = 64
    OUT TCCR0B, R16
WAIT_C_H:
    IN R17, TIFR0                 ;Doc gia tri thanh ghi bao tran
    SBRS R17, OCF0A               ;Kiem tra co bao tran
    RJMP WAIT_C_H                ;Chua tran tiep tuc dem
    OUT TIFR0, R17                ;Xoa co tran
    MOV R16, SIGN_L
    OUT OCR0A, R16
WAIT_C_L:
    IN R17, TIFR0                 ;Doc gia tri thanh ghi bao tran
    SBRS R17, OCF0A               ;Kiem tra co bao tran
    RJMP WAIT_C_L                ;Chua tran tiep tuc dem
    OUT TIFR0, R17                ;Xoa co tran
    DEC R23
    BRNE AGAIN
    LDI R23, 10
    INC R24
    CPI R24, 100                  ;Khi duty dat 100%
    BREQ CHANGE_RULE             ;Thay doi viec cong tru cac thanh ghi
                                ;Co bit0 cua R25 = 0 tang dutycycle tu
0 --> 100
CONTI:
    SBRC R25, 0                   ;Neu bang 1 giam dutycycle 100 → 0
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

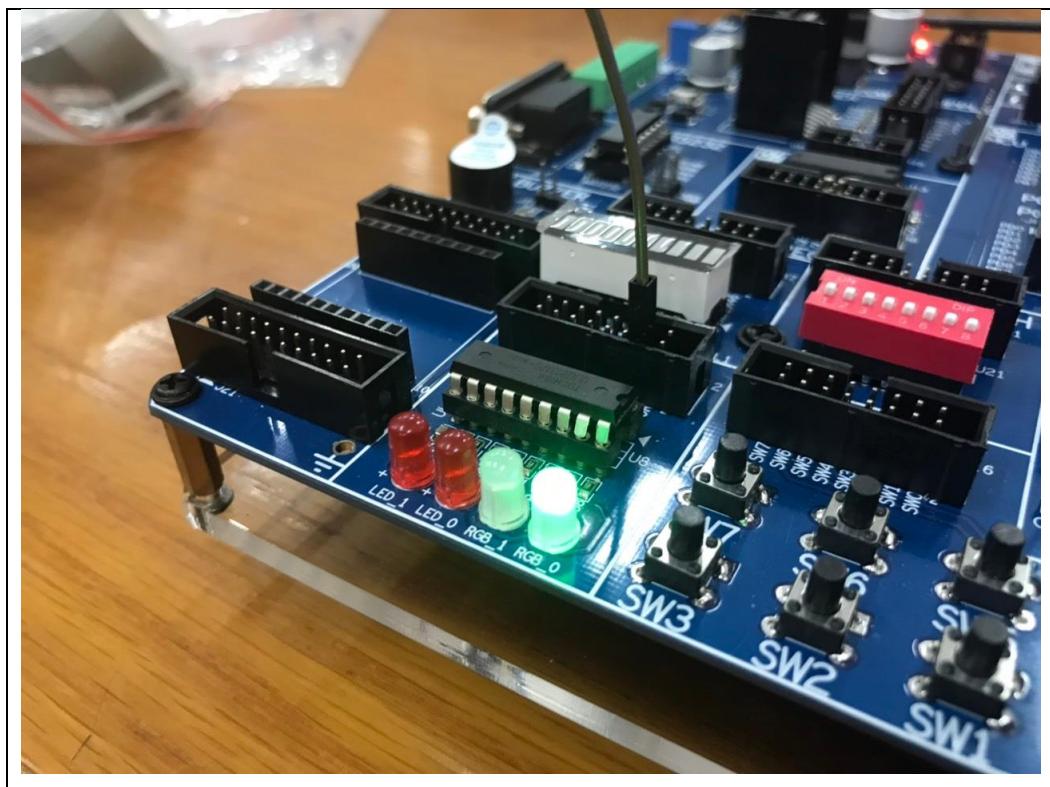
```
RJMP CHANGE_VALUE
INC SIGN_H
DEC SIGN_L
RJMP AGAIN
CHANGE_RULE:
CLR R24
LDI R26,$01
EOR R25, R26
RJMP CONTI
CHANGE_VALUE:
DEC SIGN_H
INC SIGN_L
AGAIN:
OUT OCR0A, SIGN_H
RJMP WAIT_C_H
RET
CAU_A:
LDI ZH, HIGH(DATA_A<<1)
LDI ZL, LOW(DATA_A<<1)
LDI R16, 20 ;Gia tri khai dong
WAVE_A:
OUT OCR0A, R16 ;Nap
gia tri cho bo dem
LDI R16, (1<<CS01)|(1<<CS00) ;Mode CTC va
Prescaler = 64
OUT TCCR0B, R16
WAIT_A:
IN R17, TIFR0 ;Doc gia tri thanh ghi bao tran
SBRS R17, OCF0A ;Kiem tra co bao tran
RJMP WAIT_A ;Chua tran tiep tuc dem
OUT TIFR0, R17 ;Xoa co tran
LPM R16, Z+
OUT OCR0A, R16
DEC R22
BRNE WAIT_A
LDI R22, 2
LDI ZH, HIGH(DATA_A<<1)
LDI ZL, LOW(DATA_A<<1)
RJMP WAIT_A
RET
DATA_A:
.DB 92, 30
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



LAB 2-2

GIAO TIẾP LED 7 ĐOẠN VÀ LED MA TRẬN

MỤC TIÊU:

- Giao tiếp được với LED 7 đoạn
- Giao tiếp được với LED ma trận

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 4
- Atmel-2505-Setup-and-Use-of-AVR-Timers_ApplicationNote_AVR130.pdf

BÀI 1

- a) Kết nối 1 port của AVR vào header J34. Kết nối 2 chân port khác vào tín hiệu nLE0 và nLE1 trên header J82. Set jumper để cấp nguồn cho LED 7 đoạn
- b) Sử dụng các chương trình mẫu trong tài liệu hướng dẫn thí nghiệm, viết chương trình hiển thị số 0123 lên 4 LED 7 đoạn, sử dụng timer 0 để quét LED với tần số quét 50Hz.

BÀI 2

- a) Kết nối port của AVR vào dip Switch, giả sử đó là PORTA
- b) Viết chương trình hiện giá trị PORTA * 9 lên 4 LED 7 đoạn.
- c) Thay đổi giá trị dip switch và quan sát kết quả

BÀI 3

- a) Kết nối các tín hiệu cần thiết để điều khiển LED ma trận. .
- b) Sử dụng chương trình mẫu, chỉnh sửa nếu cần thiết để hiển thị chữ ‘A’ lên LED ma trận. Quét LED ma trận sử dụng timer để tạo delay với tần số quét 25 Hz.
- c) Chỉnh sửa chương trình để đạt tần số quét là 125Hz.

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

BÀI 1

1. Trả lời các câu hỏi

- a. Để có tần số quét là 50Hz, một LED sẽ sáng 1 lần trong bao lâu?
- 1 LED sáng 1 lần trong 5ms
- b. Cấu hình timer như thế nào để có độ trễ này

Cấu hình timer0 mode NOR với giá trị nạp cho
TCNT0 là -156
Giá trị các thanh ghi của timer0:
TCNT0: -156, TCCR0A: \$00, TCCR0B: \$04

2. Mã nguồn và chú thích

```
.EQU DATA_DDR = DDRB
.EQU DATA_OUT = PORTB
.EQU LE_DDR = DDRA ; enable pin of HC573
.EQU LE = PORTA
.ORG 0
RJMP MAIN
.ORG $40

MAIN:
    SER R16
    OUT DATA_DDR, R16
    SBI LE_DDR, 0
    SBI LE_DDR, 1
    CBI LE, 0
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
CBI LE, 1 ; lock 573
LDI R18, $FF ; choose led to display

LOOP:
    LDI R17, 3 ; data to display
    LDI R19, $FE
    RCALL DISPLAY_7SEG
    RCALL SET_UP_TIMER

    LDI R17, 2
    LDI R19, $FD
    RCALL DISPLAY_7SEG
    RCALL SET_UP_TIMER

    LDI R17, 1
    LDI R19, $FB
    RCALL DISPLAY_7SEG
    RCALL SET_UP_TIMER

    LDI R17, 0
    LDI R19, $F7
    RCALL DISPLAY_7SEG
    RCALL SET_UP_TIMER
    RJMP LOOP

; -----
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
; input data R18 -- choose led to display
; input R17 -- data to display

DISPLAY_7SEG:
    PUSH R18
    ANDI R18, $0F

    OUT DATA_OUT, R18           ; enable 573 to choose led
    ;Turn off led
    SBI LE, 1
    NOP
    CBI LE, 1

    LDI ZH, HIGH(TAB << 1)     ; look-up Led7seg
    LDI ZL, LOW(TAB << 1)
    ADD ZL, R17
    BRCC SKIP
    INC ZH

SKIP: LPM R20, Z
    OUT DATA_OUT, R20
    SBI LE, 0
    NOP
    CBI LE, 0
    ;Chon led
    OUT DATA_OUT, R19
    SBI LE, 1
```

BÁO CÁO

Nhóm môn học:

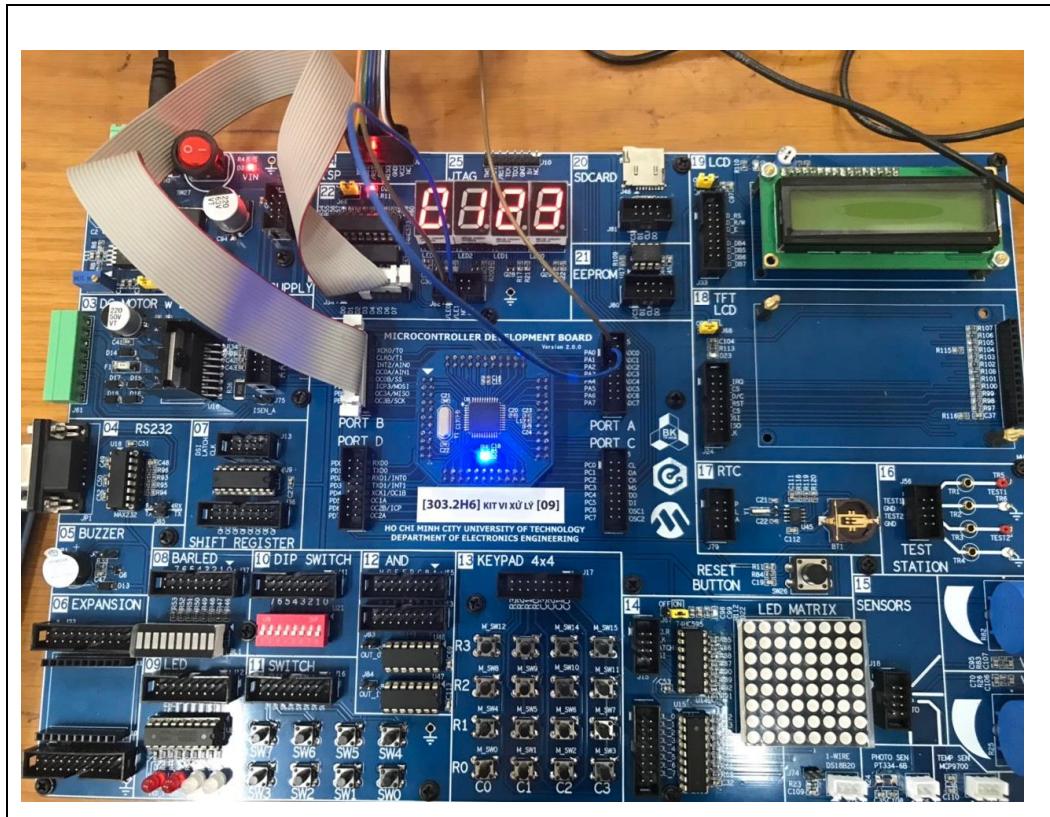
Nhóm:
Môn thí nghiệm:

```
NOP  
  
CBI LE, 1  
  
POP R18  
  
RET  
  
;  
----- USE TIMER0 TO DELAY  
  
SET_UP_TIMER: ; 1/200 = 0.05 s = 40 000 (MCs) => use CLK/256 divider  
with N = 156 pulse  
  
    LDI R16, 0  
    OUT TCCR0A, R16          ; timer 0 mod NOR  
    OUT TCCR0B, R16          ; stop timer  
  
DELAY:  
  
    LDI R16, $64            ; input (-n)  
    OUT TCNT0, R16          ; count 156 pulses  
    LDI R16, $04            ; use CLK/256 prescaler  
    OUT TCCR0B, R16          ; start timer  
  
WAIT:  
  
    SBIS TIFR0, TOV0        ; check if TOV = 1  
    RJMP WAIT               ; continue counting  
    SBI TIFR0, TOV0         ; reset TOV0  
    LDI R16, 0  
  
    OUT TCCR0B, R16          ; stop timer  
    RET  
  
TAB: .DB 0xC0, 0xF9, 0xA4, 0xB0
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:



BÀI 2

1. Trả lời các câu hỏi

a. Giá trị PORTA * 9 là số có bao nhiêu bit

- 16 bit

b. Làm thế nào để hiển thị từng chữ số lên 4 LED?

Theo đoạn code nhóm đã làm thì nhóm sẽ đưa hàng nghìn, hàng trăm, hàng chục của kết quả vào 4 ô nhớ liên tiếp sau đó dùng chương trình hiển thị LED 7 đoạn ở bài 2 chỉnh sửa cách lấy giá trị từ bộ nhớ (bài 2 ta lấy giá trị hiển thị trong bộ nhớ ROM, ở bài này

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

ta lấy giá trị trong RAM để tham chiếu đến bảng tra được lưu trong ROM và lấy giá trị cần hiển thị ra) và đưa kết quả hiển thị ra LED 7 đoạn.

2. Mã nguồn và chú thích

```
.DEF TRAM = R23
.DEF CHUC = R24
.DEF DONVI = R25
.DEF SONHO = R15
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
CALL CONFIG
START:
IN R20, PINA
COM R20
CALL SAVE_TRAM_CHUC_DONVI
CALL CACULATE ;Ket qua tinh toan se luu vao o nho
$0100 -- $0103
LOOP:
LDI R18, $FF ; TURN OFF LED
LDI XL, $00
LDI XH, $01
LDI R19, $FE
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
RCALL DISPLAY_7SEG  
RCALL SET_UP_TIMER  
  
LDI R19, $FD  
LDI XL, $01  
LDI XH, $01  
RCALL DISPLAY_7SEG  
RCALL SET_UP_TIMER  
  
LDI R19, $FB  
LDI XL, $02  
LDI XH, $01  
RCALL DISPLAY_7SEG  
RCALL SET_UP_TIMER  
  
LDI R19, $F7  
LDI XL, $03  
LDI XH, $01  
RCALL DISPLAY_7SEG  
RCALL SET_UP_TIMER  
RJMP START  
  
CONFIG:  
CLR R16  
OUT DDRA, R16 ;PortA là input  
SER R16  
OUT DDRB, R16 ;PortB là output
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
OUT PORTA, R16 ;Pull-up
LDI R16, $03 ;PortD chan 0 va 1 la output
OUT DDRD, R16
CBI PORTD, 0 ;Khoa chot
CBI PORTD, 1 ;Khoa chot
LDI XL, $00
LDI XH, $01
RET

CACULATE:
LDI R19, $09
MUL DONVI, R19
MOVW R20, R0 ;Ket qua luu vao R21:R20
PUSH TRAM ;Dung sau
PUSH CHUC ;Dung sau
CALL SAVE_TRAM_CHUC_DONVI ;Sau dong nay ta se duoc hang
don vi cua ket qua
STS $0100, DONVI ;Luu hang don vi vao o
nho nay

MOV SONHO, CHUC ;Hang chuc o kqua
truoc chinh la so nho cua phep nhan lan sau
POP CHUC ;Lay hang chuc cua
kqua nhap tu portA ra lai
MUL CHUC, R19
MOVW R20, R0
ADD R20, SONHO
CALL SAVE_TRAM_CHUC_DONVI ;Sau dong nay se duoc hang chuc
cua kqua
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

STS \$0101, DONVI vi	;Luon luu vao la hang don
MOV SONHO, CHUC truoc chinh la so nho cua phep nhan lan sau	;Hang chuc o kqua
POP TRAM kqua nhap tu portA ra lai	;Lay hang tram cua
MUL TRAM, R19	
MOVW R20, R0	
ADD R20, SONHO	
CALL SAVE_TRAM_CHUC_DONVI va hang nghin cua kqua	;Sau dong nay se duoc hang tram
STS \$0102, DONVI	
STS \$0103, CHUC	
RET	
SAVE_TRAM_CHUC_DONVI:	
MOV R18, R20	
LDI R21, 10	
CLR R22	
L1:	
INC R22	;Div10 lan 1
SUB R18, R21	
BRCC L1	
DEC R22	
ADD R18, R21	
PUSH R18	;Cat hang don vi
MOV R18, R22	;Lay phan nguyen chia 10 lan 2
CLR R22	

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

L2:

```
INC R22           ;Div10 lan 2
SUB R18, R21
BRCC L2
DEC R22
ADD R18, R21
PUSH R18          ;Cat hang chuc
MOV R18, R22 ;Lay phan nguyen chia 10 lan 3
CLR R22
```

L3:

```
INC R22           ;Div10 lan 3
SUB R18, R21
BRCC L3
DEC R22
ADD R18, R21
PUSH R18          ;Cat hang tram
POP TRAM          ;Hang tram
POP CHUC          ;Hang chuc
POP DONVI         ;Hang don vi
RET
; input data R18 -- choose led to display
DISPLAY_7SEG:
PUSH R18
ANDI R18, $0F
OUT PORTB, R18      ; enable 573 to choose led
;Turn off led
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
SBI PORTD, 1
NOP
CBI PORTD, 1
LDI ZH, HIGH(TAB << 1)      ; look-up Led7seg
LDI ZL, LOW(TAB << 1)
LD R15, X
ADD ZL, R15
LPM R17, Z
OUT PORTB, R17
SBI PORTD, 0
NOP
CBI PORTD, 0
OUT PORTB, R19
SBI PORTD, 1
NOP
CBI PORTD, 1
POP R18
RET
----- USE TIMER0 TO DELAY

SET_UP_TIMER: ; 1/200 = 0.05 s = 40 000 (MCs) => use CLK/256 divider
with N = 156 pulses
LDI R16, 0
OUT TCCR0A, R16          ; timer 0 mod NOR
OUT TCCR0B, R16          ; stop timer

DELAY:
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
LDI R16, $64          ; input (-n)
OUT TCNT0, R16        ; count 156 pulses
LDI R16, $04          ; use CLK/256 prescaler
OUT TCCR0B, R16       ; start timer

WAIT:
    SBIS TIFR0, TOV0   ; check if TOV = 1
    RJMP WAIT          ; continue counting
    SBI TIFR0, TOV0    ; reset TOV0
    LDI R16, 0

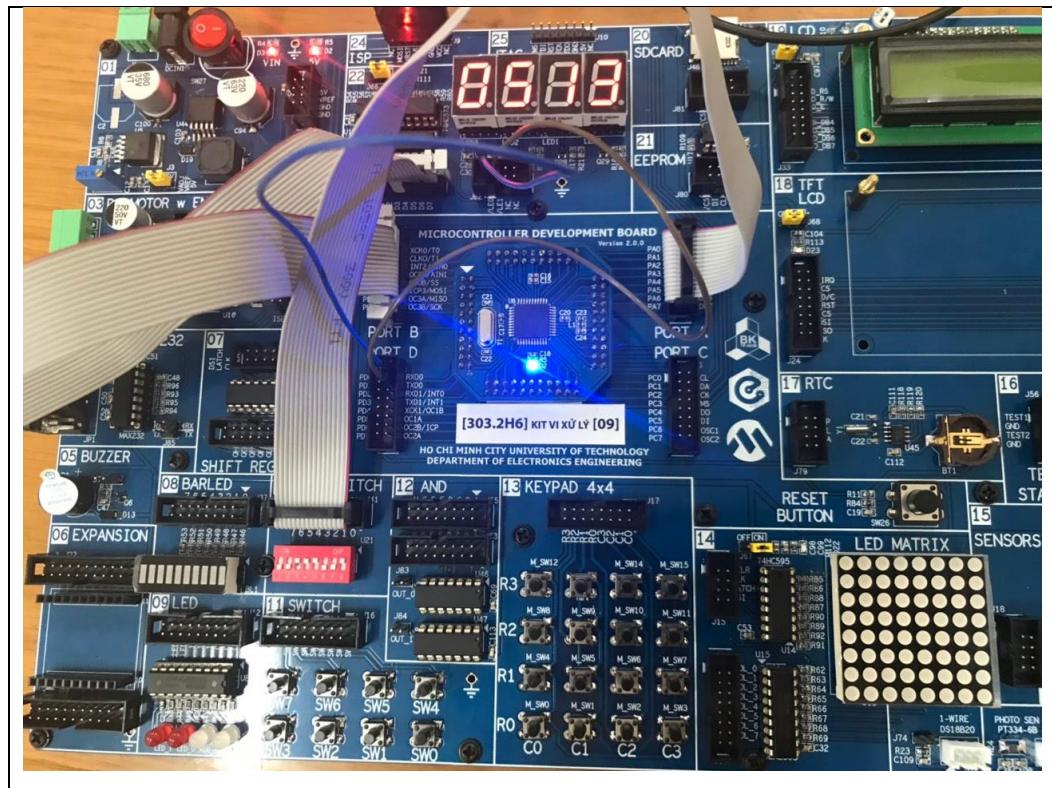
    OUT TCCR0B, R16    ; stop timer
    RET

