รายงานการวิจัย

เรื่อง

เครื่องควบคุมตำแหน่งสารตัวอย่าง

(SAMPLE CONTROLLER)

โดย

นายสมศักดิ์ เดียวสุรินทร์

ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์

มหาวิทยาลัยสงขลานครินทร์ หาดใหญ่ สงขลา

เครื่องควบคุมตำแหน่งสารตัวอย่าง
(SAMPLE CONTROLLER)

บทคัดย่อ

งานวิจัยนี้เป็นการออกแบบและสร้างอุปกรณ์ในการควบคุมตำแหน่งสารตัวอย่างที่จะนำมาวัดหาปริมาณรังสีในห้องปฏิบัติการนิวเคลียร์ สารตัวอย่างจะถูกจัดให้ตรงกับตำแหน่งหัววัดโดยการหมุนซึ่งควบคุมจากตัวควบคุม(controller) ระยะการหมุนป้อนได้จากคีย์บอร์ดบนตัวควบคุม สารตัวอย่างจะหมุนพร้อมกับจานรองรับซึ่งขับเคลื่อนด้วยสเตปเพอร์มอเตอร์(stepper motor) จำนวนสเตปต่อการหมุนหนึ่งรอบคือ400 ดังนั้นค่าละเอียดที่สุดของการหมุนคือ360/400เท่ากับ 0.9 องศา

เครื่องควบคุมตำแหน่งสารตัวอย่าง
(SAMPLE CONTROLLER)

วัตถุประสงค์

เพื่อพัฒนาอุปกรณ์การวัดในห้องปฏิบัติการนิวเคลียร์ฟิสิกส์ให้มีประสิทธิภาพ

วิธีวิจัย

การวิจัยแยกได้เป็นสองส่วนการออกแบบฮาร์ดแวร์(hardware)และการเขียนซอฟท์แวร์(software)ควบคุมการทำงาน

ฮาร์ดแวร์(hardware)

เป็นการออกแบบสร้างอุปกรณ์ อุปกรณ์ประกอบด้วย หน่วยควบคุมการหมุน(controller unit)จานวางสารตัวอย่างและจานวางสารตัวอย่างซึ่งขับเคลื่อนด้วยสเตปเพอร์มอเตอร์ วงจรสำหรับหน่วยควบคุมออกแบบโดยใช้ไมโครคอนโทรลเลอร์ PIC16F84 ของบริษัท Microchip ซึ่งเป็นหัวใจในการทำงานของหน่วยควบคุม รูปที่1 เป็นบล๊อกไดอะแกรมเครื่องควบคุมตำแหน่งสารตัวอย่าง

หน่วยควบคุม(Controller unit)

หน่วยควบคุมมีวงจรดังรูปที่2 วงจรประกอบด้วย Microcontroller PIC 16F84สองตัว U1 และU2 PIC16F84 มีหน่วยความจำขนาด 1K สำหรับเก็บโปรแกรมอยู่ภายในชิป มี I/O พอร์ทสองพอร์ท พอร์ทAมี5ขา(RAO-RA4) พอร์ทBมี8ขา(RB0-RB7) ลักษณะการจัดขาของ PIC16F84 แสดงในรูปที่2 U1เชื่อมต่อกับU2ผ่านทางพอร์ทAโดยRA1ของU1ต่อกับRA0ของU2 พอร์ทB ของU1ใช้สำหรับเชื่อมต่อทั้งหน่วยแสดงผลเซเว่นเซ็กเม้นท์สองหลักและคีย์บอร์ดแบบเมทริกซ์ 3X4 พอร์ทBมีลักษณะเป็นแบบมัลติเพลกซ์ คือพอร์ทBพอร์ทเดียวเป็นทั้งตัวรับข้อมูลจากคีย์บอร์ดและส่งข้อมูลไปที่หน่วยแสดงผล สำหรับขาRA0และRA1ต่อกับทรานซิสเตอร์Q1และQ2ตามลำดับเพื่ออีเนเบิ้ลหน่วยแสดงผล U2กำหนดให้ทำหน้าที่รับข้อมูลแบบอนุกรมจากU1และเชื่อมต่อกับU3(ULN2003A)ซึ่งเป็นตัวขับสเตปเพอร์มอเตอร์(stepper motor driver)

หน่วยควบคุมทำงานดังนี้ เมื่อมีการกดคีย์บอร์ดไมโครคอนโทรลเลอร์U1รับตัวเลขที่ถูกกด(0-9)และนำไปแสดงผลบนหน่วยแสดงผล ตัวเลขที่ถูกกดจะแสดงผลบนตัวแสดงผลตัวขวามือ

ตัวเลขที่กดครั้งก่อนจะเลื่อนไปทางซ้าย หน่วยแสดงผลเป็นตัวเลขฐานสิบสองหลักซึ่งเป็นจำนวนสเตปที่เราต้องการให้สเตปเพอร์มอเตอร์หมุน จำนวนสเตปสูงสุดของการหมุนแต่ละครั้งคือ99 จำนวนสเตปต่ำสุดคือ1

