# 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

1. 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
2. Kết nối PC0 vào oscilloscope để đo dạng sóng

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

.EQU PORT\_OUT=0

.ORG 0X00

RJMP MAIN

.ORG 0X0024

RJMP TIMER0\_OVF\_ISR

.ORG 0X0040

MAIN:

SBI DDRC,1

CBI PORTC,1

LDI R19,1

LDI R16,HIGH(RAMEND)

OUT SPH,R16

LDI R16,LOW(RAMEND)

OUT SPL,R16

LDI R16,(1<<PORT\_OUT)

OUT DDRC,R16

CBI PORTC,PORT\_OUT

LDI R16,-62

OUT TCNT0,R16

LDI R16,0

OUT TCCR0A,R16

LDI R16,$03

OUT TCCR0B,R16

SEI

LDI R16,(1<<TOIE0)

STS TIMSK0,R16

START:

RJMP START

;===========================================================

TIMER0\_OVF\_ISR:

LDI R17,0

OUT TCCR0B,R17

LDI R17,-62

OUT TCNT0,R17

IN R17,PORTC

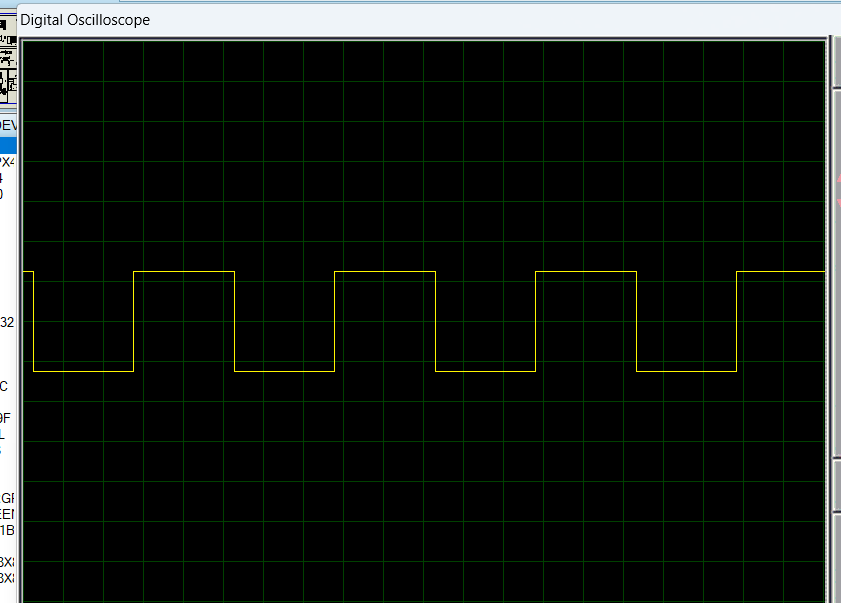
EOR R17,R19

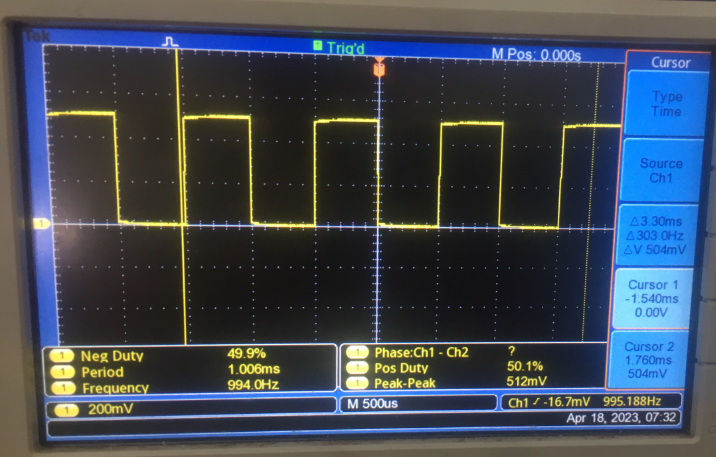
OUT PORTC,R17

LDI R16,$03

OUT TCCR0B,R16

RETI





# Bài 2

1. 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.
2. 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.