TAB:
    .DB 0XC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8, 0X80, 0X90,
    0X88, 0X83
    .DB 0XC6,0XA1,0X86,0X8E
```

BÁO CÁO

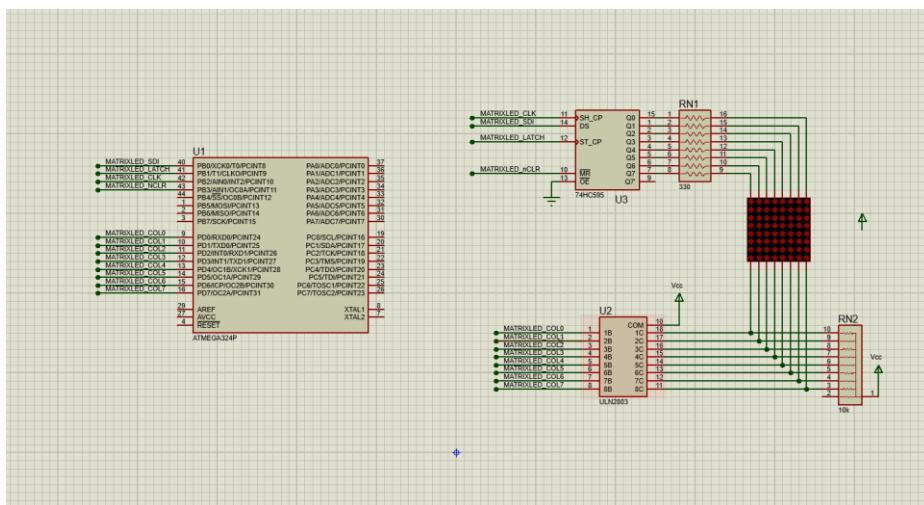
Nhóm môn học:

Nhóm:
Môn thí nghiệm:



BÀI 3

1. Trả lời các câu hỏi
 - a. Mô tả kết nối trên kit



BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

- b. Để có tần số quét 25Hz thì một cột LED sáng trong bao lâu?
Mỗi LED sáng trong 0,625ms
- c. Sự khác nhau khi quét ở tần số 25Hz và 125Hz
Trên kit thí nghiệm không thấy rõ sự khác nhau ở 2 trường hợp này
2. Mã nguồn chương trình với chú thích

```
.EQU OUTPORT=PORTB
.EQU IOSETB=DDRB
.EQU SCK=0
.EQU SDA=1
.EQU STCP=2
.EQU SHIFT=PORTA
.EQU SHIFT_DR=DDRA

.ORG 0
//Set chế độ cho các port
LDI R16,0x07
OUT SHIFT_DR,R16 ;SCK=0
CBI SHIFT,SCK ;SDA=0
CBI SHIFT,SDA
LDI R16,0xFF
OUT IOSETB,R16
START:
LDI ZH,HIGH(TABLE_A<<1) ;Z trả địa chỉ đầu bảng tra
LDI ZL,LOW(TABLE_A<<1)
LDI R19,8 ;đếm 8 lần quét
LDI R18,0x01 ;mã quét cột
LOOP:
CLR R16
OUT OUTPORT,R16 ;xóa toàn bộ
LPM R17,Z+ ;Lấy mã ký tự
RCALL SHO_8 ;xuất ký tự
OUT OUTPORT,R18 ;xuất mã quét
RCALL DELAY_5MS
CLC ;xóa cờ C chuẩn bị quay
ROL R18 ;quay trái tạo mã quét cột
kế tiếp
DEC R19 ;đếm số lần quét cột
BRNE LOOP
RJMP START

TABLE_A: .DB 0x00,0x7E,0x11,0x11,0x11,0x7E,0x00,0x00
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

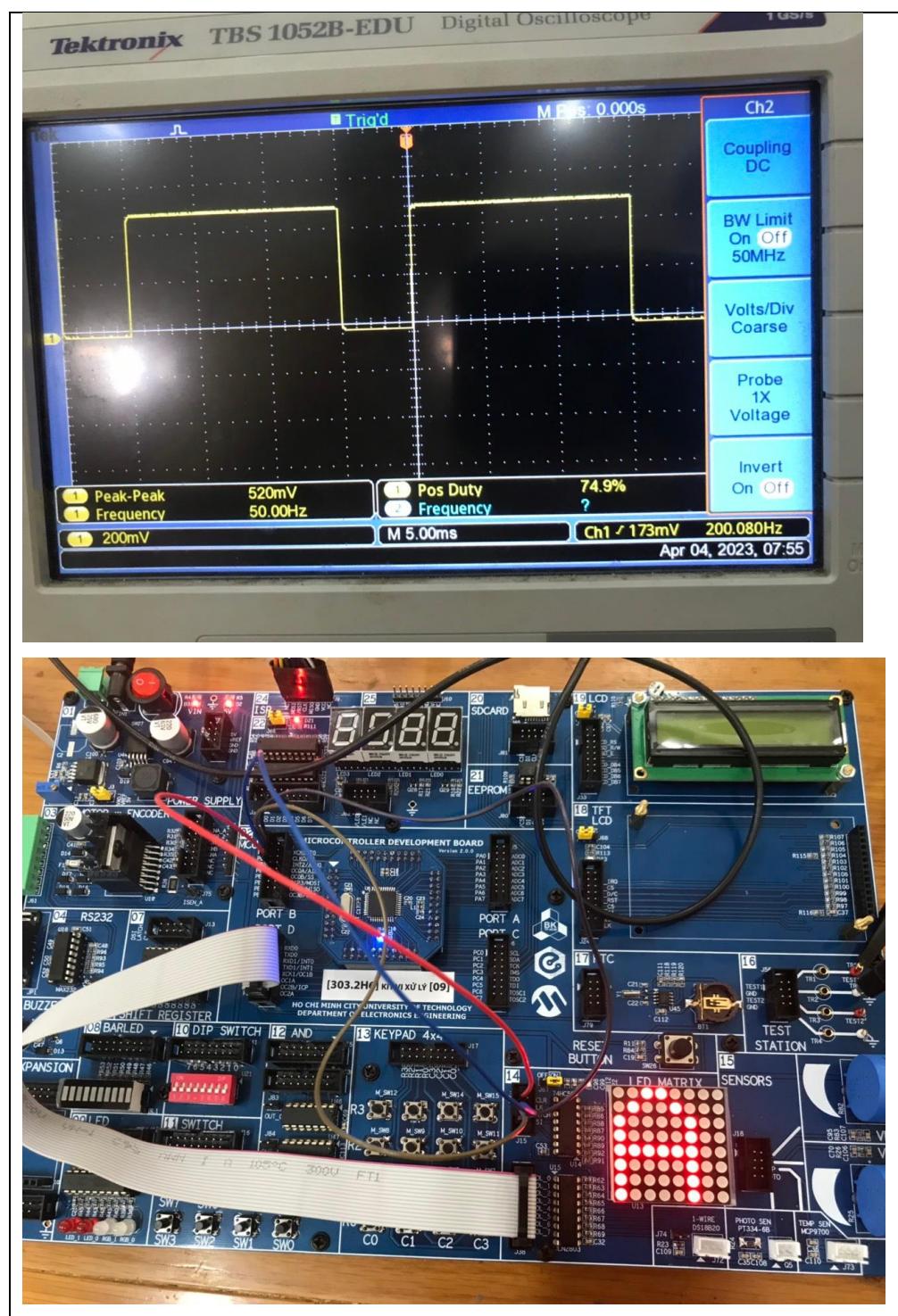
```
DELAY_5MS:
PUSH R16
DELAY:
LDI      R16,HIGH(-5000)
STS      TCNT1H,R16
LDI      R16,LOW(-5000)
STS      TCNT1L,R16
LDI      R16,$00          ;Chọn mode NOR, N=1
STS      TCCR1A,R16
LDI      R16,$01
STS      TCCR1B,R16
LOOP1:
SBIS    TIFR1,TOV1        ;Chờ cờ TOV0 = 1 báo Timer1 tràn
RJMP    LOOP1             ;Xóa cờ TOV0 về 0
SBI      TIFR1,TOV1
LDI      R16,$00
STS      TCCR1B,R16
POP    R16
RET

;SHO_8 dịch trái 8bit dát cất trong R17 ra ngõ SDA
;Input: R17
;Output: SDA nối tiếp MSB trước
SHO_8:
LDI      R16,9           ;đếm 9 lần dịch
SH_LOOP:
ROL      R17              ;quay trái qua C byte thấp
C=b7,b0=C
BRCC   BIT_0              ;C=0 nhảy đến BIT_0
SBI      SHIFT,SDA         ;dịch bit7=1 byte cao ra SDA
RJMP    NEXT
BIT_0:
CBI      SHIFT,SDA         ;dịch bit7=0 byte cao ra SDA
NEXT:
SBI      SHIFT,SCK
CBI      SHIFT,SCK
DEC    R16
BRNE   SH_LOOP
SBI      PORTA, STCP ;tạo xung dắt xuất
CBI      PORTA, STCP
RET
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:



BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

LAB 3-1

GIAO TIẾP SERIAL PORT, EEPROM, RTC

MỤC TIÊU:

- Hiểu và sử dụng được các ngoại vi UART, I2C, SPI
- Hiểu cách giao tiếp với RTC, EEPROM

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 7, 9, 11
- Atmel-2505-Setup-and-Use-of-AVR-Timers_ApplicationNote_AVR130.pdf

BÀI 1

- a) Kết nối chân TxD và RxD của UART0 vào vào tín hiệu UART_TxD0 và UART_RxD0 trên header J85 ở khối UART.
- b) Kết nối dây USB-Serial vào kit thí nghiệm
- c) Setup chương trình Hercules với baudrate 9600, 8 bit data, no parity, 1 stop, no handshake.
- d) Sử dụng các ví dụ mẫu trong tài liệu thí nghiệm, viết chương trình khởi động UART0 với các thông số như trên, chờ nhận một byte từ UART0 và phát ngược lại UART0.
- e) Dùng Hercules truyền một ký tự xuống kit và quan sát các dữ liệu nhận được để kiểm tra hoạt động chương trình.

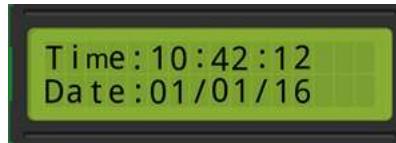
(Lưu ý: tần số xung clock cho CPU trên kit thí nghiệm là 8Mhz)

BÀI 2

- a) Kết nối các tín hiệu SDA và SCL của AVR vào các tín hiệu tương ứng trên module RTC.
Kết nối 1 chân port vào tín hiệu MFP. Kết nối LCD 16x2 vào 1 port của AVR
- b) Viết chương trình con khởi động RTC với thời gian hiện hành, cấu hình xung MFP tần số 1Hz. Sau đó cứ mỗi cạnh lên của MFP, đọc các giá trị ngày tháng năm giờ phút giây của RTC và cập nhật lên LCD

LAB 3-1

GIAO TIẾP SERIAL PORT, EEPROM, RTC



- c) Biên dịch chương trình và quan sát LCD để kiểm tra chương trình.

BÀI 3

- Kết nối các tín hiệu MOSI, SCK của port SPI từ AVR đến tín hiệu SDI và CLK của khối thanh ghi dịch. Kết nối 2 chân port khác vào tín hiệu nCLR và LATCH. Kết nối ngõ ra của thanh ghi dịch vào Bar LED
- Kết nối các tín hiệu UART như ở bài 1.
- Viết chương trình nhận 1 giá trị từ UART và xuất ra Bar Led sử dụng SPI.

BÀI 4

- Kết nối các tín hiệu MOSI, MISO, SCK của port SPI từ AVR các tín hiệu tương ứng trên header J80. Kết nối 1 chân port khác vào tín hiệu nCS.
- Kết nối các tín hiệu UART như ở bài 1.
- Kết nối 1 port vào Bar LED.
- Viết chương trình đếm số ký tự nhận được từ UART và xuất ra Bar Led, cứ mỗi lần có 1 byte nhận được, số đếm tăng lên 1 và được ghi vào EEPROM. Khi vi xử lý mất điện và có lại, số đếm được đọc ra từ EEPROM và lấy làm giá trị bắt đầu.

BÀI 5

- Kết nối các tín hiệu UART như ở bài 1.
- Kết nối 1 port vào Bar LED.
- Viết chương trình đếm số ký tự nhận được từ UART và xuất ra Bar Led, cứ mỗi lần có 1 byte nhận được, số đếm tăng lên 1 và được ghi vào EEPROM nội của AVR. Khi vi xử lý mất điện và có lại, số đếm được đọc ra từ EEPROM nội và lấy làm giá trị bắt đầu.

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

BÀI 1

1. Trả lời các câu hỏi
 - a. Với tần số là 8Mhz, baudrate thực tế sẽ sai lệch với mong muốn là 9600 như thế nào?
 - Sai số 0.16%, baudrate thực tế là 9615,385
 - b. Cờ UDRE dùng để làm gì?
 - Cờ báo khi băng 1 thì dữ liệu trong bộ đếm phát được chuyển đến thanh ghi dịch phát, khi cờ băng 0 thì không nên ghi dữ liệu
 - c. Sự khác nhau giữa hardware UART và software UART (bit-banging UART)
 - Hardware UART: phần cứng được tích sẵn trong chip vi điều khiển, không tồn tại nguyên CPU, tốc độ truyền nhanh hơn và ổn định hơn so với software
 - Software UART: Sử dụng CPU của vi điều khiển, tốc độ truyền chậm hơn, không yêu cầu phần cứng UART
 - d. Chân TxD0 và chân RxD0 của UART0 là chân port nào?
 - Là chân PD1 và PD0 của port D
 - e. Atmega324 có bao nhiêu phần cứng UART?
 - Có hai phần cứng UART0 và UART1
2. Mã nguồn chương trình với chú thích

```
.DEF DATA_RX = R18
.DEF DATA_TX = R19

.ORG 0
RJMP MAIN
.ORG $40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

CALL USART_INIT
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
CLR DATA_RX
CLR DATA_TX
START:
    CALL USART_RECEIVER_CHAR
    MOV DATA_TX, DATA_RX
    CALL USART_SEND_CHAR
    RJMP START

USART_INIT:
    LDI R16, (1<<RXEN0) | (1<<TXEN0)
    STS UCSR0B, R16
    LDI R16, $00
    STS UBRR0H, R16
    LDI R16, 51
    STS UBRR0L, R16

    LDI R16, (1<<UCSZ01) | (1<<UCSZ00)
    STS UCSR0C, R16
    RET

USART_SEND_CHAR:
    PUSH R17
    WAIT_USART_SEND:
        LDS R17, UCSR0A
        SBRS R17, UDRE0
        RJMP WAIT_USART_SEND
        STS UDR0, DATA_TX
        POP R17
    RET

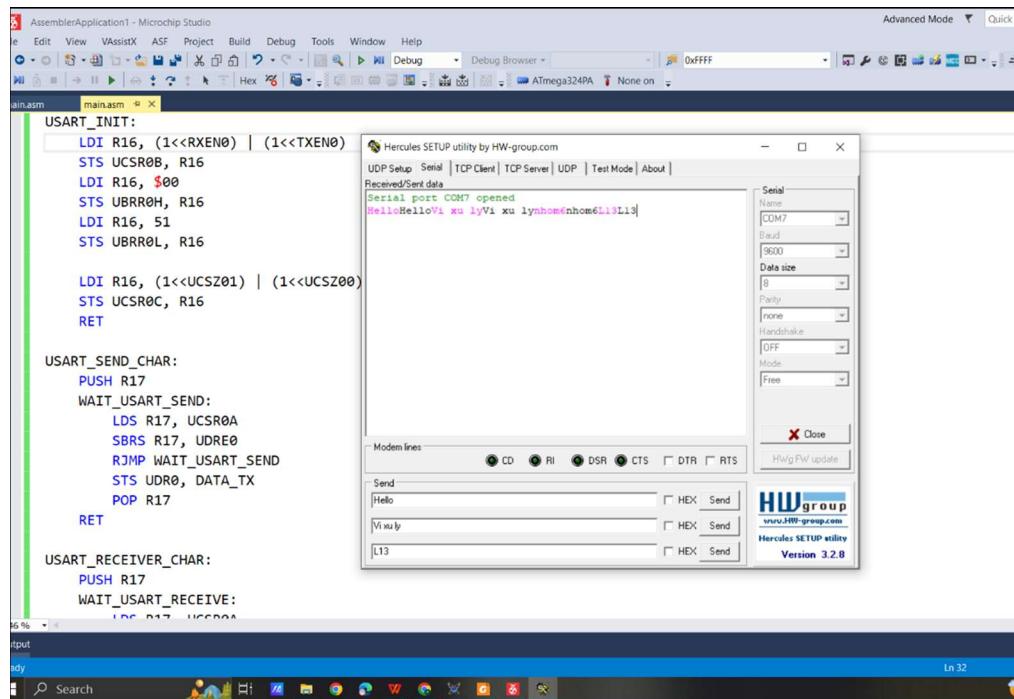
USART_RECEIVER_CHAR:
    PUSH R17
    WAIT_USART_RECEIVE:
        LDS R17, UCSR0A
        SBRS R17, RXC0
        RJMP WAIT_USART_RECEIVE
        LDS DATA_RX, UDR0
    POP R17
    RET
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 2

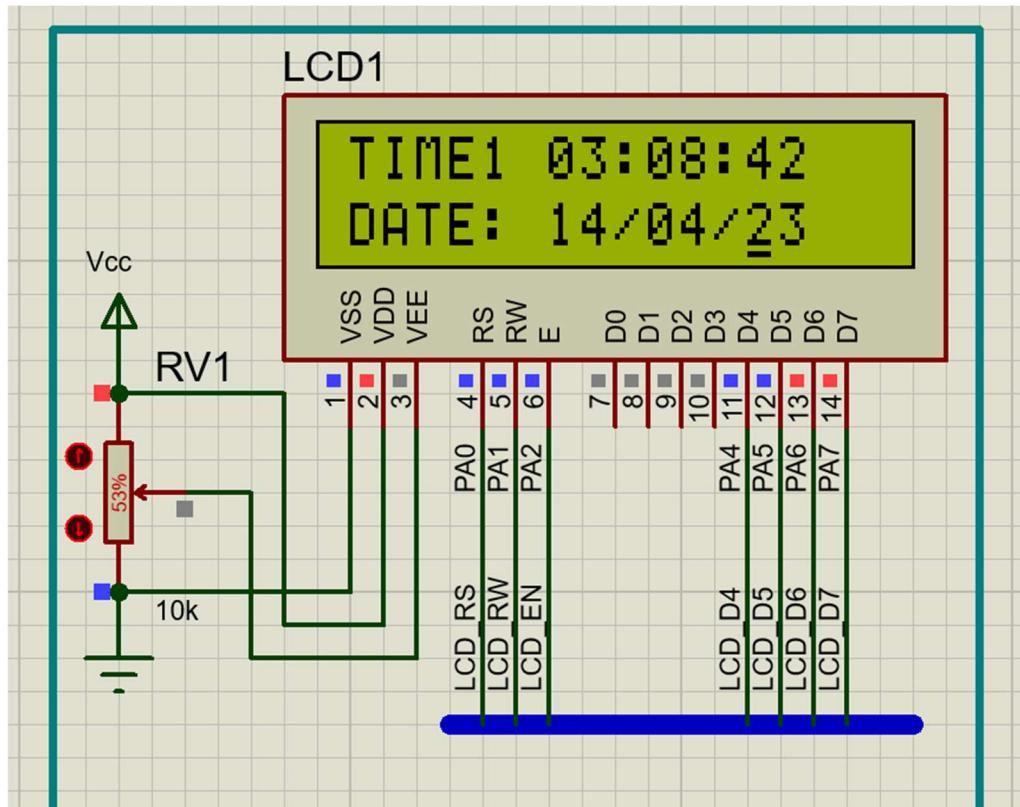
1. Trả lời các câu hỏi
 - a. Các chân SCL, SDA là chân nào của AVR?
SCL – chân PC0, SDA – chân PC1
 - b. Vẽ hình mô tả kết nối trong bài thí nghiệm