จานวางสารตัวอย่างเป็นแผ่นพลาสติกรูปวงกลมวางอยู่ในแนวราบนแกนหมุนของสเตปเพอร์มอเตอร์ มอเตอร์ที่ใช้เป็นแบบขั้วเดี่ยว(unipolar stepper moter)ซึ่งมีสายที่ต่อออกมาจากขดลวดภายในมอเตอร์ 6 สายสำหรับใช้ในการควบคุม มอเตอร์ถูกควบคุมการหมุนโดยโปรแกรมที่บรรจุภายในไมโครคอนโทรลเลอร์ U2 ผ่านตัวขับ(driver circuit)ซึ่งเป็นวงจรดาร์ลิงตันในไอซี ULN2003A(U5) มอเตอร์นี้มี 200 สเตปต่อการหมุนหนึ่งรอบสำหรับการหมุนแบบ full-step สำหรับงานนี้กำหนดการหมุนเป็นแบบ half-step ทำให้จำนวนสเตปต่อการหมุนต่อรอบเพิ่มเป็นสองเท่าคือ 400 สเตป ดังนั้นหนึ่งสเตปของการหมุนมีค่าเท่ากับ 360/400=0.9 องศา

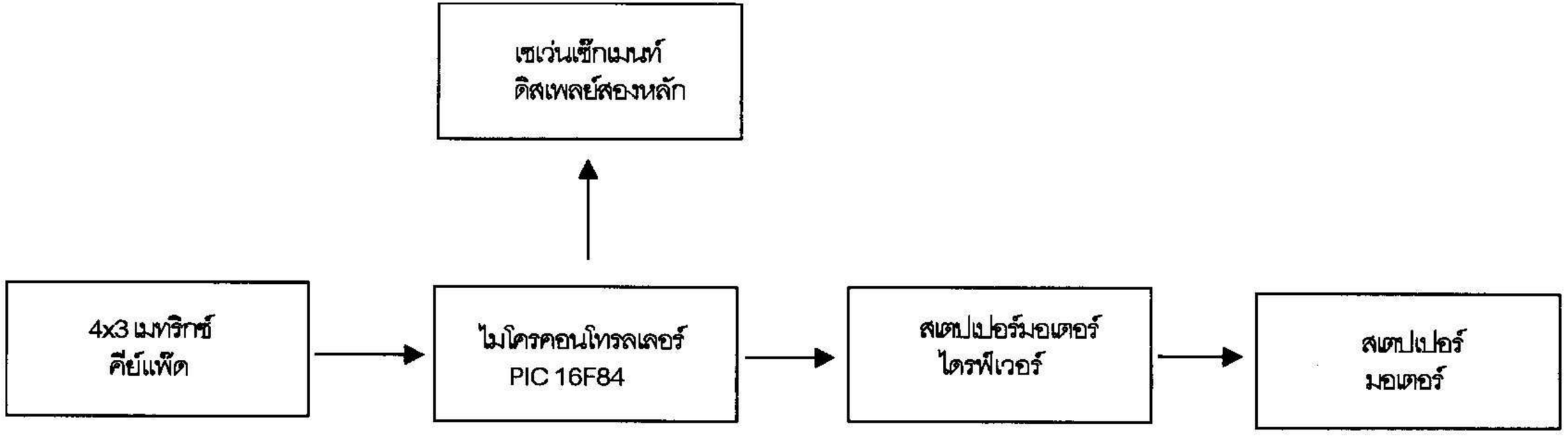
รูปที่2. แสดงลายวงจรของแผงวงจรหน่วยควบคุมและแหล่งจ่ายไฟของวงจรซึ่งรวมอยู่บนแผงเดียวกัน ส่วนวงจรและลายวงจรของหน่วยแสดงผลแสดงในรูปที่ 3.และ 4 ตามลำดับ

ซอฟท์แวร์ควบคุม(control software)

