# Mục tiêu:

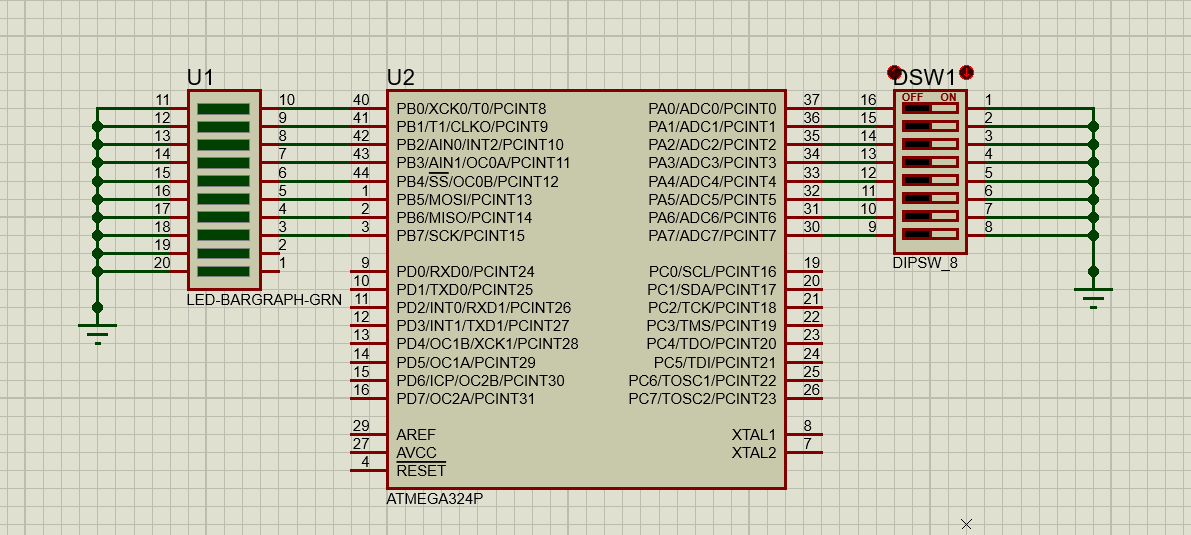
* Thực hiện các giao tiếp I/O Port, các lệnh tính toán

# Tham khảo:

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

# Bài 1

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



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

# BÀI 2

1. 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.
2. Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

# BÀI 3

1. Kết nối và thực hiện chương trình tính tích của 2 nibble cao và thấp của PORTAvà gửi ra PORT B. Coi như 2 nibble này là 2 số không dấu

VD: PORTA = 0b0111\_1111, thì PORTB = 3\*15.

1. Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

# BÀI 4

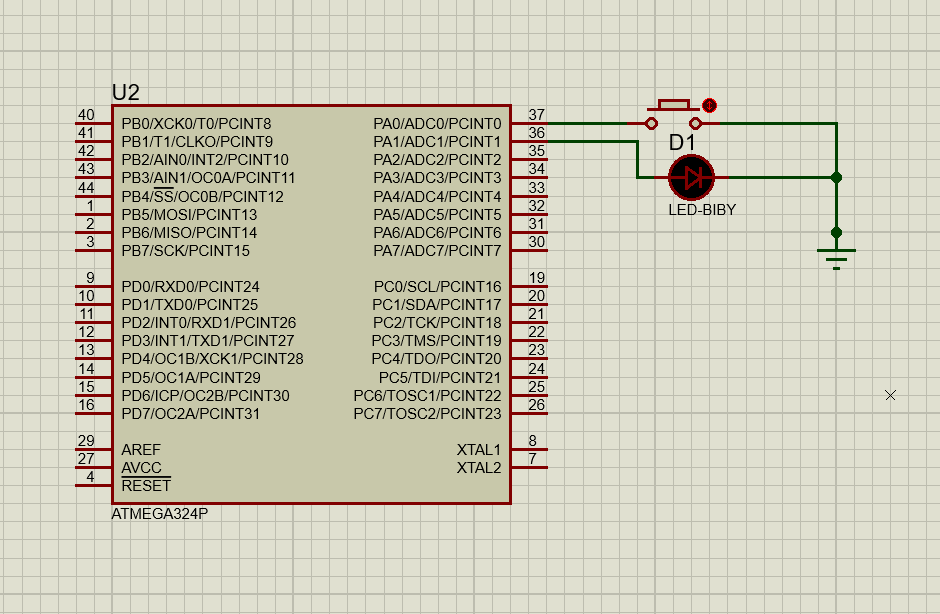
1. Kết nối và thực hiện chương trình tính tích của 2 nibble cao và thấp của PORTAvà gửi ra PORT B. Coi như 2 nibble này là 2 số có dấu

VD: PORTA = 0b0111\_1111, thì PORTB = 3\* (-1).

