

LISTING 1—TIME-DELAY ROUTINE

```
int_test_timer0

    btfss    INTCON,    TMR0IF    ; test if Timer0 IRQ was active
    goto    int_test_INT1        ;if not, check
next INT source
    bcf     INTCON,    TMR0IE    ;disable Timer0 int.
    bcf     INTCON,    TMR0IF    ; clear Timer1 H/W flag

    ;Jobs to do here:    INC the PACER_CLOCK variable

    incf    PACER_CLOCK        ;This variable is a free running
counter

    ;RELOAD TIMER0 now: (Timer0 set for 10 milliseconds pacer clock)
    ;Note: Rollover occurs after the Timer reaches 0xFFFF
    ;Required number of ticks is calculated as: 0xFFFF-(TMR0H:TMR0L)

    movlw   0x0D                ; High byte
    movwf   TMR0H                ; Reload Timer0 high
    movlw   0x61                ; LOW byte
    movwf   TMR0L                ; Reload Timer0 low.

    ;Note: for an internal clock period of 0.161002 microseconds, count 62111 clocks
    ;to make up the 10 millisecond interval.

    bcf     INTCON,    TMR0IF    ; clear Timer1 H/W flag
    bsf     INTCON,    TMR0IE    ;Re-enable the Timer0
interrupt
    bsf     TOCON,     TMR0ON    ; turn ON Timer0
module

int_test_INT1
    ;code for INT1 starts from here...
    .
    .
retfie    ;return from interrupt
```

Somewhere in the code:

```
    ;*****
    movf    PACER_CLOCK, w    ;get the current Pacer_Clock
state
    addlw   D'10'            ; to
count up to 10 Timer0 overflow periods
    movwf   TIME_VAR        ;load the time variable
with the contents of ...

    ;...PACER_CLOCK plus 10
wait_loop100
    ;do some jobs here...
    ;...
    ;...
    movf    PACER_CLOCK, w
    xorwf   TIME_VAR, w    ;check
if PACER_CLOCK was incremented 10 times
    tstfsz    WREG
    ;if zero, files are equal = Timed Out !
    goto    wait_loop100
    ;otherwise wait longer

    ;*****
    ;
```