1. Biên dịch chương trình và quan sát oscilloscope để kiểm tra chương trình.

.EQU PORT\_OUT=0

.ORG 0

CLR R20

RJMP MAIN

.ORG $001A

INC R20

CPI R20,10

BREQ CHANGE

RJMP EXIT

CHANGE:

CLR R20

IN R17,PORTC

EOR R17,R19

OUT PORTC,R17

EXIT:

RETI

.ORG $100

MAIN:

SBI DDRC,1

CBI PORTC,1

LDI R19,1

LDI R16,HIGH(RAMEND)

OUT SPH,R16

LDI R16,LOW(RAMEND)

OUT SPL,R16

LDI R16,(1<<PORT\_OUT)

OUT DDRC,R16

LDI R17,HIGH(500)

STS OCR1AH,R17

LDI R17,LOW(500)

STS OCR1AL,R17

LDI R17,0

STS TCCR1A,R17

LDI R17,0B00001010

STS TCCR1B,R17

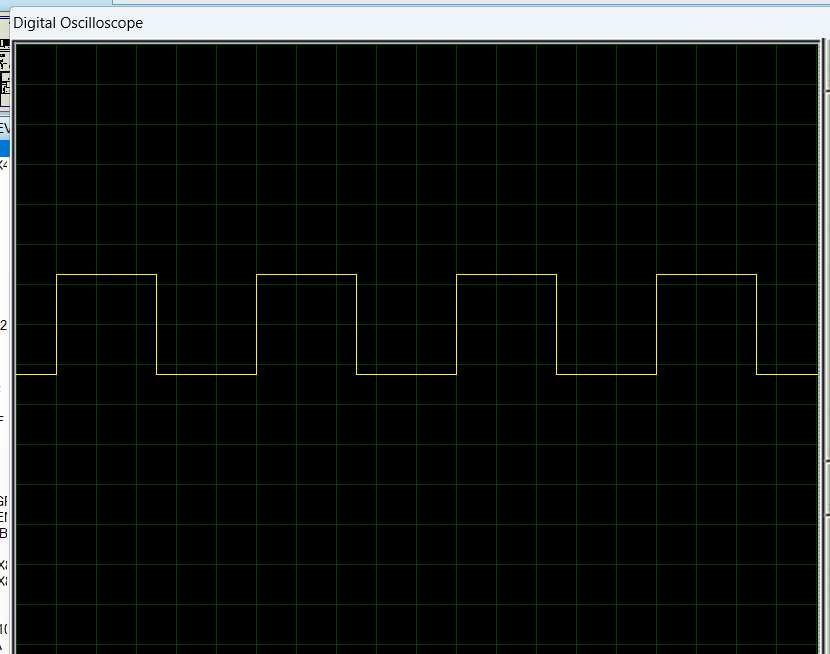
SEI

LDI R17,(1<<OCIE1A)

STS TIMSK1,R17

START:

RJMP START



# Bài 3

1. Kết nối các tín hiệu cần thiết để điều khiển khối LED 7 đoạn.
2. 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)

.EQU P\_OUT = 0

.EQU P\_OUT\_LED = PORTA

.EQU DD\_P\_OUT\_LED = DDRA

.EQU LE\_0 = 0

.EQU LE\_1 = 1

.EQU P\_CONTROL\_E = PORTC

.EQU DD\_P\_CONTROL\_E = DDRC

;------------------------------;

.DEF DATA\_DISPLAY = R21

.DEF DATA\_DISPLAY\_7SEG = R22

.DEF LED\_ACTIVE = R23

.DEF DATA\_DISPLAY\_LED = R24

.ORG 0

RJMP MAIN

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

.ORG $1A

RJMP TIMER1\_COMPA

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

.ORG $40

MAIN:

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

CALL CONFIG\_7SEG

CALL INIT\_TIMER1\_CTC

SEI ;Global interrupt

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

STS TIMSK1, R17

START:

RJMP START

DISPLAY\_7SEG:

LDI R25, $FF

OUT P\_OUT\_LED, R25 ;Turn off led

SBI P\_CONTROL\_E, LE\_1

CBI P\_CONTROL\_E, LE\_1

LDI ZH, HIGH(TABLE\_7SEG<<1)

LDI ZL, LOW(TABLE\_7SEG<<1)

ADD ZL, DATA\_DISPLAY

LPM DATA\_DISPLAY\_7SEG, Z

OUT P\_OUT\_LED, DATA\_DISPLAY\_7SEG

SBI P\_CONTROL\_E, LE\_0

CBI P\_CONTROL\_E, LE\_0

LDI ZH, HIGH(TABLE\_CONTROL<<1)