1. Thay đổi trạng thái của Dip Switch và quan sát trạng thái Bar LED

# BÀI 5

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



1. Viết chương trình bật LED nếu SW nhấn, tắt LED nếu SW nhả.

# Bài 1

1. Trả lời các câu hỏi
2. Lấy giá trị từ 2 nibble của PORTA như thế nào
3. Enable điện trở pullup như thế nào?
4. Khi Switch ở trạng thái ON/OFF, giá trị chân Port bằng bao nhiêu?
5. Khi chân port ở trạng thái 1, BAR LED sáng hay tắt?
6. 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 ;  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

1. Trả lời các câu hỏi
2. 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

1. Trả lời các câu hỏi
2. Làm thế nào lấy giá trị từ 2 nibble của PORT A
3. 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
2. Mã nguồn với chú thích

|  |
| --- |
| ldi R16, 0x00  out DDRA, R16 ; port A nhập  ldi R16, 0xFF  out DDRB, R16 ; port B xuất  out PORTA, R16 ; port A có điện trở kéo lên  loop: in R17, PINA ; đọc dữ liệu 8 bit từ port A  com R17  mov R18, R17 ; sao chép R17 sang R18  ldi R16, 0xF0 ; khởi tạo thanh ghi MASk  and R17, R16 ; lấy nibble cao lưu ở R17  swap R16 ; đảo nibble R16  and R18, R16 ; lấy nibble thấp lưu ở R18  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
2. Khi Switch ở trạng thái nhấn/nhả, giá trị chân Port bằng bao nhiêu?
3. Để LED sáng, chân port xuất ra mức logic gì?
4. Mã nguồn với chú thích

|  |
| --- |
| a) Khi Swich nhấn/nhả, giá trị chân Port bằng 0/1.  b) Để LED sáng, chân port xuất ra mức logic 1.  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 |

# 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

1. 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

1. 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

BRNE LOOP2

DEC R16

BRNE LOOP1

RET

- Chưởng 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

1. Dùng chương trình con này viết các chương trình con Delay10ms, Delay100ms, Delay1s.

- Chưởng trình con Delay10ms:

DELAY\_10MS:

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:

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

1. Dùng chương trình con Delay1s viết chương trình chớp/tắt 1 LED gắn vào PA0.

- Chưởng 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

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

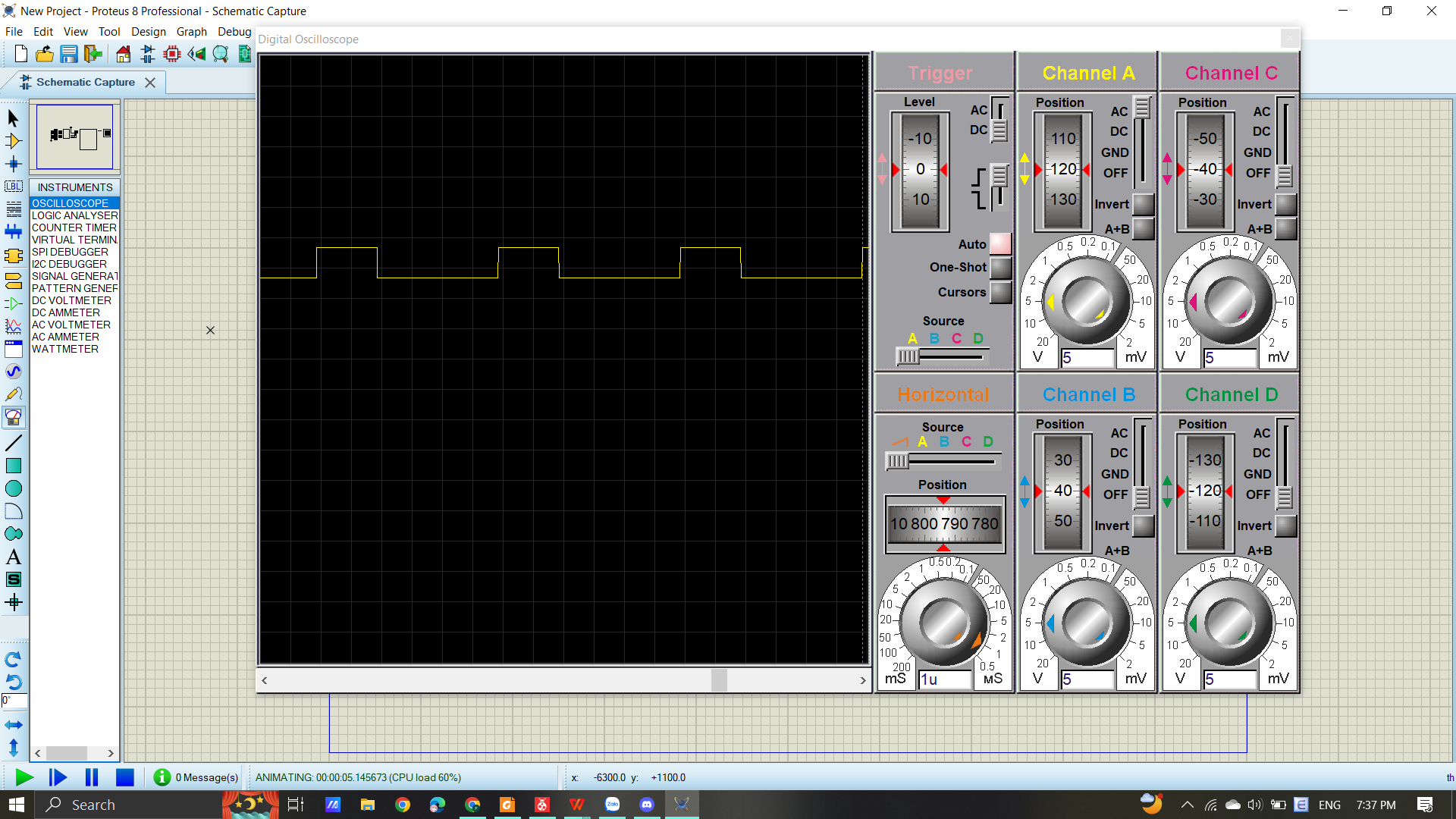
DEC R16

BRNE LOOP4

RET

# BÀI 3

1. 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.
2. Dùng các chương trình trong ví dụ mẫu trong tài liệu hướng dẫn thì nghiệm, viết chương trình tạo hiệu ứng LED sáng dần từ trái qua phải, sau đo tắt dần từ trái qua phải sau mỗi khoảng thời gian 500ms.



# Bài 1

1. Trả lời các câu hỏi
   1. Chụp ảnh dạng xung trên PA0
2. Tần số, thời gian tín hiệu bằng 1, thời gian tín hiệu bằng 0 là bao nhiêu?

- Thời gian tín hiệu bằng 1 là

- Thời gian tin hiệu bằng 0 là

1. 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 lênh CBI (1MC) là lệnh RJMP (2MC), trong lúc thực hiện lệnh RJMP thì PORTA0 có điện áp vẫn bằng 0.