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



2. Mã nguồn và chú thích

```
.DEF      REG_FLAG=R19
.DEF      COUNT=R20
.DEF      NUM_MAX=R21          ; bien dem
.DEF      NUM_MIN=R22          ; bien max
.DEF      POS_CRS=R23          ; bien min
.EQU     LCD=PORTA            ; bien vi tri con tro hien thi
.EQU     LCD_DR=DDRA           ; PORTA hien thi
.EQU     CONT=PORTD             ;PORTD dieu khien
.EQU     CONT_DR=DDRD           ;
.EQU     CONT_IN=PIND           ;
.EQU     SW_FLG=0
.EQU     RS=0                  ;bit RS
.EQU     RW=1                  ;bit RW
.EQU     E=2                  ;bit E
.EQU     SCL=0                 ;ki hieu chan SCL
.EQU     SDA=1                 ;ki hieu chan SDA
.EQU     SW1=0                 ;ki hieu chan SW1
.EQU     SW2=1                 ;ki hieu chan SW2
.EQU     ST0=7                 ;bit cho phep OSC RTC
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
.EQU      VBATEN=3           ;bit cho phep nguon du phong
.EQU      NULL=$00           ;ma ket thuc chuoi ky tu?
.EQU      CTL_BYTE=0B11011110 ;byte dieu khien truy xuat
.EQU      RTC_BUF=0X200
.ORG      0

        RJMP      MAIN
        .ORG      0X40
MAIN:   LDI       R16,HIGH(RAMEND)    ;dua stack len vung cao
        OUT      SPH,R16
        LDI       R16,LOW(RAMEND)
        OUT      SPL,R16
        LDI       R16,0xFF
        OUT      LCD_DR,R16
        LDI       R16,0X00
        OUT      LCD,R16
        CBI       CONT_DR,SW1  ;chon SW1 input
        SBI       CONT,SW1   ;dien tro keo len chan SW1
        CBI       CONT_DR,SW2  ;chon SW2 input
        SBI       CONT,SW2   ;dien tro keo len chan SW2
        LDI       R16,250    ;delay 25ms
        RCALL    DELAY_US   ;ctc delay 100?sxR16
        LDI       R16,250    ;delay 25ms
        RCALL    DELAY_US   ;ctc delay 100?sxR16
        CBI       LCD,RS    ;RS=0 ghi lenh
        LDI       R17,$30    ;ma lenh=$30 lan 1,RS=RW=E=0
        RCALL    OUT_LCD4   ;ctc ghi ra LCD
        LDI       R16,42    ;delay 4.2ms
        RCALL    DELAY_US
        CBI       LCD,RS
        LDI       R17,$30    ;ma lenh=$30 lan 2
        RCALL    OUT_LCD4
        LDI       R16,2     ;delay 200us
        RCALL    DELAY_US
        CBI       LCD,RS
        LDI       R17,$30    ;ma lenh=$30 lan 3
        RCALL    OUT_LCD4
        LDI       R16,1     ;delay 100?s
        RCALL    DELAY_US
        CBI       LCD,RS
        LDI       R17,$20    ;ma lenh=$20
        RCALL    OUT_LCD4
        LDI       R18,$28    ;Function set 2 dong font
5x8, mode 4 bit
        LDI       R19,$01    ;Clear display
        LDI       R20,$0C    ;display on,con tro off
        LDI       R21,$06    ;Entry mode set dich phai con
tro,
        RCALL   INIT_LCD4   ;ctc khai dong LCD 4 bit
        RCALL   TWI_INIT

START:
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
LDI      REG_FLAG,0          ;xoá cac co bao
LDI      R16,1
RCALL   DELAY_US
CBI      LCD,RS            ;RS=0 ghi lenh
LDI      R17,$01            ;xoá man hinh
RCALL   OUT_LCD
LDI      R16,20              ;cho 2ms sau lenh Clear display
RCALL   DELAY_US
LDI      R17,$80            ;con tro bat dau sau dong 1
RCALL   CURS_POS           ;xuat lenh ra LCD
LDI      ZH,HIGH(MSG1<<1)
LDI      ZL,LOW(MSG1<<1)
RCALL   MSG_DISP            ;ghi MSG1 ra LCD
LDI      R17,$C0            ;con tro bat dau sau dong 2
RCALL   CURS_POS           ;xuat lenh ra LCD
LDI      ZH,HIGH(MSG2<<1);Z tra cuu bang tra MSG2
LDI      ZL,LOW(MSG2<<1)
RCALL   MSG_DISP            ;ghi MSG2 ra LCD
;-----
;bit SQWEN=1,RS2:0=000 cho dao dong 1Hz
;xuat ra chon MFP
;-----
RCALL   TWI_START           ;phat xung START
LDI      R17,(CTL_BYTE|0X00);truy xuat ghi RTC_TCCR
RCALL   TWI_WRITE            ;ghi RTC+W
LDI      R17,0X07            ;dia chi thanh ghi Control
RCALL   TWI_WRITE
LDI      R17,0B01000000        ;MFP xuat xung 1Hz
RCALL   TWI_WRITE
RCALL   TWI_STOP
;-----
;cac thanh ghi 0x00-0x06 RTC
;-----
START1:
LDI      XH,HIGH(RTC_BUF);X tro sau buffer RTC
LDI      XL,LOW(RTC_BUF)
LDI      COUNT,7
RCALL   TWI_START           ;phat xung START
LDI      R17,(CTL_BYTE|0X00);truy xuat ghi RTC_TCCR
RCALL   TWI_WRITE            ;ghi RTC+W
LDI      R17,0X00            ;dia chi thanh ghi 0x00
RCALL   TWI_WRITE
RCALL   TWI_START           ;phat xung START
LDI      R17,(CTL_BYTE|0X01);truy xuat RTC_TCCR
RCALL   TWI_WRITE            ;ghi RTC+R
RTC_RD:
RCALL   TWI_READ
ST      X+,R17
DEC    COUNT
BRNE   RTC_RD
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
RCALL TWI_NAK
RCALL TWI_STOP
;-----
;Hien thi thu gio:phut:giay
;-----
START2:
LDI R17,$0C          ;xoa con tro
CBI LCD,RS
LDI R16,1            ;cho 100us
RCALL DELAY_US
RCALL OUT_LCD
LDI XH,HIGH(RTC_BUF+3);X tro buffer RTC
LDI XL,LOW(RTC_BUF+3)
LDI R17,$84           ;con tro bat dau o dong 1
RCALL CURS_POS        ;xuat lenh ra LCD
LD R17,X
ANDI R17,0X07
LDI R18,0X30           ;chuyen sang ma ASCII
ADD R17,R18
SBI LCD,RS
LDI R16,1            ;cho 100us
RCALL DELAY_US
RCALL OUT_LCD          ;hien thi ra LCD
LDI R17,0X20
SBI LCD,RS
LDI R16,1            ;cho 100us
RCALL DELAY_US
RCALL OUT_LCD          ;hien thi ra LCD
LDI COUNT,3
LDI R17,$86           ;con tro bat dau o dong 1 vi tri gio
RCALL CURS_POS        ;xuat lenh ra LCD

DISP_NXT1:
LD R17,-X              ;lay data
CPI COUNT,1             ;data=sec
BRNE D_NXT      ;khac,hien thi tiep
CBR R17,(1<<ST0)    ;xoa bit ST
D_NXT:
RCALL NUM_DISP
DEC COUNT
BREQ QUIT1
LDI R17,':'
SBI LCD,RS
LDI R16,1            ;cho 100us
RCALL DELAY_US
RCALL OUT_LCD          ;hien thi ra LCD
RJMP DISP_NXT1
;-----
;Hien thi ngay/thang/nam
;-----
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
QUIT1:
    LDI    XH,HIGH(RTC_BUF+4);X tro buffer RTC ngay
    LDI    XL,LOW(RTC_BUF+4)
    LDI    COUNT,3
    LDI    R17,$C6           ;con tro bat dau o dong 2 vi tri ngay
    RCALL CURS_POS          ;xuat lenh ra LCD

DISP_NXT2:
    LD     R17,X+
    RCALL NUM_DISP
    DEC   COUNT
    BREQ  SW_CHK
    LDI   R17,'/'
    SBI   LCD,RS
    LDI   R16,1             ;cho 100us
    RCALL DELAY_US
    RCALL OUT_LCD           ;hien thi ra LCD
    RJMP  DISP_NXT2

;-----
;RTC
;-----

SW_CHK:
    RCALL GET_SW
    SBRS REG_FLAG,SW_FLG
    RJMP START1
    CPI   R17,1
    BRNE SW_CHK
    LDI   R17,$0E           ;hien thi con tro
    CBI   LCD,RS
    LDI   R16,1             ;cho 100us
    RCALL DELAY_US
    RCALL OUT_LCD           ;xuat lenh ra LCD

RTC_SET:
    CPI   COUNT,0
    BRNE HR_CHK      ;khac,kiem tra gio
    LDI   XH,HIGH(RTC_BUF+3);X tro buffer RTC thu
    LDI   XL,LOW(RTC_BUF+3)
    LDI   NUM_MAX,7
    LDI   NUM_MIN,1
    LDI   POS_CRS,$84        ;dat con tro vi tri thu
    RCALL SET_NUM
    LD    R17,X
    SBR  R17,(1<<VBATEN)  ;cho phep nguon backup
    ST   X,R17
    RJMP RTC_SET

HR_CHK:
    CPI   COUNT,1
    BRNE MI_CHK      ;khac,kiem tra phut
    LDI   XH,HIGH(RTC_BUF+2);X tro buffer RTC gio
    LDI   XL,LOW(RTC_BUF+2)
    LDI   NUM_MAX,0X23
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
LDI    NUM_MIN,0
LDI    POS_CRS,$86          ;dat con tro vi tri gio
RCALL SET_NUM
RJMP  RTC_SET

MI_CHK:
CPI    COUNT,2
BRNE  SEC_CHK           ;khac,kiem tra giay
LDI    XH,HIGH(RTC_BUF+1);X tro buffer RTC phut
LDI    XL,LOW(RTC_BUF+1)
LDI    NUM_MAX,0X59
LDI    NUM_MIN,0
LDI    POS_CRS,$89          ;dat con tro vi tri phut
RCALL SET_NUM
RJMP  RTC_SET

SEC_CHK:
CPI    COUNT,3
        DAT_CHK           ;kiem tra ngay
LDI    XH,HIGH(RTC_BUF);X tro buffer RTC giay
LDI    XL,LOW(RTC_BUF)
LDI    NUM_MAX,0X59
LDI    NUM_MIN,0
LDI    POS_CRS,$8C      ;dat con tro vi tri giay
RCALL SET_NUM
LD    R17,X
SBR   R17,(1<<ST0)  ;dat bit ST0=1 cho phep OSC
ST    X,R17
RJMP  RTC_SET

DAT_CHK:
CPI    COUNT,4
        MO_CHK            ;khac,kiem tra thang
LDI    XH,HIGH(RTC_BUF+4);X tro buffer RTC ngay
LDI    XL,LOW(RTC_BUF+4)
LDI    NUM_MAX,0X31
LDI    NUM_MIN,1
LDI    POS_CRS,$C6      ;con tro vi tri ngay
RCALL SET_NUM
RJMP  RTC_SET

MO_CHK:
CPI    COUNT,5
        YEA_CHK           ;khac,kiem tra nam
LDI    XH,HIGH(RTC_BUF+5);X tro buffer RTC thang
LDI    XL,LOW(RTC_BUF+5)
LDI    NUM_MAX,0X12
LDI    NUM_MIN,1
LDI    POS_CRS,$C9      ;con tro vi tri thang
RCALL SET_NUM

YEA_CHK:
CPI    COUNT,6
        EXIT_CHK          ;khac,thoat
LDI    XH,HIGH(RTC_BUF+6);X tro buffer RTC nam
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
LDI      XL,LOW(RTC_BUF+6)
LDI      NUM_MAX,0X99
LDI      NUM_MIN,1
LDI      POS_CRS,$CC ;con tro vi tri nam
RCALL  SET_NUM
RJMP   RTC_SET
;-----
;Luu cac gia tri cai dat vao RTCC
;-----
EXIT_CHK:
LDI      COUNT,7 ;luu vao RTCC
LDI      XH,HIGH(RTC_BUF);X tro buffer RTC
LDI      XL,LOW(RTC_BUF)
RCALL  TWI_START ;phat xung START
LDI      R17,(CTL_BYTE|0X00);truy xuat ghi RTC
RCALL  TWI_WRITE ;ghi RTC+W
LDI      R17,0X00 ;dia chi thanh ghi giay
RCALL  TWI_WRITE ;ghi dia chi TCCR
WR_RTC:
LD      R17,X+
RCALL  TWI_WRITE ;ghi TCCR
DEC    COUNT
BRNE  WR_RTC
RCALL  TWI_STOP
RJMP   START1
;-----
;-----
-- 
GET_SW:
CBR    REG_FLAG,(1<<SW_FLG);xoa co bao nhan SW
BACK0:
LDI    R16,50 ;kiem tra SW nhan 50 lan lien
tuc
WAIT0:
IN     R17,CONT_IN
ANDI   R17,(1<<SW1)|(1<<SW2);che bit SW1,SW2
CPI    R17,(1<<SW1)|(1<<SW2);kiem tra SW nhan?
BREQ   EXIT_SW ;khong nhan thoat
DEC    R16 ;co nhan tiep tuc
BRNE  WAIT0 ;
PUSH   R17
BACK1:
LDI    R16,50 ;kiem tra sw nha 50 lan lien tuc
WAIT1:
IN     R17,CONT_IN
ANDI   R17,(1<<SW1)|(1<<SW2)
CPI    R17,(1<<SW1)|(1<<SW2)
BRNE  BACK1
DEC    R16
BRNE  WAIT1
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
POP      R17          ;phuc hoi SW
CPI      R17,(1<<SW2) ;SW1=0 nhan, SW2=1 khong nhan
BRNE    SW2_CODE      ;khong phai kiem tra SW2
LDI      R17,1         ;gan gia tri SW1
RJMP    SET_FLG       ;bao co nhan SW
SW2_CODE:
CPI      R17,(1<<SW1) ;SW2=0 nh?n, SW1=1 khong nhan
BRNE    EXIT_SW       ;khong phai thoat
LDI      R17,2         ;gan gia tri SW2
SET_FLG:
SBR      REG_FLAG,(1<<SW_FLG);dat co bao nhan SW
EXIT_SW:
RET
;-----;
;SET_NUM cai dat cac gia tri thoi gian chon qua bien COUNT
;Nhan/nha SW1 thoat
;Nhan/nha SW2 cai dat gia tri
;Su dung R17,R18,ctc CURS_POS,GET_SW
;-----;
SET_NUM:
MOV      R17,POS_CRS
RCALL   CURS_POS
SW_CHK1:
RCALL   GET_SW
SBRS    REG_FLAG,SW_FLG;co SW nhan
RJMP    SW_CHK1        ;cho nhan SW
CPI     R17,1           ;SW1 nhan?
BREQ    EXIT_NUM       ;dong, thoat
CPI     R17,2           ;SW2 nhan?
BRNE    SW_CHK1        ;khac, doc lai SW
LD      R17,X           ;nap gia tri cai dat
CPI     COUNT,3         ;cai dat giay?
BRNE    DAY_CHK         ;khac, kiem tra ngay
CBR     R17,(1<<ST0) ;dong, xoa bit ST
RJMP    PRESET          ;tien hanh dat
DAY_CHK:
CPI     COUNT,0          ;cai dat ngay?
BRNE    PRESET          ;khac, tien hanh dat
ANDI    R17,0X07         ;loc lay data ngay
PRESET:
INC     R17             ;tang gia tri them 1
MOV     R18,R17          ;cat gia tri dat
ANDI    R17,$0F           ;che lay 4 bit thap
CPI     R17,$0A           ;gia tri<10
BRCS   NON_CR          ;dong, khong tran
LDI     R17,$06           ;hieu lenh BCD
ADD     R18,R17
NON_CR:
MOV     R17,R18          ;tra ss BCD dat ve R17
CP     R17,NUM_MAX       ;so sanh MAX
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
        BRCS      DISP          ;nho hon,hien thi
        BREQ      DISP          ;bang,hien thi?
        MOV       R17,NUM_MIN   ;lon hon,tra ve hon MIN
DISP:  ST       X,R17         ;cat so BCD dat vao buffer
        RCALL    NUM_DISP     ;hien thi so BCD dat
        RJMP     SET_NUM      ;tiep tuc dat
EXIT_NUM:
INC      COUNT         ;tang bien demm vi tri cai dat
        RET
;-----
NUM_DISP:
PUSH    R17
        SWAP      R17
        ANDI     R17,0X0F      ;che lay so BCD cao
        ORI      R17,0X30      ;chuyen sang ma ASCII
        SBI      LCD,RS
        LDI      R16,1         ;cho 100us
        RCALL   DELAY_US
        RCALL   OUT_LCD       ;hien thi gia tri
        POP      R17          ;phuc hoi data
        ANDI     R17,0X0F      ;che lay so BCD thap
        ORI      R17,0X30      ;chuyen sang ma ASCII
        SBI      LCD,RS
        LDI      R16,1         ;cho 100us
        RCALL   DELAY_US
        RCALL   OUT_LCD       ;hien thi gia tri?
        RET
;-----
;MSG_DISP  hien thi chuoi ky tu ket thuc bang ma NULL dat trong Flash
ROM
;Input: Z chua dia chi sau chuoi ky t?
;Output: hien thi chuoi ky tu ra LCD tai vi tri con tro hien hanh
;Su dung R16,R17,ctc DELAY_US,OUT_LCD
;-----
MSG_DISP:
LPM     R17,Z+          ;lay ma ASCII ky tu tu Flash ROM
        CPI      R17,NULL       ;kiem tra ky tu ket thuc
        BREQ    EXIT_MSG      ;ky tu NULL thoat
        LDI      R16,1          ;cho 100us
        RCALL   DELAY_US
        SBI      LCD,RS        ;RS=1 ghi data hien thu LCD
        RCALL   OUT_LCD       ;ghi ma ASCII ky tu ra LCD
        RJMP    MSG_DISP
EXIT_MSG:
RET
;-----
;CURS_POS dat con tro tai vi tri co dia chi trong R17
;Input: R17=$80 -$8F dong 1,$C0-$CF dong 2
;       R17= dia chi vi tri con tro
;Su dung R16,ctc DELAY_US,OUT_LCD
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
;-----  
CURS_POS:  
LDI      R16,1          ;cho 100us  
RCALL   DELAY_US  
CBI      LCD,RS         ;RS=0 ghi lenh  
RCALL   OUT_LCD  
RET  
;-----  
;INIT_LCD4 khai dong LCD ghi 4 byte ma lenh theo giao tiep 4 bit  
;Function set:R18=$28 2 dong font 5x8 giao tiep 4 bit  
;Clear display:R19=$01 xoa man hinh  
;Display on/off LCDrol:R20=$0C man hinh on,con tro off  
;Entry mode set:R21=$06 dich phai con tro ,d/c DDRAM tang 1 khi ghi data  
;RS=bit0=0,RW=bit1=0  
;-----  
INIT_LCD4:  
CBI      LCD,RS         ;RS=0: ghi lenh  
MOV     R17,R18          ;R18=Function set  
RCALL   OUT_LCD          ;ghi 1 byte data ra LCD  
MOV     R17,R19          ;R19=Clear display  
RCALL   OUT_LCD  
LDI      R16,20          ;cho 2ms sau lenh Clear display  
RCALL   DELAY_US  
MOV     R17,R20          ;R20=Display LCDrol on/off  
RCALL   OUT_LCD  
MOV     R17,R21          ;R21=Entry mode set  
RCALL   OUT_LCD  
RET  
;-----  
;OUT_LCD4 ghi ma lenh/data ra LCD  
;Input: R17 chua ma lenh/data 4 bit cao  
;-----  
OUT_LCD4:  
OUT    LCD,R17  
SBI    LCD,E  
CBI    LCD,E  
RET  
;-----  
;OUT_LCD ghi 1 byte ma lenh/data ra LCD  
;chia lam 2 lan ghi 4bit  
;Input: R17 chia ma lenh/data,R16  
;bit RS=0/1:lenh/data,bit RW=0:ghi  
;Su dung ctc OUT_LCD4  
;-----  
OUT_LCD:  
LDI      R16,1          ;cho 100us  
RCALL   DELAY_US  
IN      R16,LCD          ;doc PORT LCD  
ANDI   R16,(1<<RS)       ;loc bit RS
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
PUSH      R16
PUSH      R17
ANDI      R17,$F0          ;lay 4 bit cao
OR        R17,R16          ;ghep bit RS
RCALL    OUT_LCD4         ;ghi ra LCD
LDI       R16,1            ;cho 100us
RCALL    DELAY_US         ;-----
POP       R17              ;phuc hoi R17
POP       R16              ;phuc hoi R16
SWAP      R17              ;dao 4 bit
ANDI      R17,$F0          ;lay 4 bit thap chuyen thanh cao
OR        R17,R16          ;ghep bit RS
RCALL    OUT_LCD4         ;ghi ra LCD
RET

;-----  

;DELAY_US =R16x100us(Fosc=8Mhz)
;Input:R16 he so nhan thoi gian tra 1 lan 255
;-----  

DELAY_US:
    MOV      R15,R16          ;1MC nap data cho R15
    LDI      R16,200           ;1MC su dung R16
L1:   MOV      R14,R16          ;1MC nap data cho R14
L2:   DEC      R14             ;1MC
    NOP
    BRNE    L2               ;2/1MC
    DEC      R15              ;1MC
    BRNE    L1               ;2/1MC
    RET                 ;4MC
;-----  

;TWI_INIT khai dong cang TWI
;dat toc do tuyen=100Khz
;-----  

TWI_INIT:
    LDI      R17,8            ;toc do truyen SCL=100Khz
    STS      TWBR,R17
    LDI      R17,1
    STS      TWSR,R17          ;he so dat truoc=4
    LDI      R17,(1<<TWEN) ;cho phep TWI
    STS      TWCR,R17
    RET

;-----  

TWI_START:
    LDI      R17,(1<<TWEN)|(1<<TWSTA)|(1<<TWINT);cho phep TWI,START,xoa
TWINT
    STS      TWCR,R17
WAIT_STA:
    LDS      R17,TWCR          ;??c c? TWINT
    SBRS    R17,TWINT          ;ch? c? TWINT=1 bao truy?n xong
    RJMP