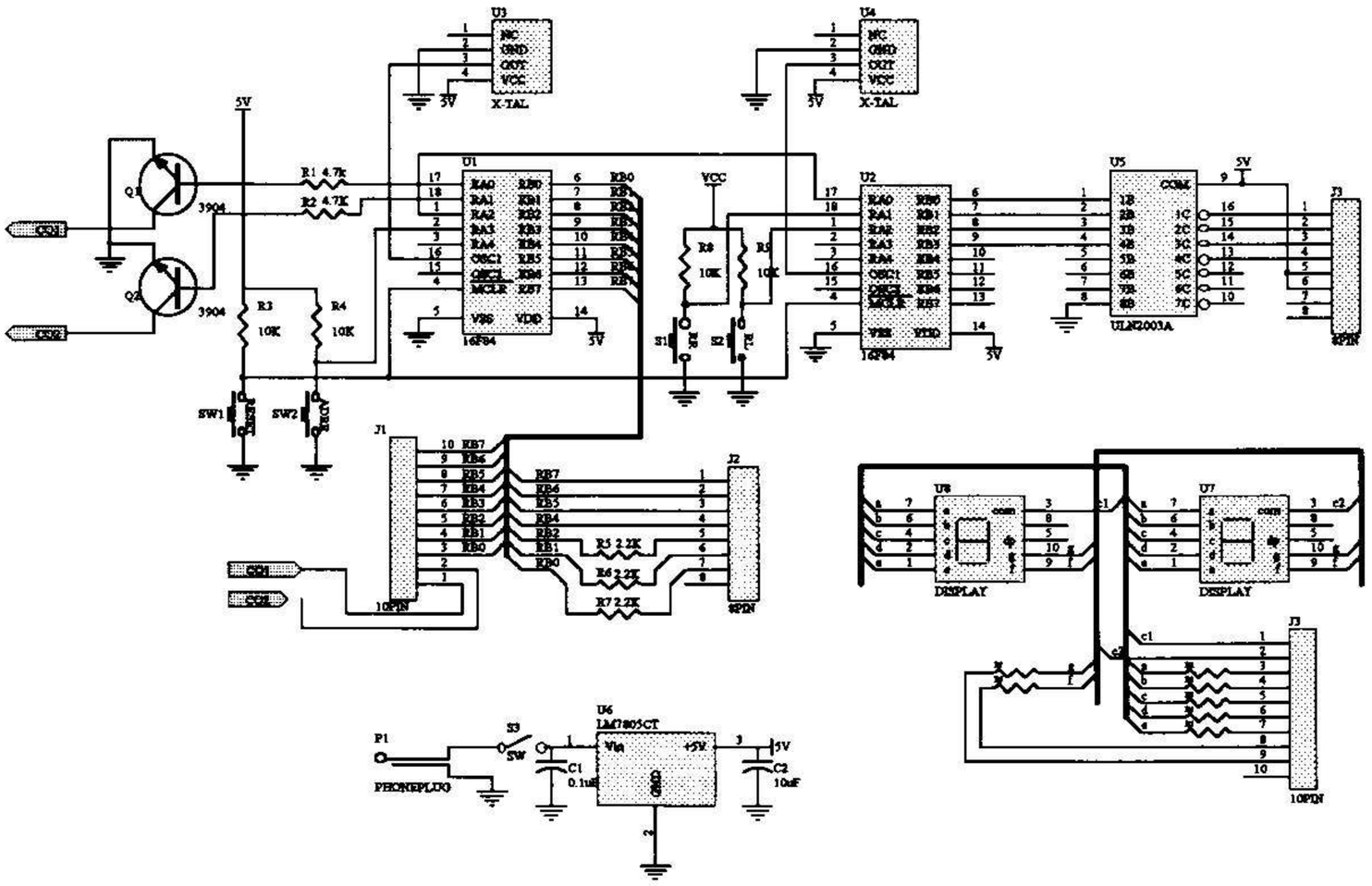
โปรแกรมสำหรับควบคุมการทำงานของหน่วยควบคุมพัฒนาโดยเขียนเป็นภาษาแอสเซมบลี้บน PC โดย เอดิเตอร์ แอสเซมเบลอร์ ซิมูเลเตอร์ และโปรแกรมเมอ ใน MPLAB4.0ซึ่งเป็นเครื่องมือทางซอฟท์แวร์(software tools)ทั้งสี่เป็นซอฟท์แวร์แพคเกจของบริษัท Microchip

โปรแกรมควบคุมการทำงานประกอบด้วยโปรแกรมหลักๆสามโปรแกรม โปรแกรมเชื่อมต่อระหว่างคีย์บอร์ดกับไมโครคอนโทรลเลอร์ โปรแกรมสำหรับขับเคลื่อนสเตปเพอร์มอเตอร์ นอกจากนี้ยังมีโปรแกรมสื่อสารแบบอนุกรมระหว่างไมโครคอนโทรลเลอร์PIC16F84สองตัว ที่ต้องใช้ไมโครคอนโทรลเลอร์สองตัวเพราะว่า PIC16F84เป็นไมโครคอนโทรลเลอร์ขนาดเล็กคือมีขาเพียง18ขาซึ่งไม่เพียงพอในการเชื่อมทั้งคีย์บอร์ด สเตปเพอร์มอเตอร์ และดิสเพลย์ในขณะเดียวกัน

โปรแกรมที่บรรจุภายในไมโครคอนโทรลเลอร์ U1 เป็นโปรแกรมสำหรับเชื่อมต่อระหว่างคีย์บอร์ดกับไมโครคอนโทรลเลอร์และยังทำหน้าที่ส่งข้อมูลที่รับมาจากคีย์บอร์ดผ่านไปสู่ไมโครคอนโทรลเลอร์ U2 การส่งข้อมูลจาก U1 ไป U2 เป็นการส่งข้อมูลแบบอนุกรม(serial data transmission) ข้อมูลส่งไปในสายที่ต่อระหว่าง U1(RA1) และ U2(RA0) เพียงสายเดียว ข้อมูลที่ U2 รับมาจาก U1 คือจำนวนสเตปที่ต้องการให้มอเตอร์หมุน จะเห็นว่าไมโครคอนโทรลเลอร์ U1 และ U2 แบ่งหน้าที่กันทำงาน U1ทำหน้าที่รับข้อมูลจากคีย์บอร์ด นำข้อมูลไปแสดงบนหน่วยแสดงผล และส่งข้อมูลให้ U2 ส่วน U2 มีหน้าที่รับข้อมูลจาก U1และควบคุมมอเตอร์ผ่านวงจรขับ ULN 2003A รายละเอียดของโปรแกรมใน U1และU2อยู่ในซอร์ซโค้ดไฟล์ DISPSTEP.ASM และRSTEP.ASM ดังแสดงในหน้า 6 และ 16 ตามลำดับ

