

```

                                DICE2
LIST P=16C54
;                               Dice2.asm       Version 1.01
;
; An Electronic Dice demonstration using the Microchip 16C54 PIC
;
;                               include "c:\upuc\mpasm121\p16c5x.inc"           ; fuse settings for
PICSTART 16B
    __CONFIG _CP_OFF & _WDT_ON & _RC_OSC
; -----
; Inputs and Outputs:
;   Port A
;       RA0 - Input - push button to start Dice Roll
;       RA1 - Input - unused
;       RA2 - Output - Drive to left LED display
;       RA3 - Output - Drive to right LED display
;   Port B
;       RB0 - Output - drives segment "b" of LED display
;       RB1 - Output - drives segment "a"
;       RB2 - Output - drives segment "f"
;       RB3 - Output - drives segment "g"
;       RB4 - Output - drives segment "dec. point"
;       RB5 - Output - drives segment "c"
;       RB6 - Output - drives segment "d"
;       RB7 - Output - drives segment "e"
; define data registers in RAM
;
;                               org      H'07'                               ; spare user RAM begins at 07h
Temp                res      1                               ; misc storage
Temp1               res      1                               ; misc storage
Displ_Pause        res      1                               ; counter for delay between
flashes
Segment            res      1                               ; pointer to next segment
Coarse_Delay       res      1                               ; used by general delay routine
Fine_Delay         res      1                               ;   , , , , ,
Seg_Count          res      1
Flash_Low          res      1                               ; used by Flash delay
Flash_High         res      1
ButtonFlag         res      1                               ; "stuck button" flag
Sleep_Count        res      1                               ; used by sleep delay
Random_1           res      1                               ; A number between 1 and six for
dice roll
Random_2           res      1
Hold_1             res      1                               ; temp store for random number
Hold_2            res      1
; define Reset Vector
;
;                               org      H'1FF'                           ; power-on-reset lands here
;                               goto     Start                             ; and jumps to start of program
;
;                               org      H'00'                           ; start code at bottom of EPROM
; first we check if this is a wakeup call (eg. were we in sleep mode previously?)
Start              btfsc    STATUS,NOT_PD                    ; check Power Down bit in status
register (1=watchdog, 0=power reset)
;                               goto     Start02                       ; ok, it was a normal power up
;
;                               btfss   PORTA,0                    ; it was a wake-up call, so is
the push button pressed?
;                               goto     Start01                       ; yes, check if it's stuck
;                               clr     ButtonFlag                    ; no, clear "button stuck" flag
;                               sleep                                     ; and go back to sleep again.

```

DICE2

```

Start01      btfsc  ButtonFlag,0      ; is button stuck down?
             sleep                    ; yes, go back to sleep again.

; next we initialise everything:
PIC from sleep                                ; watch dog timer is set to wake
Start02      movlw  B'00001111'        ; approx once every second
set to divide by 256                          ; watch dog runs via prescaler,
             option                    ; write this to Option reg.
             clrwdt                    ; reset watchdog timer.

reads push button  movlw  B'11110011'    ; RA2 & RA3 are output while RA0
             tris    PORTA              ; Switch all Port A lines high
             movlw  B'11111111'
             movwf  PORTA

(to drive the LEDs)  movlw  B'00000000'    ; switch all of Port B to outputs
are off)          tris    PORTB
                 movlw  B'00000000'    ; Switch all Port_B lines low
                 movwf  PORTB          ; (so that all the display LEDs

                 movlw  H'01'          ; initialise various registers
                 movwf  Flash_High
                 movwf  Flash_Low
                 movlw  D'50'
                 movwf  Sleep_Count
                 movlw  H'06'
                 movwf  Random_1
                 bsf   ButtonFlag,0    ; start with button flag set

; Main Program Loop

Main         clrwdt                    ; clear watchdog timer
             call   Random              ; update random number
             call   Flash               ; flash the decimal point
             btfss  PORTA,0            ; is the button pressed?
             goto  Main02              ; yes
             clrf   ButtonFlag         ; no, clear flag
             goto  Main                ; and loop back.

Main02      btfsc  ButtonFlag,0        ; is button stuck down?
             goto  Main                ; yes, loop back
             bsf   ButtonFlag,0        ; no, but now set flag

; Get two random numbers

Main03      movf   Random_1,W          ; get first random number
             movwf  Hold_1             ; and save it
             call   Flash              ; flash the decimal point
             call   Rand02             ; continue updating 2nd number
             btfss  PORTA,0            ; is the button released?
             goto  Main03              ; no, wait for it.
             movf   Random_2,W          ; get second random number
             movwf  Hold_2             ; and save it too

             call   Roll                ; roll the dice

             movlw  D'50'              ; reset sleep counter
             movwf  Sleep_Count
             movlw  H'FF'              ; set up display time
             movwf  Disp1_Pause

