

# LAB 1-1

## GIAO TIẾP I/O VÀ CÁC LỆNH TÍNH TOÁN

### MỤC TIÊU:

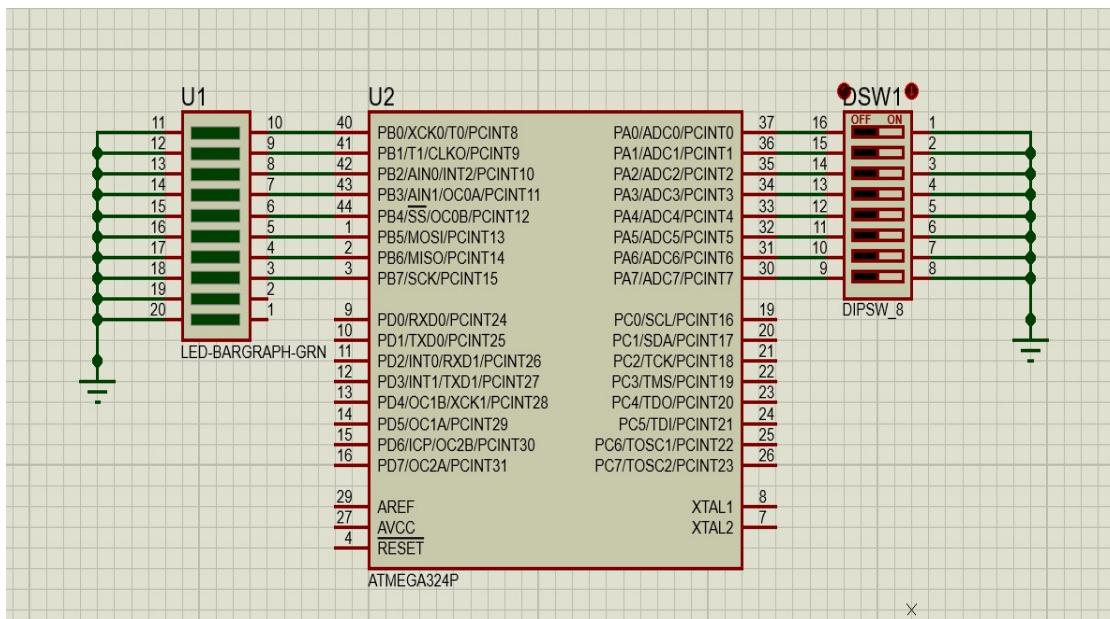
- Thực hiện các giao tiếp I/O Port, các lệnh tính toán

### THAM KHẢO:

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

### BÀI 1

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



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

### BÀI 2

- a) 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.
- b) 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

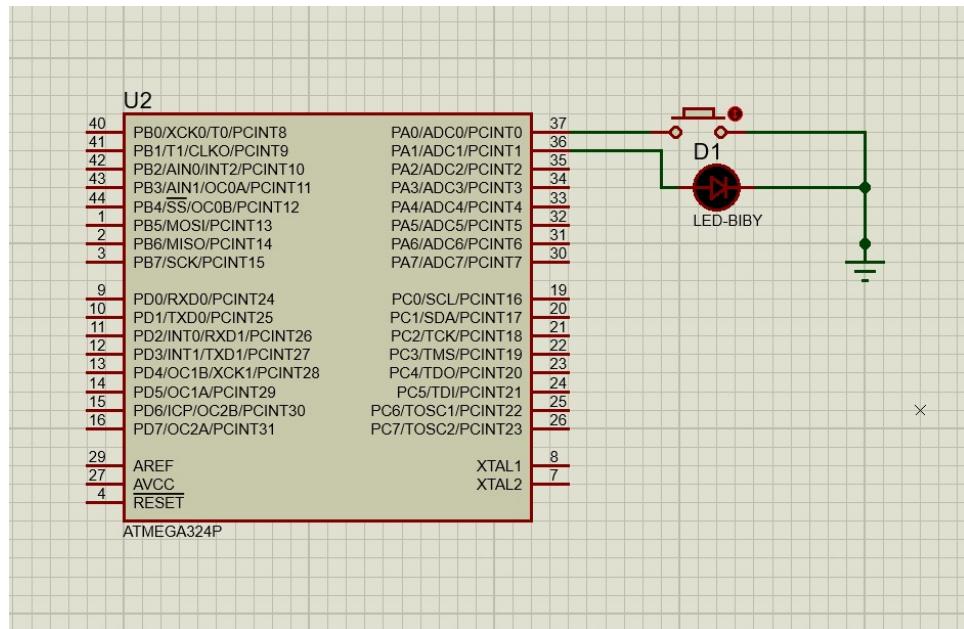
- a) Kết nối và thực hiện chương trình tính tích của 2 nibble cao và thấp của PORTA và gửi ra PORT B. Coi như 2 nibble này là 2 số không dấu  
VD: PORTA = 0b0111\_1111, thì PORTB = 3\*15.
- b) Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

### 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

1. Trả lời các câu hỏi
  - a. Lấy giá trị từ 2 nibble của PORTA như thế nào?
  - b. Enable điện trở pullup như thế nào?
  - c. Khi Switch ở trạng thái ON/OFF, giá trị chân Port bằng bao nhiêu?
  - d. Khi chân port ở trạng thái 1, BAR LED sáng hay tắt?
  - e. 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  
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 ;
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
```

## BÀI 2

- Trả lời các câu hỏi

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

## BÀI 3

- Trả lời các câu hỏi

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

- a. Làm thế nào lấy giá trị từ 2 nibble của PORT A
- b. 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 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  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
```

## BÀI 4

1. Trả lời các câu hỏi

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

## 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 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
      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

### 1. Trả lời các câu hỏi

- Khi Switch ở trạng thái nhấn/nhả, giá trị chân Port bằng bao nhiêu?
- Để LED sáng, chân port xuất ra mức logic gì?
- Mã nguồn với chú 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:
```

```
LDI 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 môn học:

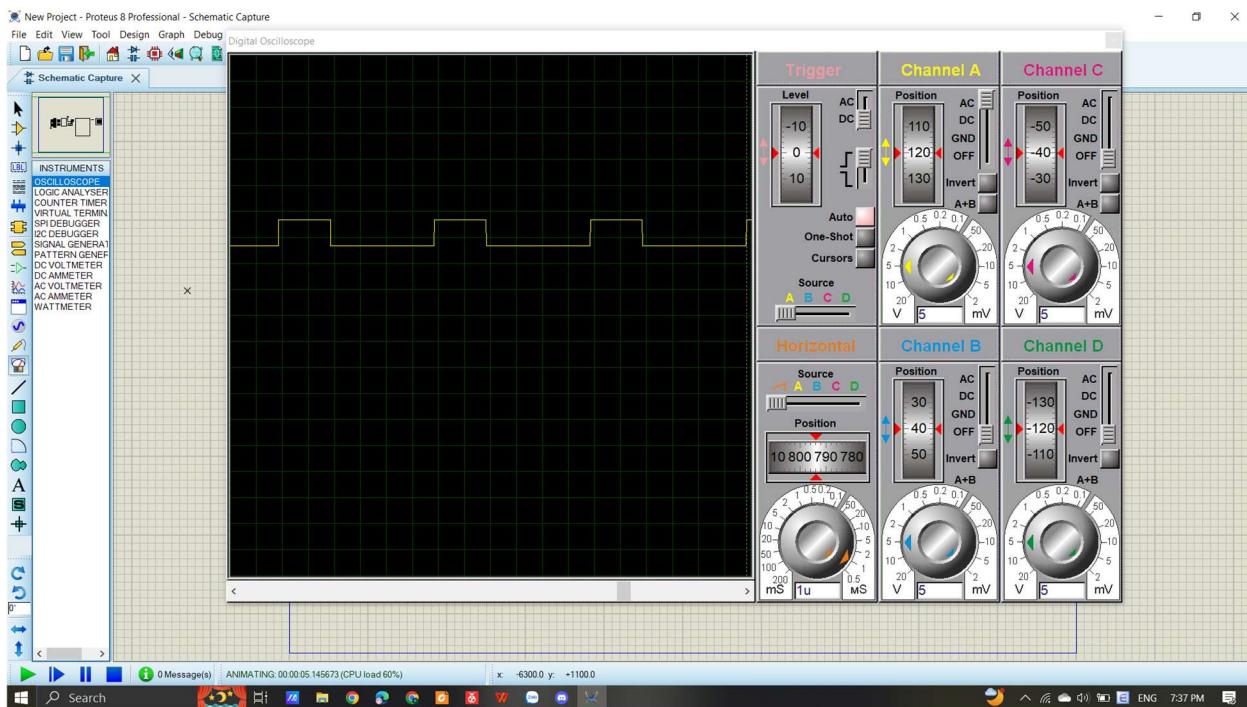
Nhóm:

Môn thí nghiệm:

## BÀI 1

### 2. Trả lời các câu hỏi

#### a. Chụp ảnh dạng xung trên PA0



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

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

---

- 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)

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

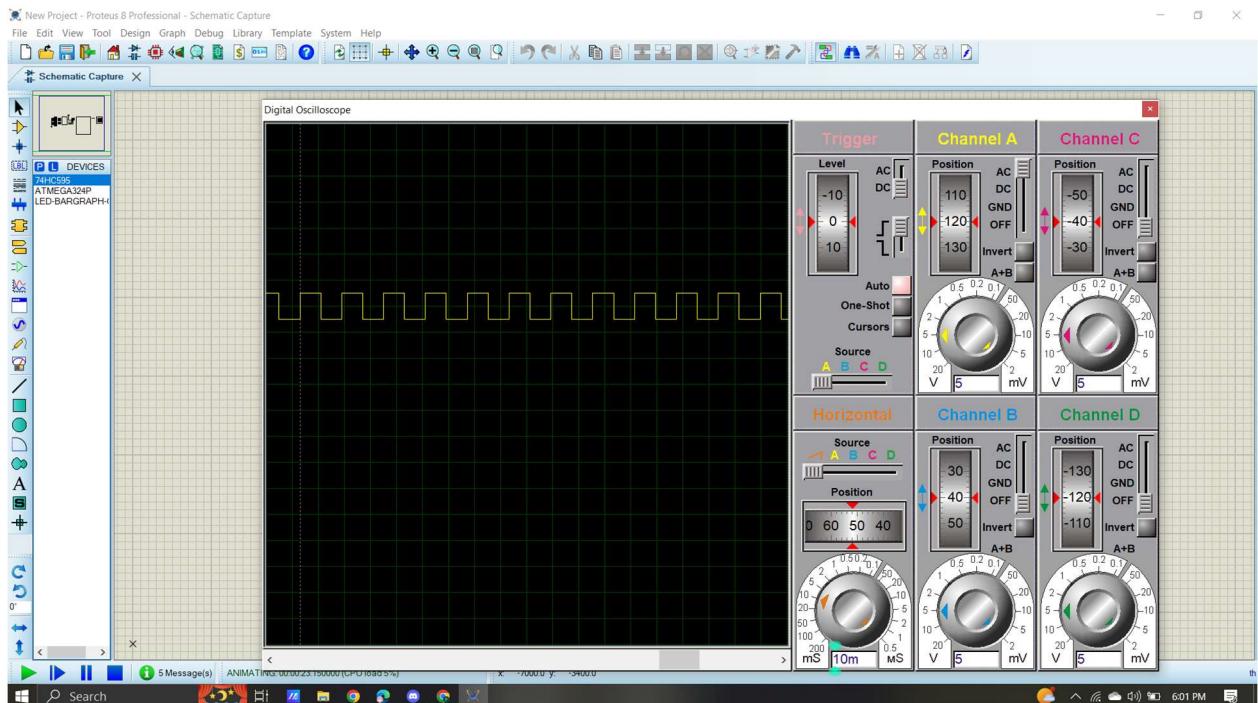
# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

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:

Nhóm:  
Môn thí nghiệm:

```
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

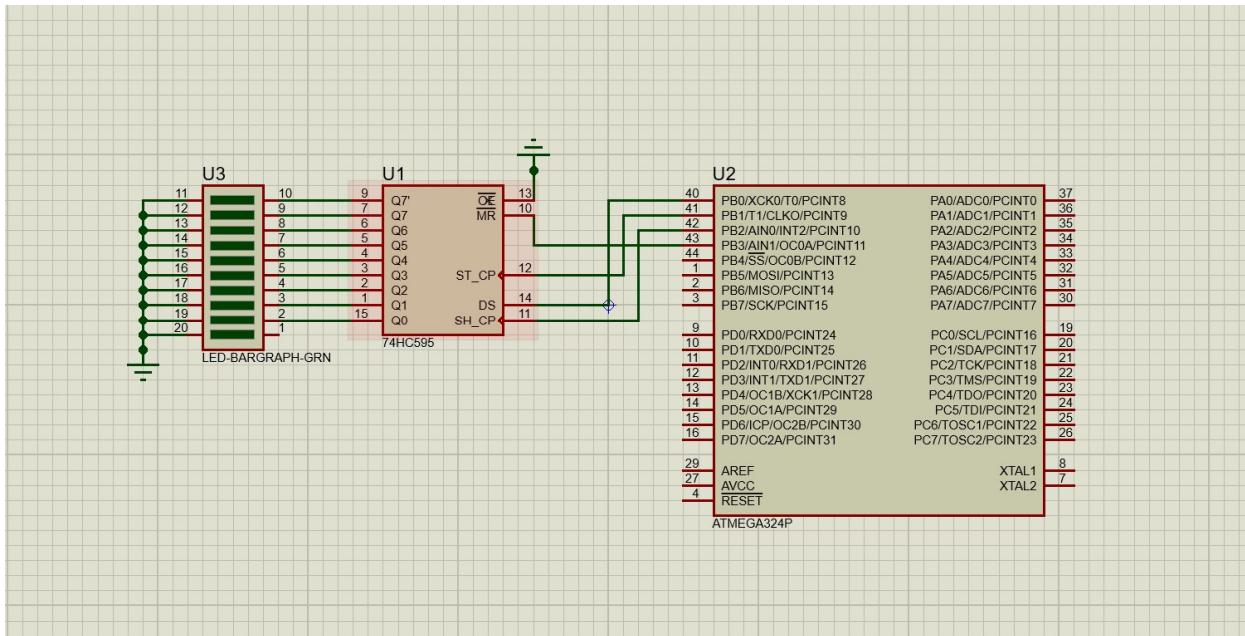
2. Trả lời các câu hỏi
  - c. Mô tả kết nối trên kit thí nghiệm

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:



- d. 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.
- e. 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
- f. 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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
.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
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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
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:
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

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

Nhóm:  
Môn thí nghiệm:

## 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:

Nhóm:  
Môn thí nghiệm:

```
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:

Nhóm:  
Môn thí nghiệm:

```
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:

Nhóm:

Môn thí nghiệm:

```
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
```

## BÀI 2

3. Trả lời các câu hỏi
  - h. Hiện tượng gì xảy ra khi không chống rung phím
  - i. Mô tả cách kết nối trên kit thí nghiệm
  - j. Mã nguồn chương trình không chống rung phím và chú thích

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

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

; Define input-output
cbi DDRA, 0 ; Set Swich 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:
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
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
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

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

LCD_wait_busy:
LCD_wait_busy:
push r18
ldi r18, 0b000000111 ; 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
in r18, LCDPORTPIN
cbi LCDPORT, LCD_EN
nop
sbi LCDPORT, LCD_EN
nop
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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

; Send data to LCD
out LCDPORT, r17
nop
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
; 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
```

k. 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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
.org 0x0000 ;\n\n;Define stack pointer\n\nldi temp, LOW(RAMEND) ; Initialize the stack pointer\n\nout SPL, temp\n\nldi temp, HIGH(RAMEND)\n\nout SPH, temp\n\n;Define LCD\n\nldi temp, 0x38 ; Function set: 8-bit interface, 2 lines, 5x7 dots\n\ncall lcd_send_command\n\nldi temp, 0x0C ; Display on, cursor off, blink off\n\ncall lcd_send_command\n\nldi temp, 0x01 ; Clear display\n\ncall lcd_send_command\n\n; Define input-output\n\ncbi DDRA, 0 ; Set Swith input is PA0\n\nsbi PORTA, 0; Pull-up resistor for PA0\n\nldi temp, 0xFF\n\nout DDRB, temp : Port B Barleds (Output)\n\nout DDRC, temp : Port C LCD (Output)\n\nloop:\n\n    sbic PORTA, 0      ; Check if PA0 = 0 (Switch pressed)\n\n    rjmp loop          ; if not, come back to loop\n\n    rcall DELAY_10ms
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
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
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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, 0b000000111 ; 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:

Nhóm:

Môn thí nghiệm:

```
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  
  
    mov r17,r16 ;save the command
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
; 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
sbi LCDPORT, LCD_EN
nop
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

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

## BÀI 3

4. Trả lời các câu hỏi
  - e. Cách kết nối các module trên bài thí nghiệm
  - f. Có hiện tượng rung phím đôi 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 đôi 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 run 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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
.equ  LCD_D5 = PINA5
.equ  LCD_D4 = PINA4
.def  LCDData = r16

;***** Program ID *****
***** MAIN *****

;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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
start:

    CALL KEY_PAD_SCAN
    MOV R24, R23
    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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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

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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
call      DELAY_10MS
call      DELAY_10MS

; Send initialization sequence
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
out LCDPORT, r17
    nop
    nop
; Pulse enable pin
    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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
; 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
sbi    LCDPORT, LCD_EN
nop
cbi    LCDPORT, LCD_EN
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
pop    r17
ret

;*****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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

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

LCD_Send_String:
    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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
pop    ZL          ; restore pointer
registers

pop    ZH

ret

;*****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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
        andi    r16,0x80
        cpi     r16,0x80
        breq   LCD_wait_busy_loop
        ldi    r16, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output
        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:
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

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

LDI R16, $0F
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

KEYPAD\_SCAN\_CHECK\_COL3:

```
SBIC PIND, 6           ;CHECK ROW 2
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)
```

# BÁO CÁO

Nhóm môn học:

Nhóm:

Môn thí nghiệm:

```
;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

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.
```

## LAB 2-1

# DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

### MỤC TIÊU:

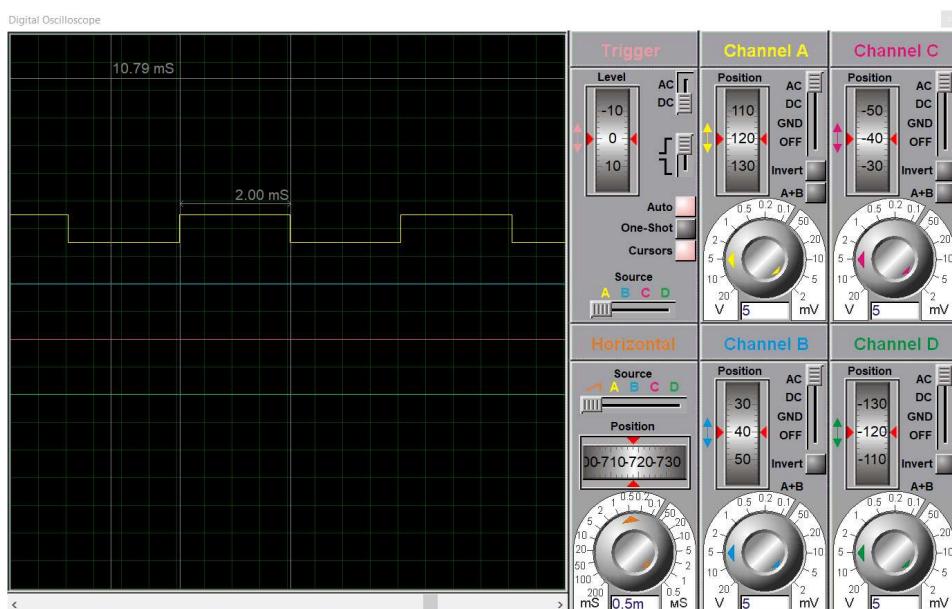
- Hiểu các mode hoạt động của timer
- Hiểu cách sử dụng timer để tạo delay và tạo xung

### 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 1 Khz 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

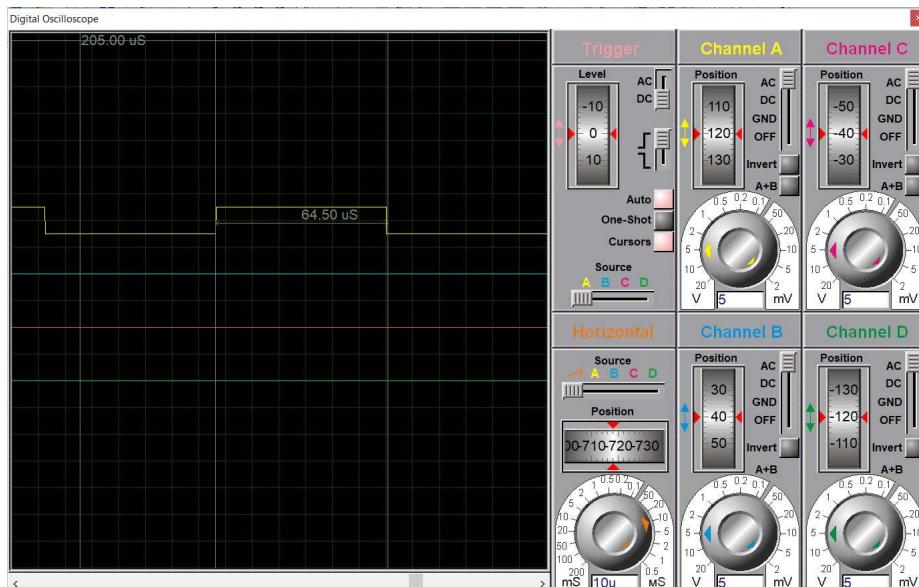


# LAB 2-1

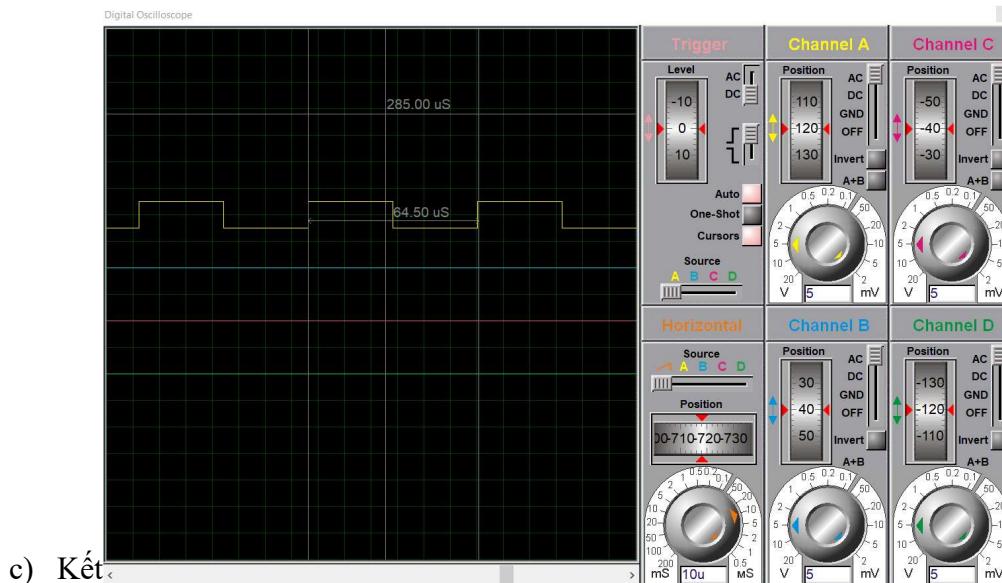
## DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

### 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.



- c) Kết

## LAB 2-1

### DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

#### BÀI 2

- Viết chương trình tạo 1 xung vuông 64 us sử dụng timer 0 ở chế độ Normal mode. Ngõ ra sử dụng chân OC0.
- 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.
- Kết nối chân OC0 ra oscilloscope và quan sát

#### BÀI 3

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

## LAB 2-1

### DÙNG TIMER ĐỂ TẠO DELAY VÀ TẠO XUNG

- 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