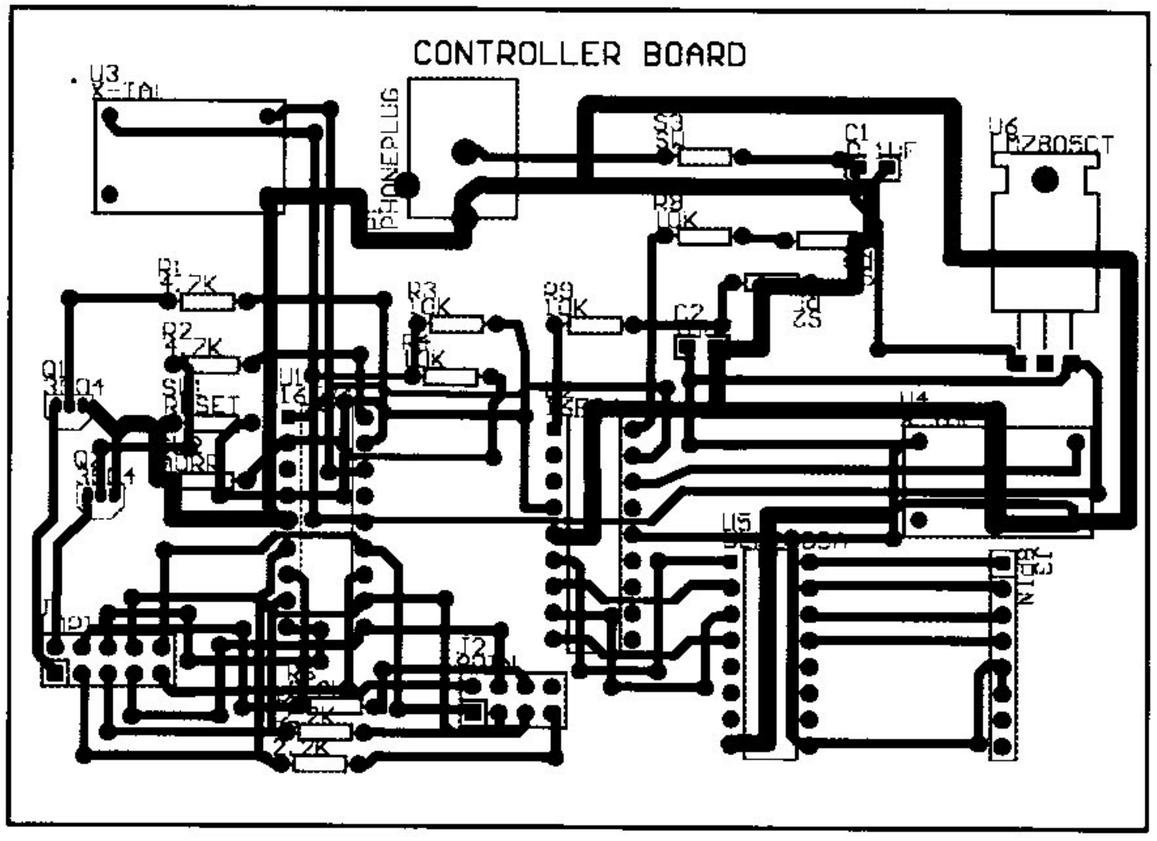


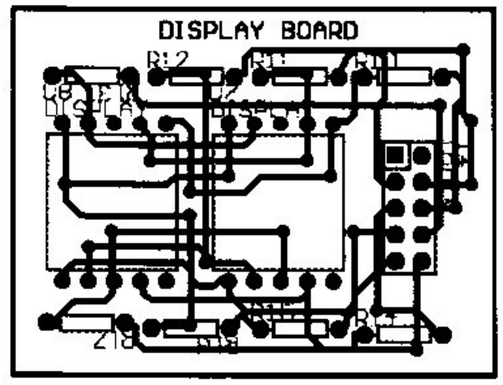
รูปที่1 บล็อกไดอะแกรมเครื่องควบคุมตำแหน่งสารตัวอย่าง



รูปที่2 วงจรควบคุม(CONTROLLER WITH DISPLAY AND POWER SUPPLY)

Title	CONTROLLER /DISPLAY/POWER SUPPLY UNIT	
Size: B	Number	Revision
Date: 26-Sep-1999		Sheet of
File: A:\TEP_KH		Drawn By: Somsak Deamarintr





รูปที่3 ลายพิมพ์วงจรสองหน้าขนาดเท่าของจริงสำหรับ CONTROLLER/DISPLAY BOARD

รายการอุปกรณ์สำหรับหน่วยควบคุม

	Part Used	PartType	Designators
1	1	0.1UF	C1
2	3	2.2K	R5 R6 R7
3	2	4.7K	R2 R1
4	2	8PIN	J3 J2
5	4	10K	R9 R8 R4 R3
6	1	10PIN	J1
7	1	10UF	C2
8	2	16F84	U2 U1
9	2	3904	Q2 Q1
10	1	ADRR	SW2
11	1	LM7805CT	U6
12	1	PHONEPLUG	P1
13	1	RESET	SW1
14	1	RL	S2
15	1	RR	S1
16	1	SW	S3
17	1	ULN2003A	U5
18	2	X-TAL	U4 U3

