**APPLICATION OPERATION**

An optical pyrometer is a device that allows non-contact measurement of temperature in the range from less than 1000° F to more than 3000° F. It operates on the principle of comparing a hot filament against a background of the object to be measured using a simple optical system similar to half of a binocular.

Normally, these devices use no electronics at all, they consist of a power rheostat, an analog meter calibrated to read temperature and on/off switch. As portability is normally a requirement, a battery source such as alkaline 'C' or 'D' cells is used to power the device.

Incorporating a PIC12C5XX into an optical pyrometer has the following benefits:

- Improved battery life from using a PWM MOSFET to control power to the filament, thus saving typically half the energy consumed.
- Elimination of the power-wasting rheostat with two inexpensive momentary pushbuttons.
- Elimination of the on/off switch.
- Automatic power-off, thus saving the batteries should the user accidentally leave the filament on.

Only four external components are required to implement the above functions, including the momentary switches. The internal RC clock and reset circuitry of the PIC12C5XX are more than adequate for this application. The internal pull-ups on inputs and the wake-up on pin change functions are extremely useful in this application.

**Functions**

Pressing either the up or down key will "wake up" the PICmicro™ MCU and turn the unit on. The power to the filament is retained from the last measurement. The power can be adjusted up or down by holding down the appropriate key. Pressing both keys at once for more than a certain amount of time (0.8 second) will turn the unit off when the keys are released. If no keys are pressed for more than a certain amount of time (2.5 minutes), the unit will turn itself off. Power draw in the "off" condition is negligible compared to battery internal self-discharge leakage.

**Author:** Spehro Pefhany  
**Barclay Instruments Inc.**  
**Burlington, Ontario, Canada**  
**email: speff@interlog.com**
GRAPHICAL HARDWARE PRESENTATION

The hardware used is shown below. No supply regulation is used since the PICmicro will operate over a wide range of supply voltage and the analog meter reads the actual average voltage on the filament. The weak pullup on GP3 also serves GP4 and GP5.

MICROCHIP TOOLS USED

Development Tools:
PISTART® Plus

Assembler/Compiler Version:
MPLAB 3.22, MPASM 1.5
APPENDIX A: SOURCE CODE

;******************************************************************************
; This program is for a PIC2C5XX microcontroller that will control a simple
; optical pyrometer using PWM and providing timed auto-off and on/off
; functions.
; (C) 1997 Spehro Pefhany, all rights reserved
; Rev. A
; - verified current in "off" state is less than 1uA
; - 15.544 msec execution for bigloop in simulator
;******************************************************************************
listp=12c509, r=HEX
include <P12C509.INC>; sfr definitions
#define POWER_TIME 0x26 ; desired time in seconds, divided by 3.96 (2.5min)
#define BOTHKEY_TIME 0x1F ; desired time in seconds, divided by 0.01554 (0.5 sec)
ctr equ 09 ; counter generates time ramp for comparison
c_valequ0A ; current value of PWM from 0..FF
repeats equ 0B; number of times to repeat the PWM loop
pwr_preequ0C; power off prescaler
pwr_ctr equ 0D ; power off counter
bothky_ctr equ 0E; timer for both keys pressed to power off

org 0

coldstart: ;******************************************************************************
; Setup options
;******************************************************************************
movwf OSCCAL ; may as well use calibrated time
movlw H'00'
btfss 7,STATUS ; if a wake up, don't reset the set value
movwf c_val
movw STATUS ; make sure that the page bit is cleared, of course it
; is supposed to be