```
    OUT TCCR0B, R17      ;Dung timer  
    RET
```

## 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

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         ;3MC  
    RCALL DELAY_32US_CTC ;3MC  
    CBI PORTB, 3         ;3MC  
    RCALL DELAY_32US_CTC ;2MC  
    RJMP START           ;2MC  
//*****TIMER1_MODE_CTC*****//
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
//*****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
        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:
```

# BÁO CÁO

Nhóm môn học: L13

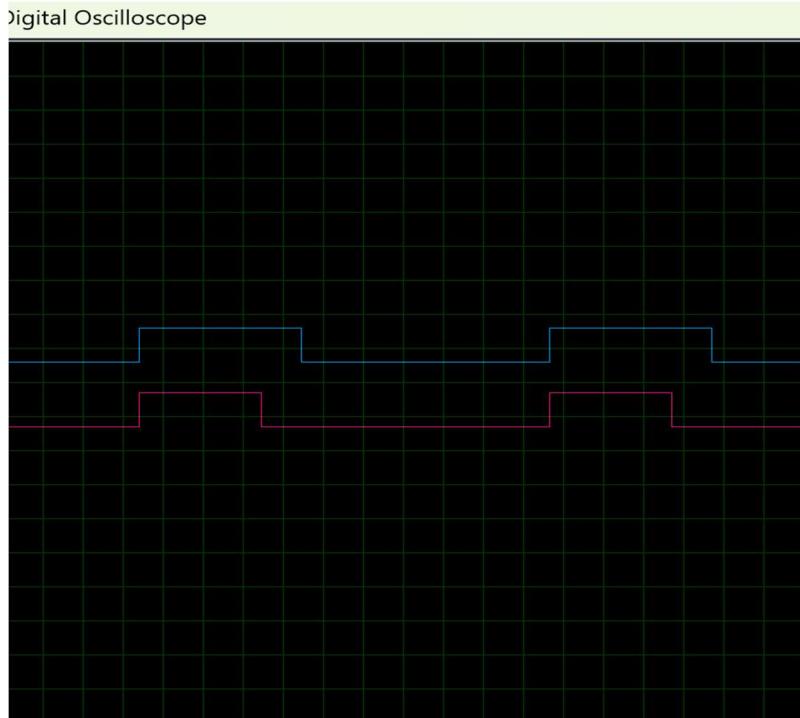
Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
                           ;4MC
RET
```

## BÀI 3

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



- b. Giải thích lý do có dạng sóng (tần số, chu kỳ làm việc, phase) như kết quả

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

-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.

## 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 RJMP MAIN .ORG 0x40  MAIN: LDI R22, 2	; Số lần lặp câu a
---	--------------------

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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 da 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
RJMP CHANGE_VALUE
INC SIGN_H
DEC SIGN_L
RJMP AGAIN
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
```

## 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 2

- d) 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
- e) 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 3

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

#### BÀI 4

- 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.

#### BÀI 2

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

## 2. 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

## 3. 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
    CBI LE, 1           ; lock 573
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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
    NOP
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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 3

### 2. 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 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.

### 3. 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:
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
IN R20, PINA
CALL SAVE_TRAM_CHUC_DONVI
CALL CACULATE ;Kết quả tính toán sẽ lưu vào ô nhớ
$0100 -- $0103

LOOP:
LDI R18, $FF ; TURN OFF LED
LDI XL, $00
LDI XH, $01
LDI R19, $FE
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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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
OUT PORTA, R16          ;Pull-up
LDI R16, $03             ;PortD chan 0 và 1 là 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              ;Kết quả lưu vào R21:R20
PUSH TRAM                ;Dung sau
PUSH CHUC                ;Dung sau
CALL SAVE_TRAM_CHUC_DONVI ;Sau dòng này ta sẽ được hàng
don vi của kết quả
STS $0100, DONVI          ;Lưu hàng đơn vị vào o
nho nay
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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

STS $0101, DONVI ;Luon luu vao la hang don
vi

MOV SONHO, CHUC ;Hang chuc o kqua
truoc chinh la so nho cua phep nhan lan sau

POP TRAM ;Lay hang tram cua
kqua nhap tu portA ra lai

MUL TRAM, R19

MOVW R20, R0

ADD R20, SONHO

CALL SAVE_TRAM_CHUC_DONVI ;Sau dong nay se duoc hang tram
va hang nghin cua kqua

STS $0102, DONVI

STS $0103, CHUC

RET

SAVE_TRAM_CHUC_DONVI:

MOV R18, R20

LDI R21, 10

CLR R22
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

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
```

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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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
    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
```

# BÁO CÁO

Nhóm môn học:

Nhóm:  
Môn thí nghiệm:

```
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:
    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:

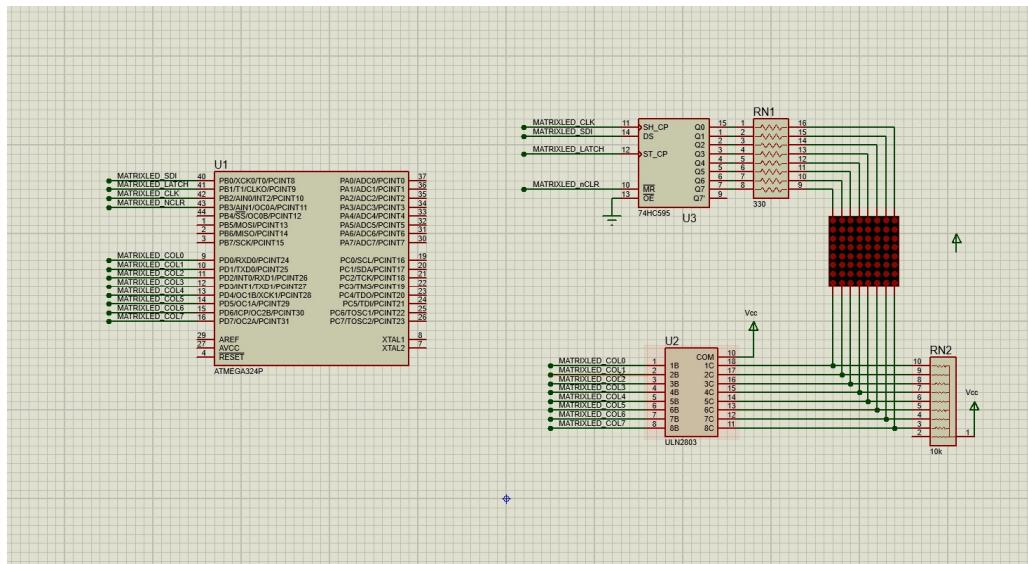
Nhóm môn học:

Môn thí nghiệm:

## BÀI 4

### 1. Trả lời các câu hỏi

#### a. Mô tả kết nối trên kit



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

### 2. Mã nguồn chương trình với chú thích

# 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
```

## BÀI 2

1. Trả lời các câu hỏi
  - a. Các chân SCL, SDA là chân nào của AVR?
  - b. Vẽ hình mô tả kết nối trong bài thí nghiệm
2. Mã nguồn và chú thích

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

## BÀI 3

### 1. Trả lời các câu hỏi

- 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:
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
RCALL USART_RECEIVER_CHAR
MOV DATA_SPI_TX, DATA_RX
CBI PORT_SPI, SS
RCALL SPI_MASTER_TRANSMIT
SBI PORT_SPI, SS
RJMP AGAIN

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 UBRR0H, R16
LDI R16, 51
STS UBRR0L, 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

;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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
;-----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  
.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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6  
Môn thí nghiệm: Vi xử lý

```
POP R16
RET

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

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
    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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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:
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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi xử lý

```
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
```

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
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6  
Môn thí nghiệm: Vi xử lý

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

```
    POP R17  
  
    RET
```

```
UART0_RX:  
  
    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)

### BÀI 2

- a) 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.
- b) 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.W
- c) Biên dịch chương trình và quan sát oscilloscope để kiểm tra chương trình.

### BÀI 3

- a) Kết nối các tín hiệu cần thiết để điều khiển khối LED 7 đoạn.

# LAB 4-1

## LẬP TRÌNH SỬ DỤNG NGẮT

- 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 kết tiếp và đo xung này trên oscilloscope.  
(Tham khảo Chương 4, tài liệu hướng dẫn thí nghiệm)

### 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 ngắt, 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.

### BÀI 5

- 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:
- Khởi động RTC với thời gian hiện hành, xuất tín hiệu MFP tần số 1 Hz.
- 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.

### BÀI 6

# LAB 4-1

## LẬP TRÌNH SỬ DỤNG NGẮT

- a) Kết nối các tín hiệu cần thiết để điều khiển LED ma trận. (gỡ bỏ các dây nối đèn LCD, Led 7 đoạn, RTC, UART).
- 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 ngắt 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.

# BÁO CÁO

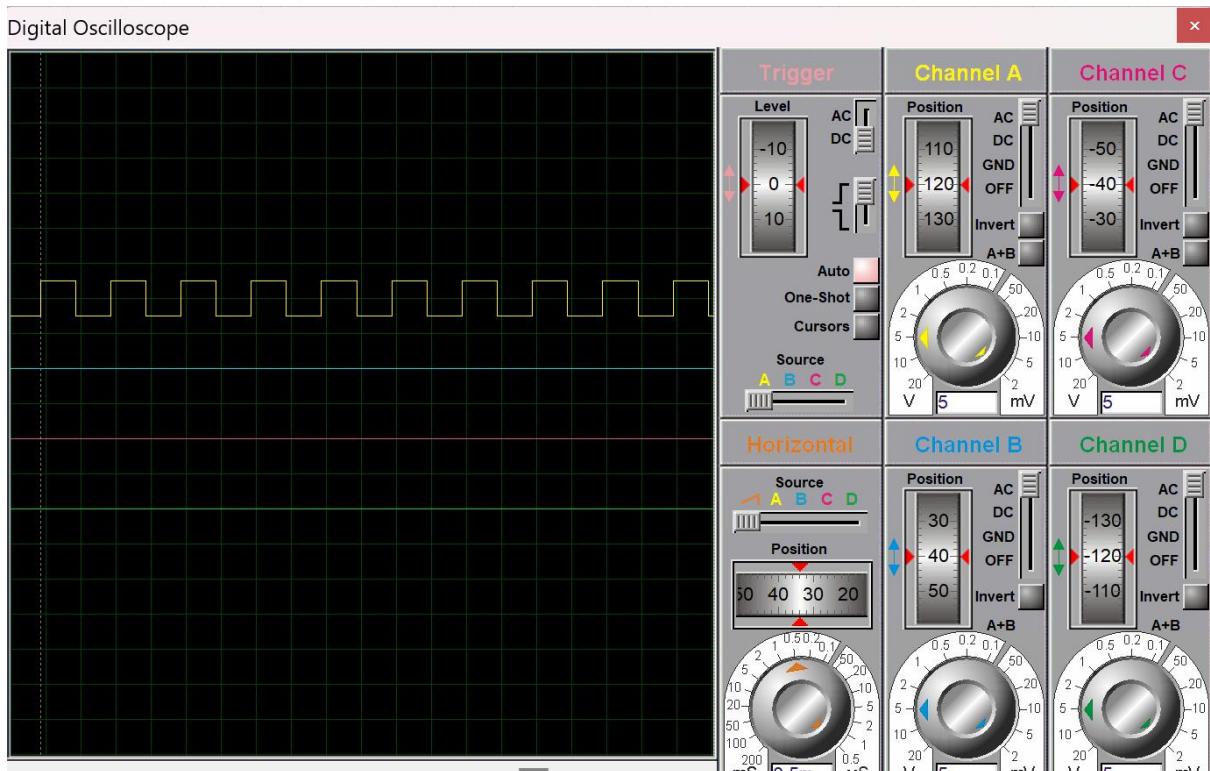
Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