รายการอุปกรณ์สำหรับหน่วยขับเคลื่อนจาน

1	1	แผ่นพลาสติกสำหรับเป็นจานวางสาร
1	1	Unipolar Stepper Motor

โปรแกรมในไมโครคอนโทรลเลอร์ U1

```
;*****************************************************************************
;This program  use method of multiplexing two 7 segment LED
;digits and a 3X4 keypad on  a PIC16F84.
;The two digits will start as '00' and when a key is hit
;it is displayed on the 7 segment leds as a decimal value 0 to 9. The last
;digit hit is always displayed on the right most led with the rest of
;the digits shifted to the left. The left most digit is deleted.
;The LEDs are updated every 20mS, the keypad is scanned at a rate of 20 mS.
;The RTCC timer is used in internal interrupt mode to generate the
;5 mS.
;
;
;
;       Program:        DISPSTEP.ASM
;
;
;*****************************************************************************
        LIST P=16F84
        ERRORLEVEL  -302
;
        include    <p16F84.inc>

TempC       equ    0x0c             ;temp general purpose regs
TempD       equ    0x0d
TempE       equ    0x0e
PABuf       equ    0x20
PBBuf       equ    0x21
Count       equ    0x0f              ;count
MsdTime     equ    0x10              ;most significant Timer
LsdTime     equ    0x11              ;Least significant Timer
KeyFlag     equ    0x12              ;flags related to key pad
keyhit      equ    0                 ;bit 0 --> key-press on
DebnceOn    equ    1                 ;bit 1 --> debounce on
noentry     equ    2                 ;no key entry = 0
ServKey     equ    3                 ;bit 3 --> service key
Debnce      equ    0x13              ;debounce counter
NewKey      equ    0x14
WBuffer     equ    0x2f
StatBuffer  equ  0x2e
OptionReg   equ  1
PCL         equ    2
```

```
;
Save_byte   equ   30
R0          equ   31              ; RAM Assignments
Save_temp   equ   32              ; temporary register
XmtReg      equ   33
Xcount      equ   34
Delaycnt    equ   35              ; delay for 104 us
mcount      equ   36
ncount      equ   37
;
;
push        macro
            movwf  WBuffer        ;save w reg in Buffer
            swapf  WBuffer, F     ;swap it
            swapf  STATUS,W       ;get status
            movwf  StatBuffer     ;save it
            endm
;
pop         macro
            swapf  StatBuffer,W   ;restore status
            movwf  STATUS         ;
            swapf  WBuffer,W      ;restore W reg
            endm
;
            org    0

            goto   Start          ;skip over interrupt vector

            org    4
;It is always a good practice to save and restore the w reg,
;and the status reg during a interrupt.
            push
            call   ServiceInterrupts
            pop
            retfie
;
Start
            call   InitPorts
            call   InitTimers
loop
            btfsc  KeyFlag,ServKey    ;key service pending
            call   ServiceKey         ;yes then service
            btfsc  PORTA,3
                goto   loop
```

```
                call    Send
                goto    loop

;ServiceKey, does the software service for a keyhit. After a key service,
;the ServKey flag is reset, to denote a completed operation.
ServiceKey
                movf    NewKey,W                ;get key value
                movwf   TempE                   ;save in TempE
                swapf   MsdTime,W               ;move MSD out
                andlw   B'11110000'             ;clr lo nibble
                movwf   MsdTime                 ;save back
                iorwf   TempE,W                 ;or with new lsd
                movwf   MsdTime
BCDtoB          movf    MsdTime,W
                movwf   R0
                clrf    Save_byte
                swapf   R0,W
                andlw   0F
                movwf   Save_byte
                call    mpy10a                  ; result = 10a
                movfw   R0
                andlw   0F
                addwf   Save_byte,F
                bcf     KeyFlag,ServKey ;reset service flag
                return

mpy10a          bcf     STATUS,C                ; multiply by 2
                rlf     Save_byte,W
                movwf   Save_temp               ;Save_temp) = 2*N

                bcf     STATUS,C                ; multiply by 2
                rlf     Save_byte, F

                bcf     STATUS,C                ; multiply by 2
                rlf     Save_byte, F

                bcf     STATUS,C                ; multiply by 2
                rlf     Save_byte, F
                                        ; (Save_byte) = 8*N
;
                movf    Save_temp,W
                addwf   Save_byte, F
```

```
        retlw   0                       ; (H_byte,L_byte) = 10*N

Send
        bcf     INTCON,7
        movf    Save_byte,W
        movwf   XmtReg
        movlw   .8
        movwf   Xcount          ;8 data bits
        bcf     PORTA,2         ;Send Start bit
        call    Delay1
X_next  bcf     STATUS,C
        rlf     XmtReg,F        ;Shift data bit
        btfsc   STATUS,C        ;Test the bit to be transmitted
        bsf     PORTA,2         ;Bit is one
        btfss   STATUS,C
        bcf     PORTA,2
        call    DelayX
        decfsz  Xcount,F        ; If count = 0, then transmit a stop bit
        goto    X_next          ; NO shift next bit
        bsf     PORTA,2         ;Send Stop Bit
        call    Delay1
        bsf     INTCON,7
        return

DelayX  movlw   .64
        goto    save

Delay1  movlw   .65
        goto    save

save    movwf   Delaycnt
redo_1  decfsz  Delaycnt,F
        goto    redo_1
        return

pause   movlw   0x3E            ;M
        movwf   mcount          ;to M counter
loadn   movlw   0x3E            ;N
        movwf   ncount          ;to N counter
decn    decfsz  ncount,F        ;decrement N
        goto    decn            ;again
        decfsz  mcount,F        ;decrement M
        goto    loadn           ;again
```

```
                return                  ;done
;
;

;
InitPorts
                bsf     STATUS,RP0      ;select pg 1

                movlw   B'00001000'
                movwf   TRISA           ;RA3 = input
                clrf    TRISB           ;make RB0-7 outputs
                bcf     STATUS,RP0      ;select page 0
                clrf    PORTB
                movlw   .6
                movwf   PORTA

                return
;
;
;The clock speed is 4.096Mhz. Dividing internal clk. by a 32 prescaler,
;the rtcc will be incremented every 31.25uS. If rtcc is preloaded
;with 96, it will take (256-96)*31.25uS to overflow i.e. 5mS. So the
;end result is that we get a rtcc interrupt every 5mS.
InitTimers
                clrf    MsdTime         ;clr timers

                clrf    KeyFlag         ;clr all flags
                bsf     STATUS,RP0      ;select pg 1
                movlw   B'10000100'     ;assign ps to rtcc
                movwf   OptionReg       ;ps = 32
                bcf     STATUS,RP0      ;select pg 0
                movlw   B'00100000'     ;enable rtcc interrupt
                movwf   INTCON          ;
                movlw   .176            ;preload rtcc
                movwf   TMR0            ;start counter
                retfie
;
ServiceInterrupts
                btfsc   INTCON,T0IF     ;rtcc interrupt?
                goto    ServiceTMR0     ;yes then service
                clrf    INTCON          ;else clr all int
                bsf     INTCON,T0IE
                return
;
```

```
ServiceTMR0
        movlw   .176            ;initialize rtcc
        movwf   TMR0
        bcf     INTCON,T0IF     ;clr int flag