movlH'FB'; only GP2 is an output, other are inputs
trisGPIO; do it

movlH'00'; set up for weak pullups and wake up on pin change
option
;******************************************************************************
; Now set up the power off timer to full time
;******************************************************************************
movlH'FF'
movwf pwr_pre; set up prescaler
movlH POWER_TIME
movwf pwr_ctr; set up counter
;******************************************************************************
; And the bothky timer
;******************************************************************************
movlw BOTHKEY_TIME
movwf bothky_ctr; set up counter
;******************************************************************************
; The central PWM loop for controlling the filament intensity
;
; We only check for zero outside the tight inner loops
bigloop:
    movlwD'20' ; number of times to repeat
    movwfrepeats
    movfc_val,w; get the current value
    btfscSTATUS,2; skip if not zero
    gotoiszero
reploop:
    ; Inner PWM loop.. if c_val is 0, don't even start
    ;
    ; The below gives exactly the same time to completion, regardless of
    ; on and off times, with some overhead where the output is off-- except
    ; for 0 input, must simulate that.
    ;*****************************************************************************
    movfc_val,w; do it again for the loop
    movwfctr
    bsfGPIO,2; turn the output on
onloop:
    decfszctr,f
    gotoonloop
    bcfGPIO,2; turn the output off
comfc_val,w; get complement of current value
    movwfctr
    incfcctr,f; add 1 to it
offloop:
    decfszctr,f
    gotooffloop
    decfszrepeats,f
    goto reploop; repeat multiple times
    gotocontinue
    ;*****************************************************************************
    ; Here we handle the special case of a zero c_val
    ;
    ; We even the time out exactly, even though it isn't critical in this case
    ;*****************************************************************************
    iszero:
    reploop1:
        bcfGPIO,2; just to be sure
        movlwH'00'; counter
        movwfctr
    offloop1:
        decfszctr,f
        gotooffloop1
        nop ; even out the delay to make it exactly the
        ; same as the controlled on/off time loops
        nop
        nop ; even it out exactly
        decfszrepeats,f
        goto reploop1; repeat multiple times
        nop ; even time out exactly
    continue:
    ;*****************************************************************************
    ; The "extra" delay loop for limiting the maximum light intensity
    ; with new batteries
    ;*****************************************************************************
    ; code goes in here, if required
    ;*****************************************************************************
    ; Now poll keys, check for time-out of power-off timer and
    ; do any key operations required.
    ;*****************************************************************************
    btfssGPIO,0; skip if no up key
    gotoupkeyp
    btfssGPIO,1; skip if no down key
goto downkeyp
;******************************************************************************
; Check to see if we are in timeout situation on bothkey timer
;******************************************************************************
movf bothky_ctr,f
btfsc STATUS,2 ; if not timed out, then continue
gotosnooze
;******************************************************************************
; Reset the timer
;******************************************************************************
movlw BOTHKEY_TIME
movwf bothky_ctr ; reset the timer for both keys pressed
;******************************************************************************
; There are no keys pressed, so count the power-off timer down and
; delay so it is the same as the other paths
;
; We hit this roughly every 16ms. power off delay should be 2-3 minutes,
; so we need a count of 9,400 to get 2.5 minutes, a divide by 255 will
; give roughly 4 second per secondary count.
;******************************************************************************
decfsz pwr_pre ; count down prescaler
gotoback
gotodecount
back:nop
nop
nop
nop
gotobigloop
deccount:
  nop ; keep time same as other cycles
decfsz pwr_ctr,f; count down main counter
gotobigloop
;******************************************************************************
; Now we have the power-off timer timeout, so we put the micro to
; sleep. It will wake upon a pin change.
;******************************************************************************
snooze:
  bcf GPIO,2 ; just to be sure
  movf GPIO,w ; read all pins, as manual recommends
  sleep ; goes to reset on wake-up
;******************************************************************************
; upkeyp handles an up key
;******************************************************************************
upkeyp:btfss GPIO,1 ; this keeps time same as downkeyp as well
gotobothkeyp
cmpc_val,w; see if it was FF
btfsc STATUS,2; skip if it wasn't FF
decfc_val,f
incfc_val,f
gotocommon1
;******************************************************************************
; downkeyp handles a down key
;******************************************************************************
downkeyp:
  movf c_val,w; see if it is zero
  btfsc STATUS,2; skip if it wasn't zero
  incf c_val,f
decfc_val,f
gotocommon1
;******************************************************************************
; bothkeyp handles situation where both keys are pressed
;
bothkeyp:
    movf bothky_ctr, w; check if already zero
    btfss STATUS, 2
    decf bothky_ctr, f; if not already zero, decrement
    nop ; keep same time as others
    goto common
;*****************************************************************************
; Common ending for upkeyp and downkeyp and bothkeyp
;*****************************************************************************
common1:nop; keep all the times the same
common:
;*****************************************************************************
; Set up the power off timer to full time again
;*****************************************************************************
    movlw H'FF'
    movwf pwr_pre; set up prescaler
    movlw POWER_TIME
    movwf pwr_ctr; set up counter
    goto bigloop

end