```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
RET
;-----
TWI_WRITE:
STS      TWDR,R17          ;ghi data
        LDI      R17,(1<<TWEN)|(1<<TWINT);cho phep TWI,xoa TWINT
        STS      TWCR,R17
WAIT_WR:
LDS      R17,TWCR          ;ch? c? TWINT=1 bao truy?n xong
        SBRS    R17,TWINT
        RJMP    WAIT_WR
        RET
;-----
TWI_READ:
LDI      R17,(1<<TWEN)|(1<<TWINT)|(1<<TWEA);cho phep TWI,xoa TWINT
        STS      TWCR,R17
WAIT_RD:
LDS      R17,TWCR          ;cho co TWINT=1 bao truyen xong
        SBRS    R17,TWINT
        RJMP    WAIT_RD
        LDS     R17,TWDR
        RET
;-----
TWI_NAK:
LDI      R17,(1<<TWEN)|(1<<TWINT);cho phep TWI,xoa TWINT,tra NAK
        STS      TWCR,R17
WAIT_NAK:
LDS      R17,TWCR          ;cho co TWINT=1 bao truyen xong
        SBRS    R17,TWINT
        RJMP    WAIT_NAK
        RET
;-----
TWI_STOP:
LDI      R17,(1<<TWEN)|(1<<TWSTO)|(1<<TWINT);cho phep TWI,xoa
TWINT,STOP
        STS      TWCR,R17
        RET
;-----
.ORG    0X200
MSG1:   .DB      "THU ",$00
MSG2:   .DB      "NGAY ",$00
```

BÀI 3

1. Trả lời các câu hỏi

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

Theo datasheet của 74HC595, tần số cao nhất của xung nhịp đưa vào 74595 là bao nhiêu?

Tần số cao nhất là khoảng 100Mhz

Với clock là 8Mhz thì SPI của Atmega328 có tốc độ cao nhất là bao nhiêu?

Vì tốc độ SPI không được lớn hơn $\frac{1}{4}$ tốc độ xung nhịp cho chip nên tốc độ SPI max = Fosc / 16 = $\frac{1}{2}$ MHz.

2. Mã nguồn và chú thích

```
.EQU DD_MOSI = 5
.EQU SS = 4
.EQU DD_SCK = 7
.EQU DD_MISO = 6
.EQU DDR_SPI = DDRB
.EQU PORT_SPI = PORTB
.DEF DATA_SPI_TX = R16
.DEF DATA_SPI_RX = R17
;-----UART-----
.DEF DATA_RX = R18
.DEF DATA_TX = R19

.ORG 0
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
START:
RCALL SPI_MASTER_INIT
RCALL USART_INIT
CLR DATA_SPI_TX
CLR DATA_SPI_RX
CLR DATA_TX
CLR DATA_RX
SBI PORT_SPI, SS
AGAIN:
RCALL USART_RECEIVER_CHAR
MOV DATA_SPI_TX, DATA_RX
CBI PORT_SPI, SS
RCALL SPI_MASTER_TRANSMIT
SBI PORT_SPI, SS
RJMP AGAIN
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6
Môn thí nghiệm: Vi xử lý

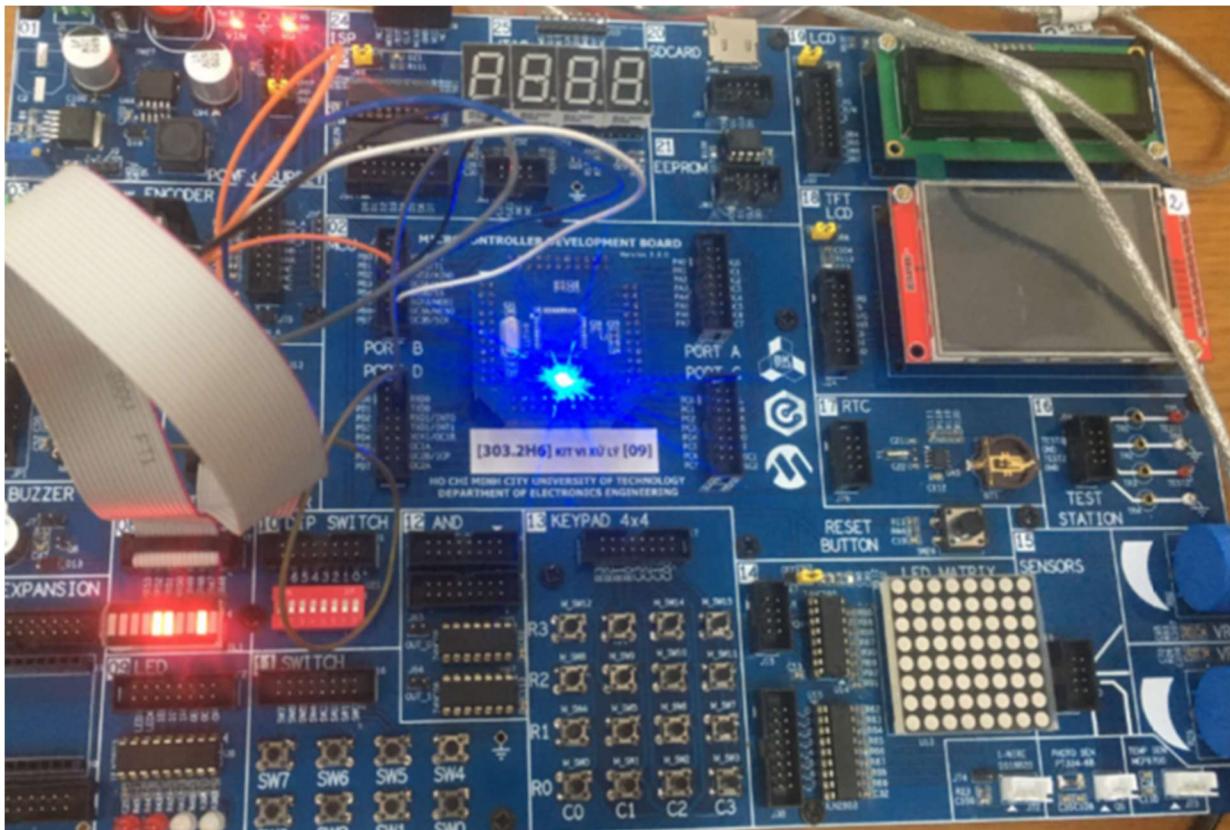
```
SPI_MASTER_INIT:  
    LDI R17, (1<<DD_MOSI)|(1<<DD_SCK)|(1<<SS)  
    OUT DDR_SPI, R17  
    SBI PORT_SPI, DD_MISO  
    LDI R17, (1 << SPE0) | (1 << MSTR0) | (1 << SPR00)  
    OUT SPCR0, R17  
    RET  
  
SPI_MASTER_TRANSMIT:  
    PUSH R17  
    OUT SPDR0, DATA_SPI_TX  
    WAIT_MASTER_TRANSMIT:  
        IN R17, SPSR0  
        SBRS R17, SPIF0  
        RJMP WAIT_MASTER_TRANSMIT  
    IN DATA_SPI_RX, SPDR0  
    POP R17  
    RET  
  
USART_INIT:  
    LDI R16, (1<<RXEN0) | (1<<TXEN0)  
    STS UCSR0B, R16  
    LDI R16, $00  
    STS UBRRLH, R16  
    LDI R16, 51  
    STS UBRRL, R16  
    LDI R16, (1<<UCSZ01) | (1<<UCSZ00)  
    STS UCSR0C, R16  
    RET  
  
USART_RECEIVER_CHAR:  
    PUSH R17  
    WAIT_USART_RECEIVE:  
        LDS R17, UCSR0A  
        SBRS R17, RXC0  
        RJMP WAIT_USART_RECEIVE  
        LDS DATA_RX, UDR0  
    POP R17  
    RET
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý



BÀI 4

1. Trả lời các câu hỏi
 - a. Dung lượng của EEPROM 25AA1024 là bao nhiêu?
 - Dung lượng là 1 megabit tương đương 128 kilobyte (KB)
 - b. Theo datasheet, tần số nhanh nhất của xung CK đưa vào EEPROM này là bao nhiêu?
 - Tần số hoạt động tối đa là 20MHz
2. Mã nguồn và chú thích

```
.ORG 0
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
;USE UART0, Baudrate 9600, 8 bit data, no parity, 1 stop  
;USE SPI, MASTER TRANSMIT  
  
.DEF DATA_TX      = R16  
.DEF DATA_RX      = R25  
.DEF DATA_SPI_TX  = R18  
.DEF DATA_SPI_RX  = R19  
.DEF COUNT         = R21  
  
;  
-----SPI-----  
.EQU DD_SS        = 4  
.EQU DD_MOSI      = 5  
.EQU DD_MISO      = 6  
.EQU DD_SCK       = 7  
.EQU DDR_SPI      = DDRB  
.EQU PORT_SPI     = PORTB  
  
;  
-----EEPROM-----  
  
.EQU WREN         = $06 ;SET THE WRITE ENABLE LATCH  
.EQU WRDI         = $04 ;RESET THE WRITE ENABLE LATCH  
.EQU RDSR         = $05 ;READ STATUS REGISTER  
.EQU WRSR         = $01 ;WRITE STATUS REGISTER  
.EQU READ          = $03 ;READ DATA FROM MEMORY  
.EQU WRITE         = $02 ;WRIE DATA TO MEMORY  
.EQU PE            = $42 ;PAGE ERASE
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
.EQU MEMADD3      =0X00 ; DIA CHI BO NHO BYTE 3
.EQU MEMADD21     =0X0100 ; DIA CHI BO NHO BYTE 21
.EQU WIP          = 0

;INIT
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

RCALL SPI0_MASTER_INIT
RCALL UART0_INIT
CLR DATA_RX
CLR DATA_TX
CLR DATA_SPI_TX
CLR DATA_SPI_RX
LDI COUNT, 0
SBI PORT_SPI, DD_SS
SER R16
OUT DDRA, R16
//    RCALL EEPROM_READ
//MOV COUNT, DATA_SPI_RX

MAIN:
RCALL EEPROM_DEL
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6
Môn thí nghiệm: Vi xử lý

```
RCALL EEPROM_WRITE
RCALL EEPROM_READ
OUT PORTA, DATA_SPI_RX
RCALL UART0_RX
INC COUNT
RJMP MAIN
SPI0_MASTER_INIT:
PUSH R16
LDI R16, (1 << DD_MOSI) | (1 << DD_SCK) | (1 << DD_SS)
OUT DDR_SPI, R16
CLR R16
OUT SPCR0, R16
LDI R16, (1 << SPE0) | (1 << MSTR0) | (1 << SPR00)
OUT SPCR0, R16
LDI R16, (1 << SPI2X0)
OUT SPSR0, R16
POP R16
RET

SPI_MASTER_TRANSMIT:
PUSH R20
OUT SPDR0, DATA_SPI_TX
WAIT_TRANSMIT:
IN R20, SPSR0
SBRS R20, SPIF0
RJMP WAIT_TRANSMIT
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
IN DATA_SPI_RX, SPDR0
POP R20

RET

EEPROM_WRITE:
LDI DATA_SPI_TX, WREN
CBI PORT_SPI, DD_SS
RCALL SPI_MASTER_TRANSMIT
SBI PORT_SPI, DD_SS
LDI DATA_SPI_TX, WRITE
CBI PORT_SPI, DD_SS
RCALL SPI_MASTER_TRANSMIT

LDI DATA_SPI_TX, MEMADD3
RCALL SPI_MASTER_TRANSMIT
LDI DATA_SPI_TX, HIGH(MEMADD21)
RCALL SPI_MASTER_TRANSMIT
LDI DATA_SPI_TX, LOW(MEMADD21)
RCALL SPI_MASTER_TRANSMIT
MOV DATA_SPI_TX, COUNT
RCALL SPI_MASTER_TRANSMIT
SBI PORT_SPI, DD_SS

CHECK_IN_PROCESS:
CBI PORT_SPI, DD_SS
LDI DATA_SPI_TX, RDSR
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
RCALL SPI_MASTER_TRANSMIT  
SBRC DATA_SPI_RX, WIP  
RJMP CHECK_IN_PROCESS  
SBI PORT_SPI, DD_SS  
RET
```

EEPROM_DEL:

```
LDI DATA_SPI_TX, WREN  
CBI PORT_SPI, DD_SS  
RCALL SPI_MASTER_TRANSMIT  
SBI PORT_SPI, DD_SS  
  
LDI DATA_SPI_TX, PE  
CBI PORT_SPI, DD_SS  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, MEMADD3  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, HIGH(MEMADD21)  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, LOW(MEMADD21)  
RCALL SPI_MASTER_TRANSMIT  
MOV DATA_SPI_TX, COUNT  
RCALL SPI_MASTER_TRANSMIT  
SBI PORT_SPI, DD_SS
```

CHECK_IN_PROCESS_DEL:

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
CBI PORT_SPI, DD_SS  
LDI DATA_SPI_TX, RDSR  
RCALL SPI_MASTER_TRANSMIT  
SBRC DATA_SPI_RX, WIP  
RJMP CHECK_IN_PROCESS_DEL  
SBI PORT_SPI, DD_SS  
  
RET  
  
EEPROM_READ:  
LDI DATA_SPI_TX, READ  
CBI PORT_SPI, DD_SS  
  
RCALL SPI_MASTER_TRANSMIT  
  
LDI DATA_SPI_TX, MEMADD3  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, HIGH(MEMADD21)  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, LOW(MEMADD21)  
RCALL SPI_MASTER_TRANSMIT  
LDI DATA_SPI_TX, $FF  
RCALL SPI_MASTER_TRANSMIT  
SBI PORT_SPI, DD_SS  
RET
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6
Môn thí nghiệm: Vi xử lý

UART0_INIT:

```
PUSH R16
LDI R16, (1 << TXEN0) | (1 << RXEN0)      STS UCSR0B, R16

LDI R16, (1 << UCSZ00) | (1 << UCSZ01)
STS UCSR0C, R16
LDI R16, $00
STS UBRR0H, R16
LDI R16, 51
STS UBRR0L, R16
POP R16
```

RET

UART0_TX:

```
PUSH R17
UART0_TX_WAIT:
    LDS R17, UCSR0A
    SBRS R17, UDRE0
    RJMP UART0_TX_WAIT
STS UDR0, DATA_TX

POP R17
RET
```

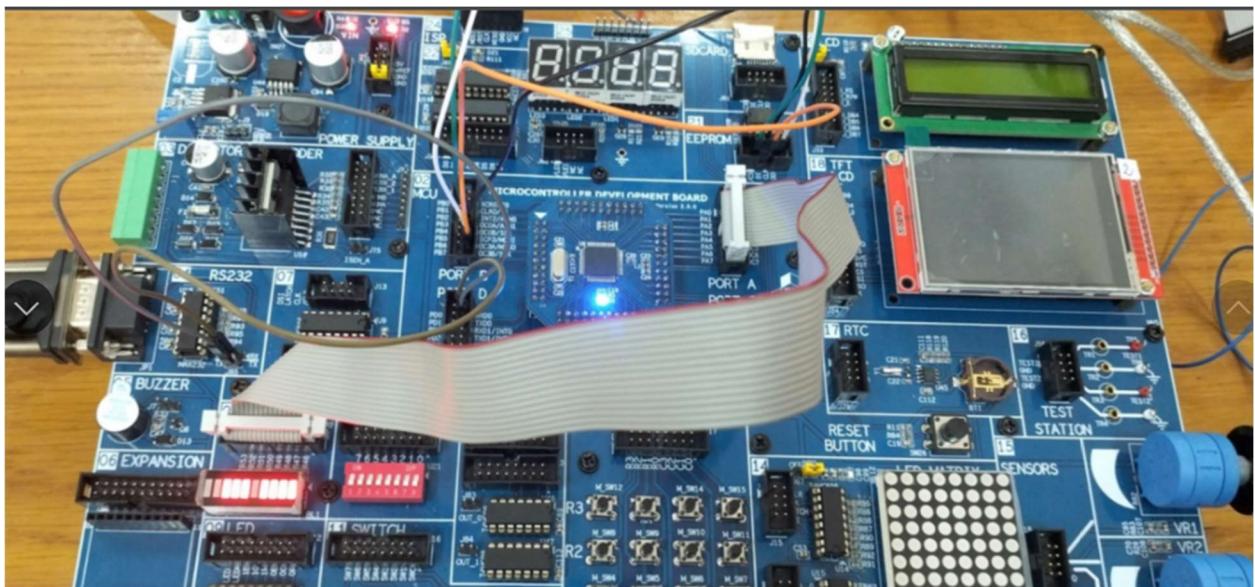
UART0_RX:

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6
Môn thí nghiệm: Vi xử lý

```
PUSH R18  
  
UART0_RX_WAIT:  
  
    LDS R18, UCSR0A  
  
    SBRS R18, RXC0  
  
    RJMP UART0_RX_WAIT  
  
    LDS DATA_RX, UDR0  
  
    POP R18  
  
    RET
```



BÀI 5

1. Trả lời các câu hỏi
 - a. Atmega324PA có dung lượng EEPROM là bao nhiêu? **1KB**
 - b. Liệt kê sự khác nhau giữa SRAM và EEPROM

EEPROM có thể giữ lại giá trị trong bộ nhớ kể cả khi ngắt nguồn điện, SRAM thì không.

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6
Môn thí nghiệm: Vi xử lý

c) Liệt kê sự khác nhau giữa Flash và EEPROM

Flash được sử dụng để lưu trữ chương trình trong khi EEPROM được sử dụng để lưu trữ dữ liệu có thể thay đổi.

- Flash được sử dụng để lưu trữ các chương trình lớn hơn, trong khi EEPROM được sử dụng để lưu trữ các dữ liệu nhỏ hơn.
- Flash chỉ có thể được xóa và ghi lại một lần khi cần thiết để cập nhật chương trình, trong khi EEPROM có thể xóa và ghi lại nhiều lần.
- Flash có thể được ghi lại theo khối, trong khi EEPROM được ghi lại từng byte một.
- Thời gian truy cập của Flash là nhanh hơn so với EEPROM.
- Flash yêu cầu năng lượng lớn hơn để xóa và ghi lại so với EEPROM.