LDI ZL, LOW(TABLE\_CONTROL<<1)

ADD ZL, LED\_ACTIVE

LPM DATA\_DISPLAY\_LED, Z

OUT P\_OUT\_LED, DATA\_DISPLAY\_LED

SBI P\_CONTROL\_E, LE\_1

CBI P\_CONTROL\_E, LE\_1

RET

TIMER1\_COMPA:

IN R18, PORTC

EOR R18, R16

OUT PORTC, R18

CALL DISPLAY\_7SEG

INC DATA\_DISPLAY

INC LED\_ACTIVE

CPI LED\_ACTIVE, 4

BREQ RESET\_LED

RJMP DONE

RESET\_LED:

LDI DATA\_DISPLAY, 1

LDI LED\_ACTIVE, 0

DONE:

RETI

CONFIG\_7SEG:

SBI DDRC, 0 ;PC0 is out

LDI DATA\_DISPLAY, 1

LDI LED\_ACTIVE, 0

LDI R16, (1 << P\_OUT)

SER R17

OUT DD\_P\_OUT\_LED, R17 ;PortA is output

SBI DD\_P\_CONTROL\_E, LE\_0 ;PB0 is LE0 that is output

SBI DD\_P\_CONTROL\_E, LE\_1 ;PB1 is LE1 that is output

CBI P\_CONTROL\_E, LE\_0 ;Block latch

CBI P\_CONTROL\_E, LE\_1 ;Block latch

RET

INIT\_TIMER1\_CTC:

LDI R17, HIGH(39999)

STS OCR1AH, R17

LDI R17, LOW(39999)

STS OCR1AL, R17

LDI R17, $00

STS TCCR1A, R17 ;Mode CTC

LDI R17, (1 << WGM12) | (1 << CS10) ;Mode CTC, timer run with prescaler = 1

STS TCCR1B, R17

RET

.ORG $200

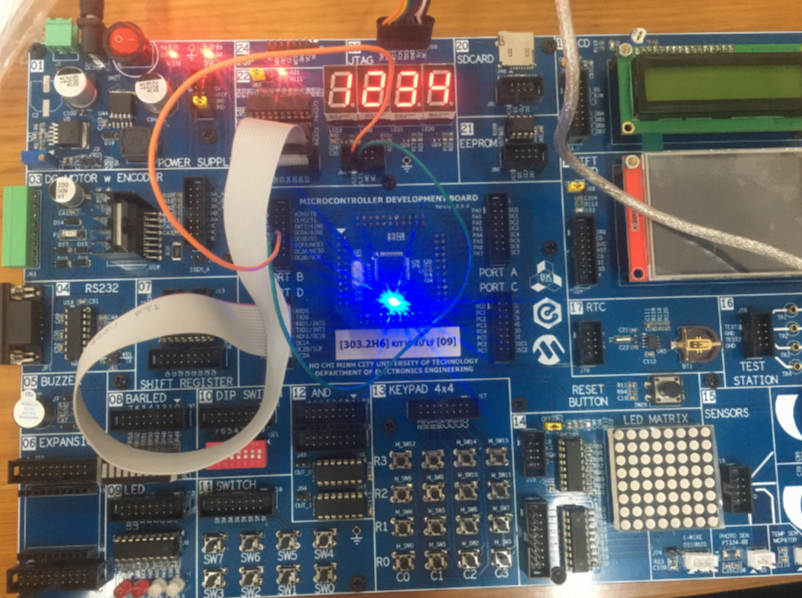
TABLE\_7SEG:

.DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83

.DB 0XC6,0XA1,0X86,0X8E

TABLE\_CONTROL:

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



# Bài 4

1. Kết nối thêm các tín hiệu cần thiết để điều khiển LCD ký tự
2. Kết nối tín hiệu UART ra khối RS232 và kết nối dây USB-Serial.
3. 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.

Phần quét LED giữ nguyên như ở bài 3.