```

```

                                DICE2
                                movlw  H'20'           ; pause
                                call    Delay
; Display left digit by itself for a sec or so
Output      clrwdt                ; reset watchdog timer
            movlw  B'11111011'    ; Switch display One ON
            movwf  PORTA
            movf   Hold_1,W        ; get the first random number
            call   Display
            decfsz Displ_Pause,F    ; has timing loop finished?
            goto   Output          ; no, do it all again.
; now display both digits
Out01       movlw  H'FF'           ; set up display time
            movwf  Displ_Pause
            clrwdt                ; reset watchdog timer
            movlw  B'11111011'    ; Switch display One ON
            movwf  PORTA
            movf   Hold_1,W        ; get the first random number
            call   Display
; Now display Right Digit
            movlw  B'11110111'    ; Switch display Two ON
            movwf  PORTA
            movf   Hold_2,W        ; get the 2nd random number
            call   Display
            decfsz Displ_Pause,F    ; has timing loop finished?
            goto   Out01          ; no, do it all again.
            movlw  H'FF'           ; restart flash timer
            movwf  Flash_Low
            movlw  H'1F'
            movwf  Flash_High
            goto   Main           ; and go back to beginning
;-----
; Display digits: As each LED display has only one (common) dropping resistor,
; we must "Multiplex" the pattern on to each display, one segment at
; a time. The eye smooths all this into seeing complete digits.
; Enter with number to be displayed in "w"
Display     call    Look_Up_Digit   ; go get digit pattern
            movwf  Temp            ; save it
Disp01      movlw  H'07'           ; set up for seven segments
            movwf  Segment
Disp03      movf   Segment,W        ; get next segment pattern
            call   Look_Up_Seg
            andwf  Temp,W          ; mask out this segment
            movwf  PORTB          ; and display it
illuminated
Disp04      movlw  H'0F'           ; set up for a delay
            movwf  Fine_Delay      ; so we can see the segment
            decfsz Fine_Delay,F    ; is the delay finished?
            goto   Disp04
            decfsz Segment,F      ; is this the last segment?
            goto   Disp03         ; no, do it again.
            clrf   PORTB          ; clear display
            retlw  0

```

DICE2

```

;-----
; General purpose Delay routine: Feed multiples of 10mS in via "w" reg
Delay      movwf   Coarse_Delay
Delay01    movlw   H'FF'
           movwf   Fine_Delay
Delay02    clrwdt          ; reset watchdog timer
           decfsz  Fine_Delay,F      ; this loop takes approx 10mSec
           goto    Delay02          ; (300kHz/2/4/256)
           decfsz  Coarse_Delay,F
           goto    Delay01
           retlw   0

```

```

;-----
; Flash segments sequentially to simulate dice rolling.