# Bài 2

1. Trả lời các câu hỏi
2. 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)

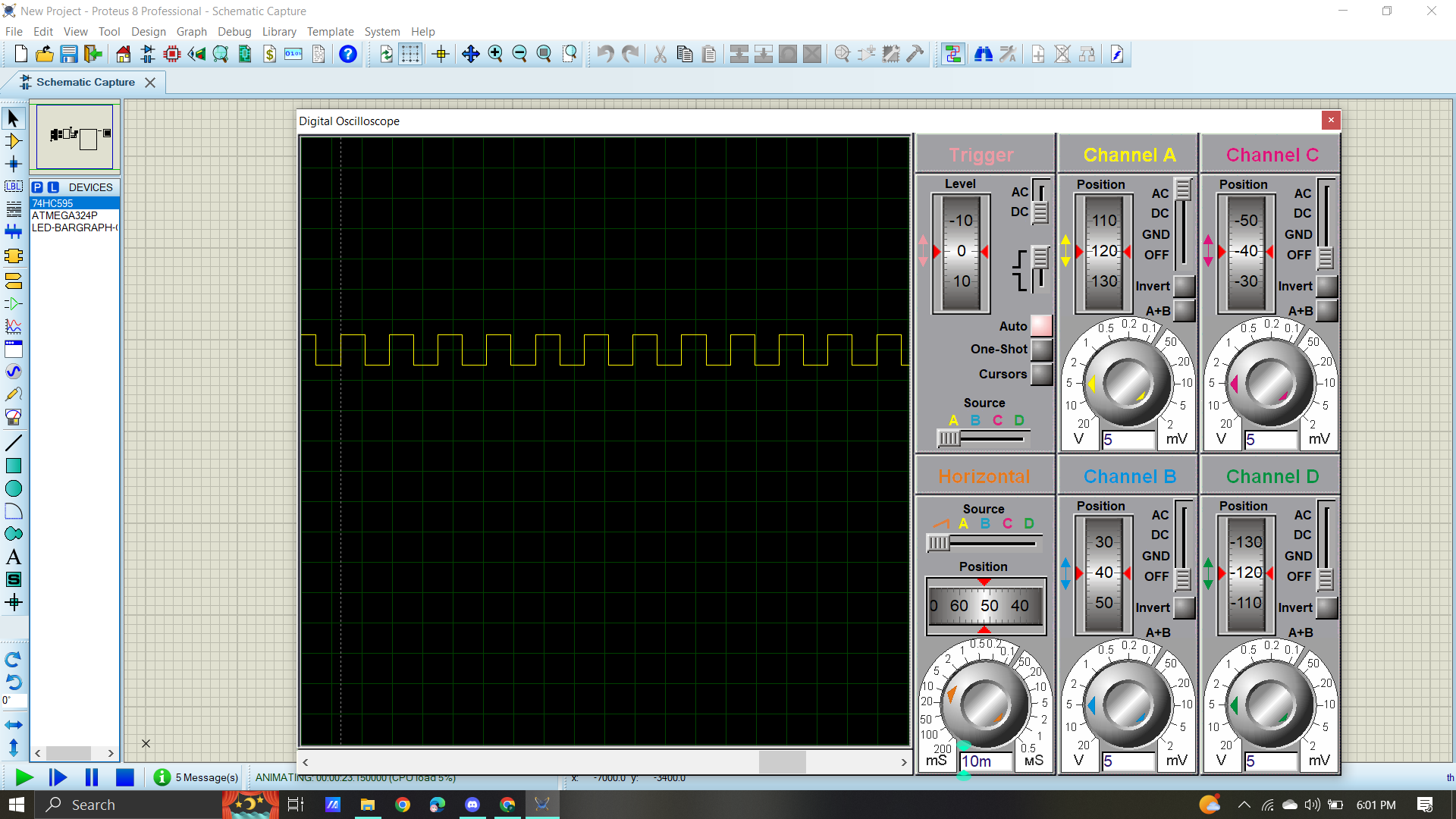
NOP (1 MC)

BRNE LOOP2 (1/2 MC)

DEC R16 (1 MC)

BRNE LOOP1 (1/2 MC)

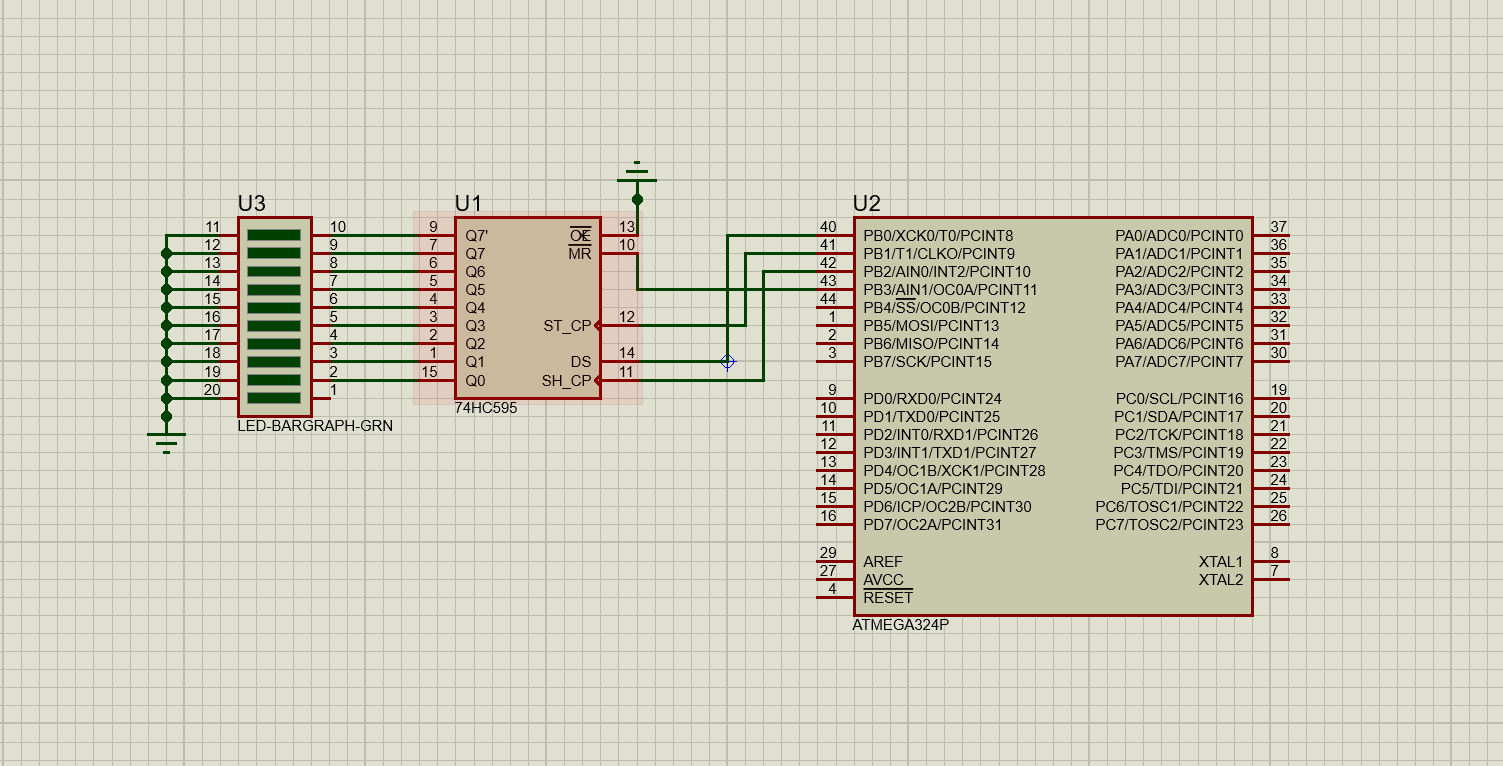
RET (4 MC)