## BÀI 1

1. Trả lời các câu hỏi
  - a. Với chế độ Normal mode, mỗi khi vào ngắt OverFlow ta có phải khởi động lại thanh ghi đếm hay không?  
- Khi timer 1 tràn, trong chương trình phục vụ ngắt đảo chân PC0 và đặt lại giá trị cho thanh ghi đếm
  - b. Giải thích các giá trị ghi vào các thanh ghi cấu hình timer và prescaler.
2. Mã nguồn chương trình với chú thích



# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
; T = 1 ms -> T/2 = 0.5ms -> Mode NOR, N = 128, TCNT1 = -31
.EQU P_OUT = 0
.ORG 0
.RJMP MAIN
.ORG 0x1E ; Vecto ngắt tràn Timer1
.RJMP TIMER1_OVER_ISR
.ORG 0x40

MAIN:
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R16, (1 << P_OUT)
    OUT DDRB, R16 ; PC0 ngõ ra
    LDI R17, 0x00
    STS TCNT1H, R17
    LDI R17, -31 ; Nạp giá trị đếm
    STS TCNT1L, R17
    LDI R17, 0x00 ; WG01:00 = 00 Timer0 mode NOR
    STS TCCR1A, R17
    LDI R17, 0x04 ; WG02 = 0, CS02:00 = 001 : Timer 0
NOR mode, N = 128
    STS TCCR1B, R17
    SEI ; Set cờ I lên 1, cho phép ngắt toàn
cục
    LDI R17, (1 << TOIE1) ; Cho phép ngắt tràn Timer1
    STS TIMSK1, R17
START: RJMP START
; -----
TIMER1_OVER_ISR: ; ISR chạy khi cờ báo tràn Timer0 set
lên 1
    LDI R17, 0x00
    STS TCCR1B, R17 ; Dừng Timer
    LDI R17, 0x00
    STS TCNT1H, R17
    LDI R17, -31 ; Nạp giá trị đếm
    STS TCNT1L, R17
    IN R17, PORTC ; Lấy trạng thái PORTC
    EOR R17, R16 ; Đảo bit PC1
    OUT PORTB, R17
    LDI R17, 0x04 ; Timer1 NOR mode, N = 128
    STS TCCR1B, R17
    RETI
```

## BÀI 2

- Trả lời các câu hỏi

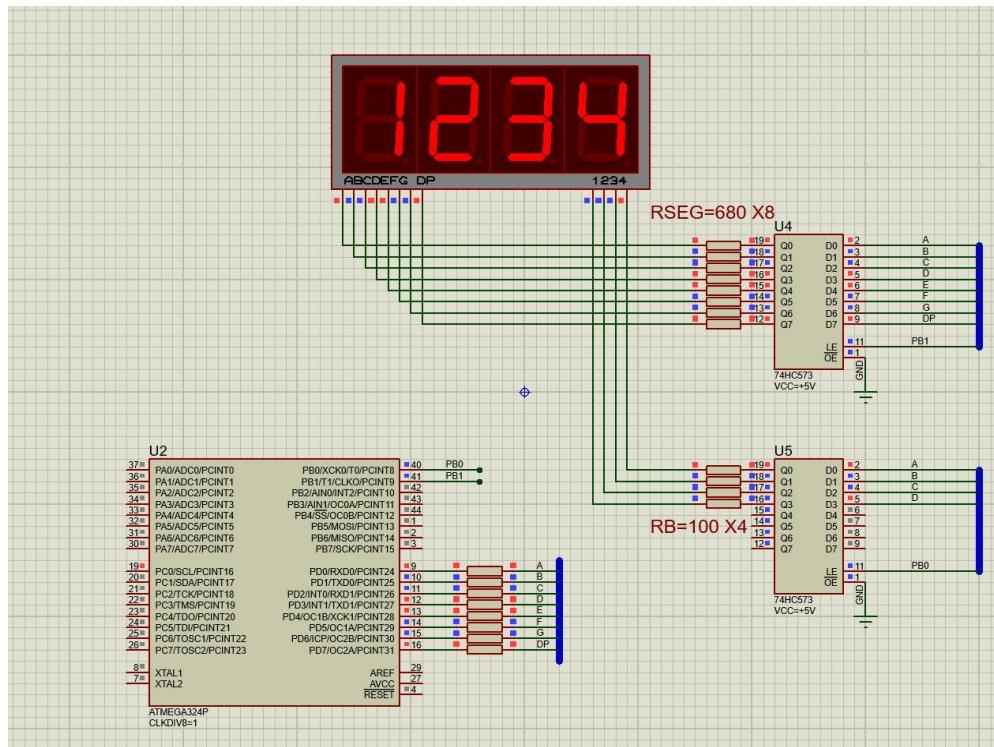
# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

- a. Với chế độ CTC mode, mỗi khi vào ngắt COMPARE\_MATCH ta có phải khởi động lại thanh ghi đếm hay không?
  - b. Điểm tiện lợi của chế độ này so với cách cấu hình ở bài 1 là như thế nào?
  - c. Giải thích các giá trị ghi vào các thanh ghi cấu hình timer và prescaler.
2. Mã nguồn và chú thích



a)

```
; T = 0.01s -> T/2 = 5ms -> Mode CTC, N = 64, n = 645, OCR1A = TOP = 624
.EQU P_OUT = 0
.ORG 0
RJMP MAIN
.ORG 0x1E ; Vecto ngắt tràn Timer1
RJMP TIMER1_OVER_ISR
.ORG 0x40
MAIN:
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
LDI    R16, HIGH(RAMEND)
OUT   SPH, R16
LDI    R16, LOW(RAMEND)
OUT   SPL, R16
LDI    R16, (1 << P_OUT)
OUT   DDRB, R16           ; PC0 ngõ ra
LDI    R17, HIGH(624)
STS   OCR1AH, R17
LDI    R17, LOW(624)
LDI    R17, 0x02           ; WG01:00 = 00 Timer1 mode CTC
STS   TCCR1A, R17
LDI    R17, 0x03           ; WG02 = 0, CS02:00 = 011 : Timer 1 NOR
mode, N = 64
STS   TCCR1B, R17
SEI
LDI    R17, (1 << TOIE1) ; Set cờ I lên 1, cho phép ngắt toàn cục
STS   TIMSK1, R17         ; Cho phép ngắt tràn Timer1
START: RJMP START
;-----
TIMER1_OVER_ISR:
LDI    R17, 0x00
STS   TCCR1B, R17         ; Dừng Timer
IN    R17, PORTC          ; Lấy trạng thái PORTC
EOR   R17, R16            ; Đảo bit PC1
OUT   PORTB, R17
LDI    R17, 0x04           ; Timer1 CTC mode, N = 64
STS   TCCR1B, R17
RETI

b)

; COMPARE_MATCH
; T = 0.01s -> T/2 = 5ms -> Mode CTC, N = 64, n = 125, OCR1A = TOP = 124
.EQU  P_OUT = 0
.DEF  COUNT = R24          ; Thành ghi đếm số lần ngắt
.ORG  0
RJMP  MAIN
.ORG  0x1E                ; Vecto ngắt tràn Timer1
RJMP  TIMER1_OVER_ISR
.ORG  0x40

MAIN:
LDI    R16, HIGH(RAMEND)
OUT   SPH, R16
LDI    R16, LOW(RAMEND)
OUT   SPL, R16
LDI    R16, (1 << P_OUT)
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
OUT    DDRB, R16           ; PC0 ngõ ra
LDI    R17, HIGH(124)
STS    OCR1AH, R17
LDI    R17, LOW(124)
LDI    R17, 0x02           ; WG01:00 = 00 Timer1 mode CTC
STS    TCCR1A, R17
LDI    R17, 0x03           ; WG02 = 0, CS02:00 = 011 : Timer 1 NOR
mode, N = 64
STS    TCCR1B, R17
SEI
LDI    R17, (1 << TOIE1)   ; Set cờ I lên 1, cho phép ngắt toàn cục
STS    TIMSK1, R17
START: RJMP   START
;-----
TIMER1_OVER_ISR:
INC    COUNT
CPI    COUNT, 5
BREQ  INVERT
RETI
INVERT:
IN     R17, PORTC          ; Lấy trạng thái PORTC
EOR    R17, R16            ; Đảo bit PC1
OUT    PORTB, R17
LDI    R24, 0               ; Reset số lần đếm tràn lại
0
RETI
```

## BÀI 3

1. Trả lời các câu hỏi
  - a. Để đạt được tần số quét 50Hz thì 1 LED sẽ sáng trong bao lâu?
  - b. Khi đó, chân PC0 (đảo mỗi khi chuyển LED) sẽ có tần số là bao nhiêu?
  - c. Số lần xảy ra ngắt cần thiết để chuyển LED là bao nhiêu?
2. Mã nguồn và chú thích

```
.org 0x0000 ; interrupt vector table
rjmp reset_handler ; reset
.org 0x001A
rjmp timer1_COMP_ISR
reset_handler:
; initialize stack pointer
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ldi r16, high(RAMEND)
out SPH, r16
ldi r16, low(RAMEND)
out SPL, r16
ldi r16, (1<<PCIE0)
sts PCICR, r16
call initTimer1CTC
call led7seg_portinit
call Led7seg_buffer_init
; enable global interrupts
sei
main:
jmp main
; Lookup table for 7-segment codes
table_7seg_data:
.DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
.DB 0XC6,0XA1,0X86,0X8E
; Lookup table for LED control
table_7seg_control:
.DB 0b00001110,0b00001101, 0b00001011, 0b00000111
; J34 connect to PORTD
; nLE0 connect to PB4
; nLE1 connect to PB5
; Output: None
.equ LED7SEGPORT = PORTD
.equ LED7SEGDIR = DDRD
.equ LED7SEGLatchPORT = PORTB
.equ LED7SEGLatchDIR = DDRB
.equ nLE0Pin = 4
.equ nLE1Pin = 5
.dseg
.org SRAM_START ;starting address is 0x100
LED7segValue: .byte 4 ;store the BCD value to display
LED7segIndex: .byte 1
.cseg
.align 2
;init the Led7seg buffer
Led7seg_buffer_init:
push r20
ldi r20,3 ;LED index start at 3
ldi r31,high(LED7segIndex)
ldi r30,low(LED7segIndex)
st z,r20
ldi r20,0
ldi r31,high(LED7segValue)
ldi r30,low(LED7segValue)
st z+,r20 ;display value is 0-1-2-3
inc r20
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
st z+,r20
inc r20
st z+,r20
inc r20
st z+,r20
pop r20
ret
led7seg_portinit:
push r20
ldi r20, 0b11111111 ; SET led7seg PORT as output
out LED7SEGDIR, r20
in r20, LED7SEGLatchDIR ; read the Latch Port direction register
ori r20, (1<<nLE0Pin) | (1 << nLE1Pin)
out LED7SEGLatchDIR,r20
pop r20
ret;
;Display a value on a 7-segment LED using a lookup table
; Input: R27 contains the value to display
; R26 contain the LED index (3..0)
; J34 connect to PORTD
; nLE0 connect to PB4
; nLE1 connect to PB5
; Output: None
display_7seg:
push r16 ; Save the temporary register
; Look up the 7-segment code for the value in R18
; Note that this assumes a common anode display, where a HIGH output
turns OFF
the segment
; If using a common cathode display, invert the values in the table
above
ldi zh,high(table_7seg_data<<1) ;
ldi zl,low(table_7seg_data<<1) ;
clr r16
add r30, r27
adc r31,r16
lpm r16, z
out LED7SEGPORt,r16
sbi LED7SEGLatchPORT,nLE0Pin
nop
cbi LED7SEGLatchPORT,nLE0Pin
ldi zh,high(table_7seg_control<<1) ;
ldi zl,low(table_7seg_control<<1) ;
clr r16
add r30, r26
adc r31,r16
lpm r16, z
out LED7SEGPORt,r16
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
sbi LED7SEGLatchPORT,nLE1Pin
nop
cbi LED7SEGLatchPORT,nLE1Pin
pop r16 ; Restore the temporary register
ret ; Return from the function
initTimer1CTC:
push r16
ldi r16, high(10000) ; Load the high byte into the temporary register
sts OCR1AH, r16 ; Set the high byte of the timer 1 compare value
ldi r16, low(10000) ; 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 0b00000101 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
push r26
push r27
ldi r31,high(LED7segIndex)
ldi r30,low(LED7segIndex)
ld r16,z
mov r26,r16
ldi r31,high(LED7segValue)
ldi r30,low(LED7segValue)
add r30,r16
clr r16
adc r31,r16
ld r27,z
call display_7seg
cpi r26,0
brne timer1_COMP_ISR_CONT
ldi r26,4 ;if r16 = 0, reset to 3
timer1_COMP_ISR_CONT:
dec r26 ;else, decrease
ldi r31,high(LED7segIndex)
ldi r30,low(LED7segIndex)
st z,r26
pop r27
pop r26
pop r16
reti
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