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

;---------LCD-----------;

.EQU RS = 0

.EQU RW = 1

.EQU EN = 2

.EQU LCD = PORTB

.EQU LCD\_DR = DDRB

.EQU LCD\_IN = PINB

.DEF DATA\_DISPLAY\_LCD = R18

;---------LCD-----------;

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

.EQU P\_OUT\_LED = PORTA

.EQU DD\_P\_OUT\_LED = DDRA

.EQU LE\_0 = 0

.EQU LE\_1 = 1

.EQU P\_CONTROL\_E = PORTC

.EQU DD\_P\_CONTROL\_E = DDRC

.DEF DATA\_DISPLAY = R21

.DEF DATA\_DISPLAY\_7SEG = R22

.DEF LED\_ACTIVE = R23

.DEF DATA\_DISPLAY\_LED = R24

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

;---------MAIN---------;

.DEF COUNT\_L = R19

.DEF COUNT\_H = R20

.DEF OPD3 = R2 ;So du

.DEF OPD1\_H = R16 ;HIGH(So bi chia)

.DEF OPD1\_L = R17 ;LOW(So bi chia)

.DEF OPD2 = R3 ;So chia

.DEF DONVI = R4

.DEF CHUC = R5

.DEF TRAM = R6

.DEF NGHIN = R7

.ORG 0

RJMP MAIN

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

.ORG $1A

RJMP TIMER1\_COMPA

.ORG $28

RJMP USART0\_RX

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

.ORG $40

MAIN:

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

CALL CONFIG\_LCD

CALL CONFIG\_7SEG

CALL CONFIG\_USART

CALL INIT\_TIMER1\_CTC

CLR DATA\_DISPLAY\_LCD

CLR DATA\_DISPLAY

CLR COUNT\_L

CLR COUNT\_H

LDI XL, $00

LDI XH, $02

SEI ;Enable interrupt globally

START:

CALL DISPLAY\_LCD

RJMP START

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

CONFIG\_USART:

LDI R16, 0

STS UBRR0H, R16

LDI R16, 51

STS UBRR0L, R16

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

STS UCSR0C, R16

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

STS UCSR0B, R16

RET

USART0\_RX:

LDS DATA\_DISPLAY\_LCD, UDR0

CALL CLEAR\_LCD

LDI R25, 1

LDI R16, 0

ADD COUNT\_L, R25

ADC COUNT\_H, R16

MOV OPD1\_L, COUNT\_L

MOV OPD1\_H, COUNT\_H

CALL DIV16\_8

MOV DONVI, OPD3

STS $203, DONVI

CALL DIV16\_8

MOV CHUC, OPD3

STS $202, CHUC

CALL DIV16\_8

MOV TRAM, OPD3

STS $201, TRAM

CALL DIV16\_8

MOV NGHIN, OPD3

STS $200, NGHIN

// When count to 1000 --> reset counter

CPI COUNT\_L, $E8

BREQ CONTI\_SS

RJMP DONE\_RX

CONTI\_SS:

CPI COUNT\_H, $03

BREQ RESET\_COUNTER

RJMP DONE\_RX

RESET\_COUNTER:

CLR COUNT\_L

CLR COUNT\_H

DONE\_RX:

RETI

DIV16\_8:

PUSH R19

LDI R19, 10

MOV OPD2, R19

LDI R19, 16

CLR OPD3

SH\_NXT:

CLC

LSL OPD1\_L

ROL OPD1\_H

ROL OPD3

BRCS OV\_C

SUB OPD3, OPD2

BRCC GT\_TH

ADD OPD3, OPD2

RJMP NEXT

OV\_C:

SUB OPD3, OPD2

GT\_TH:

SBR OPD1\_L, 1

NEXT:

DEC R19

BRNE SH\_NXT

POP R19

RET

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

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

CONFIG\_LCD:

SER R16 ;Portc is output LCD

OUT LCD\_DR, R16

CBI LCD, RS

CBI LCD, RW

CBI LCD, EN

LDI R16, 250

CALL DELAY\_US

LDI R16, 250

CALL DELAY\_US

CBI LCD, RS

LDI R17, $30

CALL OUT\_LCD4

LDI R16, 42

CALL DELAY\_US

CBI LCD, RS

LDI R17, $30

CALL OUT\_LCD4

LDI R16,2

CALL DELAY\_US

CBI LCD, RS

LDI R17, $32

CALL OUT\_LCD4\_2

LDI R18, $28

LDI R19, $01

LDI R20, $0C

LDI R21, $06

CALL INIT\_LCD4

RET

INIT\_LCD4:

CBI LCD, RS ;Write Instruction

MOV R17, R18 ;R18 is function set

CALL OUT\_LCD4\_2

MOV R17, R19 ;R19 is clear display

CALL OUT\_LCD4\_2

LDI R16, 20 ;Wait 2ms after clearing

CALL DELAY\_US

MOV R17, R20

CALL OUT\_LCD4\_2

MOV R17, R21

CALL OUT\_LCD4\_2

RET

DISPLAY\_LCD:

CBI LCD, RS

LDI R17, $80

CALL OUT\_LCD4\_2

MOV R17, DATA\_DISPLAY\_LCD

SBI LCD, RS

CALL OUT\_LCD4\_2

RET

CLEAR\_LCD:

PUSH R17

PUSH R16

CBI LCD,RS

LDI R17, $01

CALL OUT\_LCD4\_2

LDI R16, 20

CALL DELAY\_US

POP R16

POP R17

RET

OUT\_LCD4:

OUT LCD, R17

SBI LCD, EN

CBI LCD, EN

RET

OUT\_LCD4\_2:

LDI R16, 1 ;Wait 100us

CALL DELAY\_US

IN R16, LCD

ANDI R16, (1<<RS)

PUSH R16 ;Push bit RS to stack

PUSH R17

ANDI R17, $F0 ;Get 4 high bit

OR R17, R16

CALL OUT\_LCD4

LDI R16, 1 ;Wait 100us

CALL DELAY\_US

POP R17

POP R16

SWAP R17

ANDI R17, $F0

OR R17, R16

CALL OUT\_LCD4

RET

DELAY\_US:

MOV R15, R16

LDI R16, 200

L1:

MOV R14, R16

L2:

DEC R14

NOP

BRNE L2

DEC R15

BRNE L1

RET

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

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

CONFIG\_7SEG:

LDI LED\_ACTIVE, 0

LDI DATA\_DISPLAY, 0

SER R17

OUT DD\_P\_OUT\_LED, R17 ;PortA is output

SBI DD\_P\_CONTROL\_E, LE\_0 ;PB0 is LE0 that is output

SBI DD\_P\_CONTROL\_E, LE\_1 ;PB1 is LE1 that is output

CBI P\_CONTROL\_E, LE\_0 ;Block latch

CBI P\_CONTROL\_E, LE\_1 ;Block latch

RET

DISPLAY\_7SEG:

LDI R25, $FF

OUT P\_OUT\_LED, R25 ;Turn off led

SBI P\_CONTROL\_E, LE\_1

CBI P\_CONTROL\_E, LE\_1

LDI R25, 0

LDI ZH, HIGH(TABLE\_7SEG<<1)

LDI ZL, LOW(TABLE\_7SEG<<1)

ADD ZL, DATA\_DISPLAY

ADC ZH, R25

LPM DATA\_DISPLAY\_7SEG, Z

OUT P\_OUT\_LED, DATA\_DISPLAY\_7SEG

SBI P\_CONTROL\_E, LE\_0

CBI P\_CONTROL\_E, LE\_0

LDI ZH, HIGH(TABLE\_CONTROL<<1)

LDI ZL, LOW(TABLE\_CONTROL<<1)

ADD ZL, LED\_ACTIVE

ADC ZH, R25

LPM DATA\_DISPLAY\_LED, Z

OUT P\_OUT\_LED, DATA\_DISPLAY\_LED

SBI P\_CONTROL\_E, LE\_1

CBI P\_CONTROL\_E, LE\_1

RET

TIMER1\_COMPA:

LD DATA\_DISPLAY, X+

CALL DISPLAY\_7SEG

INC LED\_ACTIVE

CPI LED\_ACTIVE, 4

BREQ RESET\_LED

RJMP DONE

RESET\_LED:

LDI LED\_ACTIVE, 0

LDI XL, $00

LDI XH, $02

DONE:

RETI

INIT\_TIMER1\_CTC:

LDI R17, HIGH(39999)

STS OCR1AH, R17

LDI R17, LOW(39999)

STS OCR1AL, R17

LDI R17, $00

STS TCCR1A, R17 ;Mode CTC

LDI R17, (1 << WGM12) | (1 << CS10) ;Mode CTC, timer run with prescaler = 64

STS TCCR1B, R17

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

STS TIMSK1, R17

RET

TABLE\_7SEG:

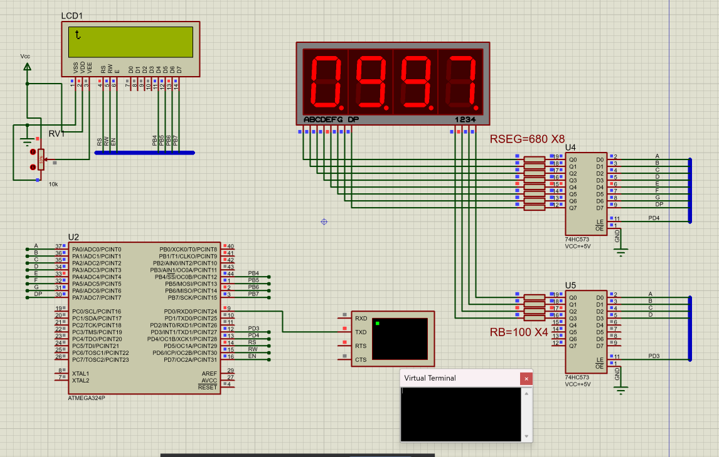
.DB 0XC0, 0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X8

.DB 0XC6,0XA1,0X86,0X8E

TABLE\_CONTROL:

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

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

****

# Bài 5

1. 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:
2. Khởi động RTC với thời gian hiện hành, xuất tín hiệu MFP tần số 1 Hz.
3. 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

1. Kết nối các tín hiệu cần thiết để điều khiển LED ma trận. (gỡ bỏ các dây nối đến LCD, Led 7 đoạn, RTC, UART).
2. 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.
3. Chỉnh sửa chương trình để đạt tần số quét là 125Hz.
4. Viết chương trình để hiển thị Logo Trường ĐH Bách Khoa lên LED ma trận.

CODE CÂU C:

.org 0x0000 ; interrupt vector table

rjmp reset\_handler ; reset

.org 0x001A

rjmp timer1\_COMP\_ISR

reset\_handler:

; initialize stack pointer

ldi r16, high(RAMEND)

out SPH, r16

ldi r16, low(RAMEND)

out SPL, r16

call shiftregister\_initport

call shiftregister\_cleardata

call initTimer1CTC

; enable global interrupts

sei

call ledmatrix\_portinit

main:

jmp main

.equ clearSignalPort = PORTB ; Set clear signal port to PORTB

.equ clearSignalPin = 3 ; Set clear signal pin to pin 3 of PORTB

.equ shiftClockPort = PORTB ; Set shift clock port to PORTB

.equ shiftClockPin = 2 ; Set shift clock pin to pin 2 of PORTB

.equ latchPort = PORTB ; Set latch port to PORTB

.equ latchPin = 1 ; Set latch pin to pin 1 of PORTB

.equ shiftDataPort = PORTB ; Set shift data port to PORTB

.equ shiftDataPin = 0 ; Set shift data pin to pin 0 of PORTB

; Initialize ports as outputs

shiftregister\_initport:

push r24

ldi r24, (1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin);

out DDRB, r24 ; Set DDRB to output

pop r24

ret

shiftregister\_cleardata:

cbi clearSignalPort, clearSignalPin ; Set clear signal pin to low ; Wait for a short time

sbi clearSignalPort, clearSignalPin ; Set clear signal pin to high

ret

; Shift out data

;shift out R27 to bar led

shiftregister\_shiftoutdata:

push r18

cbi shiftClockPort, shiftClockPin ;

ldi r18, 8 ; Shift 8 bits

shiftloop:

sbrc r27, 7 ; Check if the MSB of shiftData is 1

sbi shiftDataPort, shiftDataPin ; Set shift data pin to high

sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high

lsl r27 ; Shift left

cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low

cbi shiftDataPort, shiftDataPin ; Set shift data pin to low

dec r18

brne shiftloop

; Latch data

sbi latchPort, latchPin ; Set latch pin to high

cbi latchPort, latchPin ; Set latch pin to low

pop r18

ret

;Lookup table for collumn control

ledmatrix\_col\_control: .DB 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01