1. Hình ảnh xung 1Khz trên PA0.
2. 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:  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

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



1. 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℃ khi Vcc = 6V.

1. 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 nổi thêm một IC dịch vào, pin DS của IC dịch thứ 2 nối với pin Q7’ của IC dịch thứ

1. Mã nguồn với chú thích

|  |
| --- |
| .include "m324padef.inc" ; Include Atmega324pa definitions  .def shiftData = r20 ; Define the shift data register  .equ clearSignalPort = PORTB ; Set clear signal port to PORTB  .equ clearSignalPin = 3 ; Set clear signal pin to pin 0 of PORTB  .equ shiftClockPort = PORTB ; Set shift clock port to PORTB  .equ shiftClockPin = 2 ; Set shift clock pin to pin 1 of PORTB  .equ latchPort = PORTB ; Set latch port to PORTB  .equ latchPin = 1 ; Set latch pin to pin 0 of PORTB  .equ shiftDataPort = PORTB ; Set shift data port to PORTB  .equ shiftDataPin = 0 ; Set shift data pin to pin 3 of PORTB  main:  call initport  ldi shiftData,0x55  call cleardata  recall:  call shiftoutdata  rjmp recall  ; Initialize ports as outputs  initport:  ldi r24, (1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin)  out DDRB, r24 ; Set DDRB to output  ret  ldi shiftData,0x55  cleardata:  cbi clearSignalPort, clearSignalPin ; Set clear signal pin to low  ; Wait for a short time  sbi clearSignalPort, clearSignalPin ; Set clear signal pin to high  ret  ; Shift out data  shiftoutdata:  cbi shiftClockPort, shiftClockPin  ldi r18, 8  ldi r17, 8  sbi shiftDataPort, shiftDataPin ; Set shift data pin to high  shift\_on:  sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high  cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low  dec r18  breq begin\_shift\_off  call DELAY\_500MS  sbi latchPort, latchPin ; Set latch pin to high  cbi latchPort, latchPin ; Set latch pin to low  rjmp shift\_on  begin\_shift\_off:  cbi shiftDataPort, shiftDataPin ; Set shift data pin to low  shift\_off:  sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high  cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low  dec r17  breq Retired  call DELAY\_500MS  sbi latchPort, latchPin ; Set latch pin to high  cbi latchPort, latchPin ; Set latch pin to low  rjmp shift\_off  Retired:  ret  DELAY\_500MS:  ldi R16, 50  LOOP3:  ldi R17, 80  LOOP1:  ldi R18, 250  LOOP2:  dec R18  nop  brne LOOP2  dec R17  brne LOOP1  dec R16  brne LOOP3  ret |

# 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

1. Kết nối một PORT của AVR vào J33 (Header điều khiển LCD) trên kit thí nghiệm.
2. 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

1. 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
2. Viết chương trình đếm số lần nhấn nút và xuất kết quả ra barled, đồng thời xuất ra LCD (không chống rung)
3. Thêm tính năng chống rung phím vào chương trình
4. Thực hiện chương trình, nhấn/nhả nút và quan sát kết quả