        btfsc   PORTA,0         ;if msb on then do
        call    ScanKeys        ;do a quick key scan
        call    UpdateDisplay   ;update display
        return
;
;
;ScanKeys, scans the 4X4 keypad matrix and returns a key value in
;NewKey (0 - B) if a key is pressed, if not it clears the keyhit flag.
;Debounce for a given keyhit is also taken care of.
;The rate of key scan is 20mS with a 4.096Mhz clock.
ScanKeys
        btfss   KeyFlag,DebnceOn    ;debounce on?
        goto    Scan1               ;no then scan keypad
        decfsz  Debnce, F           ;else dec debounce time
        return                      ;not over then return
        bcf     KeyFlag,DebnceOn    ;over, clr debounce flag
        return                      ;and return
Scan1
        call    SavePorts           ;save port values
        movlw   B'11110111'         ;init TempD
        movwf   TempD
ScanNext
        movf    PORTB,W             ;read to init port
        bcf     INTCON,RBIF         ;clr flag
        rrf     TempD, F            ;get correct column
        btfss   STATUS,C            ;if carry set?
        goto    NoKey               ;no then end
        movf    TempD,W             ;else output
        movwf   PORTB               ;low column scan line
        nop
        btfss   INTCON,RBIF         ;flag set?
        goto    ScanNext            ;no then next
        btfsc   KeyFlag,keyhit      ;last key released?
        goto    SKreturn            ;no then exit
        bsf     KeyFlag,keyhit      ;set new key hit
        swapf   PORTB,W             ;read port
        movwf   TempE               ;save in TempE
        call    GetKeyValue         ;get key value 0 - B
        movwf   NewKey
       movlw    0x0a
```

```
                subwf   NewKey,w
                btfsc   STATUS,C
                goto    AorB
                incf    NewKey,f
                goto    self
AorB            btfss   STATUS,2
                movlw   0B

                movwf   NewKey                  ;save as New key
self            bsf     KeyFlag,ServKey         ;set service flag
                bsf     KeyFlag,DebnceOn        ;set flag
                movlw   2
                movwf   Debnce                  ;load debounce time
SKreturn
                call    RestorePorts            ;restore ports
                return
;
NoKey
                bcf     KeyFlag,keyhit          ;clr flag
                goto    SKreturn
;
;GetKeyValue gets the key as per the following layout
;
;               Col1    Col2    Col3
;               (RB2)   (RB1)   (RB0)
;
;Row1(RB4)          0       1       2
;
;Row2(RB5)          3       4       5
;
;Row3(RB6)          6       7       8
;
;Row4(RB7)          9       A       B
;
GetKeyValue
                clrf    TempC
                btfss   TempD,2                 ;first col.
                goto    RowValEnd
                incf    TempC, F
                btfss   TempD,1                 ;second col.
                goto    RowValEnd
                incf    TempC,F                 ;last col

RowValEnd
```

```
                btfss   TempE,0          ;top row?
                goto    GetValCom           ;yes then get 0,1,2
                btfss   TempE,1           ;2nd row?
                goto    Get345            ;yes the get 3,4,5,
                btfss   TempE,2           ;3rd row?
                goto    Get678            ;yes then get 6,7,8
Get9ab
                movlw   3
                addwf   TempC,F
Get678
                movlw   6
                addwf   TempC,F
                goto    GetValCom               ;do common part
Get345
                movlw   3
                addwf   TempC,F
GetValCom
                movf    TempC,W
                addwf   PCL, F
                retlw   0
                retlw   1
                retlw   2
                retlw   3
                retlw   4
                retlw   5
                retlw   6
                retlw   7
                retlw   8
                retlw   9
                retlw   0a
                retlw   0b

;
;SavePorts, saves the porta and portb condition during a key scan
;operation.
SavePorts
                movf    PORTA,W         ;Get sink value
                movwf   PABuf           ;save in buffer
                bcf     PORTA,0         ;disable all sinks
                bcf     PORTA,1
                movf    PORTB,W         ;get port b
                movwf   PBBuf           ;save in buffer
                movlw   0xff            ;make all high
                movwf   PORTB           ;on port b
```

```
        bsf     STATUS,RP0      ;select page 1
        bcf     OptionReg,7             ;enable pull ups
        movlw   B'11110000'             ;port b hi nibble inputs
        movwf   TRISB                   ;lo nibble outputs
        bcf     STATUS,RP0              ;page 0
        return
;
;RestorePorts, restores the condition of porta and portb after a
;key scan operation.
RestorePorts
                movf    PBBuf,W                 ;get port n
                movwf   PORTB
                movf    PABuf,W                 ;get port a value
                movwf   PORTA
                bsf     STATUS,RP0              ;select page 1
                bsf     OptionReg,7             ;disable pull ups
                clrf    TRISA                   ;make port a outputs
                bsf     TRISA,3                 ;exept RA3
                clrf    TRISB                   ;as well as PORTB
                bcf     STATUS,RP0              ;page 0
                return
;
;
UpdateDisplay
                movf    PORTA,W                 ;present sink value in w
                bcf     PORTA,0                 ;disable all digits sinks
                bcf     PORTA,1

                andlw   0x03
                movwf   TempC                   ;save sink value in tempC

                bsf     TempC,2                 ;preset for lsd sink
                rrf     TempC, F                ;determine next sink value
                btfss   STATUS,C                ;c=1?
                bcf     TempC,1                 ;no then reset LSD sink
                btfsc   TempC,0                 ;else see if Msd
                goto    UpdateMsd               ;yes then do Msd
UpdateLsd
                movf    MsdTime,W               ;get Lsd in w
                andlw   0x0f                    ;       /
                goto    DisplayOut
UpdateMsd
                swapf   MsdTime,W               ;get 2nd Lsd in w
                andlw   0x0f                    ;mask rest
```

```
DisplayOut
            call   LedTable          ;get digit output
            movwf  PORTB             ;drive leds
            movf   TempC,W           ;get sink value in w
            iorwf  PORTA
            return
;
;
LedTable
            addwf  PCL, F            ;add to PC low
            retlw  B'00111111'       ;led drive for 0
            retlw  B'00000110'       ;led drive for 1
            retlw  B'01011011'       ;led drive for 2
            retlw  B'01001111'       ;led drive for 3
            retlw  B'01100110'       ;led drive for 4
            retlw  B'01101101'       ;led drive for 5
            retlw  B'01111101'       ;led drive for 6
            retlw  B'00000111'       ;led drive for 7
            retlw  B'01111111'       ;led drive for 8
            retlw  B'01100111'       ;led drive for 9
            retlw  B'01110111'       ;led drive for A
            retlw  B'01111100'       ;led drive for b

           end
```

