To control LED blinking, Hayden Callender has developed a program that

1. Toggles the top three LEDs every half second in sequence: bottom, middle, top.
2. Uses a 4 MHz crystal for a 1 microsecond internal clock period.
3. Uses Timer2 to obtain a ten millisecond loop time.
4. Toggles bit 5 of PORTA each time around the mainline loop (to test loop time).
5. Echoes RPG "encoder emulator" outputs to the bottom two LEDs in response to presses of the "INC" and "DEC" pushbuttons.

Program hierarchy:

Mainline
  Initial
  Blink
    BlinkTable
  LoopTime

IntService
  RPG

```
list P=PIC16F877, F=INHX8M, C=160, N=77, ST=OFF, MM=OFF, R=DEC, X=OFF
#include P16F877.inc
__config(_CP_OFF & _PWRTE_ON & _XT_OSC & _WDT_OFF & _BODEN_OFF)

Equates:

Bank0RAM equ H'20' ;Start of Bank 0 RAM area
MaxCount equ 100 ;Number of loops in half a second

Variables:

cblock Bank0RAM
  W_TEMP ;Temporary storage for W during interrupts
  STATUS_TEMP ;Temporary storage for STATUS during interrupts
  BLNKCNT ;Loop counter for blinking LEDs every half sec.
  TEMP ;Temporary variable for BlinkTable subroutine
endc

Macro definitions:

MOVLF macro literal,dest
  movlw literal
  movwf dest
endm

MOVFF macro source,dest
  movf source,W
  movwf dest
endm

Vectors:

org H'000' ;Reset vector
  nop

org H'004' ;Interrupt vector
  goto IntService ;Branch to interrupt service routine
```
This subroutine reads PORTD and retains only the upper three LED bits. It uses them to access table. It returns in W the bits of PORTD to be toggled.

BlinkTable

```asm
MOVFF PORTD, W ; Copy present state of LEDs to TEMP
andlw B'00001111' ; Keep only bits to be shifted
addwf PCL, F ; Change PC with PCLATH and offset in W
retlw B'00000001', 0000->0001 forward (initialize)
retlw B'00000011', 0001->0010 to right
retlw B'00000110', 0010->0100 to left
retlw B'00000010', 0100->0001 reinitialize to forward
retlw B'00000010', 0101->0001 reinitialize to forward
retlw B'00000111', 0111->0001 reinitialize to forward
retlw B'00000110', 0111->0001 reinitialize to forward
retlw B'00001100', 1000->0001 backward to forward
retlw B'00000100', 1001->0001 reinitialize to forward
retlw B'00001111', 1011->0001 reinitialize to forward
retlw B'00001001', 1100->0001 reinitialize to forward
retlw B'00001000', 1101->0001 reinitialize to forward
retlw B'00001011', 1110->0001 reinitialize to forward
retlw B'00001010', 1111->0001 reinitialize to forward
```

;;;;;; End of Tables

;;;;;; Mainline program

Mainline

```asm
call Initial ; Initialize everything
MainLoop
call Blink ; Blink upper 4 LEDs
call LoopTime ; Force loop time to be ten milliseconds
goto MainLoop
```

;;;;;; Initial subroutine

```asm
Initial
bsf STATUS, RP0 ; Set register access to bank 1
MOVLF B'00000100', ADCON1 ; Select PORTA pins for ADC or digital I/O
MOVLF B'00000111', TRISA ; Set I/O for PORTA
MOVLF B'00001111', TRISB ; Set I/O for PORTB
MOVLF B'11010111', TRISC ; Set I/O for PORTC
clf TRISD ; Set I/O for PORTD
MOVLF B'00000100', TRISE ; Set I/O for PORTE
MOVLF 249, PR2 ; Set up Timer2 for a looptime of 10 ms
```

bcf STATUS, RP0 ; Set register access back to bank 0
MOVLF B'01001101', T2CON ; Finish set up of Timer2 (see page 62)
clf PORTD ; Turn off LEDs
MOVLF B'11010000', INTCON ; Enable RB0/INT interrupts (see page 98)
return

;;;;;; Blink subroutine

```asm
Blink
```

;;;;;; This subroutine blinks a new LED every 1.5 second.
decsz  BLNKCNT, F    ; Decrement loop counter and return if not zero

goto   BlinkEnd

movlw  MaxCount       ; Reinitialize BLNKCNT

movwf  BLNKCNT

call   BlinkTable      ; Get bits to change into W

xorwf  PORTD, F        ; and toggle them into PORTD

BlinkEnd

return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; This subroutine waits for Timer2 to complete its ten millisecond count sequence.

LoopTime

btfss  PIR1, TMR2IF    ; Check whether ten milliseconds are up

goto   LoopTime

bcf    PIR1, TMR2IF    ; Clear flag

movlw  B'00100000'     ; Toggle PORTA, bit 5

 xorwf  PORTA, F       ; for testing loop time

return

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; This interrupt service routine fields all interrupts. It first sets aside W and STATUS. It assumes that direct addressing will not be used in the mainline code to access Bank 1 addresses (once the Initial subroutine has been executed and interrupts enabled). It polls each possible interrupt source to determine whether it needs service.

IntService

end