; Lookup table for font

ledmatrix\_Font\_A: .DB 0b11111100, 0b00010010, 0b00010001, 0b00010001, 0b00010010, 0b11111100, 0b00000000, 0b00000000

; J38 connect to PORTD

; clear signal pin to pin 0 of PORTB

; shift clock pin to pin 1 of PORTB

; latch pin to pin 0 of PORTB

; shift data pin to pin 3 of PORTB

; Output: None

.equ LEDMATRIXPORT = PORTD

.equ LEDMATRIXDIR = DDRD

.dseg

.org SRAM\_START ;starting address is 0x100

LedMatrixBuffer : .byte 8

LedMatrixColIndex : .byte 1

.cseg

.align 2

ledmatrix\_portinit:

push r20

push r21

ldi r20, 0b11111111 ; SET port as output

out LEDMATRIXDIR, r20

ldi r20,0 ;col index start at 0

ldi r31,high(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

st z,r20

ldi r20,0

ldi r31,high(ledmatrix\_Font\_A << 1) ;Z register point to fontA value

ldi r30,low(ledmatrix\_Font\_A << 1)

ldi r29,high(LedMatrixBuffer) ; Y register point to fontA value

ldi r28,low(LedMatrixBuffer)

ldi r20,8

ledmatrix\_portinit\_loop: ;copy font to display buffer

lpm r21,z+

st y+,r21

dec r20

cpi r20,0

brne ledmatrix\_portinit\_loop

pop r21

pop r20

ret

; Display a Collumn of Led Matrix

; Input: R27 contains the value to display

; R26 contain the Col index (3..0)

; Output: None

ledmatrix\_display\_col:

push r16 ; Save the temporary register

push r27

clr r16

out LEDMATRIXPORT,r16

call shiftregister\_shiftoutdata

ldi r31,high(ledmatrix\_col\_control << 1)

ldi r30,low(ledmatrix\_col\_control << 1)

clr r16

add r30,r26

adc r31,r16

lpm r27,z

out LEDMATRIXPORT,r27

pop r27

pop r16 ; Restore the temporary register

ret ; Return from the function

initTimer1CTC:

push r16

ldi r16, high(4000) ; Load the high yte into the temporary register

sts OCR1AH, r16 ; Set the high byte of the timer 1 compare value

ldi r16, low(4000) ; Load the low byte into the temporary register

sts OCR1AL, r16 ; Set the low byte of the timer 1 compare value

ldi r16, (1 << CS10)| (1<< WGM12) ; Load the value 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(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

ld r16,z

mov r26,r16

ldi r31,high(LedMatrixBuffer)

ldi r30,low(LedMatrixBuffer)

add r30,r16

clr r16

adc r31,r16

ld r27,z

call ledmatrix\_display\_col

inc r26

cpi r26,8

brne timer1\_COMP\_ISR\_CONT

ldi r26,0 ;if r26 = 8, reset to 0

timer1\_COMP\_ISR\_CONT:

ldi r31,high(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

st z,r26

pop r27

pop r26

pop r16

reti

CODE CÂU D:

.org 0x0000 ; interrupt vector table

rjmp reset\_handler ; reset

.org 0x001A

rjmp timer1\_COMP\_ISR

reset\_handler:

; initialize stack pointer

ldi r16, high(RAMEND)

out SPH, r16

ldi r16, low(RAMEND)

out SPL, r16

call shiftregister\_initport

call shiftregister\_cleardata

call initTimer1CTC

; enable global interrupts

sei

call ledmatrix\_portinit

main:

jmp main

.equ clearSignalPort = PORTB ; Set clear signal port to PORTB

.equ clearSignalPin = 3 ; Set clear signal pin to pin 3 of PORTB

.equ shiftClockPort = PORTB ; Set shift clock port to PORTB

.equ shiftClockPin = 2 ; Set shift clock pin to pin 2 of PORTB

.equ latchPort = PORTB ; Set latch port to PORTB

.equ latchPin = 1 ; Set latch pin to pin 1 of PORTB

.equ shiftDataPort = PORTB ; Set shift data port to PORTB

.equ shiftDataPin = 0 ; Set shift data pin to pin 0 of PORTB

; Initialize ports as outputs

shiftregister\_initport:

push r24

ldi r24, (1<<clearSignalPin)|(1<<shiftClockPin)|(1<<latchPin)|(1<<shiftDataPin);