2. Mã nguồn chương trình với chú thích

```
.DEF COUNTER = R18
.DEF DATA_UART_RX = R19
.EQU P_OUT = PORTA
.EQU DD_OUT = DDRA

.ORG 0
MAIN:
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    CLR COUNTER
    SER R16
    OUT DD_OUT, R16
    LDI R20, $01
    LDI R21, $00
    RCALL USART_INIT

START:
    RCALL WRITE_TO_EEPROM
    RCALL READ_FROM_EEPROM
    OUT P_OUT, COUNTER
    RCALL USART_RECEIVER_CHAR
    INC COUNTER
    RJMP START

USART_INIT:
    LDI R16, $00
```

BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
STS UBRR0H, R16
LDI R16, 51
STS UBRR0L, R16

LDI R16, (1<<RXEN0)
STS UCSR0B, R16
LDI R16, (1<<UCSZ01)|(1<<UCSZ00)
STS UCSR0C, R16
RET

USART_RECEIVER_CHAR:
PUSH R17
WAIT_UART_RECEIVE:
LDS R17, UCSR0A
SBRS R17, RXC0
RJMP WAIT_UART_RECEIVE
LDS DATA_UART_RX, UDR0
POP R17
RET

WRITE_TO_EEPROM:
WAIT_ENABLE_WRITE:
SBIC EECR, EEPE
RJMP WAIT_ENABLE_WRITE
OUT EEARH, R21
OUT EEARL, R20
OUT EEDR, COUNTER
SBI EECR, EEMPE
SBI EECR, EEEPE
RCALL DELAY_5MS
RET

READ_FROM_EEPROM:
WAIT_READ:
SBIC EECR, EEEPE
RJMP WAIT_READ
OUT EEARH, R21
OUT EEARL, R20
SBI EECR, EERE
IN COUNTER, EEDR
RET

DELAY_5MS:
LDI R20, 5
LP2:
LDI R21, 250
LP1:
NOP
DEC R21
BRNE LP1
DEC R20
```

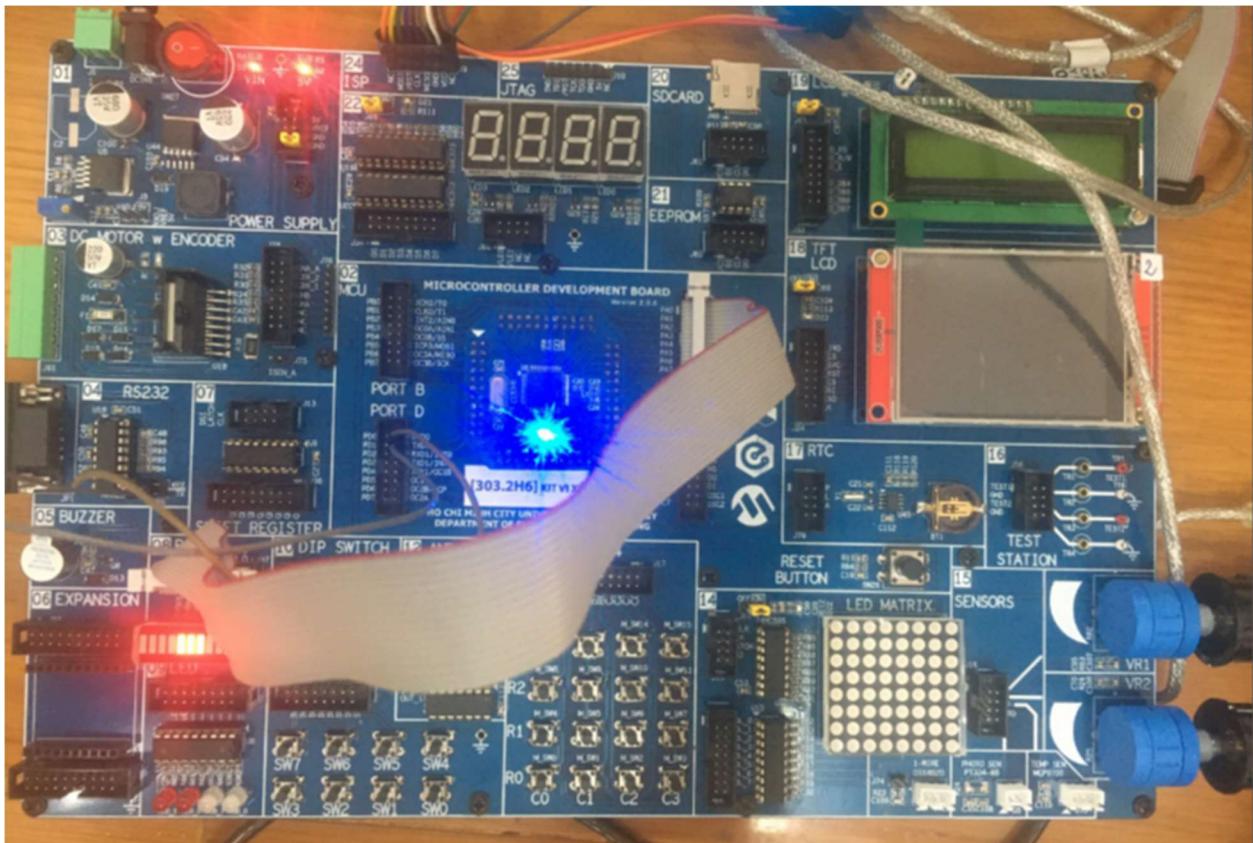
BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

BRNE LP2
RET



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

MỤC TIÊU:

- Hiểu và sử dụng được ngắt ngoài, ngắt timer, UART
- Quét LED 7 đoạn và LED ma trận sử dụng ngắt timer

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 3, 4, 5, 7
- Atmel-2505-Setup-and-Use-of-AVR-Timers_ApplicationNote_AVR130.pdf

BÀI 1

- a) Lập trình tạo một xung tần số 1 Khz trên chân PC0 sử dụng ngắt timer 1 overflow. Khi timer 1 tràn, trong chương trình phục vụ ngắt đảo chân PC0 và đặt lại giá trị cho thanh ghi đếm
- b) Kết nối PC0 vào oscilloscope để đo dạng sóng
(Lưu ý: tần số xung clock cho CPU trên kit thí nghiệm là 8Mhz)

```
.EQU PORT_OUT=0
.ORG 0X00
RJMP MAIN
.ORG 0X0024
RJMP TIMER0_OVF_ISR
.ORG 0X0040
MAIN:
SBI DDRC,1
CBI PORTC,1
LDI R19,1
LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,(1<<PORT_OUT)
OUT DDRC,R16
CBI PORTC,PORT_OUT
```

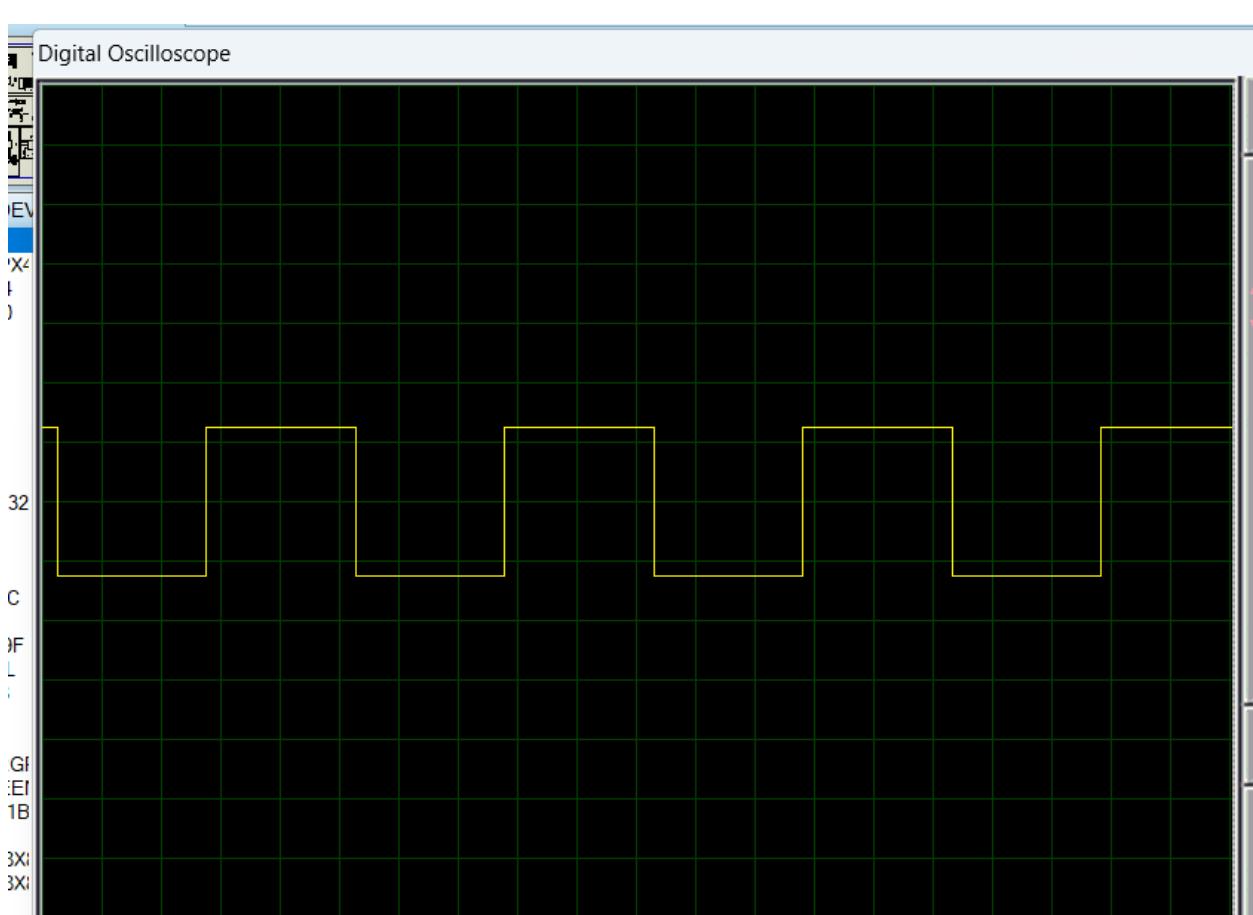
LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
LDI R16,-62
OUT TCNT0,R16
LDI R16,0
OUT TCCR0A,R16
LDI R16,$03
OUT TCCR0B,R16
SEI
LDI R16,(1<<TOIE0)
STS TIMSK0,R16
START:
RJMP START
;=====
TIMER0_OVF_ISR:
LDI R17,0
OUT TCCR0B,R17
LDI R17,-62
OUT TCNT0,R17
IN R17,PORTC
EOR R17,R19
OUT PORTC,R17
LDI R16,$03
OUT TCCR0B,R16
RETI
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT



BÀI 2

- Lặp lại bài 1 sử dụng timer 1 ở mode CTC, sử dụng ngắt COMPARE_MATCH, tạo ra 1 xung với tần số 100 Hz trên chân PC0.
- Cấu hình timer để tạo ra ngắt COMPARE_MATCH sau mỗi 1ms, trong ngắt sử dụng 1 số đếm để đếm số lần xảy ra ngắt và điều khiển chân PC0 để tạo ra xung tần số 100 Hz.
HD: Mỗi lần xảy ra ngắt cộng số đếm lên 1, nếu số đếm bằng 5 thì đảo PC0 và reset số đếm về 0.
- Biên dịch chương trình và quan sát oscilloscope để kiểm tra chương trình.

```
.EQU PORT_OUT=0  
.ORG 0  
CLR R20
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

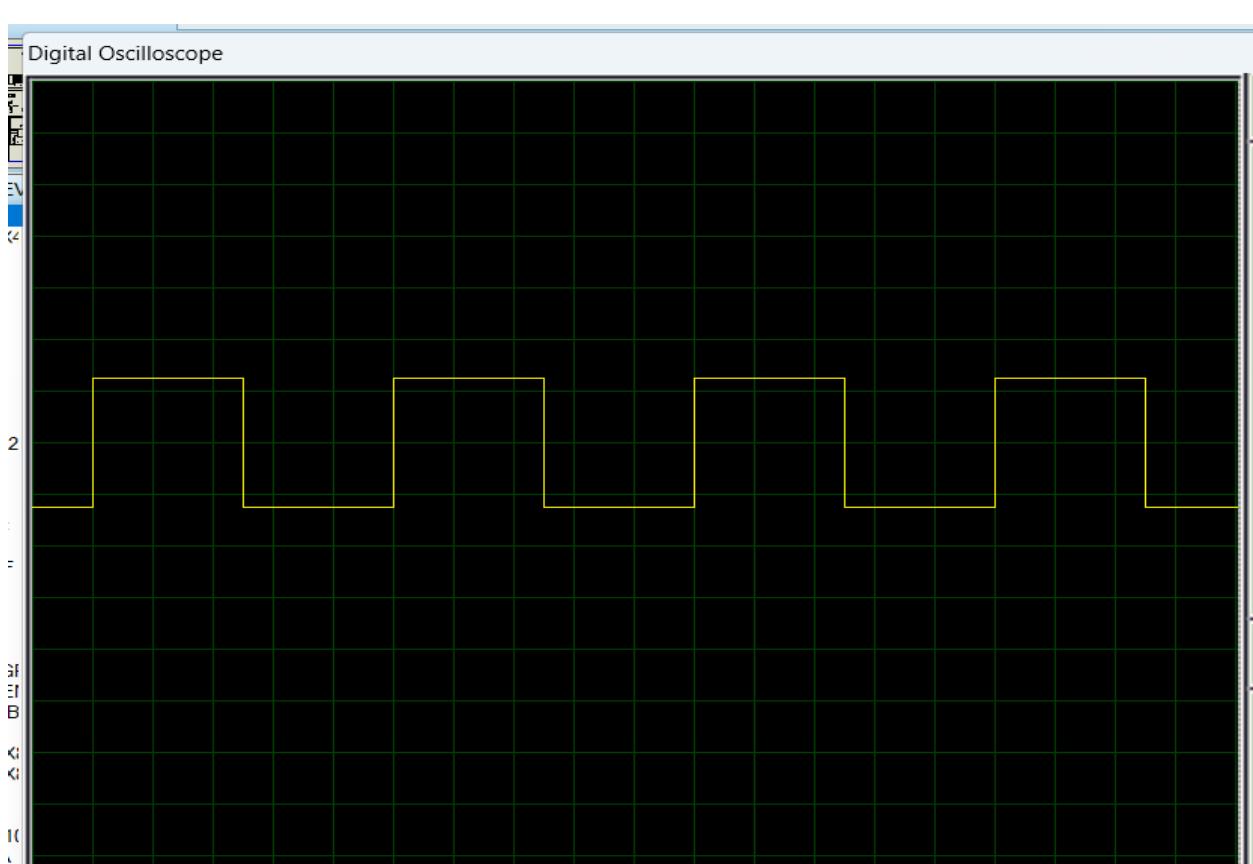
RJMP MAIN

```
.ORG $001A
INC R20
CPI R20,10
BREQ CHANGE
RJMP EXIT
CHANGE:
CLR R20
IN R17,PORTC
EOR R17,R19
OUT PORTC,R17
EXIT:
RETI
```

```
.ORG $100
MAIN:
SBI DDRC,1
CBI PORTC,1
LDI R19,1
LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,(1<<PORT_OUT)
OUT DDRC,R16
LDI R17,HIGH(500)
STS OCR1AH,R17
LDI R17,LOW(500)
STS OCR1AL,R17
LDI R17,0
STS TCCR1A,R17
LDI R17,0B00001010
STS TCCR1B,R17
SEI
LDI R17,(1<<OCIE1A)
STS TIMSK1,R17
START:
RJMP START
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

BÀI 3

- a) Kết nối các tín hiệu cần thiết để điều khiển khói LED 7 đoạn.
- b) Sử dụng ngắt COMPARE_MATCH của timer 1 như ở Bài 2 để xuất số 1-2-3-4 ra 4 LED 7 đoạn với tần số quét 50 Hz. Để đo tần số quét, đảo chân PC0 mỗi lần chuyển sang LED két tiếp và đo xung này trên oscilloscope.

(Tham khảo Chương 4, tài liệu hướng dẫn thí nghiệm)

```
.EQU P_OUT = 0
.EQU P_OUT_LED = PORTA
.EQU DD_P_OUT_LED = DDRA
.EQU LE_0 = 0
.EQU LE_1 = 1
.EQU P_CONTROL_E = PORTC
.EQU DD_P_CONTROL_E = DDRC

;-----;
.DEF DATA_DISPLAY = R21
.DEF DATA_DISPLAY_7SEG = R22
.DEF LED_ACTIVE = R23
.DEF DATA_DISPLAY_LED = R24

.ORG 0
RJMP MAIN
;----Interrupt vecto table-----;
.ORG $1A
RJMP TIMER1_COMPA
;----Interrupt vecto table-----;

.ORG $40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

CALL CONFIG_7SEG
CALL INIT_TIMER1_CTC
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
SEI ;Global interrupt
LDI R17, (1 << OCIE1A) ;Enable interrupt at A channel
STS TIMSK1, R17

START:
    RJMP START

DISPLAY_7SEG:
    LDI R25, $FF
    OUT P_OUT_LED, R25 ;Turn off led
    SBI P_CONTROL_E, LE_1
    CBI P_CONTROL_E, LE_1

    LDI ZH, HIGH(TABLE_7SEG<<1)
    LDI ZL, LOW(TABLE_7SEG<<1)
    ADD ZL, DATA_DISPLAY

    LPM DATA_DISPLAY_7SEG, Z
    OUT P_OUT_LED, DATA_DISPLAY_7SEG
    SBI P_CONTROL_E, LE_0
    CBI P_CONTROL_E, LE_0

    LDI ZH, HIGH(TABLE_CONTROL<<1)
    LDI ZL, LOW(TABLE_CONTROL<<1)
    ADD ZL, LED_ACTIVE

    LPM DATA_DISPLAY_LED, Z
    OUT P_OUT_LED, DATA_DISPLAY_LED
    SBI P_CONTROL_E, LE_1
    CBI P_CONTROL_E, LE_1
    RET

TIMER1_COMPA:
    IN R18, PORTC
    EOR R18, R16
    OUT PORTC, R18
    CALL DISPLAY_7SEG
    INC DATA_DISPLAY
    INC LED_ACTIVE
    CPI LED_ACTIVE, 4
    BREQ RESET_LED
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
RJMP DONE
RESET_LED:
    LDI DATA_DISPLAY, 1
    LDI LED_ACTIVE, 0
DONE:
    RETI

CONFIG_7SEG:
    SBI DDRC, 0 ;PC0 is out
    LDI DATA_DISPLAY, 1
    LDI LED_ACTIVE, 0
    LDI R16, (1 << P_OUT)
    SER R17
    OUT DD_P_OUT_LED, R17 ;PortA is output
    SBI DD_P_CONTROL_E, LE_0 ;PB0 is LE0 that is output
    SBI DD_P_CONTROL_E, LE_1 ;PB1 is LE1      that is output
    CBI P_CONTROL_E, LE_0 ;Block latch
    CBI P_CONTROL_E, LE_1 ;Block latch
    RET

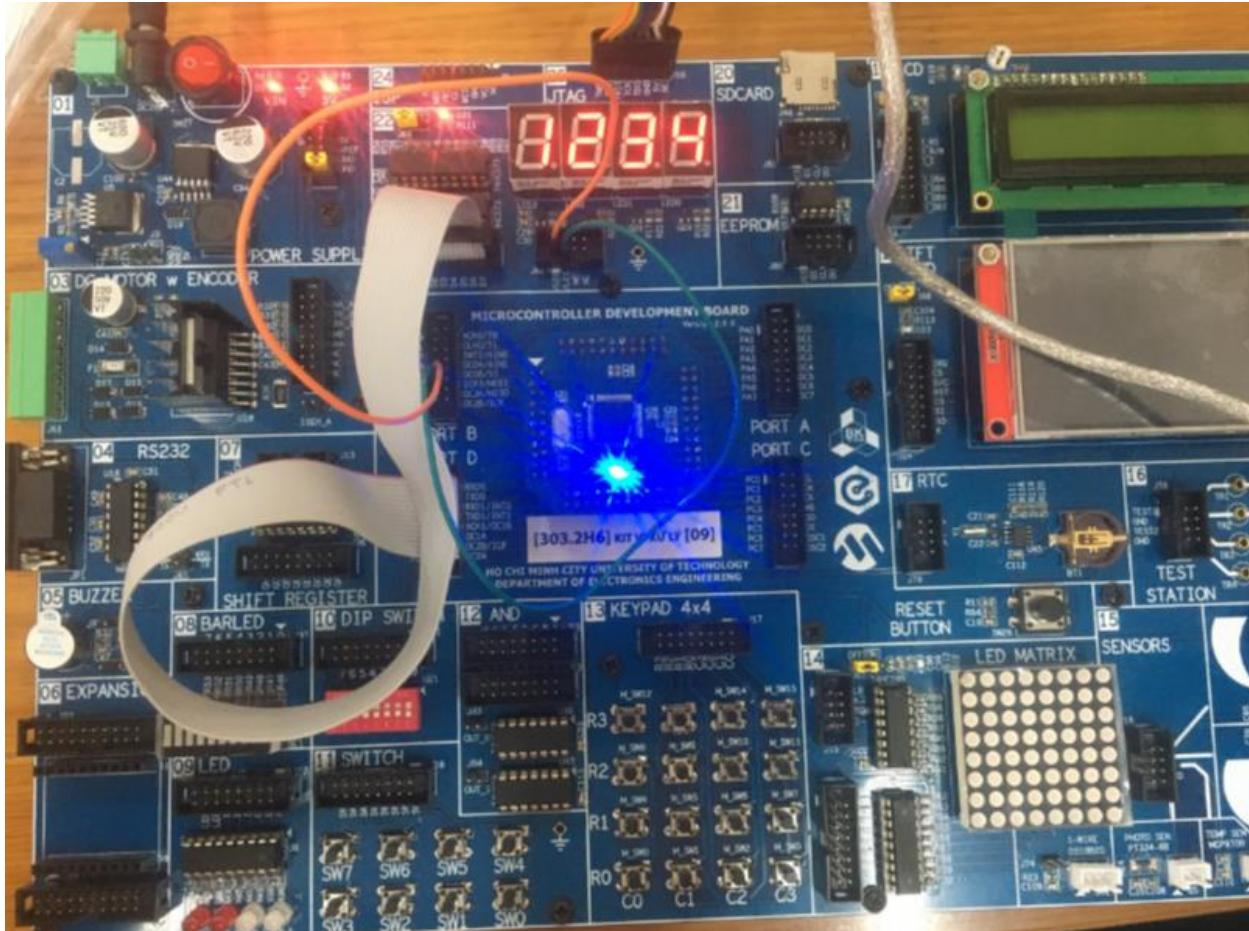
INIT_TIMER1_CTC:
    LDI R17, HIGH(39999)
    STS OCR1AH, R17
    LDI R17, LOW(39999)
    STS OCR1AL, R17
    LDI R17, $00
    STS TCCR1A, R17
    ;Mode CTC
    LDI R17, (1 << WGM12) | (1 << CS10) ;Mode CTC, timer run with
prescaler = 1
    STS TCCR1B, R17
    RET

.ORG $200
TABLE_7SEG:
    .DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
    .DB 0XC6,0XA1,0X86,0X8E

TABLE_CONTROL:
    .DB 0b00001110, 0b00001101, 0b00001011, 0b00000111
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

BÀI 4

- Kết nối thêm các tín hiệu cần thiết để điều khiển LCD ký tự
- Kết nối tín hiệu UART ra khói RS232 và kết nối dây USB-Serial.
- Viết chương trình nhận ký tự từ UART sử dụng ngắn, xuất ký tự đó ra LCD ở vị trí đầu tiên bên trái, đồng thời đếm số ký tự nhận được và xuất ra 4 LED 7 đoạn. Khi nhận được hơn 1000 ký tự thì reset về 0.