## BÀI 4

1. Trả lời các câu hỏi
  - a. Mô tả các kết nối trên kit.
  - b. Mô tả cách cấu hình UART
  - c. Chuyển đổi số đếm thành số BCD để ghi vào bộ đếm hiển thị LED như thế nào?
2. Mã nguồn và chú thích

```
.org 0
rjmp ioset
.org $28
rjmp uart0_rx_isr
.org $40
ioset:
    ldi r16,0xff
    out ddra,r16
    ldi r16,0xf0
    out ddrb,r16
    ldi r16,0xf8
    out ddrc,r16
    ldi r16,0
    out portb,r16
    out portd,r16
    out porta,r16
stack:
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ldi r16,high(ramend)
out sph,r16
ldi r16,low(ramend)
out spl,r16

main:
rcall set_power_lcd
rcall set_monitor
ldi r16,0
sts ubrr0h,r16
ldi r16,51
sts ubrr0l,r16
ldi r16,6
sts ucsr0c,r16
ldi r20,0
ldi r21,9
ldi r22,9
ldi r23,8
ldi zh,high(number<<1)
ldi zl,low(number<<1)
sei
ldi r16,0x90
sts ucsr0b,r16
start:
xuat_donvi:
    mov r30,r23
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
lpm r24,z  
ldi r16,1  
rcall selected_led_7segs  
rcall out_led_7segs  
ldi r17,100  
rcall delay_100us  
  
xuat_chuc:  
    mov r30,r22  
    lpm r24,z  
    ldi r16,2  
    rcall selected_led_7segs  
    rcall out_led_7segs  
    ldi r17,100  
    rcall delay_100us  
  
xuat_tram:  
    mov r30,r21  
    lpm r24,z  
    ldi r16,4  
    rcall selected_led_7segs  
    rcall out_led_7segs  
    ldi r17,100  
    rcall delay_100us  
  
xuat_nghin:  
    mov r30,r20  
    lpm r24,z
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ldi r16,8
rcall selected_led_7segs
rcall out_led_7segs
ldi r17,100
rcall delay_100us
rjmp start

uart0_rx_isr:
lds r18,udr0
push r18
rcall start_lcd
pop r18
rcall out_data
rcall count_led_7segs
reti

set_power_lcd:
ldi r17,200
rcall delay_100us
ldi r18,0x30
rcall out_lenh
ldi r17,50
rcall delay_100us
ldi r18,0x30
rcall out_lenh
ldi r17,1
rcall delay_100us
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ldi r18,0x30
rcall out_lenh
ldi r17,1
rcall delay_100us
ldi r18,0x20
rcall out_lenh
set_monitor:
ldi r17,20
rcall delay_100us
ldi r18,0x24
rcall out_2lenh
ldi r17,20
rcall delay_100us
ldi r18,0x01
rcall out_2lenh
ldi r17,20
rcall delay_100us
ldi r18,0x0c
rcall out_2lenh
ldi r17,20
rcall delay_100us
ldi r18,0x06
rcall out_2lenh
ldi r17,20
rcall delay_100us
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ret

start_lcd:
    ldi r17,1
    rcall delay_100us
    ldi r18,1
    rcall out_2lenh
    ldi r17,20
    rcall delay_100us
    ldi r18,0x80
    rcall out_2lenh
    ret

out_lenh:
    cbi portd,5
    out portb,r18
    sbi portd,7
    cbi portd,7
    ret

out_2lenh:
    rcall out_lenh
    ldi r17,1
    rcall delay_100us
    swap r18
    rcall out_lenh
    ret

out_data:
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
sbi portd,5
out portb,r18
sbi portd,7
cbi portd,7
ldi r17,1
rcall delay_100us
swap r18
out portb,r18
sbi portd,7
cbi portd,7
ret

delay_100us:
l1: ldi r19,200
again:
nop
dec r19
brne again
dec r17
brne l1
ret

selected_led_7segs:
out porta,r16
sbi portd,3
cbi portd,3
ret
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
out_led_7segs:  
    out porta,r24  
    sbi portd,4  
    cbi portd,4  
    ret  
  
count_led_7segs:  
    cpi r20,1  
    breq reset_zero  
    cpi r23,9  
    breq reset_donvi  
    inc r23  
    rjmp end_funtion  
  
reset_donvi:  
    ldi r23,0  
    cpi r22,9  
    breq reset_chuc  
    inc r22  
    rjmp end_funtion  
  
reset_chuc:  
    ldi r22,0  
    cpi r21,9  
    breq reset_tram  
    inc r21  
    rjmp end_funtion  
  
reset_tram:
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
ldi r21,0
inc r20
rjmp end_funtion

reset_zero:
    ldi r20,0
    ldi r21,0
    ldi r22,0
    ldi r23,0

end_funtion:
    ret
.org $400
number: .db 0x40,0x79,0x24,0x30,0x19,0x12,0x02,0x78,0x00,0x10
```

## BÀI 5

1. Trả lời các câu hỏi
  - a. Mô tả kết nối trên kit
  - b. Để xảy ra ngắt 1 s 1 lần thì ngắt ngoài cấu hình như thế nào?
  - c. Hiện tượng gì sẽ xảy ra nếu ta cấu hình ngắt ngoài tích cực theo mức thấp?
2. Mã nguồn chương trình với chú thích

## BÀI 6

1. Trả lời các câu hỏi

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

---

- a. Mô tả kết nối trên kit
  - b. Để có tần số quét 25Hz thì một cột LED sáng trong bao lâu?
  - c. Sự khác nhau khi quét ở tần số 25Hz và 125Hz
2. Mã nguồn chương trình với chú thích

## LAB 4-2

### GIAO TIẾP I/O VÀ CÁC LỆNH TÍNH TOÁN

#### MỤC TIÊU:

- Hiểu và sử dụng được ngắt timer
- Hiểu cách điều khiển và đo tốc độ động cơ

#### THAM KHẢO:

- Tài liệu hướng dẫn thí nghiệm, chương 12
- Atmel-2505-Setup-and-Use-of-AVR-Timers\_ApplicationNote\_AVR130.pdf

#### BÀI 1

Yêu cầu:

- a. Viết chương trình điều khiển tốc độ động cơ DC dùng PWM với tần số 1 KHz, sử dụng timer 0. Điều khiển tốc độ tăng/giảm sử dụng 2 nút nhấn, mỗi lần nhấn nút tăng/giảm duty cycle 5%. Cho phép động cơ chạy/dừng và điều khiển động cơ quay thuận/ngược bằng 2 switch trên dip switch.
  - Kết nối động cơ vào kit thí nghiệm
  - Kết nối tín hiệu từ 2 switch trên dipswitch vào 2 chân port của AVR
  - Kết nối tín hiệu từ 2 nút nhấn vào 2 chân port của AVR
  - Kết nối tín hiệu từ chân OC0B ra 1 kênh đo của khối test point
  - Kết nối tín hiệu từ 2 chân port điều khiển chiều quay thuận/ngược ra led đơn để kiểm tra trạng thái.
- b. Biên dịch, thực thi và kiểm tra hoạt động của chương trình bằng cách đo dạng sóng trên oscilloscope và quan sát trạng thái các LED khi thay đổi dip switch và khi nhấn các nút nhấn tăng/giảm tốc độ.

## LAB 4-2

### GIAO TIẾP I/O VÀ CÁC LỆNH TÍNH TOÁN

- c. Kết nối tín hiệu PWM vào MOTOR\_ENABLE, tín hiệu điều khiển chiều quay vào MOTOR\_CTRL1, MOTOR\_CTRL2 trên J76 của khối DC\_MOTOR.
  
- d. Kiểm tra hoạt động của hệ thống.
- e. Đo dạng sóng từ hai tín hiệu A-B của encoder và so sánh trong hai trường hợp động cơ quay thuận hay ngược.

#### BÀI 2

Yêu cầu:

Thêm vào bài 1 ở trên chức năng đo tốc độ động cơ và hiển thị tốc độ, chiều quay lên LCD.

Tín hiệu từ 1 kênh của encoder đưa vào ngõ vào clock cho timer 2, sử dụng timer 2 ở chế độ dùng clock ngoài. Khi timer 2 tràn sẽ sinh ra một ngắt, khi đó ta tăng 1 số đếm lên để đếm số lần tràn của timer.

Hai kênh từ encoder cũng được đưa vào 2 chân port để xác định chiều quay thuận/ngược.

Thời gian 1 s tạo ra sử dụng ngắt timer 1. Khi ngắt xảy ra, đếm số lượng xung từ encoder trong vòng 1 s và tính ra tốc độ động cơ, đưa lên LCD. Reset các số đếm để bắt đầu lại quá trình đo

#### BÀI 1

- 3. Trả lời các câu hỏi
  - c. Mô tả cách kết nối trên kit
  - d. Chụp ảnh dạng xung của 2 kênh encoder trong trường hợp quay thuận và quay nghịch.
- 4. Mã nguồn chương trình với chú thích

```
.EQU MOTOR_PORT=DDRB
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
.EQU MOTOR_DIRECT=PORTB
.EQU MOTOR_CTR_1=1
.EQU MOTOR_CTR_2=2
.EQU MOTOR_EN_A=0

.EQU SPEED_PORT=DDRC
.EQU SPEED_DIRECT=PORTC
.EQU INC_BUT=0
.EQU DEC_BUT=1

.EQU CONTROL_PORT=DDRD
.EQU CONTROL_DIRECT=PORTD
.EQU RIGHT_BUT=1
.EQU LEFT_BUT=0

.ORG 0X00
RJMP MAIN
.ORG 0X0C
RJMP PCINT
.ORG 0X0E
RJMP PDINT
.ORG 0X40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

```
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16,
(1<<MOTOR_CTR_1)|(1<<MOTOR_CTR_2)|(1<<MOTOR_EN_A)
OUT MOTOR_PORT, R16
OUT MOTOR_DIRECT, R16
LDI R16, (1<<INC_BUT)|(1<<DEC_BUT)
OUT SPEED_PORT, R16
LDI R16, (1<<RIGHT_BUT)|(1<<LEFT_BUT)
OUT CONTROL_PORT, R16

LDI R16, 0X00
OUT TCNT0, R16
LDI R19, 0X7E
OUT OCR0A, R19
LDI R16, 0X01
OUT TCCR0A, R16
LDI R16, 0X11
OUT TCCR0B, R16
SEI
LDI R16, (1<<PCIE2)|(1<<PCIE3)
STS PCICR, R16