โปรแกรมในไมโครคอนโทรลเลอร์ U2

```
;*****************************************************************************
;
;This program is contained in U2 which using for receive the no of turn from U1 and also
;driving the  stepper motor from portb through the  stepper motor driver (ULN 2003A)
;       PROGRAM:    RSTEP.ASM
;
;*****************************************************************************
    LIST P=16F84
      ERRORLEVEL  -302
;
      include    <p16F84.inc>
Count           equ    0x10          ;counter for #of bits transmiteed
RcvReg          equ    0x11          ;data received
Count1          equ    0x13          ;counter for delay

MCOUNT          equ     0x0c         ;counter for delay30
NCOUNT          equ     0x0d
L_stepcount     equ     0x20         ;#of step counts for left turn
L_index         equ     0x21         ;current count for left turn
L_save          equ     0x22          ;temporaly store of current count for left turn
R_stepcount     equ     0x23         ;#of step counts for right turn
R_index         equ     0x24         ;current count for right turn
R_save          equ     0x25          ;temporaly store of current count for right turn

                org    0

Start           call    Delay4
                call    Delay4

                call    InitPorts
Again           call    Talk              ;receive serial data
                movf    RcvReg,W

C_left          btfsc   PORTA,1           ;key pressed?
                goto    C_right           ;No
                goto    L                 ;turn left
C_right         btfsc   PORTA,2           ;key pressed?
                goto    C_left            ;No
                goto    R                 ;turn right
```

```
L       movf    RcvReg,W        ;store # of step count
        movwf   L_stepcount
        movf    L_save,W
        xorlw   0x08
        btfss   STATUS,2        ;8 step turn?
        goto    L_here          ;No
        clrf    L_index         ;Yes,initialize current count
        clrf    L_save
        goto    L_there
L_here  movf    L_save,W        ;store current count
L_there call    L_turn

        goto    Again

R       movf    RcvReg,W
        movwf   R_stepcount
        movf    R_save,W
        xorlw   0x08
        btfss   STATUS,2        ;8 step turn?
        goto    R_here          ;No
        clrf    R_index         ;Yes,initialize current count
        clrf    R_save
        goto    R_there
R_here  movf    R_save,W        ;store current count
R_there call    R_turn

        goto    Again

InitPorts
        bsf     STATUS,RP0      ;select pg 1
        movlw   0x1B
        movwf   TRISA           ;make RA0-4 inputs
        clrf    TRISB           ;make RB0-7 outputs
        bcf     STATUS,RP0      ;select page 0
        clrf    PORTB           ;make all outputs low
        clrf    L_index
        clrf    R_index
        clrf    L_save
        clrf    R_save
        return

Talk    clrf    RcvReg          ;Clear all bits of RcvReg
```

```
            btfsc   PORTA,0        ;check for start bit
            goto    User           ;delay for 104/2 us
            call    Delay4         ;delay for 104 + 104/4 us

Rcvr        movlw   0x08           ; 8 data bits
            movwf   Count
Rv_next     bcf     STATUS,C       ;clear carry bit
            rlf     RcvReg,F       ;to set for MSB first
            btfsc   PORTA,0        ;Is the bit a zero or one?
            bsf     RcvReg,0       ;bit is a one
            call    DelayY
            decfsz  Count,F
            goto    Rv_next
            return                 ;reception done

Delay4      movlw   .82
            goto    save
                                   ; 104 us for 9600 baud
DelayY      movlw   .64
            goto    save


User        movlw   .34
            movwf   Count1
redo_r1     decfsz  Count1,F       ;104/2us delay
            goto    redo_r1
            goto    Talk


save        movwf   Count1
redo_x1     decfsz  Count1,F
            goto    redo_x1
            retlw   0


L_turn      bcf     STATUS,2
L_repeat    call    L_table        ;send bit pattern to portb
            movwf   PORTB
            call    Delay30        ;pause
            incf    L_index,F
            decf    L_stepcount,F
            btfss   STATUS,2       ;Is this last step?
            goto    L_next         ;No
```

```
        movf    L_index,W       ;Yes,clear current count then exit
        movwf   L_save
        goto    L_exit
L_next  btfss   L_index,3       ;Is the current 8?
        goto    L_skip          ;No
        clrf    L_index         :Yes,clear current count then repeat
L_skip  movf    L_index,W       ;save current count count then rerpeat
        goto    L_repeat
L_exit  return


R_turn  bcf     STATUS,2
R_repeat call   R_table         ;send bit patternto portb
        movwf   PORTB
        call    Delay30         ;pause
        incf    R_index,F
        decf    R_stepcount,F
        btfss   STATUS,2        ;Is this last step?
        goto    R_next          ;No
        movf    R_index,W       ;Yes,clear current count then exit
        movwf   R_save
        goto    R_exit
R_next  btfss   R_index,3
        goto    R_skip
        clrf    R_index
R_skip  movf    R_index,W
        goto    R_repeat
R_exit  return


Delay30 movlw   0x20
        movwf   MCOUNT
M_dec   movlw   0xff
        movwf   NCOUNT          ;duration delay for each step
N_dec   decfsz  NCOUNT,F
        goto    N_dec
        decfsz  MCOUNT,F
        goto    M_dec
        return

L_table addwf   PCL,F

        retlw   0xF4
        retlw   0xF6            ;bit pattern for half step left turn
```

```
        retlw   0xF2
        retlw   0xF3
        retlw   0xF1
        retlw   0xF9

R_table addwf   PCL,F
        retlw   0xF9
        retlw   0xF1
        retlw   0xF3
        retlw   0xF2            ;bit pattern for half step right turn
        retlw   0xF6
        retlw   0xF4
        retlw   0xFC
        retlw   0xF8

        end
```