# Bài 3

1. 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.
2. 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
3. Dùng chương trình con này, viết chương trình thực hiện việc quét phím và xuất giá trị đọc được lên bar led và LCD.
4. Thực hiện chương trình, quan sát kết quả

# Bài 1

1. Trả lời các câu hỏi
2. LCD phân biệt command và data bằng cách nào?
3. Ngoài cách đọc bit BUSY, còn cách nào để đảm bảo là LCD rảnh khi gửi dữ liệu/command?
4. Mô tả kết nối trên kit thí nghiệm.
5. 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  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  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  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,"Nhom: ","06",$00 |

# Bài 2

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

|  |
| --- |
| .include "m328pdef.inc" ; Define Atmega file  .def temp = r18 ; Define a temporary register  .def count = r16 ; Define count variable  .org 0x0000 ;  ;Define stack pointer  ldi temp, LOW(RAMEND) ; Initialize the stack pointer  out SPL, temp  ldi temp, HIGH(RAMEND)  out SPH, temp  ;Define LCD  ldi temp, 0x38 ; Function set: 8-bit interface, 2 lines, 5x7 dots  call lcd\_send\_command  ldi temp, 0x0C ; Display on, cursor off, blink off  call lcd\_send\_command  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:  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  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  pop r17  ret  LCD\_wait\_busy:  LCD\_wait\_busy:  push r18  ldi r18, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  out LCDPORTDIR, r18  ldi r18,0b11110010  ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r18  nop  LCD\_wait\_busy\_loop:  sbi LCDPORT, LCD\_EN  nop  nop  in r18, LCDPORTPIN  cbi LCDPORT, LCD\_EN  nop  sbi LCDPORT, LCD\_EN  nop  nop cbi LCDPORT, LCD\_EN  nop  andi r18,0x80  cpi r16=8,0x80  breq LCD\_wait\_busy\_loop  ldi r18, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output out LCDPORTDIR, r18  ldi r18,0b00000000  ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r16  pop r16  ret  ;module lcd\_send\_data (data is r16-count)  lcd\_send\_data:  push r17  call LCD\_wait\_busy ;check if LCD is busy  mov r17,r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17,0xF0  ori r17,0x01  ; Send data to LCD  out LCDPORT, r17  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  ; Delay for command execution ;send the lower nibble  nop  swap r16  andi r16,0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16,0xF0  ori r16,0x01  ; Send command to LCD  out LCDPORT, r16  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  pop r17  ret |

1. Mã nguồn chương trình có chống rung và chú thích

|  |
| --- |
| include "m328pdef.inc" ; Define Atmega file  .def temp = r18 ; Define a temporary register  .def count = r16 ; Define count variable  .org 0x0000 ;  ;Define stack pointer  ldi temp, LOW(RAMEND) ; Initialize the stack pointer  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:  sbic PORTA, 0 ; Check if PA0 = 0 (Switch pressed)  rjmp loop ; if not, come back to loop  rcall DELAY\_10ms  sbic PORTA, 0 ; Check if PA0 = 0 again  rjmp loop ; if PA0=1, come back loop  inc count ; If yes, increment count  out PORTB, count ; Output count to bar LEDs  call lcd\_send\_data ; sent count to LCD  rjmp loop  ; module DELAY\_10ms  DELAY\_10ms:  LDI R21, 80  LOOP1: LDI R22, 250  LOOP2: DEC R22  NOP  BRNE LOOP2  DEC R21  BRNE LOOP1  RET  ; module lcd\_send\_command (command code in r18)  lcd\_send\_command:  push r17  call LCD\_wait\_busy ; check if LCD is busy  mov r17,r18 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17,0xF0 ; Send command to LCD  out LCDPORT, r17  nop  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  swap r18 andi r18,0xF0 ; Send command to LCD  out LCDPORT, r18  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  pop r17  ret  LCD\_wait\_busy:  LCD\_wait\_busy:  push r18  ldi r18, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  out LCDPORTDIR, r18  ldi r18,0b11110010  ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r18  nop  LCD\_wait\_busy\_loop:  sbi LCDPORT, LCD\_EN  nop  nop  in r18, LCDPORTPIN  cbi LCDPORT, LCD\_EN  nop  sbi LCDPORT, LCD\_EN  nop  nop cbi LCDPORT, LCD\_EN  nop  andi r18,0x80  cpi r16=8,0x80  breq LCD\_wait\_busy\_loop  ldi r18, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output out LCDPORTDIR, r18  ldi r18,0b00000000  ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r16  pop r16  ret  ;module lcd\_send\_data (data is r16-count)  lcd\_send\_data:  push r17 call LCD\_wait\_busy ;check if LCD is busy  mov r17,r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17,0xF0  ori r17,0x01  ; Send data to LCD  out LCDPORT, r17  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  ; Delay for command execution ;send the lower nibble  nop  swap r16  andi r16,0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16,0xF0  ori r16,0x01  ; Send command to LCD  out LCDPORT, r16  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  pop r17  ret |