START:
RJMP START
```

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

---

PCINT:

IN R16, PORTC

SBRC R16, 0

RJMP INCREASE

RJMP SSS

RETI

SSS:

SBRS R16, 1

RETI

RJMP DECREASE

INCREASE:

INC R19

OUT OCR0A, R19

RETI

DECREASE:

DEC R19

OUT OCR0A, R19

RETI

PDINT:

IN R16, PORTD

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

---

SBRC R16, 0

RJMP AAA

RJMP DDD

RETI

AAA:

SBRS R16, 1

RJMP CASE\_3

RJMP CASE\_4

DDD:

SBRS R16, 1

RJMP CASE\_1

RJMP CASE\_2

CASE\_1:

LDI R16, (0<<MOTOR\_CTR\_1)|(0<<MOTOR\_CTR\_2)

OUT MOTOR\_PORT, R16

RETI

CASE2:

LDI R16, (0<<MOTOR\_CTR\_1)|(1<<MOTOR\_CTR\_2)

OUT MOTOR\_PORT, R16

RETI

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

CASE3:

LDI R16, (1<<MOTOR\_CTR\_1)|(0<<MOTOR\_CTR\_2)

OUT MOTOR\_PORT, R16

RETI

CASE4:

LDI R16, (1<<MOTOR\_CTR\_1)|(1<<MOTOR\_CTR\_2)

OUT MOTOR\_PORT, R16

RETI

## BÀI 2

### 3. Trả lời các câu hỏi

- a. Mô tả kết nối trên
- b. Với động cơ trong thí nghiệm, công thức tính tốc độ như thế nào?
- c. Timer 1 có thể được sử dụng để tạo ngắt sau mỗi 1 s hay không?
- d. Cấu hình timer 1 như thế nào?
- e. Thanh ghi đếm của timer 2 là thanh ghi bao nhiêu bit?
- f. Để sử dụng timer 2 đếm số xung trong vòng 1s, có xảy ra trường hợp tràn timer 2 không?

HD: từ tốc độ tối đa của động cơ, tỉ số truyền, số xung/vòng ta tính ra số xung tối đa trong 1s.

- g. Nếu có khả năng tràn timer 2, ta phải làm gì để đảm bảo tính đúng số xung encoder?

### 4. Mã nguồn và chú thích

# BÁO CÁO

Nhóm môn học: L13

Nhóm: 6

Môn thí nghiệm: Vi Xử Lý

---

# 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<sub>REF</sub> là điện áp nội V<sub>CCA</sub>. 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

```
.EQU ADC_PORT=PORTA
.EQU ADC_DR=DDRA
.EQU ADC_IN=PIN4
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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 TACKITU
MOV R17, R1
RCALL TACKITU
LDI R17, ' '
RCALL PHAT
RCALL DELAY1S
RJMP START ;tiếp tục chuyển đổi
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

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
NUM: LDI R18,0X30
CHAR: ADD R18,R17
RET
TACHKITU :
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

- d) Thay đổi điện áp đưa vào ADC0, đo bằng VOM và so sánh với kết quả lấy mẫu ADC, điều 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<sub>REF</sub> (header J57), sử dụng VOM.

```
.EQU ADC_PORT=PORTA
.EQU ADC_DR=DDRA
.EQU ADC_IN=PINB
.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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
OUT DDRB, R16
OUT DDRC, 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,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 p?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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
RCALL TACKKITU
MOV R17,R1
RCALL TACKKITU
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
MOV R17,R18
SBI CONT,RS
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
NOP
BRNE LP_3
DEC R17
BRNE LP_2
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
MOV R12,R0
MOV R13,R1
ADD R12,R11
CLR R0
ADC R13,R0 ;R13:R12:R10
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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)
;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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
OR R17,R16
RCALL OUT_LCD
RET
OUT_LCD:
OUT LCD,R17 ;1MC,ghi l?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:
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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 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
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:
```

# LAB 5-1

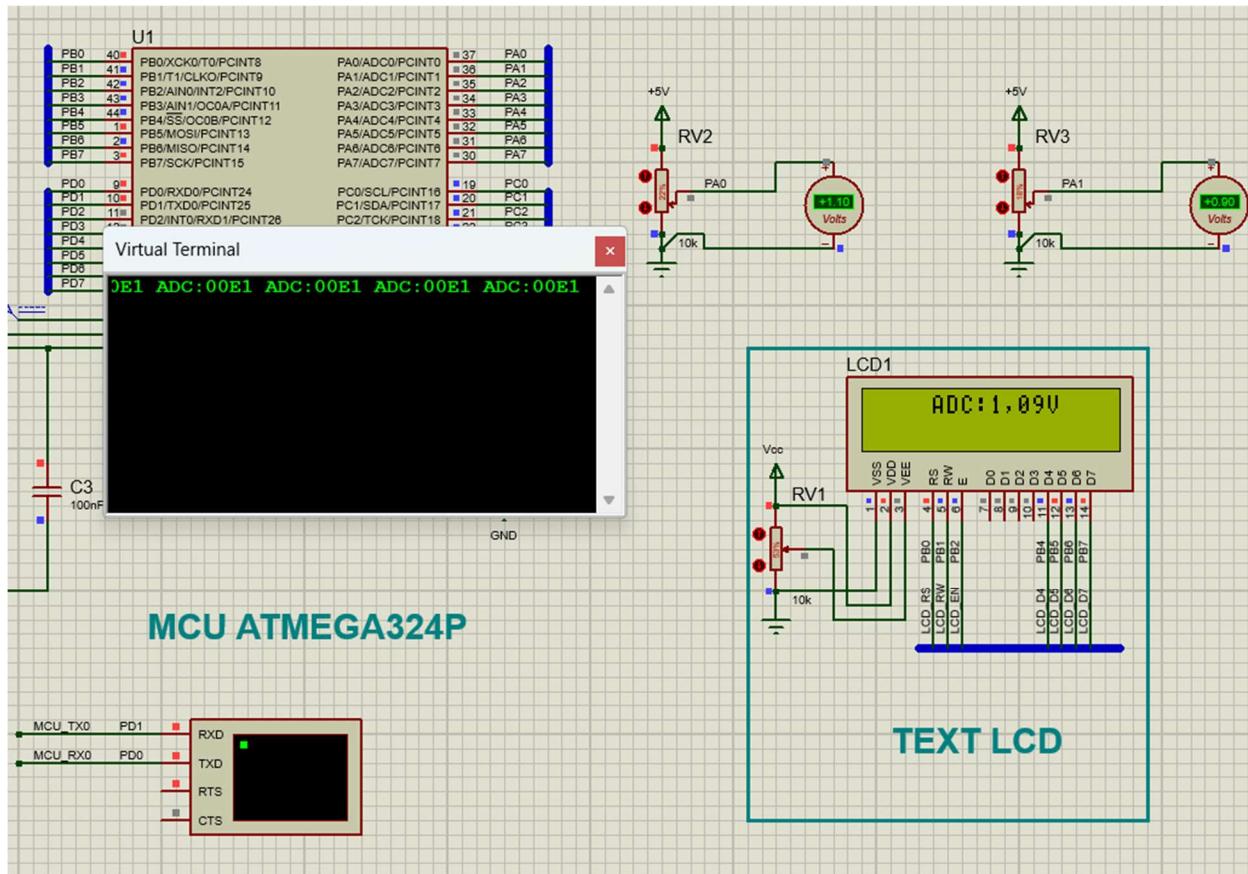
## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC



# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

### 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<sub>CCA</sub>, gửi lên máy tính các giá trị ADC đã hiệu chỉnh, và xuất ra LCD giá trị điện áp đo được
- (Đọc kỹ tài liệu: AVR120: Characterization and Calibration of the ADC on an AVR)

```
.org 0
rjmp ioset
.org $40
ioset:
ldi r16,0xf7
out ddrb,r16
ldi r16,0
out ddra,r16
out portb,r16
ldi r16,0xff
out ddrc,r16
stack:
ldi r16,high(ramend)
out sph,r16
ldi r16,low(ramend)
out spl,r16
set_timer1:
ldi r16,0
sts tccr1a,r16
ldi r16,0x7a
sts ocr1ah,r16
sts ocr1bh,r16
ldi r16,0x11
sts ocr1al,r16
sts ocr1bl,r16
ldi r16,12
sts tccr1b,r16
set_uart:
ldi r16,24
sts ucsr0b,r16
ldi r16,6
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
sts ucsr0c,r16
ldi r16,0
sts ubrr0h,r16
ldi r16,51
sts ubrr0l,r16
set_mode_adc:
ldi r16,64
sts admux,r16
ldi r16,166
sts adcsra,r16
ldi r16,5
sts adcsrb,r16
main:
rcall set_lcd_power
rcall set_lcd_screen
start:
lds r16,adcsra
sbrs r16,adif
rjmp start
sts adcsra,r16
rcall start_lcd
lds r17,adcl
lds r18,adch
rcall xuat_volt_lcd
out portc,r18
rcall out_hercules
in r16,tifr1
out tifr1,r16
rjmp start
;=====
uart_trans:
lds r16,ucsr0a
sbrs r16,udre0
rjmp uart_trans
sts udr0,r18
ret
;=====
out_hercules:
ldi r28,15
ldi r29,10
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi zh,high(hex<<1)
ldi zl,low(hex<<1)
mov r0,r17
mov r1,r18
ldi r18,'A'
rcall uart_trans
ldi r18,'D'
rcall uart_trans
ldi r18,'C'
rcall uart_trans
ldi r18,':'
rcall uart_trans
ldi r18,'_'
rcall uart_trans
ldi r18,'0'
rcall uart_trans
ldi r18,'x'
rcall uart_trans
ldi r18,'0'
rcall uart_trans
mov r30,r1
lpm r18,z
rcall uart_trans
mov r1,r0
swap r1
and r1,r28
cp r1,r29
brlo xuat1
rcall address_hex1
xuat1:
mov r30,r1
lpm r18,z
rcall uart_trans
ldi zh,high(hex<<1)
and r0,r28
mov r1,r0
cp r1,r29
brlo xuat2
rcall address_hex1
xuat2:
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
mov r30,r1
lpm r18,z
rcall uart_trans
ldi r18,' '
rcall uart_trans
ret
=====
address_hex1:
ldi zh,high(hex1<<1)
ldi zl,low(hex1<<1)
sub r1,r29
ret
=====
set_lcd_power:
ldi r16,250
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,50
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,10
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,10
rcall delay_100us
ldi r19,0x20
rcall out_lcd
ret
=====
set_lcd_screen:
ldi r16,20
rcall delay_100us
ldi r19,0x24
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x01
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x0c
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x06
rcall out_2lenh
ldi r16,20
rcall delay_100us
ret
=====
start_lcd:
ldi r16,2
rcall delay_100us
ldi r19,1
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x83
rcall out_2lenh
ret
=====
out_lcd:
cbi portb,0
out portb,r19
sbi portb,2
cbi portb,2
ret
=====
out_2lenh:
push r19
andi r19,240
rcall out_lcd
ldi r16,1
rcall delay_100us
pop r19
swap r19
andi r19,240
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall out_lcd
ret
;=====
out_data:
push r19
andi r19,240
out portb,r19
sbi portb,0
sbi portb,2
cbi portb,2
ldi r16,1
rcall delay_100us
pop r19
swap r19
andi r19,240
out portb,r19
sbi portb,0
sbi portb,2
cbi portb,2
ret
;=====
delay_100us:
11: ldi r25,200
again:
nop
dec r25
brne again
dec r16
brne 11
ret
;=====
hex_to_dec:
clr r8
ldi r16,10
loop1:
cp r9,r16
brlo end_funtion
sub r9,r16
inc r8
rjmp loop1
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
end_funtion:  
ret  
=====  
xuat_volt_lcd:  
clr r2  
clr r3  
clr r4  
clr r5  
clr r6  
clr r7  
bit_0:  
sbrs r17,0  
rjmp bit_1  
rjmp solve_bit0  
bit_1:  
sbrs r17,1  
rjmp bit_2  
rjmp solve_bit1  
bit_2:  
sbrs r17,2  
rjmp bit_3  
rjmp solve_bit2  
bit_3:  
sbrs r17,3  
rjmp bit_4  
rjmp solve_bit3  
bit_4:  
sbrs r17,4  
rjmp bit_5  
rjmp solve_bit4  
bit_5:  
sbrs r17,5  
rjmp bit_6  
rjmp solve_bit5  
bit_6:  
sbrs r17,6  
rjmp bit_7  
rjmp solve_bit6  
bit_7:  
sbrs r17,7
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rjmp bit_8
rjmp solve_bit7
bit_8:
sbrs r18,0
rjmp bit_9
rjmp solve_bit8
bit_9:
sbrs r18,1
rjmp end_function1
rjmp solve_bit9
end_function1:
rcall xuat_volt
ret
solve_bit0:
ldi r26,8
mov r15,r26
mov r14,r26
ldi r26,4
mov r13,r26
clr r12
clr r11
clr r10
rcall bit_n
rjmp bit_1
solve_bit1:
ldi r26,7
mov r15,r26
mov r14,r26
ldi r26,9
mov r13,r26
clr r12
clr r11
clr r10
rcall bit_n
rjmp bit_2
solve_bit2:
ldi r26,3
mov r15,r26
ldi r26,5
mov r14,r26
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi r26,9
mov r13,r26
ldi r26,1
mov r12,r26
clr r11
clr r10
rcall bit_n
rjmp bit_3
solve_bit3:
ldi r26,6
mov r15,r26
ldi r26,0
mov r14,r26
ldi r26,9
mov r13,r26
ldi r26,3
mov r12,r26
clr r11
clr r10
rcall bit_n
rjmp bit_4
solve_bit4:
ldi r26,3
mov r15,r26
ldi r26,1
mov r14,r26
ldi r26,8
mov r13,r26
ldi r26,7
mov r12,r26
clr r11
clr r10
rcall bit_n
rjmp bit_5
solve_bit5:
ldi r26,5
mov r15,r26
ldi r26,2
mov r14,r26
ldi r26,6
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
mov r13,r26
ldi r26,5
mov r12,r26
ldi r26,1
mov r11,r26
clr r10
rcall bit_n
rjmp bit_6
solve_bit6:
clr r15
ldi r26,5
mov r14,r26
ldi r26,2
mov r13,r26
ldi r26,1
mov r12,r26
ldi r26,3
mov r11,r26
clr r10
rcall bit_n
rjmp bit_7
solve_bit7:
clr r15
clr r14
ldi r26,5
mov r13,r26
ldi r26,2
mov r12,r26
ldi r26,6
mov r11,r26
clr r10
rcall bit_n
rjmp bit_8
solve_bit8:
clr r15
clr r14
ldi r26,0
mov r13,r26
ldi r26,5
mov r12,r26
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi r26,2
mov r11,r26
ldi r26,1
mov r10,r26
rcall bit_n
rjmp bit_9
solve_bit9:
clr r15
clr r14
ldi r26,0
mov r13,r26
ldi r26,0
mov r12,r26
ldi r26,5
mov r11,r26
ldi r26,2
mov r10,r26
rcall bit_n
rjmp end_function1
=====
bit_n:
add r7,r15
add r6,r14
add r5,r13
add r4,r12
add r3,r11
add r2,r10
vitri_0:
mov r9,r7
rcall hex_to_dec
mov r7,r9
add r6,r8
vitri_1:
mov r9,r6
rcall hex_to_dec
mov r6,r9
add r5,r8
vitri_2:
mov r9,r5
rcall hex_to_dec
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
mov r5,r9
add r4,r8
vitri_3:
mov r9,r4
rcall hex_to_dec
mov r4,r9
add r3,r8
vitri_4:
mov r9,r3
rcall hex_to_dec
mov r3,r9
add r2,r8
end_function2:
ret
=====
xuat_volt:
ldi r26,48
add r2,r26
mov r19,r2
rcall out_data
add r3,r26
mov r19,r3
rcall out_data
add r4,r26
mov r19,r4
rcall out_data
add r5,r26
mov r19,r5
rcall out_data
ldi r19,'.'
rcall out_data
add r6,r26
mov r19,r6
rcall out_data
add r7,r26
mov r19,r7
rcall out_data
ldi r19,'m'
rcall out_data
ldi r19,'V'
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall out_data
ret
=====
.org $400
hex: .db "0123456789"
.org $500
hex1: .db "ABCDEF"
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

### 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.

```
.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 ;đ/c đầu SRAM lưu số BCD (kq chuyển từ số 16 bit)
.DEF OPD1_L=R24 ;byte thấp của số nhị phân 16 bit
.DEF OPD1_H=R25 ;byte cao của số nhị 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
LDI R16, 0x00 ;PortA input
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
OUT ADC_DR, R16
OUT PORTD, R16 ;output=0x0000
OUT PORTB, R16
LDI R16, 0b01001001 ;Vref=2.56V, SE ADC0 ADC 1
STS ADMUX, R16 ; x1,dịch phải
LDI R16, 166
;cho phép ADC,bắt đầu chuyển đổi,
;mode tự chạy, fCKADC=125Khz
STS ADCSRA, R16
ldi r16,5
sts adcsrb,r16
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) ;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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
MOV R17,R0
RCALL TACKITU
MOV R17,R1
RCALL TACKITU
LDI R17, ''
RCALL PHAT
LDS R1, ADCL ;đọc byte thấp ADC
LDS R0, ADCH ;đọc 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 lệnh
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
MOV R17,R18
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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 ;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
LP_1: LDI R17,160
LP_2: LDI R18,50
LP_3: DEC R18
NOP
BRNE LP_3
DEC R17
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
BRNE LP_2
DEC R16
BRNE LP_1
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
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
MOV R12,R0
MOV R13,R1
ADD R12,R11
CLR R0
ADC R13,R0 ;R13:R12:R10
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
RET
SHIFT_R:
LSR R12
BST R13,0
BLD R12,7
LSR R13
MOV R24,R12
MOV R25,R13
RET
;BIN16_BCD5DG chuyển đổi số nhị phân 16 bit sang số BCD 5 digit
;Inputs: OPD1_H=R25:OPD1_L=R24 chứa số nhị phân 16 bit
;Outputs: BCD_BUF:BCD_BUF+4:địa chỉ SRAM chứa 5 digit BCD từ cao đến
thấp
;Sử dụng R17,COUNT,X,ctc DIV16_8
;-----
BIN16_BCD5DG:
LDI XH,HIGH(BCD_BUF);X trỏ địa chỉ đầu buffer BCD
LDI XL,LOW(BCD_BUF)
LDI COUNT,5 ;đếm số byte bộ nhớ
LDI R17,0X00 ;nạp giá trị 0
LOOP_CL:ST X+,R17 ;xóa buffer bộ nhớ
DEC COUNT ;đếm đủ 5 byte
BRNE LOOP_CL
LDI OPD2,10 ;nạp số chia (SC)
DIV_NXT:
RCALL DIV16_8 ;chia số nhị phân 16 bit cho số nhị phân 8 bit
ST -X,OPD3 ;cất số dư vào buffer
CPI OPD1_L,0 ;thương số=0?
BRNE DIV_NXT ;khác 0 chia tiếp
RET
;-----
;DIV16_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2 (Xem giải thuật
chia ở Chương 0)
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
; OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=thương số
; OPD3=DS(GPR0-31)
;Sử dụng COUNT(GPR16-31)
;-----
DIV16_8: LDI COUNT,16 ;COUNT=đếm 16
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
CLR OPD3 ;xóa dư số
SH_NXT: CLC ;C=0=bit thương số
LSL OPD1_L ;dịch trái SBC L,bit0=C=thương số
ROL OPD1_H ;quay trái SBC H,C=bit7
ROL OPD3 ;dịch bit7 SBC H vào dư số
BRCS OV_C ;tràn bit C=1,chia được
SUB OPD3,OPD2 ;trừ dư số với số chia
BRCC GT_TH ;C=0 chia được
ADD OPD3,OPD2 ;C=1 không chia được,không trừ
RJMP NEXT
OV_C: SUB OPD3,OPD2 ;trừ dư số với số chia
GT_TH: SBR OPD1_L,1 ;chia được,thương số=1
NEXT: DEC COUNT ;đếm số lần dịch SBC
BRNE SH_NXT ;chưa đủ tiếp tục dịch 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
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:
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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 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
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:
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