out DDRB, r24 ; Set DDRB to output

pop r24

ret

shiftregister\_cleardata:

cbi clearSignalPort, clearSignalPin ; Set clear signal pin to low ; Wait for a short time

sbi clearSignalPort, clearSignalPin ; Set clear signal pin to high

ret

; Shift out data

;shift out R27 to bar led

shiftregister\_shiftoutdata:

push r18

cbi shiftClockPort, shiftClockPin ;

ldi r18, 8 ; Shift 8 bits

shiftloop:

sbrc r27, 7 ; Check if the MSB of shiftData is 1

sbi shiftDataPort, shiftDataPin ; Set shift data pin to high

sbi shiftClockPort, shiftClockPin ; Set shift clock pin to high

lsl r27 ; Shift left

cbi shiftClockPort, shiftClockPin ; Set shift clock pin to low

cbi shiftDataPort, shiftDataPin ; Set shift data pin to low

dec r18

brne shiftloop

; Latch data

sbi latchPort, latchPin ; Set latch pin to high

cbi latchPort, latchPin ; Set latch pin to low

pop r18

ret

;Lookup table for collumn control

ledmatrix\_col\_control: .DB 0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01

; Lookup table for font

ledmatrix\_Font\_A: .DB 0b01100000, 0b10010000, 0b10001100, 0b01000010, 0b01000010, 0b10001100, 0b10010000, 0b01100000

; J38 connect to PORTD

; clear signal pin to pin 0 of PORTB

; shift clock pin to pin 1 of PORTB

; latch pin to pin 0 of PORTB

; shift data pin to pin 3 of PORTB

; Output: None

.equ LEDMATRIXPORT = PORTD

.equ LEDMATRIXDIR = DDRD

.dseg

.org SRAM\_START ;starting address is 0x100

LedMatrixBuffer : .byte 8

LedMatrixColIndex : .byte 1

.cseg

.align 2

ledmatrix\_portinit:

push r20

push r21

ldi r20, 0b11111111 ; SET port as output

out LEDMATRIXDIR, r20

ldi r20,0 ;col index start at 0

ldi r31,high(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

st z,r20

ldi r20,0

ldi r31,high(ledmatrix\_Font\_A << 1) ;Z register point to fontA value

ldi r30,low(ledmatrix\_Font\_A << 1)

ldi r29,high(LedMatrixBuffer) ; Y register point to fontA value

ldi r28,low(LedMatrixBuffer)

ldi r20,8

ledmatrix\_portinit\_loop: ;copy font to display buffer

lpm r21,z+

st y+,r21

dec r20

cpi r20,0

brne ledmatrix\_portinit\_loop

pop r21

pop r20

ret

; Display a Collumn of Led Matrix

; Input: R27 contains the value to display

; R26 contain the Col index (3..0)

; Output: None

ledmatrix\_display\_col:

push r16 ; Save the temporary register

push r27

clr r16

out LEDMATRIXPORT,r16

call shiftregister\_shiftoutdata

ldi r31,high(ledmatrix\_col\_control << 1)

ldi r30,low(ledmatrix\_col\_control << 1)

clr r16

add r30,r26

adc r31,r16

lpm r27,z

out LEDMATRIXPORT,r27

pop r27

pop r16 ; Restore the temporary register

ret ; Return from the function

initTimer1CTC:

push r16

ldi r16, high(4000) ; Load the high yte into the temporary register

sts OCR1AH, r16 ; Set the high byte of the timer 1 compare value

ldi r16, low(4000) ; Load the low byte into the temporary register

sts OCR1AL, r16 ; Set the low byte of the timer 1 compare value

ldi r16, (1 << CS10)| (1<< WGM12) ; Load the value 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(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

ld r16,z

mov r26,r16

ldi r31,high(LedMatrixBuffer)

ldi r30,low(LedMatrixBuffer)

add r30,r16

clr r16

adc r31,r16

ld r27,z

call ledmatrix\_display\_col

inc r26

cpi r26,8

brne timer1\_COMP\_ISR\_CONT

ldi r26,0 ;if r26 = 8, reset to 0

timer1\_COMP\_ISR\_CONT:

ldi r31,high(LedMatrixColIndex)

ldi r30,low(LedMatrixColIndex)

st z,r26

pop r27

pop r26

pop r16

reti