# Mục tiêu:

* Hiểu và sử dụng được ADC của AVR
* Hiểu cách sử dụng ADC để đo đạc

# 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

1. 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:
2. Chọn điện áp VREF là điện áp nội VCCA . 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
3. 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=PINA

.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, B output

OUT DDRD, R16

OUT DDRB, R16

LDI R16, 0x00 ;PortA input

OUT ADC\_DR, R16

OUT PORTD, R16 ;output=0x0000

OUT PORTB, R16

LDI R16, 0b01000000 ;Vref=AVcc=5V, SE ADC0

STS ADMUX, R16 ; x1,dịch phải

LDI R16, 0b10000110 ;cho phép ADC,mode 1 lần.

STS ADCSRA, R16 ;f(ADC)=fosc/64=125Khz

RCALL SETUART

START:

LDS R16, ADCSRA

ORI R16, (1<<ADSC) ;bắt đầu chuyển đổi

STS ADCSRA, R16

WAIT:

LDS R16, ADCSRA ;đọc cờ ADIF

SBRS R16, ADIF ;cờ ADIF=1 chuyển đổi xong

RJMP WAIT ;chờ cờ ADIF=1

STS ADCSRA, R16 ;xóa cờ ADIF

LDS R1, ADCL ;đọc byte thấp ADC

LDS R0, ADCH ;đọc byte cao ADC

LDI R17,'A'

RCALL PHAT

LDI R17,'D'

RCALL PHAT

LDI R17,'C'

RCALL PHAT

LDI R17,':'

RCALL PHAT

MOV R17,R0

RCALL TACHKITU

MOV R17,R1

RCALL TACHKITU

LDI R17,' '

RCALL PHAT

RCALL DELAY1S

RJMP START ;tiếp tục chuyển đổi

SETUART:

LDI R16, (1<<TXEN0) ;cho phép phát

STS UCSR0B, R16

LDI R16, (1<<UCSZ01)|(1<<UCSZ00)

;8-bit data, không parity, 1 stop bit

STS UCSR0C, R16

LDI R16, 0x00

STS UBRR0H, R16

LDI R16, 51 ;9600 baud rate

STS UBRR0L, R16

RET

PHAT:

LDS R16,UCSR0A

SBRS R16,UDRE0 ; KIEM TRA CO TRONG KHONG

RJMP PHAT ; NEU CHUA TRONG THI TIEP TUC KIEM TRA LAI

STS UDR0,R17 ; KHI TRONG THI CHEP DU LIEU VAO UDR0

RET

DELAY1S:

LDI R16,200

LP1: LDI R17,160

LP2: LDI R18,50

LP3: DEC R18

NOP

BRNE LP3

DEC R17

BRNE LP2

DEC R16

BRNE LP1

RET

;HEX\_ASC chuyển từ mã Hex sang mã ASCII

;Input R17=mã Hex,Output R18=mã ASCII

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

HEX\_ASC:CPI R17,0X0A

BRCS NUM

LDI R18,0X37

RJMP CHAR

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

1. Thay đổi điện áp đưa vào ADC0, đo bằng VOM và so sánh với kết quả lấy mẫu ADC, điền vào bảng trong báo cáo
2. 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.
3. 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.
4. Đo điện áp trên chân VREF (header J57), sử dụng VOM.

.EQU ADC\_PORT=PORTA

.EQU ADC\_DR=DDRA

.EQU ADC\_IN=PINA

.EQU LCD=PORTB ;PORTB data

.EQU LCD\_IN=PINB

.EQU LCD\_DR=DDRB

.EQU CONT=PORTB ;PORTB ?i?u khi?n

.EQU CONT\_DR=DDRB

.EQU CONT\_OUT=PORTB ;

.EQU CONT\_IN=PINB ;