```

```

Roll      movlw   H'02'          ; set up for two rolls
           movwf   Temp1
           movlw   H'08'          ; set up segment delay
           movwf   Temp

Roll00    movlw   B'11110111'     ; Switch display Two ON
           movwf   PORTA
           movlw   B'00000010'    ; select segment "a"
           movwf   PORTB          ; write pattern to port B
           movf    Temp,w         ; wait a while
           call    Delay
           movlw   B'00000001'    ; "b"
           movwf   PORTB
           movf    Temp,w
           call    Delay
           movlw   B'00100000'    ; "c"
           movwf   PORTB
           movf    Temp,w
           call    Delay
           movlw   B'01000000'    ; "d"
           movwf   PORTB
           movf    Temp,w
           call    Delay

           movlw   B'11111011'    ; Switch display One ON
           movwf   PORTA
           movlw   B'01000000'    ; "d" again
           movwf   PORTB
           movf    Temp,w
           call    Delay
           movlw   B'10000000'    ; "e"
           movwf   PORTB
           movf    Temp,w
           call    Delay
           movlw   B'00000100'    ; "f"
           movwf   PORTB
           movf    Temp,w
           call    Delay
           movlw   B'00000010'    ; back to "a"
           movwf   PORTB
           movf    Temp,w
           call    Delay

           decfsz  Temp1,F        ; is this the last roll?
           goto    Roll00         ; no, go around again
           clrf   PORTB          ; yes, switch off LEDs
           retlw   0

```

```

;-----
; Decimal Point Flash routine. Used to show that PIC is "awake".
; If the button hasn't been pressed for fifty flashes, the processor goes

```

DICE2

; to sleep to conserve power.

```
Flash      decfsz   Flash_Low,F      ; has low timer run out?
           retlw    0                ; no, return
           movlw   H'FF'            ; yes, reset timer
           movwf   Flash_Low
           decfsz  Flash_High,F      ; has high timer run out?
           retlw   0                ; no, return
           movlw   H'1F'            ; yes, reset timer
           movwf   Flash_High
           movlw   B'11111011'      ; Switch display One ON
           movwf   PORTA
           movlw   B'00010000'      ; set up bit pattern for dec
point
           movwf   PORTB            ; write it out
           movlw   H'05'
           call    Delay            ; and pause a bit
           clrfs   PORTB           ; switch it off again
```

; is it sleepy time yet?

```
           decfsz  Sleep_Count,F    ; has count expired?
           retlw   0                ; no.
           movlw   H'FF'            ; yes, shut down all ports
           tris    PORTA            ; (make them inputs
           tris    PORTB            ; to conserve power)
           sleep   ; and goto sleep!
```

; Random number generator: We constantly decrement two numbers to use as the
; seed for the dice roll. As this happens about 4000 times per second
; it is impossible to guess which number will be present when the
; button is pressed (for display 1) and released (for display 2).

```
Random     decfsz   Random_1,F      ; has it got to zero?
           goto    Rand01          ; no, skip it
           movlw   H'06'            ; yes, reset it to 6
           movwf   Random_1
           goto    Rand02

Rand01     nop                    ; add some delays so that we
spend equal time
           nop                    ; in branch, to make roll truly
random.
```

; now do 2nd Number

```
Rand02     decfsz   Random_2,F      ; has it got to zero?
           goto    Rand03          ; no, skip it
           movlw   H'06'            ; yes, reset it to 6
           movwf   Random_2
           retlw   0                ; return

Rand03     nop                    ; add some more delays
           nop
           retlw   0                ; return
```

; Look-up tables for display bit patterns: These use a "computed goto" technique.
; The value in "w" is added to the program counter to cause program execution
; to jump to the appropriate line. The "Return-with-Literal-in-w" instruction
; returns the required value.

```
Look_Up_Seg  addwf   PCL,F          ; return bit pattern for successive segments
           nop                    ; (we skip the "zero" value)
```

```

                                DICE2
retlw B'00000010'                ; a
retlw B'00000001'                ; b
retlw B'00100000'                ; c
retlw B'01000000'                ; d
retlw B'10000000'                ; e
retlw B'00000100'                ; f
retlw B'00001000'                ; g

; -----
Look_Up_Digit   addwf   PCL,F      ; return bit patterns for each digit (from 1
-> 6)           nop              ; (we skip the "zero" digit)

                                ;
                                ;           a
retlw B'00100001'                ; '1'           xxxxxxx
retlw B'11001011'                ; '2'           f x           x b
retlw B'01101011'                ; '3'           x           g x
retlw B'00101101'                ; '4'           xxxxxx
retlw B'01101110'                ; '5'           e x           x c
retlw B'11101110'                ; '6'           x           x
                                ;           xxxxxx
                                ;           d
                                ;
                                ; bits       76543210
                                ; segments   edcXgfab   (X = dec

point)

; -----
END
->

```