Hướng dẫn:

Sử dụng ngắt UART để nhận ký tự. Trong ngắt UART tăng biến đếm (16 bit), chuyển số này thành số BCD và ghi vào 4 byte **LED7segValue**. Ví dụ số đếm đang là 500, ta ghi 0-5-0-0 vào 4 byte này.

Phản quét LED giữ nguyên như ở bài 3.

Tham khảo chương 4, phần 4.4, tài liệu hướng dẫn thí nghiệm.

```
;-----LCD-----;
.EQU RS = 0
.EQU RW = 1
.EQU EN = 2
.EQU LCD = PORTB
.EQU LCD_DR = DDRB
.EQU LCD_IN = PINB
.DEF DATA_DISPLAY_LCD = R18
;-----LCD-----;

;-----7SEG-----;

.EQU P_OUT_LED = PORTA
.EQU DD_P_OUT_LED = DDRA
.EQU LE_0 = 0
.EQU LE_1 = 1
.EQU P_CONTROL_E = PORTC
.EQU DD_P_CONTROL_E = DDRC

.DEF DATA_DISPLAY = R21
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
.DEF DATA_DISPLAY_7SEG = R22
.DEF LED_ACTIVE = R23
.DEF DATA_DISPLAY_LED = R24

;-----7SEG-----;

;-----MAIN-----;
.DEF COUNT_L = R19
.DEF COUNT_H = R20
.DEF OPD3 = R2 ;So du
                ;HIGH(So bi chia)
.DEF OPD1_H = R16 ;LOW(So bi chia)
.DEF OPD1_L = R17 ;So chia
.DEF OPD2 = R3
.DEF DONVI = R4
.DEF CHUC = R5
.DEF TRAM = R6
.DEF NGHIN = R7

.ORG 0
RJMP MAIN

;-----Interrupt vecto table-----;

.ORG $1A
RJMP TIMER1_COMPA
.ORG $28
RJMP USART0_RX

;-----Interrupt vecto table-----;

.ORG $40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

CALL CONFIG_LCD
CALL CONFIG_7SEG
CALL CONFIG_USART
CALL INIT_TIMER1_CTC
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
CLR DATA_DISPLAY_LCD
CLR DATA_DISPLAY
CLR COUNT_L
CLR COUNT_H
LDI XL, $00
LDI XH, $02

SEI ;Enable interrupt globally

START:
CALL DISPLAY_LCD
RJMP START

;---Functions for UART-----

CONFIG_USART:
LDI R16, 0
STS UBRR0H, R16
LDI R16, 51
STS UBRR0L, R16

LDI R16, (1 << UCSZ00) | (1 << UCSZ01) ;8bit data 1 stop
bit, no-parity
STS UCSR0C, R16

LDI R16, (1 << RXEN0) | (1 << RXCIE0) ;Enable receive and
enable to interrupt receive
STS UCSR0B, R16
RET

USART0_RX:
LDS DATA_DISPLAY_LCD, UDR0
CALL CLEAR_LCD

LDI R25, 1
LDI R16, 0
ADD COUNT_L, R25
ADC COUNT_H, R16
MOV OPD1_L, COUNT_L
MOV OPD1_H, COUNT_H
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
CALL DIV16_8
MOV DONVI, OPD3
STS $203, DONVI
CALL DIV16_8
MOV CHUC, OPD3
STS $202, CHUC
CALL DIV16_8
MOV TRAM, OPD3
STS $201, TRAM
CALL DIV16_8
MOV NGHIN, OPD3
STS $200, NGHIN

// When count to 1000 --> reset counter
    CPI COUNT_L, $E8
    BREQ CONTI_SS
    RJMP DONE_RX
CONTI_SS:
    CPI COUNT_H, $03
    BREQ RESET_COUNTER
    RJMP DONE_RX

RESET_COUNTER:
    CLR COUNT_L
    CLR COUNT_H
DONE_RX:
    RETI

DIV16_8:
    PUSH R19
    LDI R19, 10
    MOV OPD2, R19
    LDI R19, 16
    CLR OPD3
SH_NXT:
    CLC
    LSL OPD1_L
    ROL OPD1_H
    ROL OPD3
    BRCS OV_C
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
SUB OPD3, OPD2
BRCC GT_TH
ADD OPD3, OPD2
RJMP NEXT

OV_C:
    SUB OPD3, OPD2
GT_TH:
    SBR OPD1_L, 1
NEXT:
    DEC R19
    BRNE SH_NXT
    POP R19
    RET

;---Functions for UART-----;

;---Functions for LCD-----;

CONFIG_LCD:
    SER R16                                ;Portc is output LCD
    OUT LCD_DR, R16

    CBI LCD, RS
    CBI LCD, RW
    CBI LCD, EN

    LDI R16, 250
    CALL DELAY_US
    LDI R16, 250
    CALL DELAY_US
    CBI LCD, RS
    LDI R17, $30
    CALL OUT_LCD4

    LDI R16, 42
    CALL DELAY_US
    CBI LCD, RS
    LDI R17, $30
    CALL OUT_LCD4
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
LDI R16,2
CALL DELAY_US
CBI LCD, RS
LDI R17, $32
CALL OUT_LCD4_2

LDI R18, $28
LDI R19, $01
LDI R20, $0C
LDI R21, $06

CALL INIT_LCD4
RET

INIT_LCD4:
    CBI LCD, RS           ;Write Instruction
    MOV R17, R18           ;R18 is function set
    CALL OUT_LCD4_2

    MOV R17, R19           ;R19 is clear display
    CALL OUT_LCD4_2
    LDI R16, 20             ;Wait 2ms after clearing
    CALL DELAY_US

    MOV R17, R20
    CALL OUT_LCD4_2
    MOV R17, R21
    CALL OUT_LCD4_2
RET

DISPLAY_LCD:
    CBI LCD, RS
    LDI R17, $80
    CALL OUT_LCD4_2

    MOV R17, DATA_DISPLAY_LCD
    SBI LCD, RS
    CALL OUT_LCD4_2
RET
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

CLEAR_LCD:

```
PUSH R17  
PUSH R16  
CBI LCD,RS  
LDI R17, $01  
CALL OUT_LCD4_2  
LDI R16, 20  
CALL DELAY_US  
POP R16  
POP R17  
RET
```

OUT_LCD4:

```
OUT LCD, R17  
SBI LCD, EN  
CBI LCD, EN  
RET
```

OUT_LCD4_2:

```
LDI R16, 1 ;Wait 100us  
CALL DELAY_US  
IN R16, LCD  
ANDI R16, (1<<RS)  
PUSH R16 ;Push bit RS to stack  
PUSH R17  
ANDI R17, $F0 ;Get 4 high bit  
OR R17, R16  
CALL OUT_LCD4  
LDI R16, 1 ;Wait 100us  
CALL DELAY_US  
POP R17  
POP R16  
SWAP R17  
ANDI R17, $F0  
OR R17, R16  
CALL OUT_LCD4  
RET
```

DELAY_US:

```
MOV R15, R16  
LDI R16, 200
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
L1:  
    MOV R14, R16  
L2:  
    DEC R14  
    NOP  
    BRNE L2  
    DEC R15  
    BRNE L1  
    RET  
;---Functions for LCD-----;  
  
;---Functions for 7SEG-----;  
  
CONFIG_7SEG:  
    LDI LED_ACTIVE, 0  
    LDI DATA_DISPLAY, 0  
  
    SER R17  
    OUT DD_P_OUT_LED, R17      ;PortA is output  
    SBI DD_P_CONTROL_E, LE_0   ;PB0 is LE0 that is output  
    SBI DD_P_CONTROL_E, LE_1   ;PB1 is LE1      that is output  
    CBI P_CONTROL_E, LE_0     ;Block latch  
    CBI P_CONTROL_E, LE_1     ;Block latch  
    RET  
  
DISPLAY_7SEG:  
    LDI R25, $FF  
    OUT P_OUT_LED, R25          ;Turn off led  
    SBI P_CONTROL_E, LE_1  
    CBI P_CONTROL_E, LE_1  
    LDI R25, 0  
  
    LDI ZH, HIGH(TABLE_7SEG<<1)  
    LDI ZL, LOW(TABLE_7SEG<<1)  
    ADD ZL, DATA_DISPLAY  
    ADC ZH, R25  
  
    LPM DATA_DISPLAY_7SEG, Z  
    OUT P_OUT_LED, DATA_DISPLAY_7SEG  
    SBI P_CONTROL_E, LE_0  
    CBI P_CONTROL_E, LE_0
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
LDI ZH, HIGH(TABLE_CONTROL<<1)
LDI ZL, LOW(TABLE_CONTROL<<1)
ADD ZL, LED_ACTIVE
ADC ZH, R25

LPM DATA_DISPLAY_LED, Z
OUT P_OUT_LED, DATA_DISPLAY_LED
SBI P_CONTROL_E, LE_1
CBI P_CONTROL_E, LE_1
RET

TIMER1_COMPA:
LD DATA_DISPLAY, X+
CALL DISPLAY_7SEG

INC LED_ACTIVE
CPI LED_ACTIVE, 4
BREQ RESET_LED
RJMP DONE

RESET_LED:
LDI LED_ACTIVE, 0
LDI XL, $00
LDI XH, $02

DONE:
RETI

INIT_TIMER1_CTC:
LDI R17, HIGH(39999)
STS OCR1AH, R17
LDI R17, LOW(39999)
STS OCR1AL, R17

LDI R17, $00
STS TCCR1A, R17
;Mode CTC
LDI R17, (1 << WGM12) | (1 << CS10)           ;Mode CTC, timer run
with prescaler = 64
STS TCCR1B, R17

LDI R17, (1 << OCIE1A)           ;Enable interrupt at A channel
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

STS TIMSK1, R17
RET

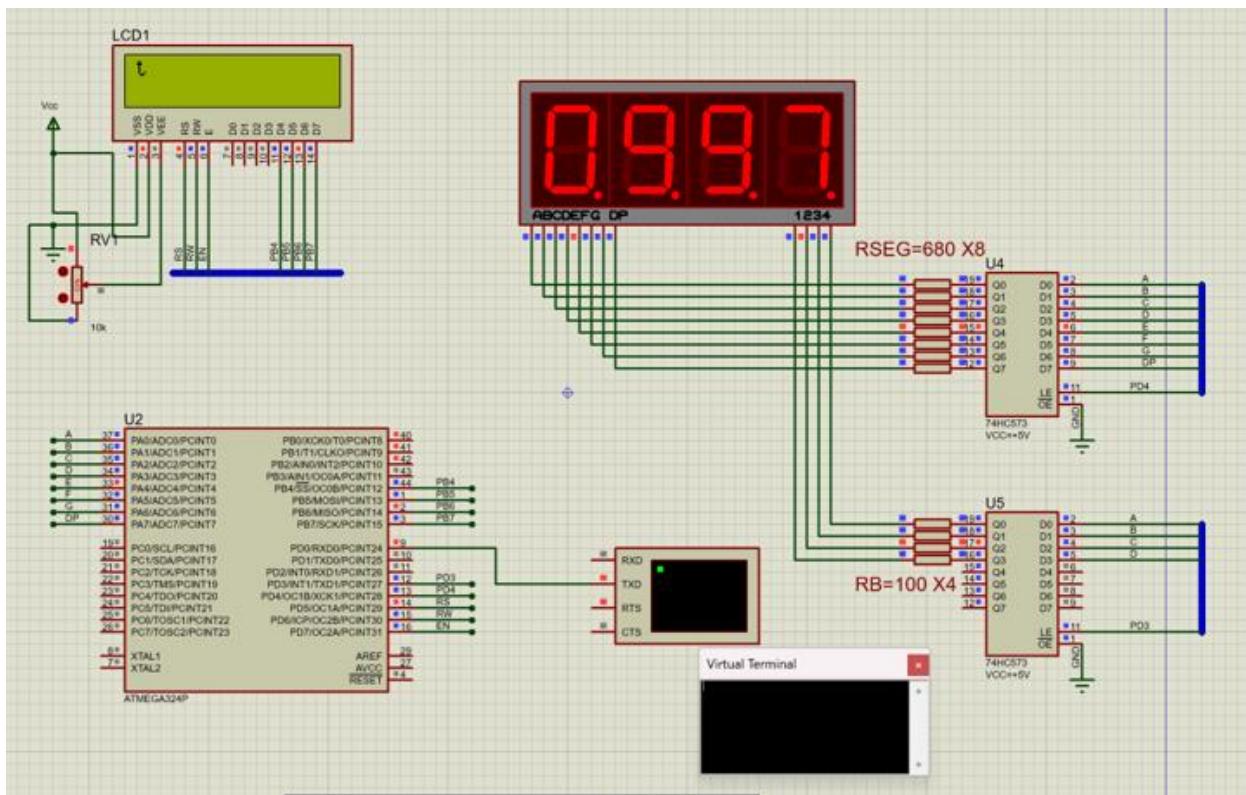
TABLE_7SEG:

```
.DB 0XC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8, 0X80, 0X90, 0X88, 0X8
.DB 0XC6, 0XA1, 0X86, 0X8E
```

TABLE_CONTROL:

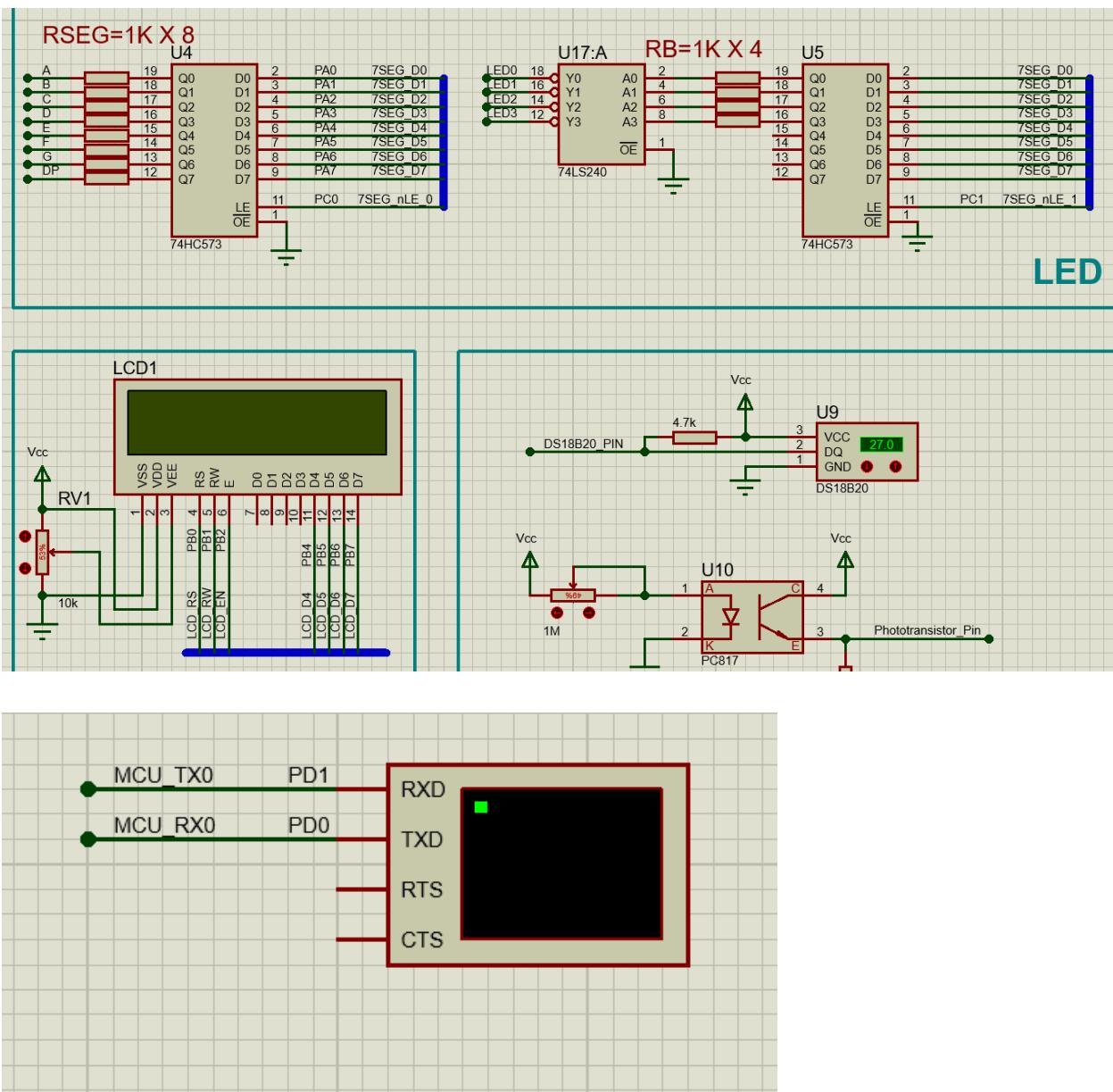
```
.DB 0b00000111, 0b00001011, 0b00001101, 0b00001110
```

;---Functions for 7SEG-----;



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT



LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

BÀI 5

- a) Kết nối thêm các tín hiệu I2C vào module RTC. Đưa tín hiệu MFP của RTC vào 1 chân ngắt ngoài. Thêm vào bài 4 các chức năng sau:
- b) Khởi động RTC với thời gian hiện hành, xuất tín hiệu MFP tần số 1 Hz.
- c) Sử dụng ngắt ngoài của AVR, cứ mỗi khi có ngắt thì đọc thời gian giờ:phút:giây từ RTC và xuất ra dòng 2 của LCD

Lưu ý: Các chức năng ở bài 4 vẫn được giữ nguyên.