.EQU RS=0 ;bit RS

.EQU RW=1 ;bit RW

.EQU E=2 ;bit E

.EQU BCD\_BUF=0X200 ;?/c ??u SRAM l?u s? BCD (kq chuy?n t? s? 16 bit)

.DEF OPD1\_L=R24 ;byte th?p c?a s? nh? phân 16 bit

.DEF OPD1\_H=R25 ;byte cao c?a s? nh? phân 16 bit

.DEF OPD2=R22

.DEF OPD3=R23

.DEF COUNT=R18

.ORG 0

RJMP MAIN

.ORG 0X40

MAIN:

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

LDI R16, 0xFF ;PortD, C output

OUT DDRB, R16

OUT DDRC, R16

LDI R16, 0x00 ;PortA input

OUT ADC\_DR, R16

OUT PORTD, R16 ;output=0x0000

OUT PORTB, R16

LDI R16, 0b01000000 ;Vref=AVcc=5V, SE ADC0

STS ADMUX, R16 ; x1,d?ch ph?i

LDI R16, 0b10000110 ;cho phép ADC,mode 1 l?n.

STS ADCSRA, R16 ;f(ADC)=fosc/64=125Khz

RCALL SETUART

LDI R16,0X07

OUT CONT\_DR,R16 ;khai báo PB0,PB1,PB2 là output

CBI CONT,RS ;RS=PB0=0

CBI CONT,RW ;RW=PB1=0 truy xu?t ghi

CBI CONT,E ;E=PB2=0 c?m LCD

LDI R16,0XFF

OUT LCD\_DR,R16 ;khai báo outport

RCALL RESET\_LCD ;ctc reset LCD

RCALL INIT\_LCD4 ;ctc kh?i ??ng LCD 4 bit

START:

LDS R16, ADCSRA

ORI R16, (1<<ADSC) ;b?t ??u chuy?n ??i

STS ADCSRA, R16

WAIT:

LDS R16, ADCSRA ;??c c? ADIF

SBRS R16, ADIF ;c? ADIF=1 chuy?n ??i xong

RJMP WAIT ;ch? c? ADIF=1

STS ADCSRA, R16 ;xóa c? ADIF

LDS R1, ADCL ;??c byte th?p ADC

LDS R0, ADCH ;??c byte cao ADC

LDI R17,'A'

RCALL PHAT

LDI R17,'D'

RCALL PHAT

LDI R17,'C'

RCALL PHAT

LDI R17,':'

RCALL PHAT

MOV R17,R0

RCALL TACHKITU

MOV R17,R1

RCALL TACHKITU

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

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

DEC R16

BRNE LP\_1

RET

;HEX\_ASC chuy?n t? 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

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? phân 16 bit sang s? BCD 5 digit

;Inputs: OPD1\_H=R25:OPD1\_L=R24 ch?a s? nh? phân 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? phân 16 bit cho s? nh? phân 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? phân 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

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:

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 l?nh

LDI R17,$30 ;mã l?nh=$30 l?n 1

RCALL OUT\_LCD

LDI R16,42 ;delay 4.2ms

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$30 ;mã l?nh=$30 l?n 2

RCALL OUT\_LCD

LDI R16,2 ;delay 200?s

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$32 ;mã l?nh=$32

RCALL OUT\_LCD4

RET

INIT\_LCD4:

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$24 ;Function set - giao ti?p 4 bit, 1 dòng, font 5x8

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$01 ;Clear display

RCALL OUT\_LCD4

LDI R16,20 ;ch? 2ms sau l?nh Clear display

RCALL DELAY\_US

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$0C ;Display on/off control

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$06 ;Entry mode set

RCALL OUT\_LCD4

RET

DELAY\_US:

PUSH R15

PUSH R14

MOV R15,R16 ;1MC n?p data cho R15

LDI R16,200 ;1MC s? d?ng R16

L1:

MOV R14,R16 ;1MC n?p data cho R14

L2:

DEC R14 ;1MC

NOP ;1MC

BRNE L2 ;2/1MC

DEC R15 ;1MC

BRNE L1 ;2/1MC

POP R14

POP R15

RET ;4MC

DELAY: ;1s=32\*250\*250

LDI R25,32

LP3: LDI R26,250

LP2: LDI R27,250

LP1: NOP

DEC R27

BRNE LP1

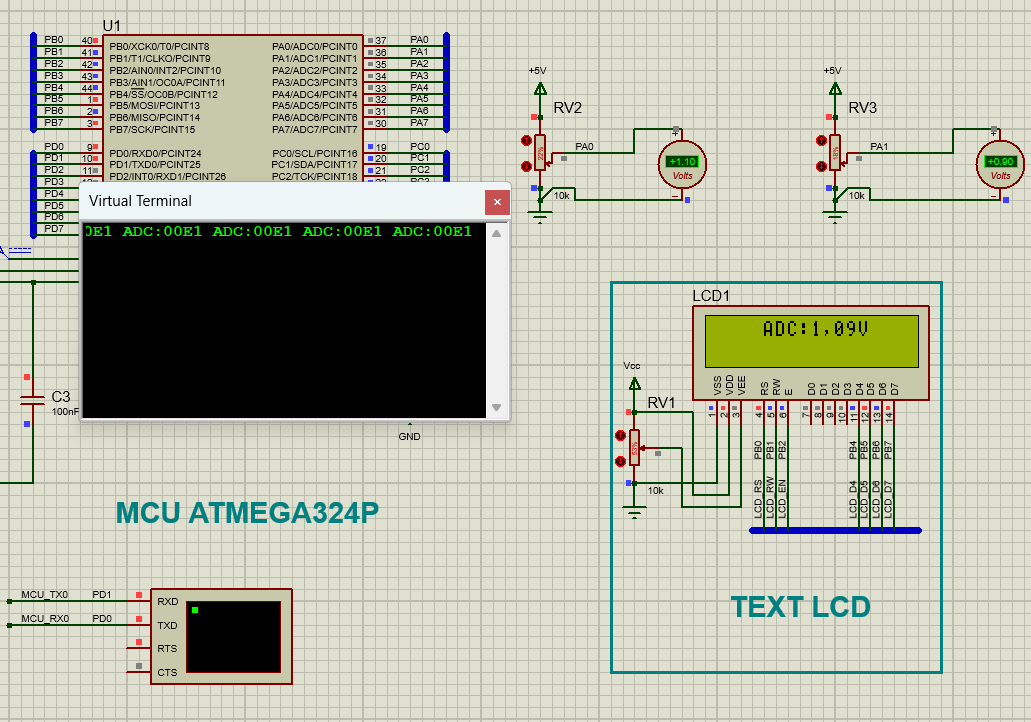
DEC R26

BRNE LP2

DEC R25

BRNE LP3

RET



# Bài 2: đo offset error và gain error

1. Tính toán các sai số offset error và gain error của ADC.
2. Viết lại chương trình với yêu cầu như ở câu e bài 1 với ADC đã được tính toán hiệu chỉnh. Vref = VCCA, 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

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

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:

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

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

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:

l1: ldi r25,200

again:

nop

dec r25

brne again

dec r16

brne l1

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

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

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

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

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

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

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'

rcall out\_data

ret

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

.org $400

hex: .db "0123456789"

.org $500

hex1: .db "ABCDEF"

# Bài 3: Đo adc ở chế độ vi sai

1. Chỉnh kênh VR1 ở mức điện áp 2.5V, đưa vào ADC0.
2. 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.
3. 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

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

MOV R17,R0

RCALL TACHKITU

MOV R17,R1

RCALL TACHKITU

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 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

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

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

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

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 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:

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 l?nh

LDI R17,$30 ;mã l?nh=$30 l?n 1

RCALL OUT\_LCD

LDI R16,42 ;delay 4.2ms

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$30 ;mã l?nh=$30 l?n 2

RCALL OUT\_LCD

LDI R16,2 ;delay 200?s

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$32 ;mã l?nh=$32

RCALL OUT\_LCD4

RET

INIT\_LCD4:

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$24 ;Function set - giao ti?p 4 bit, 1 dòng, font 5x8

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$01 ;Clear display

RCALL OUT\_LCD4

LDI R16,20 ;ch? 2ms sau l?nh Clear display

RCALL DELAY\_US

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$0C ;Display on/off control

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$06 ;Entry mode set

RCALL OUT\_LCD4

RET

DELAY\_US:

PUSH R15

PUSH R14

MOV R15,R16 ;1MC n?p data cho R15

LDI R16,200 ;1MC s? d?ng R16

L1:

MOV R14,R16 ;1MC n?p data cho R14

L2:

DEC R14 ;1MC

NOP ;1MC

BRNE L2 ;2/1MC

DEC R15 ;1MC

BRNE L1 ;2/1MC

POP R14

POP R15

RET ;4MC

DELAY: ;1s=32\*250\*250

LDI R25,32

LP3: LDI R26,250

LP2: LDI R27,250

LP1: NOP

DEC R27

BRNE LP1

DEC R26

BRNE LP2

DEC R25

BRNE LP3

RET

# Bài 4: Đo offset error và gain error ở mode visai

1. Kết nối cả ADC1 và ADC0 vào điện áp ADC\_VR1. Chỉnh điện áp này về 1V. Ghi nhận giá trị ADC đo được. Đây chính là Offset error.
2. Từ giá trị offset error, tính ra gain error từ Bảng 4.
3. 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

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

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

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

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

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

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:

l1: ldi r25,200

again:

nop

dec r25

brne again

dec r16

brne l1

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

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

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

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

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

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

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:

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"

# Bài 5: Đo nhiệt độ sử dụng mcp9701

1. Kết nối cảm biến vào header J73.
2. Kết nối tín hiệu điện áp V\_TEMP trên header J18 tới ADC0
3. 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=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

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

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

RCALL TACHKITU

MOV R17,R1

RCALL TACHKITU

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 lenh

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

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

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

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:

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

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

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 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:

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 l?nh

LDI R17,$30 ;mã l?nh=$30 l?n 1

RCALL OUT\_LCD

LDI R16,42 ;delay 4.2ms

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$30 ;mã l?nh=$30 l?n 2

RCALL OUT\_LCD

LDI R16,2 ;delay 200?s

RCALL DELAY\_US

CBI CONT,RS

LDI R17,$32 ;mã l?nh=$32

RCALL OUT\_LCD4

RET

INIT\_LCD4:

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$24 ;Function set - giao ti?p 4 bit, 1 dòng, font 5x8

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$01 ;Clear display

RCALL OUT\_LCD4

LDI R16,20 ;ch? 2ms sau l?nh Clear display

RCALL DELAY\_US

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$0C ;Display on/off control

RCALL OUT\_LCD4

CBI CONT,RS ;RS=0 ghi l?nh

LDI R17,$06 ;Entry mode set

RCALL OUT\_LCD4

RET

DELAY\_US:

PUSH R15

PUSH R14

MOV R15,R16 ;1MC n?p data cho R15

LDI R16,200 ;1MC s? d?ng R16

L1:

MOV R14,R16 ;1MC n?p data cho R14

L2:

DEC R14 ;1MC

NOP ;1MC

BRNE L2 ;2/1MC

DEC R15 ;1MC

BRNE L1 ;2/1MC

POP R14

POP R15

RET ;4MC

DELAY: ;1s=32\*250\*250

LDI R25,32

LP3: LDI R26,250

LP2: LDI R27,250

LP1: NOP

DEC R27

BRNE LP1

DEC R26

BRNE LP2

DEC R25

BRNE LP3

RET

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