# Bài 3

1. Trả lời các câu hỏi
2. Cách kết nối các module trên bài thí nghiệm
3. Có hiện tượng rung phím đối với bàn phím ma trận hay không? Nếu có thì xử lý bằng cách nào?
4. Trình bày mã nguồn chương trình và chú thích.

|  |
| --- |
| f. Có hiện tượng run phím đối với bàn phím ma trận. Một trong những cách xử lý là dùng các chương trình tạo trễ một khoảng thời gian sau khi nhận tín hiệu nhấn/nhả.  g. Mã nguồn của chương trình (có chống 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  .equ LCD\_D5 = PINA5  .equ LCD\_D4 = PINA4  .def LCDData = r16  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program ID \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;PORTD -> CONTROL KEYPAD  ;PORTC -> BAR LED  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  reset\_handler:  CALL LCD\_Init  SER R16  OUT DDRC, R16  LDI ZL, 0  LDI ZH, 7  LDI R16, $30  MOV R10, R16 ;DIGIT -> ASCII  LDI R16, $37  MOV R11, R16 ;ALPHA -> ASCII  CLR R15  ; display the first line of information  start:  CALL KEY\_PAD\_SCAN  MOV R24, R23  OUT PORTC, R24  CPI R24, 0XFF  BREQ CLEAR  CPI R24, 10  BRCC ALPHA  ADD R24, R10 ;ASCII -> DIGIT  ST Z+, R24 ;DATA  ST Z, R15 ;END LINE  LDI ZL, 0  LDI ZH, 7  CLR R16  CLR R17  CALL LCD\_Move\_Cursor  CALL LCD\_Send\_String  RJMP start  ALPHA:  ADD R24, R11 ;ASCII -> ALPHA  ST Z+, R24 ;DATA  ST Z, R15 ;END LINE  LDI ZL, 0  LDI ZH, 7  CLR R16  CLR R17  CALL LCD\_Move\_Cursor  CALL LCD\_Send\_String  rjmp start  CLEAR:  ldi r16, 0x01 ; Clear Display  call LCD\_Send\_Command  rjmp start  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ;\*\*\*\*\*\*\*\*\*\*\*\*INIT\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_Init:  ; Set up data direction register for Port A  ldi r16, 0b11110111 ; set PA7-PA4 as outputs, PA2-PA0 as output  out LCDPORTDIR, r16  ; Wait for LCD to power up  call DELAY\_10MS  call DELAY\_10MS    ; Send initialization sequence  ldi r16, 0x02 ; Function Set: 4-bit interface  call LCD\_Send\_Command  ldi r16, 0x28 ; Function Set: enable 5x7 mode for chars  call LCD\_Send\_Command  ldi r16, 0x0C ; Display Control: Display OFF, Cursor OFF  call LCD\_Send\_Command  ldi r16, 0x01 ; Clear Display  call LCD\_Send\_Command  ldi r16, 0x80 ; Clear Display  call LCD\_Send\_Command  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SEND CMD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_Send\_Command:  push r17  call LCD\_wait\_busy ; check if LCD is busy  mov r17,r16 ;save the command  ; Set RS low to select command register  ; Set RW low to write to LCD  andi r17,0xF0  ; Send command to LCD  out LCDPORT, r17  nop  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  swap r16  andi r16,0xF0  ; Send command to LCD  out LCDPORT, r16  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  pop r17  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*SEND DATA\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_Send\_Data:  push r17  call LCD\_wait\_busy ;check if LCD is busy  mov r17,r16 ;save the command  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r17,0xF0  ori r17,0x01  ; Send data to LCD  out LCDPORT, r17  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  ; Delay for command execution  ;send the lower nibble  nop  swap r16  andi r16,0xF0  ; Set RS high to select data register  ; Set RW low to write to LCD  andi r16,0xF0  ori r16,0x01  ; Send command to LCD  out LCDPORT, r16  nop  ; Pulse enable pin  sbi LCDPORT, LCD\_EN  nop  cbi LCDPORT, LCD\_EN  pop r17  