วิธีใช้เครื่องมือ

1 ป้อนไฟจากแหล่งจ่ายไฟ 9 โวลต์โดยใช้หัวเสียบแบบ JACK ด้านล่างเครื่องมือ หน่วยแสดงผลจะแสดงเลข 00สองหลัก

2ป้อนจำนวนสเตปที่ต้องการให้มอเตอร์หมุนจากแผงคีย์บอร์ดซึ่งเป็นเลขฐานสิบจำนวนไม่เกิน 99

3 กดปุ่ม START ทุกๆครั้งก่อนกดปุ่ม RIGHT TURN หรือ LEFT TURNในแต่ละครั้งเพื่อต้องการหมุนซ้ายหรือขวา

การทดสอบเครื่องมือ

1.ทำเครื่องหมายบนจานโดยใช้ดินสอหรือปากาขีดบนกระดาษแล้วนำไปติดกับจานโดยใช้เทปไส

2.จ่ายไฟแก่เครื่องควบคุมแล้วป้อนจำนวนสเต็ปที่ต้องการหมุนเป็น 80

3.กดปุ่ม START กดปุ่ม RIGHT TURN หรือ LEFT TURN เมื่อมอเตอร์หมุนได้ครบ 80 สเต็ปมันจะหยุด ทำเช่นเดียวกัน(ไม่ต้องกดปุ่ม START)อีกสี่ครั้ง ถ้าเครื่องหมายที่ทำไว้กลับมาอยู่ที่เดิมแสดงว่าเครื่องควบคุมทำงานถูกต้อง เพราะมอเตอร์หมุนได้ครบรอบคือ 400 สเต็ปพอดี

ผลการทดสอบ

เครื่องควบคุมทำงานได้ถูกต้อง ความผิดพลาดจะเกิดขึ้นได้ 0.45 องศา(ครึ่งสเต็ป) ในกรณีเมื่อมีการหมุนกลับทิศ

สรุปผล

เครื่องควบคุมนี้นอกจากจะทำให้เกิดความสะดวกในการควบคุมตำแหน่งของสารในห้องปฏิบัติการนิวเคลียร์แล้ว ผู้วิจัยเห็นว่าสามารนำเครื่องนี้ไปประยุกต์ใช้กับห้องปฏิบัติการอื่นๆที่การทดลองจำเป็นต้องมีการหมุนอุปกรณ์เป็นจำนวนองศาที่ค่อนข้างแน่นอนได้ เพียงแต่ดัดแปลงจานวางสารตัวอย่างให้เหมาะสมกับอุปกรณ์โดยไม่ต้องยุ่งเกี่ยวตัวควบคุมเลย

เอกสารอ้างอิง

1. PIC16F84" **8-bit CMOS EEPROM Microcontroller** " (MICROCHIP data book)

2. AN529"**Multiplexing LED Drive and 4x4 Keypad Sampling**"(MICROCHIP application note 1997 Microchip Technology Inc.)

3. AN555 "**Software Implementation of Asynchronous Serial I/O**(MICROCHIP application note 1996 Microchip Technology Inc.)

4. ไกรวุฒิ โรจน์ประเสิฐสุด " **บอร์ดควบคุมสเต็ปเปอร์มอเตอร์ด้วยเครื่องพีซี**" (วารสารเซมิคอนดักเตอร์ ฉบับที่ 146 กุมภาพันธ์-มีนาคม 2538)