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

- a) Kết nối cá 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.
- b) Từ giá trị offset error, tính ra gain error từ **Error! Reference source not found..**
- c) Viết lại chương trình ở câu c bài 3, với giá trị ADC được hiệu chỉnh

```
.org 0
rjmp ioset
.org $40
ioset:
ldi r16,0xf7
out ddrb,r16
ldi r16,0
out ddra,r16
out portb,r16
ldi r16,0xff
out ddrc,r16
stack:
ldi r16,high(ramend)
out sph,r16
ldi r16,low(ramend)
out spl,r16
set_uart:
ldi r16,24
sts ucsr0b,r16
ldi r16,6
sts ucsr0c,r16
ldi r16,0
sts ubrr0h,r16
ldi r16,51
sts ubrr0l,r16
set_mode_adc:
ldi r16,201
sts admux,r16
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi r16,166
sts adcsra,r16
ldi r16,5
sts adcsrb,r16
rcall set_lcd_power
rcall set_lcd_screen
main:
lds r16,adcsra
ori r16,(1<<adsc)
sts adcsra,r16
start:
lds r16,adcsra
sbrs r16,adif
rjmp start
sts adcsra,r16
rcall start_lcd
lds r17,adcl
lds r18,adch
out portc,r18
rcall xuat_volt_lcd
rcall out_hercules
rcall delay_1s
rjmp main
;=====
uart_trans:
lds r16,ucsr0a
sbrs r16,udre0
rjmp uart_trans
sts udr0,r18
ret
;=====
out_hercules:
ldi r28,15
ldi r29,10
ldi zh,high(hex<<1)
ldi zl,low(hex<<1)
mov r0,r17
mov r1,r18
ldi r18,'A'
rcall uart_trans
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi r18,'D'
rcall uart_trans
ldi r18,'C'
rcall uart_trans
ldi r18,':'
rcall uart_trans
ldi r18,' '
rcall uart_trans
ldi r18,'0'
rcall uart_trans
ldi r18,'x'
rcall uart_trans
ldi r18,'0'
rcall uart_trans
mov r30,r1
lpm r18,z
rcall uart_trans
mov r1,r0
swap r1
and r1,r28
cp r1,r29
brlo xuat1
rcall address_hex1
xuat1:
mov r30,r1
lpm r18,z
rcall uart_trans
ldi zh,high(hex<<1)
and r0,r28
mov r1,r0
cp r1,r29
brlo xuat2
rcall address_hex1
xuat2:
mov r30,r1
lpm r18,z
rcall uart_trans
ldi r18,' '
rcall uart_trans
ret
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
;=====
address_hex1:
ldi zh,high(hex1<<1)
ldi zl,low(hex1<<1)
sub r1,r29
ret
;=====
set_lcd_power:
ldi r16,250
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,50
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,10
rcall delay_100us
ldi r19,0x30
rcall out_lcd
ldi r16,10
rcall delay_100us
ldi r19,0x20
rcall out_lcd
ret
;=====
set_lcd_screen:
ldi r16,20
rcall delay_100us
ldi r19,0x24
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x01
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x0c
rcall out_2lenh
ldi r16,20
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall delay_100us
ldi r19,0x06
rcall out_2lenh
ldi r16,20
rcall delay_100us
ret
=====
start_lcd:
ldi r16,2
rcall delay_100us
ldi r19,1
rcall out_2lenh
ldi r16,20
rcall delay_100us
ldi r19,0x83
rcall out_2lenh
ret
=====
out_lcd:
cbi portb,0
out portb,r19
sbi portb,2
cbi portb,2
ret
=====
out_2lenh:
push r19
andi r19,240
rcall out_lcd
ldi r16,1
rcall delay_100us
pop r19
swap r19
andi r19,240
rcall out_lcd
ret
=====
out_data:
push r19
andi r19,240
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
out portb,r19
sbi portb,0
sbi portb,2
cbi portb,2
ldi r16,1
rcall delay_100us
pop r19
swap r19
andi r19,240
out portb,r19
sbi portb,0
sbi portb,2
cbi portb,2
ret
=====
delay_100us:
11: ldi r25,200
again:
nop
dec r25
brne again
dec r16
brne 11
ret
=====
hex_to_dec:
clr r8
ldi r16,10
loop1:
cp r9,r16
brlo end_funtion
sub r9,r16
inc r8
rjmp loop1
end_funtion:
ret
=====
xuat_volt_lcd:
clr r2
clr r3
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
clr r4
clr r5
clr r6
clr r7
bit_0:
sbrs r17,0
rjmp bit_1
rjmp solve_bit0
bit_1:
sbrs r17,1
rjmp bit_2
rjmp solve_bit1
bit_2:
sbrs r17,2
rjmp bit_3
rjmp solve_bit2
bit_3:
sbrs r17,3
rjmp bit_4
rjmp solve_bit3
bit_4:
sbrs r17,4
rjmp bit_5
rjmp solve_bit4
bit_5:
sbrs r17,5
rjmp bit_6
rjmp solve_bit5
bit_6:
sbrs r17,6
rjmp bit_7
rjmp solve_bit6
bit_7:
sbrs r17,7
rjmp bit_8
rjmp solve_bit7
bit_8:
sbrs r18,0
rjmp bit_9
rjmp solve_bit8
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
bit_9:  
    sbrs r18,1  
    rjmp end_function1  
    rjmp solve_bit9  
end_function1:  
    rcall xuat_volt  
    ret  
solve_bit0:  
    ldi r26,0  
    mov r15,r26  
    ldi r26,5  
    mov r14,r26  
    ldi r26,2  
    mov r13,r26  
    clr r12  
    clr r11  
    clr r10  
    rcall bit_n  
    rjmp bit_1  
solve_bit1:  
    ldi r26,0  
    mov r15,r26  
    ldi r26,0  
    mov r14,r26  
    ldi r26,5  
    mov r13,r26  
    clr r12  
    clr r11  
    clr r10  
    rcall bit_n  
    rjmp bit_2  
solve_bit2:  
    ldi r26,0  
    mov r15,r26  
    ldi r26,0  
    mov r14,r26  
    ldi r26,0  
    mov r13,r26  
    ldi r26,1  
    mov r12,r26
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
clr r11
clr r10
rcall bit_n
rjmp bit_3
solve_bit3:
ldi r26,0
mov r15,r26
ldi r26,0
mov r14,r26
ldi r26,0
mov r13,r26
ldi r26,2
mov r12,r26
clr r11
clr r10
rcall bit_n
rjmp bit_4
solve_bit4:
ldi r26,0
mov r15,r26
ldi r26,0
mov r14,r26
ldi r26,0
mov r13,r26
ldi r26,4
mov r12,r26
clr r11
clr r10
rcall bit_n
rjmp bit_5
solve_bit5:
ldi r26,0
mov r15,r26
ldi r26,0
mov r14,r26
ldi r26,0
mov r13,r26
ldi r26,8
mov r12,r26
ldi r26,0
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
mov r11,r26
clr r10
rcall bit_n
rjmp bit_6
solve_bit6:
clr r15
ldi r26,0
mov r14,r26
ldi r26,0
mov r13,r26
ldi r26,6
mov r12,r26
ldi r26,1
mov r11,r26
clr r10
rcall bit_n
rjmp bit_7
solve_bit7:
clr r15
clr r14
ldi r26,0
mov r13,r26
ldi r26,2
mov r12,r26
ldi r26,3
mov r11,r26
clr r10
rcall bit_n
rjmp bit_8
solve_bit8:
clr r15
clr r14
ldi r26,0
mov r13,r26
ldi r26,4
mov r12,r26
ldi r26,6
mov r11,r26
ldi r26,0
mov r10,r26
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall bit_n
rjmp bit_9
solve_bit9:
clr r15
clr r14
ldi r26,0
mov r13,r26
ldi r26,8
mov r12,r26
ldi r26,2
mov r11,r26
ldi r26,1
mov r10,r26
rcall bit_n
rjmp end_function1
=====
bit_n:
add r7,r15
add r6,r14
add r5,r13
add r4,r12
add r3,r11
add r2,r10
vitri_0:
mov r9,r7
rcall hex_to_dec
mov r7,r9
add r6,r8
vitri_1:
mov r9,r6
rcall hex_to_dec
mov r6,r9
add r5,r8
vitri_2:
mov r9,r5
rcall hex_to_dec
mov r5,r9
add r4,r8
vitri_3:
mov r9,r4
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
rcall hex_to_dec
mov r4,r9
add r3,r8
vitri_4:
mov r9,r3
rcall hex_to_dec
mov r3,r9
add r2,r8
end_function2:
ret
=====
xuat_volt:
ldi r26,48
add r2,r26
mov r19,r2
rcall out_data
add r3,r26
mov r19,r3
rcall out_data
add r4,r26
mov r19,r4
rcall out_data
add r5,r26
mov r19,r5
rcall out_data
ldi r19,'.'
rcall out_data
add r6,r26
mov r19,r6
rcall out_data
add r7,r26
mov r19,r7
rcall out_data
ldi r19,'m'
rcall out_data
ldi r19,'V'
rcall out_data
ret
=====
delay_1s:
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
ldi r20,80
a1: ldi r21,100
a2: ldi r22,250
a3:
nop
dec r22
brne a3
dec r21
brne a2
dec r20
brne a1
ret
=====
.org $400
hex: .db "0123456789"
.org $500

hex1: .db "ABCDEF"
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

### BÀI 5: ĐO NHIỆT ĐỘ SỬ DỤNG MCP9701

- a) Kết nối cảm biến vào header J73.
- b) Kết nối tín hiệu điện áp V\_TEMP trên header J18 tới ADC0
- c) 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.

```
.EQU ADC_PORT=PORTA
.EQU ADC_DR=DDRA
.EQU ADC_IN=PIN1
.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 ;đ/c đầu SRAM lưu số BCD (kq chuyển từ số 16 bit)
.DEF OPD1_L=R24 ;byte thấp của số nhị phân 16 bit
.DEF OPD1_H=R25 ;byte cao của số nhị 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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
OUT DDRB, R16
OUT DDRC, 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 phái
LDI R16, 0b10000110 ;cho phép ADC,mode 1 lần.
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 khởi động LCD 4 bit
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
RCALL TACKKITU
MOV R17,R1
RCALL TACKKITU
LDI R17, ' '
RCALL PHAT
LDS R1, ADCL ;đọc byte thấp ADC
LDS R0, ADCH ;đọc byte cao ADC
MOV R17,R0
MOV R16,R1
LDI R20,250
RCALL MUL_MATCH
RCALL SHIFT_R
;RCALL BIN16_BCD5DG
MOV R4,R25
MOV R5,R24
MOV R16,R4
MOV R17,R5
LDI R20,20
RCALL MUL_MATCH
RCALL SHIFT_R
MOV R31,R24
MOV R30,R25 ; R30h:R31L
RCALL TRU16BIT
MOV R24,R31
MOV R25,R30
LDI R22,20
RCALL DIV16_8
RCALL BIN16_BCD5DG
XUAT_LCD:
CBI CONT,RS ;RS=0 ghi lệnh
LDI R17,$85 ;con trỏ bắt đầu dòng 1 vị trí số 5
RCALL OUT_LCD4
LDI R17,'T'
SBI CONT,RS
RCALL OUT_LCD4
LDI R17,'='
SBI CONT,RS
RCALL OUT_LCD4
LDI R17, ' '
SBI CONT,RS
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
RCALL OUT_LCD4
LDS R17,0X202 ; XUAT HANG TRAM
RCALL HEX_ASC
MOV R17,R18
SBI CONT,RS
RCALL OUT_LCD4
LDS R17,0X203 ; XUAT HANG CHUC
RCALL HEX_ASC
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
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
LP_1: LDI R17,160
LP_2: LDI R18,50
LP_3: DEC R18
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
NOP
BRNE LP_3
DEC R17
BRNE LP_2
DEC R16
BRNE LP_1
RET
;HEX_ASC chuyển từ mã Hex sang mã ASCII
;Input R17=mã Hex,Output R18=mã ASCII
;-----
HEX_ASC:
CPI R17,0X0A
BRC$ 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:
MUL R16,R20
MOV R10,R0
MOV R11,R1
MUL R17,R20
MOV R12,R0
MOV R13,R1
ADD R12,R11
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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 chuyển đổi số nhị phân 16 bit sang số BCD 5 digit
;Inputs: OPD1_H=R25:OPD1_L=R24 chứa số nhị phân 16 bit
;Outputs: BCD_BUF:BCD_BUF+4:địa chỉ SRAM chứa 5 digit BCD từ cao đến
thấp
;Sử dụng R17,COUNT,X,ctc DIV16_8
;-----
BIN16_BCD5DG:
LDI XH,HIGH(BCD_BUF);X trả địa chỉ đầu buffer BCD
LDI XL,LOW(BCD_BUF)
LDI COUNT,5 ;đếm số byte bộ nhớ
LDI R17,0X00 ;nạp giá trị 0
LOOP_CL:ST X+,R17 ;xóa buffer bộ nhớ
DEC COUNT ;đếm đủ 5 byte
BRNE LOOP_CL
LDI OPD2,10 ;nạp số chia (SC)
DIV_NXT:
RCALL DIV16_8 ;chia số nhị phân 16 bit cho số nhị phân 8 bit
ST -X,OPD3 ;cất số dư vào buffer
CPI OPD1_L,0 ;thương số=0?
BRNE DIV_NXT ;khác 0 chia tiếp
RET
;-----
;DIV16_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2 (Xem giải thuật
chia ở Chương 0)
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
; OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=thương số
; OPD3=DS(GPR0-31)
;Sử dụng COUNT(GPR16-31)
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
;-----  
DIV16_8: LDI COUNT,16 ;COUNT=đếm 16  
CLR OPD3 ;xóa dư số  
SH_NXT: CLC ;C=0=bit thương số  
LSL OPD1_L ;dịch trái SBC L,bit0=C=thương số  
ROL OPD1_H ;quay trái SBC H,C=bit7  
ROL OPD3 ;dịch bit7 SBC H vào dư số  
BRCS OV_C ;tràn bit C=1,chia được  
SUB OPD3,OPD2 ;trừ dư số với số chia  
BRCC GT_TH ;C=0 chia được  
ADD OPD3,OPD2 ;C=1 không chia được,không trừ  
RJMP NEXT  
OV_C: SUB OPD3,OPD2 ;trừ dư số với số chia  
GT_TH: SBR OPD1_L,1 ;chia được,thương số=1  
NEXT: DEC COUNT ;đếm số lần dịch SBC  
BRNE SH_NXT ;chưa đủ tiếp tục dịch 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  
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
RET
RESET_LCD:
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 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
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
```

# LAB 5-1

## LẬP TRÌNH SỬ DỤNG ADC

---

```
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
TRU16BIT : ; R30:R31 - R18:R19 = R30:R31
LDI R18,0X01
LDI R19,0X90
sub r31, r19 ;trừ byte thấp của số thứ hai khỏi byte thấp của số thứ
nhất
sbc r30, r18
RET
```