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SET\_CURSOR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_Move\_Cursor:  cpi r16,0 ;check if first row  brne LCD\_Move\_Cursor\_Second  andi r17, 0x0F  ori r17,0x80  mov r16,r17  ; Send command to LCD  call LCD\_Send\_Command  ret  LCD\_Move\_Cursor\_Second:  cpi r16,1 ;check if second row  brne LCD\_Move\_Cursor\_Exit ;else exit  andi r17, 0x0F  ori r17,0xC0  mov r16,r17  ; Send command to LCD  call LCD\_Send\_Command  LCD\_Move\_Cursor\_Exit:  ; Return from function  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SEND STRING\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_Send\_String:  push ZH ; preserve pointer registers  push ZL  push LCDData  ; fix up the pointers for use with the 'lpm' instruction  // lsl ZL ; shift the pointer one bit left for the lpm instruction  // rol ZH  ; write the string of characters  LCD\_Send\_String\_01:  LD LCDData, Z+ ; get a character  cpi LCDData, 0 ; check for end of string  breq LCD\_Send\_String\_02 ; done  ; arrive here if this is a valid character  call LCD\_Send\_Data ; display the character  rjmp LCD\_Send\_String\_01 ; not done, send another character  ; arrive here when all characters in the message have been sent to the LCD module  LCD\_Send\_String\_02:  pop LCDData  pop ZL ; restore pointer registers  pop ZH  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LCD\_WAIT\_BUSY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  LCD\_wait\_busy:  push r16  ldi r16, 0b00000111 ; set PA7-PA4 as input, PA2-PA0 as output  out LCDPORTDIR, r16  ldi r16,0b11110010 ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r16  nop  LCD\_wait\_busy\_loop:  sbi LCDPORT, LCD\_EN  nop  nop  in r16, LCDPORTPIN  cbi LCDPORT, LCD\_EN  nop  sbi LCDPORT, LCD\_EN  nop  nop  cbi LCDPORT, LCD\_EN  nop  andi r16,0x80  cpi r16,0x80  breq LCD\_wait\_busy\_loop  ldi r16, 0b11110111 ; set PA7-PA4 as output, PA2-PA0 as output  out LCDPORTDIR, r16  ldi r16,0b00000000 ; set RS=0, RW=1 for read the busy flag  out LCDPORT, r16  pop r16  ret  ;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DELAY10MS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;  DELAY\_10MS:  LDI R16,10  LOOP2:  LDI R17,250  LOOP1:  NOP  DEC R17  BRNE LOOP1  DEC R16  BRNE LOOP2  RET  KEY\_PAD\_SCAN:  ;PD\_0 -> PD\_3: OUTPUT, COL  ;PD\_4 -> PD\_7: INPUT, ROW  LDI R16, $0F  OUT DDRD, R16  LDI R16, $F0  OUT PORTD, R16  CALL BUTTON    LDI R22, 0B11110111 ;INITIAL COLUMN MASK  LDI R23, 0 ;INITIAL PRESSED ROW VALUE  LDI R24, 3 ;SCANNING COLUMN INDEX  KEYPAD\_SCAN\_LOOP:  OUT PORTD, R22  SBIC PIND, 4 ;CHECK ROW 0  RJMP KEYPAD\_SCAN\_CHECK\_COL2  RJMP KEYPAD\_SCAN\_FOUND ;ROW 0 IS PRESSED  KEYPAD\_SCAN\_CHECK\_COL2:  SBIC PIND, 5 ;CHECK ROW 1  RJMP KEYPAD\_SCAN\_CHECK\_COL3  LDI R23, 1 ;ROW 1 IS PRESSED  RJMP KEYPAD\_SCAN\_FOUND  KEYPAD\_SCAN\_CHECK\_COL3:  SBIC PIND, 6 ;CHECK ROW 2  RJMP KEYPAD\_SCAN\_CHECK\_COL4  LDI R23, 2 ;ROW 2 IS PRESSED  RJMP KEYPAD\_SCAN\_FOUND  KEYPAD\_SCAN\_CHECK\_COL4:  SBIC PIND, 7 ;CHECK ROW 3  RJMP KEYPAD\_SCAN\_NEXT\_ROW  LDI R23, 3 ;ROW 3 IS PRESSED  RJMP KEYPAD\_SCAN\_FOUND  KEYPAD\_SCAN\_NEXT\_ROW:    CPI R24, 0  BREQ KEYPAD\_SCAN\_NOT\_FOUND  ROR R22  DEC R24  RJMP KEYPAD\_SCAN\_LOOP  KEYPAD\_SCAN\_FOUND:  ; combine row and column to get key value (0-15)  ;key code = row\*4 + col  LSL R23 ; shift row value 4 bits to the left  LSL R23  ADD R23, R24 ; add row value to column value  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. |