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

BÀI 6

- a) Kết nối các tín hiệu cần thiết để điều khiển LED ma trận. (gỡ bỏ các dây nối đến LCD, Led 7 đoạn, RTC, UART).
- b) Sử dụng chương trình mẫu, chỉnh sửa nêu cần thiết để hiển thị chữ ‘A’ lên LED ma trận. Quét LED ma trận sử dụng ngắn timer với tần số quét 25 Hz.
- c) Chỉnh sửa chương trình để đạt tần số quét là 125Hz.
- d) Viết chương trình để hiển thị Logo Trường ĐH Bách Khoa lên LED ma trận.

CODE CÂU C:

```
.org 0x0000 ; interrupt vector table
rjmp reset_handler ; reset

.org 0x001A
rjmp timer1_COMP_ISR
reset_handler:
    ; initialize stack pointer
    ldi r16, high(RAMEND)
    out SPH, r16
    ldi r16, low(RAMEND)
    out SPL, r16
    call shiftregister_initport
    call shiftregister_cleardata
    call initTimer1CTC
    ; enable global interrupts
    sei
    call ledmatrix_portinit
main:
    jmp main

.equ clearSignalPort = PORTB      ; Set clear signal port to PORTB
.equ clearSignalPin = 3           ; Set clear signal pin to pin 3 of
PORTB
.equ shiftClockPort = PORTB       ; Set shift clock port to PORTB
.equ shiftClockPin = 2             ; Set shift clock pin to pin 2 of PORTB
.equ latchPort = PORTB           ; Set latch port to PORTB
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
.equ latchPin = 1           ; Set latch pin to pin 1 of PORTB
.equ shiftDataPort = PORTB   ; Set shift data port to PORTB
.equ shiftDataPin = 0         ; Set shift data pin to pin 0 of PORTB

; Initialize ports as outputs
shiftregister_initport:
    push r24
    ldi      r24,
(1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin)
;
    out      DDRB, r24        ; Set DDRB to output
    pop      r24
    ret

shiftregister_cleardata:
    cbi clearSignalPort, clearSignalPin      ; Set clear signal pin
to low                         ; Wait for a short time
    sbi clearSignalPort, clearSignalPin      ; Set clear signal pin
to high
    ret
; Shift out data
;shift out R27 to bar led
shiftregister_shiftoutdata:
    push r18
    cbi shiftClockPort, shiftClockPin        ;
    ldi r18, 8                      ; Shift 8 bits
shiftloop:
    sbrc r27, 7       ; Check if the MSB of shiftData is 1
    sbi shiftDataPort, shiftDataPin    ; Set shift data pin to high
    sbi shiftClockPort, shiftClockPin  ; Set shift clock pin to
high
    lsl r27          ; Shift left
    cbi shiftClockPort, shiftClockPin  ; Set shift clock pin to
low
    cbi shiftDataPort, shiftDataPin    ; Set shift data pin to low
    dec r18
    brne shiftloop
; Latch data
    sbi latchPort, latchPin     ; Set latch pin to high
    cbi latchPort, latchPin     ; Set latch pin to low
    pop  r18
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
ret

;Lookup table for column control
ledmatrix_col_control: .DB 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02,
0x01
; Lookup table for font
ledmatrix_Font_A:      .DB      0b11111100, 0b00010010, 0b00010001,
0b00010001, 0b00010010, 0b11111100, 0b00000000, 0b00000000
;           J38 connect to PORTD
; clear signal pin to pin 0 of PORTB
; shift clock pin to pin 1 of PORTB
; latch pin to pin 0 of PORTB
; shift data pin to pin 3 of PORTB
; Output: None

.equ LEDMATRIXPORT = PORTD
.equ LEDMATRIXDIR = DDRD
.dsseg
.org SRAM_START          ;starting address is 0x100
    LedMatrixBuffer : .byte 8
    LedMatrixColIndex : .byte 1
.csseg
.align 2
ledmatrix_portinit:
    push r20
    push r21
    ldi     r20, 0b11111111 ; SET port as output
    out    LEDMATRIXDIR, r20

    ldi     r20,0                 ;col index start at 0
    ldi     r31,high(LedMatrixColIndex)
    ldi     r30,low(LedMatrixColIndex)
    st      z,r20
    ldi     r20,0
    ldi     r31,high(ledmatrix_Font_A << 1) ;Z register point to
fontA value
    ldi     r30,low(ledmatrix_Font_A << 1)
ldi     r29,high(LedMatrixBuffer) ; Y register point to fontA
value
    ldi     r28,low(LedMatrixBuffer)
    ldi     r20,8
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
ledmatrix_portinit_loop: ;copy font to display
buffer
    lpm      r21,z+
    st       y+,r21
    dec     r20
    cpi     r20,0
    brne   ledmatrix_portinit_loop
    pop     r21
    pop     r20
    ret

; Display a Column of Led Matrix
; Input: R27 contains the value to display
;         R26 contain the Col index (3..0)
; Output: None
ledmatrix_display_col:
    push r16 ; Save the temporary register
    push r27
    clr     r16
    out    LEDMATRIXPORT,r16
    call   shiftregister_shiftoutdata

    ldi     r31,high(ledmatrix_col_control << 1)
    ldi     r30,low(ledmatrix_col_control << 1)
    clr     r16
    add    r30,r26
    adc     r31,r16
    lpm     r27,z
    out    LEDMATRIXPORT,r27
    pop     r27
    pop     r16 ; Restore the temporary register
    ret ; Return from the function

initTimer1CTC:
    push   r16
    ldi   r16, high(4000) ; Load the high byte into the temporary
register
    sts   OCR1AH, r16      ; Set the high byte of the timer 1
compare value
    ldi   r16, low(4000)   ; Load the low byte into the temporary
register
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
sts OCR1AL, r16      ; Set the low byte of the timer 1 compare
value
ldi r16, (1 << CS10) | (1<< WGM12) ; Load the value 0b000000101
into the temporary register
sts TCCR1B, r16      ;
ldi r16, (1 << OCIE1A); Load the value 0b000000010 into the
temporary register
sts TIMSK1, r16      ; Enable the timer 1 compare A interrupt
pop r16
ret

timer1_COMP_ISR:
push r16
push r26
push r27
ldi      r31,high(LedMatrixColIndex)
ldi      r30,low(LedMatrixColIndex)
ld       r16,z
mov     r26,r16
ldi      r31,high(LedMatrixBuffer)
ldi      r30,low(LedMatrixBuffer)
add     r30,r16
clr     r16
adc     r31,r16
ld       r27,z
call   ledmatrix_display_col

inc      r26
cpi      r26,8
brne   timer1_COMP_ISR_CONT
ldi      r26,0 ;if r26 = 8, reset to 0
timer1_COMP_ISR_CONT:
ldi      r31,high(LedMatrixColIndex)
ldi      r30,low(LedMatrixColIndex)
st       z,r26

pop     r27
pop     r26
pop     r16
reti
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

CODE CÂU D:

```
.org 0x0000 ; interrupt vector table
rjmp reset_handler ; reset

.org 0x001A
rjmp timer1_COMP_ISR
reset_handler:
    ; initialize stack pointer
    ldi r16, high(RAMEND)
    out SPH, r16
    ldi r16, low(RAMEND)
    out SPL, r16
    call shiftregister_initport
    call shiftregister_cleardata
    call initTimer1CTC
    ; enable global interrupts
    sei
    call ledmatrix_portinit
main:
    jmp main

.equ clearSignalPort = PORTB      ; Set clear signal port to PORTB
.equ clearSignalPin = 3            ; Set clear signal pin to pin 3 of
PORTB
.equ shiftClockPort = PORTB       ; Set shift clock port to PORTB
.equ shiftClockPin = 2             ; Set shift clock pin to pin 2 of PORTB
.equ latchPort = PORTB           ; Set latch port to PORTB
.equ latchPin = 1                 ; Set latch pin to pin 1 of PORTB
.equ shiftDataPort = PORTB        ; Set shift data port to PORTB
.equ shiftDataPin = 0              ; Set shift data pin to pin 0 of PORTB

; Initialize ports as outputs
shiftregister_initport:
    push r24
    ldi r24,
(1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin)
;
    out DDRB, r24                  ; Set DDRB to output
    pop r24
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
ret

shiftregister_cleardata:
    cbi clearSignalPort, clearSignalPin      ; Set clear signal pin
to low                         ; Wait for a short time
    sbi clearSignalPort, clearSignalPin      ; Set clear signal pin
to high
    ret
; Shift out data
;shift out R27 to bar led
shiftregister_shiftoutdata:
    push r18
    cbi shiftClockPort, shiftClockPin      ;
    ldi r18, 8                          ; Shift 8 bits
shiftloop:
    sbrc r27, 7           ; Check if the MSB of shiftData is 1
    sbi shiftDataPort, shiftDataPin      ; Set shift data pin to high
    sbi shiftClockPort, shiftClockPin   ; Set shift clock pin to
high
    lsl r27                ; Shift left
    cbi shiftClockPort, shiftClockPin   ; Set shift clock pin to
low
    cbi shiftDataPort, shiftDataPin      ; Set shift data pin to low
    dec r18
    brne shiftloop
; Latch data
    sbi latchPort, latchPin      ; Set latch pin to high
    cbi latchPort, latchPin      ; Set latch pin to low
    pop r18
    ret

;Lookup table for column control
ledmatrix_col_control: .DB 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02,
0x01
; Lookup table for font
ledmatrix_Font_A:     .DB          0b01100000, 0b10010000, 0b10001100,
0b01000010, 0b01000010, 0b10001100, 0b10010000, 0b01100000
;           J38 connect to PORTD
; clear signal pin to pin 0 of PORTB
; shift clock pin to pin 1 of PORTB
; latch pin to pin 0 of PORTB
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
; shift data pin to pin 3 of PORTB
; Output: None

.equ LEDMATRIXPORT = PORTD
.equ LEDMATRIXDIR = DDRD
.dsseg
.org SRAM_START ;starting address is 0x100
    LedMatrixBuffer : .byte 8
    LedMatrixColIndex : .byte 1
.csseg
.align 2
ledmatrix_portinit:
    push r20
    push r21
    ldi     r20, 0b11111111 ; SET port as output
    out    LEDMATRIXDIR, r20

    ldi     r20,0           ;col index start at 0
    ldi     r31,high(LedMatrixColIndex)
    ldi     r30,low(LedMatrixColIndex)
    st      z,r20
    ldi     r20,0
    ldi     r31,high(ledmatrix_Font_A << 1) ;Z register point to
fontA value
    ldi     r30,low(ledmatrix_Font_A << 1)
ldi     r29,high(LedMatrixBuffer) ; Y register point to fontA
value
    ldi     r28,low(LedMatrixBuffer)
    ldi     r20,8
ledmatrix_portinit_loop:          ;copy font to display
buffer
    lpm    r21,z+
    st     y+,r21
    dec    r20
    cpi    r20,0
    brne  ledmatrix_portinit_loop
    pop    r21
    pop    r20
    ret

; Display a Column of Led Matrix
; Input: R27 contains the value to display
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
;           R26 contain the Col index (3..0)
; Output: None
ledmatrix_display_col:
    push r16 ; Save the temporary register
    push r27
    clr     r16
    out    LEDMATRIXPORT,r16
    call   shiftregister_shiftoutdata

    ldi      r31,high(ledmatrix_col_control << 1)
    ldi      r30,low(ledmatrix_col_control << 1)
    clr     r16
    add     r30,r26
    adc     r31,r16
    lpm     r27,z
    out    LEDMATRIXPORT,r27
    pop     r27
    pop     r16 ; Restore the temporary register
    ret ; Return from the function

initTimer1CTC:
    push   r16
    ldi r16, high(4000) ; Load the high byte into the temporary
register
    sts  OCR1AH, r16      ; Set the high byte of the timer 1
compare value
    ldi r16, low(4000)   ; Load the low byte into the temporary
register
    sts  OCR1AL, r16      ; Set the low byte of the timer 1 compare
value
    ldi r16, (1 << CS10)| (1<< WGM12) ; Load the value 0b000000101
into the temporary register
    sts  TCCR1B, r16      ;
    ldi r16, (1 << OCIE1A); Load the value 0b00000010 into the
temporary register
    sts  TIMSK1, r16      ; Enable the timer 1 compare A interrupt
    pop  r16
    ret

timer1_COMP_ISR:
    push  r16
```

LAB 4-1

LẬP TRÌNH SỬ DỤNG NGẮT

```
push r26
push r27
ldi    r31,high(LedMatrixColIndex)
ldi    r30,low(LedMatrixColIndex)
ld     r16,z
mov   r26,r16
ldi    r31,high(LedMatrixBuffer)
ldi    r30,low(LedMatrixBuffer)
add   r30,r16
clr   r16
adc   r31,r16
ld    r27,z
call  ledmatrix_display_col

inc   r26
cpi   r26,8
brne timer1_COMP_ISR_CONT
ldi   r26,0 ;if r26 = 8, reset to 0
timer1_COMP_ISR_CONT:
ldi    r31,high(LedMatrixColIndex)
ldi    r30,low(LedMatrixColIndex)
st     z,r26

pop   r27
pop   r26
pop   r16
reti
```

LAB 5-1

LẬP TRÌNH SỬ DỤNG ADC

MỤC TIÊU

- Hiểu và sử dụng được ADC của AVR
- Hiểu cách sử dụng ADC để đo đặc

THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 11
- AN2538-ADC-of-megaAVR-in-SingleEnded-Mode-00002538A.pdf
- AVR120: Characterization and Calibration of the ADC on an AVR

BÀI 1: ĐO TÍN HIỆU SINGLE END

- a) Kết nối hai tín hiệu ADC_VR1 và ADC_VR2 từ header J86 vào hai ngõ vào ADC0 và ADC1. Kết nối UART0 với khối RS232 và kết nối cáp USB-Serial vào máy tính. Kết nối ADC_VR1 và ADC_VR2 vào các test point trên header J56. Lưu ý không cắm nhầm vào các pin header GND. Viết chương trình thực hiện các việc sau:
 - b) Chọn điện áp V_{REF} là điện áp nội V_{CCA}. Khởi động UART với cấu hình tự chọn. (Lưu ý cấu hình phần mềm Hercules trên máy tính tương tự). Click chuột phải vào màn hình Hercules để chọn HEX Enable
 - c) Viết chương trình thực hiện lấy mẫu tín hiệu đưa vào ADC0 và gửi lên máy tính sử dụng UART0 với khung truyền như sau sau mỗi 1s. Thời gian 1s tạo ra bằng hàm delay hoặc timer.
0x55 ADCH ADCL 0xFF
 - d) Thay đổi điện áp đưa vào ADC0, đo bằng VOM và so sánh với kết quả lấy mẫu ADC, điền vào bảng trong báo cáo
 - e) Kết nối LCD vào 1 port của AVR, bổ sung vào chương trình đã viết chức năng tính toán điện áp đưa vào và hiển thị lên LCD.
 - f) Thay đổi điện áp tham chiếu là điện áp 2.56V bên trong. Lặp lại các bước c, d, e, giả định điện áp tham chiếu chính xác là 2.56V.
 - g) Đo điện áp trên chân V_{REF} (header J57), sử dụng VOM.

BÀI 2: ĐO OFFSET ERROR VÀ GAIN ERROR

- a) Tính toán các sai số offset error và gain error của ADC.
- b) Viết lại chương trình với yêu cầu như ở câu e bài 1 với ADC đã được tính toán hiệu chỉnh. Vref = V_{CCA}, gửi lên máy tính các giá trị ADC đã hiệu chỉnh, và xuất ra LCD giá trị điện áp đo được

LAB 5-1

LẬP TRÌNH SỬ DỤNG ADC

➤ (Đọc kỹ tài liệu: AVR120: Characterization and Calibration of the ADC on an AVR)

BÀI 3: ĐO ADC Ở CHẾ ĐỘ VI SAI

- Chỉnh kênh VR1 ở mức điện áp 2.5V, đưa vào ADC0.
- Viết chương trình khởi động ADC ở chế độ vi sai với 2 kênh ngõ vào là ADC0 và ADC1, độ lợi khuếch đại là 10, điện áp tham chiếu 2.56V. Khởi động ADC ở chế độ FreeRunning.
- Viết chương trình hiển thị giá trị điện áp VR1 lên LCD, đồng thời gửi kết quả đo ADC lên máy tính như ở bài 1 sau mỗi 1 s như ở bài 1.

BÀI 4: ĐO OFFSET ERROR VÀ GAIN ERROR Ở MODE VISAI

- Kết nối cả ADC1 và ADC0 vào điện áp ADC_VR1. Chỉnh điện áp này về 1V. Ghi nhận giá trị ADC đo được. Đây chính là Offset error.
- Từ giá trị offset error, tính ra gain error từ **Error! Reference source not found.**.
- Viết lại chương trình ở câu c bài 3, với giá trị ADC được hiệu chỉnh

BÀI 5: ĐO NHIỆT ĐỘ SỬ DỤNG MCP9701

- Kết nối cảm biến vào header J73.
- Kết nối tín hiệu điện áp V_TEMP trên header J18 tới ADC0
- Viết chương trình đo giá trị điện áp V_TEMP với các tham số hiệu chỉnh như ở bài 1, tính ra giá trị nhiệt độ và hiển thị lên LCD.

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

BÀI 1

1. Trả lời các câu hỏi

- a. Ở chế độ Single Conversion, làm thế nào bắt đầu một chu kỳ lấy mẫu và kiểm tra xem bao giờ nó thực hiện xong?

Ở chế độ Single Conversion:

+ Để bắt đầu một chu kỳ lấy mẫu ta set bit ADEN (bit 7 của thanh ghi ADCSRA) để cho phép ADC làm việc, set bit ADSC (bit 6 của thanh ghi ADCSRA) để bắt đầu chu kỳ lấy mẫu (bắt đầu chuyển đổi).

+ Để kiểm tra xem bao giờ chu kỳ lấy mẫu được thực thi xong, ta xét cờ ADIF (bit 4 của thanh ghi ADCSRA), với ADIF = 1 báo đã thực thi xong

- b. Lựa chọn kênh ADC như thế nào?

Lựa chọn kênh ADC bằng cách set 5 bit MUX4:0 (bit 4:0) của thanh ghi ADMUX,

- c. Atmega324 có kênh đo nhiệt độ bên trong chip hay không?

Không có kênh cố định nào dung để chuyển đổi ADC Vout. Kênh đọc điện áp Vout được đưa ra từ cảm biến nhiệt độ ta có thể dùng một trong các kênh ADC từ kênh ADC7 đến ADC0 làm ngõ vào chuyển đổi ADC.

- d. Với $V_{REF}=V_{CCA}$, công thức tính ra điện áp ngõ vào từ ADC là gì?

Với $V_{REF} = V_{CCA}$, ta có:

$$V_{IN} = \frac{ADCval}{1024}.5$$

- e. Với $V_{REF}=2.56V$, công thức tính ra điện áp ngõ vào từ ADC là gì?

Với $V_{REF} = 2.56V$, ta có:

$$V_{IN} = \frac{ADCval}{1024}.2,56$$

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

- f. Ghi nhận giá trị ADC so với kết quả đo từ đồng hồ đo. Tính giá trị điện áp theo giá trị thu được từ ADC với $V_{REF}=V_{CCA}$.

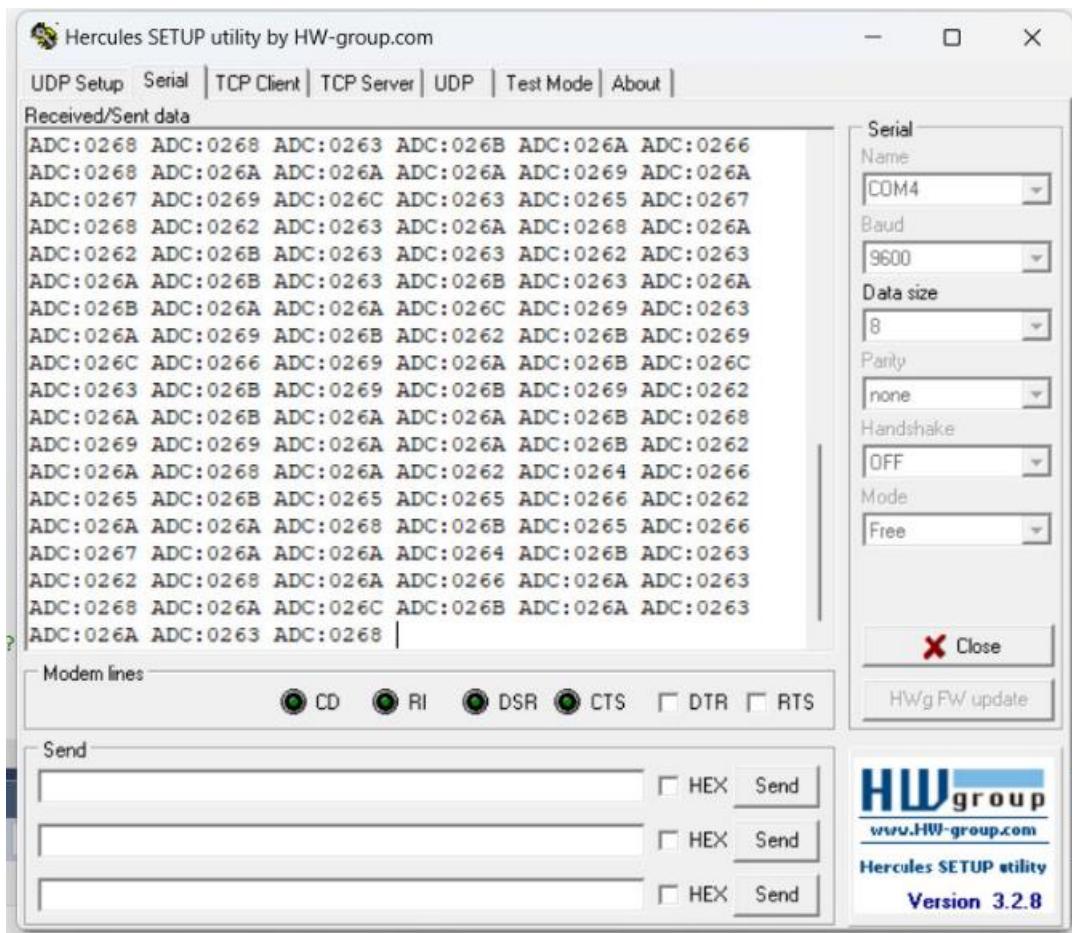
V_{ADC0} (V)	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
ADCH- ADCL	0001	0066	00C7	0135	019C	0204	0265	02D1	033A	039C	03FD
ADCH- ADCL (lý thuyết)	0000	0067	00CD	0133	019A	0200	0266	02CD	0333	039A	0400
Sai số (LSB)	0,004	0,004	0,029	0,009	0,009	0,019	0,004	0,019	0,034	0,009	0,014
	88	88	30	77	77	53	88	53	18	77	65



BÁO CÁO

Nhóm môn học:

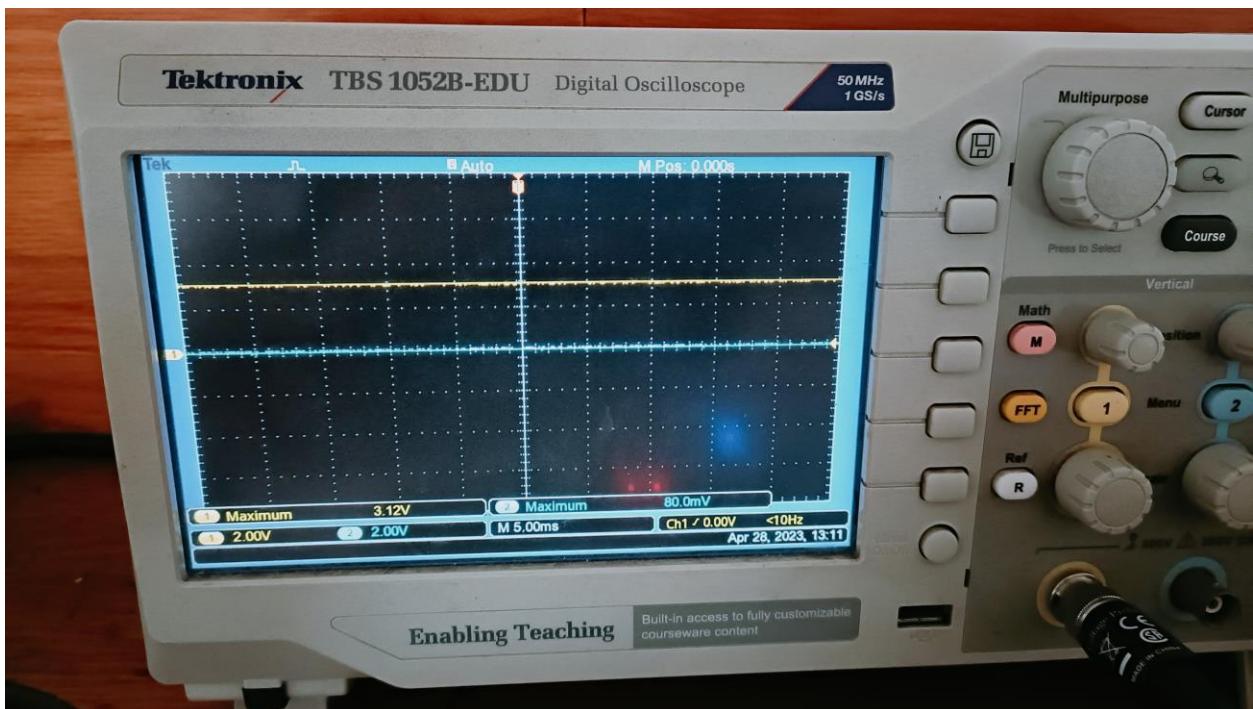
Nhóm:
Môn thí nghiệm:



BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:



2. Mã nguồn chương trình với chú thích
 - a) Tính toán và xuất giá trị Vin ra LCD, đồng thời gửi ADCL và ADCH lên Hercules dưới dạng mã Hex:

```
.EQU ADC_PORT=PORTA
.EQU ADC_DR=DDRA
.EQU ADC_IN=PINA
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
OUT SPL, R16
LDI R16, 0xFF ;PortD, B output
OUT DDRD, R16
OUT DDRB, R16
LDI R16, 0x00 ;PortA input
OUT ADC_DR, R16
OUT PORTD, R16 ;output=0x0000
OUT PORTB, R16
LDI R16, 0b01000000 ;Vref=AVcc=5V, SE ADC0
STS ADMUX, R16 ; x1,dịch phải
LDI R16, 0b10000110 ;cho phép ADC, mode 1 lần.
STS ADCSRA, R16 ;f(ADC)=fosc/64=125Khz
RCALL SETUART

START:
LDS R16, ADCSRA
ORI R16, (1<<ADSC) ;bắt đầu chuyển đổi
STS ADCSRA, R16

WAIT:
LDS R16, ADCSRA ;đọc cờ ADIF
SBRS R16, ADIF ;cờ ADIF=1 chuyển đổi xong
RJMP WAIT ;chờ cờ ADIF=1
STS ADCSRA, R16 ;xóa cờ ADIF
LDS R1, ADCL ;đọc byte thấp ADC
LDS R0, ADCH ;đọc byte cao ADC
LDI R17, 'A'
RCALL PHAT
LDI R17, 'D'
RCALL PHAT
LDI R17, 'C'
RCALL PHAT
LDI R17, ':'
RCALL PHAT
MOV R17, R0
RCALL TACHKITU
MOV R17, R1
RCALL TACHKITU
LDI R17, ' '
RCALL PHAT
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
RCALL DELAY1S
RJMP START ;tiếp tục chuyển đổi
SETUART:
LDI R16, (1<<TXEN0) ;cho phép phát
STS UCSR0B, R16
LDI R16, (1<<UCSZ01)|(1<<UCSZ00)
;8-bit data, không parity, 1 stop bit
STS UCSR0C, R16
LDI R16, 0x00
STS UBRR0H, R16
LDI R16, 51 ;9600 baud rate
STS UBRR0L, R16
RET
PHAT:
LDS R16, UCSR0A
SBRS R16, UDRE0 ; KIEM TRA CO TRONG KHONG
RJMP PHAT ; NEU CHUA TRONG THI TIEP TUC KIEM TRA LAI
STS UDR0, R17 ; KHI TRONG THI CHEP DU LIEU VAO UDR0
RET
DELAY1S:
LDI R16, 200
LP1: LDI R17, 160
LP2: LDI R18, 50
LP3: DEC R18
NOP
BRNE LP3
DEC R17
BRNE LP2
DEC R16
BRNE LP1
RET
;HEX_ASC chuyển từ mã Hex sang mã ASCII
;Input R17=mã Hex, Output R18=mã ASCII
;-----
HEX_ASC:CPI R17, 0X0A
BRCS NUM
LDI R18, 0X37
RJMP CHAR
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
NUM: LDI R18,0X30
CHAR: ADD R18,R17
RET
TACHKITU :
MOV R15,R17
LDI R16,0XF0
AND R17,R16 ; GIU LAI BIT CAO
SWAP R17
RCALL HEX_ASC
MOV R17,R18
RCALL PHAT
MOV R17,R15
ANDI R17,0X0F
RCALL HEX_ASC
MOV R17,R18
RCALL PHAT
RET
```

BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

Xuất giá trị ADC ra BARLED (ADCL) và 2 LED đơn (ADCH), đồng thời gửi ADCL và ADCH lên Hercules dưới dạng mã Hex:

```
.EQU ADC_PORT=PORTA
.EQU ADC_DR=DDRA
.EQU ADC_IN=PINA
.EQU LCD=PORTB ;PORTB data
.EQU LCD_IN=PINB
.EQU LCD_DR=DDRB
.EQU CONT=PORTB ;PORTB ?i?u khi?n
.EQU CONT_DR=DDRB
.EQU CONT_OUT=PORTB ;
.EQU CONT_IN=PINB ;
.EQU RS=0 ;bit RS
.EQU RW=1 ;bit RW
.EQU E=2 ;bit E
.EQU BCD_BUF=0X200 ;?/c ??u SRAM l?u s? BCD (kq chuy?n t? s? 16 bit)
.DEF OPD1_L=R24 ;byte th?p c?a s? nh? phân 16 bit
.DEF OPD1_H=R25 ;byte cao c?a s? nh? phân 16 bit
.DEF OPD2=R22
.DEF OPD3=R23
.DEF COUNT=R18
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, 0xFF ;PortD, C output
OUT DDRB, R16
OUT DDRC, R16
```

BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
LDI R16, 0x00 ;PortA input
OUT ADC_DR, R16
OUT PORTD, R16 ;output=0x0000
OUT PORTB, R16
LDI R16, 0b01000000 ;Vref=AVcc=5V, SE ADC0
STS ADMUX, R16 ; x1,d?ch ph?i
LDI R16, 0b10000110 ;cho phép ADC, mode 1 l?n.
STS ADCSRA, R16 ;f(ADC)=fosc/64=125Khz
RCALL SETUART
LDI R16, 0X07
OUT CONT_DR, R16 ;khai báo PB0,PB1,PB2 là output
CBI CONT,RS ;RS=PB0=0
CBI CONT,RW ;RW=PB1=0 truy xu?t ghi
CBI CONT,E ;E=PB2=0 c?m LCD
LDI R16, 0xFF
OUT LCD_DR, R16 ;khai báo outport
RCALL RESET_LCD ;ctc reset LCD
RCALL INIT_LCD4 ;ctc kh?i ??ng LCD 4 bit
START:
LDS R16, ADCSRA
ORI R16, (1<<ADSC) ;b?t ??u chuy?n ??i
STS ADCSRA, R16
WAIT:
LDS R16, ADCSRA ;??c c? ADIF
SBRS R16, ADIF ;c? ADIF=1 chuy?n ??i xong
RJMP WAIT ;ch? c? ADIF=1
STS ADCSRA, R16 ;xóa c? ADIF
LDS R1, ADCL ;??c byte th?p ADC
LDS R0, ADCH ;??c byte cao ADC
LDI R17, 'A'
RCALL PHAT
LDI R17, 'D'
RCALL PHAT
LDI R17, 'C'
RCALL PHAT
LDI R17, ':'
RCALL PHAT
MOV R17, R0
```

BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
RCALL TACKITU
MOV R17,R1
RCALL TACKITU
LDI R17,' '
RCALL PHAT
LDS R1, ADCL ;??c byte th?p ADC
LDS R0, ADCH ;??c byte cao ADC
MOV R17,R0
MOV R16,R1
RCALL MUL_MATCH
RCALL SHIFT_R
RCALL BIN16_BCD5DG
XUAT_LCD:
CBI CONT,RS ;RS=0 ghi lenh
LDI R17,$84 ;con tr? b?t ??u ? dòng 1 v? trí th? 1
RCALL OUT_LCD4
LDI R17,'A'
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,'D'
SBI CONT,RS
RCALL OUT_LCD4
SBI CONT,RS
LDI R17,'C'
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,':'
RCALL OUT_LCD4
LDS R17,0X202 ; XUAT HANG TRAM
RCALL HEX_ASC
MOV R17,R18
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,44 ;xuat ','
SBI CONT,RS
RCALL OUT_LCD4
LDS R17,0X203 ; XUAT HANG CHUC
RCALL HEX_ASC
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
MOV R17,R18
SBI CONT,RS
RCALL OUT_LCD4
LDS R17,0X204 ; XUAT HANG DON VI
RCALL HEX_ASC
MOV R17,R18
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,'V'
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,10
SBI CONT,RS
RCALL OUT_LCD4
RCALL DELAY1S
RJMP START ;ti?p t?c chuy?n ??i
SETUART:
LDI R16, (1<<TXEN0) ;cho phép phát
STS UCSR0B, R16
LDI R16, (1<<UCSZ01)|(1<<UCSZ00)
;8-bit data, không parity, 1 stop bit
STS UCSR0C, R16
LDI R16, 0x00
STS UBRR0H, R16
LDI R16, 51 ;9600 baud rate
STS UBRR0L, R16
RET
PHAT:
LDS R16,UCSR0A
SBRS R16,UDRE0 ; KIEM TRA CO TRONG KHONG
RJMP PHAT ; NEU CHUA TRONG THI TIEP TUC KIEM TRA LAI
STS UDR0,R17 ; KHI TRONG THI CHEP DU LIEU VAO UDR0
RET
DELAY1S:
LDI R16,200
LP_1: LDI R17,160
LP_2: LDI R18,50
LP_3: DEC R18
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
NOP
BRNE LP_3
DEC R17
BRNE LP_2
DEC R16
BRNE LP_1
RET
;HEX_ASC chuyen t? m? Hex sang m? ASCII
;Input R17=m? Hex,Output R18=m? ASCII
;-----
HEX_ASC:
CPI R17,0X0A
BRCS NUM
LDI R18,0X37
RJMP CHAR
NUM: LDI R18,0X30
CHAR: ADD R18,R17
RET
TACHKITU :
MOV R15,R17
LDI R16,0XF0
AND R17,R16 ; GIU LAI BIT CAO
SWAP R17
RCALL HEX_ASC
MOV R17,R18
RCALL PHAT
MOV R17,R15
ANDI R17,0X0F
RCALL HEX_ASC
MOV R17,R18
RCALL PHAT
RET
MUL_MATCH:
LDI R20,250
MUL R16,R20
MOV R10,R0
MOV R11,R1
MUL R17,R20
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
MOV R12,R0
MOV R13,R1
ADD R12,R11
CLR R0
ADC R13,R0 ;R13:R12:R10
RET
SHIFT_R:
LSR R12
BST R13,0
BLD R12,7
LSR R13
MOV R24,R12
MOV R25,R13
RET
;BIN16_BCD5DG chuy?n ??i s? nh? ph?n 16 bit sang s? BCD 5 digit
;Inputs: OPD1_H=R25:OPD1_L=R24 ch?a s? nh? ph?n 16 bit
;Outputs: BCD_BUF:BCD_BUF+4:??a ch? SRAM ch?a 5 digit BCD t? cao ??n
th?p
;S? d?ng R17,COUNT,X,ctc DIV16_8
;-----
BIN16_BCD5DG:
LDI XH,HIGH(BCD_BUF);X tr? ??a ch? ??u buffer BCD
LDI XL,LOW(BCD_BUF)
LDI COUNT,5 ;??m s? byte b? nh?
LDI R17,0X00 ;n?p giá tr? 0
LOOP_CL:ST X+,R17 ;xóa buffer b? nh?
DEC COUNT ;??m ?? 5 byte
BRNE LOOP_CL
LDI OPD2,10 ;n?p s? chia (SC)
DIV_NXT:
RCALL DIV16_8 ;chia s? nh? ph?n 16 bit cho s? nh? ph?n 8 bit
ST -X,OPD3 ;c?t s? d? vào buffer
CPI OPD1_L,0 ;th??ng s?=0?
BRNE DIV_NXT ;khác 0 chia ti?p
RET
;-----
;DIV16_8 chia s? nh? ph?n 16 bit OPD1 cho 8 bit OPD2 (Xem gi?i thu?t
chia ? Ch??ng 0)
```

BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
; OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=th??ng s?
; OPD3=DS(GPR0-31)
;S? d?ng COUNT(GPR16-31)
;-----
DIV16_8: LDI COUNT,16 ;COUNT=??m 16
CLR OPD3 ;xóa d? s?
SH_NXT: CLC ;C=0=bit th??ng s?
LSL OPD1_L ;d?ch trái SBC L,bit0=C=th??ng s?
ROL OPD1_H ;quay trái SBC H,C=bit7
ROL OPD3 ;d?ch bit7 SBC H vào d? s?
BRCS OV_C ;tràn bit C=1,chia ???c
SUB OPD3,OPD2 ;tr? d? s? v?i s? chia
BRCC GT_TH ;C=0 chia ???c
ADD OPD3,OPD2 ;C=1 không chia ???c,không tr?
RJMP NEXT
OV_C: SUB OPD3,OPD2 ;tr? d? s? v?i s? chia
GT_TH: SBR OPD1_L,1 ;chia ???c,th??ng s?=1
NEXT: DEC COUNT ;??m s? l?n d?ch SBC
BRNE SH_NXT ;ch?a ?? ti?p t?c d?ch bit
RET
OUT_LCD4:
LDI R16,1
RCALL DELAY_US
IN R16,CONT
ANDI R16,(1<<RS)
PUSH R16
PUSH R17
ANDI R17,$F0
OR R17,R16
RCALL OUT_LCD
LDI R16,1
RCALL DELAY_US
POP R17
POP R16
SWAP R17
ANDI R17,$F0
```

BÁO CÁO

Nhóm môn học:

Nhóm:
Môn thí nghiệm:

```
OR R17,R16
RCALL OUT_LCD
RET
OUT_LCD:
OUT LCD,R17 ;1MC,ghi 1?nh/data ra LCD
SBI CONT,E ;2MC,xu?t xung cho phép LCD
CBI CONT,E ;2MC,PWEH=2MC=250ns,tDSW=3MC=375ns
RET
RESET_LCD:
LDI R16,250 ;delay 25ms
RCALL DELAY_US ;ctc delay 100?s x R16
LDI R16,250 ;delay 25ms
RCALL DELAY_US ;ctc delay 100?s x R16
CBI CONT,RS ;RS=0 ghi 1?nh
LDI R17,$30 ;mã 1?nh=$30 1?n 1
RCALL OUT_LCD
LDI R16,42 ;delay 4.2ms
RCALL DELAY_US
CBI CONT,RS
LDI R17,$30 ;mã 1?nh=$30 1?n 2
RCALL OUT_LCD
LDI R16,2 ;delay 200?s
RCALL DELAY_US
CBI CONT,RS
LDI R17,$32 ;mã 1?nh=$32
RCALL OUT_LCD4
RET
INIT_LCD4:
CBI CONT,RS ;RS=0 ghi 1?nh
LDI R17,$24 ;Function set - giao ti?p 4 bit, 1 dòng, font 5x8
RCALL OUT_LCD4
CBI CONT,RS ;RS=0 ghi 1?nh
LDI R17,$01 ;Clear display
RCALL OUT_LCD4
LDI R16,20 ;ch? 2ms sau 1?nh Clear display
RCALL DELAY_US
CBI CONT,RS ;RS=0 ghi 1?nh
LDI R17,$0C ;Display on/off control
```

BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
RCALL OUT_LCD4
CBI CONT,RS ;RS=0 ghi 1?nh
LDI R17,$06 ;Entry mode set
RCALL OUT_LCD4
RET
DELAY_US:
PUSH R15
PUSH R14
MOV R15,R16 ;1MC n?p data cho R15
LDI R16,200 ;1MC s? d?ng R16
L1:
MOV R14,R16 ;1MC n?p data cho R14
L2:
DEC R14 ;1MC
NOP ;1MC
BRNE L2 ;2/1MC
DEC R15 ;1MC
BRNE L1 ;2/1MC
POP R14
POP R15
RET ;4MC
DELAY: ;1s=32*250*250
LDI R25,32
LP3: LDI R26,250
LP2: LDI R27,250
LP1: NOP
DEC R27
BRNE LP1
DEC R26
BRNE LP2
DEC R25
BRNE LP3
RET
```