```
' {$STAMP BS2p}
' This program implements a 60-hour countdown timer with a user-friendly
' rotary-encoder (twist-knob) interface and LCD Serial Backpack display.
' When first powered up, the display shows "00:00:00" and waits for
' the user to twist the knob to set the hours. Clockwise increases the
' setting, counter-clockwise reduces it. When the hours are set, the
' user pushes the knob in to set the minutes and seconds in the same
' way. Once the seconds are set, pushing the knob in one more time
' starts the timer. The display counts down to zero, then turns on the
' output.
' This application relies on an external timer as an accurate source of
' 2-Hz 'ticks.' Typical accuracy is within 2-3 seconds over the maximum
' timing period of 59:59:59 (almost 60 hours). Another interesting
' feature of the application is its control of the rotary-encoder power
' supply. Since the encoder's LEDs draw almost 20 mA of current, the
' program shuts them off when they're not needed and thereby conserves
' battery power.
' P3 is the OUTPUT pin controlled by the timer.
' P2 provides power to the rotary encoder LEDs.
' ================================
' Variables and constants.
' ================================
old VAR Byte  ' Previous bit pattern of rotary encoder.
new VAR Byte  ' Current " " " " "
directn VAR Bit ' Direction of knob rotation.
cnt VAR Byte ' Number dialed in by encoder.
hours VAR Byte ' Timer hours setting.
minutes VAR Byte ' Timer minutes setting.
seconds VAR Byte ' Timer seconds setting.
temp VAR Byte ' Temporary variable used by display routine.
prnPos VAR Byte ' Printing position on LCD screen.
btn VAR Byte ' Workspace variable for Button command.
case VAR Byte ' Offset for Branch command.
I CON 254 ' LCD Backpack instruction prefix (see note).
CLR CON 1 ' LCD Backpack clear-screen instruction.
N2400 CON 17405 'Set P0 baudrate at 2400.
' NOTE: This program is written for the rev3A Backpack firmware,
' which uses an instruction prefix, rather than a toggle. The new
' firmware makes this latest Backpack "reset proof" since the
' controller can always put the LCD into a known state by clearing
' the screen (and optionally also turning the cursor on/off).
' ================================
' Main Program Start
' ================================
Begin:
IF DIR3 = 1 THEN repeat
repeat:
READ 0, hours
READ 1, minutes
READ 2, seconds
SEROUT 0,N2400,[DEC hours]
SEROUT 0,N2400,[DEC minutes]
SEROUT 0,N2400,[DEC seconds]
LOW 3 ' Turn off the output pin.
HIGH 2 ' Turn on power to encoder LEDs.
PAUSE 1000 ' Wait a sec for LCD initialization.
```

```
SEROUT 0,N2400,[I,CLR] ' Clear the LCD screen
new = INS & $C0 ' Get initial state of encoder pins.
prnPos = 132 ' Set print position to 4 (128+4)
GOSUB Display ' Put 0s on the display.
' ================================
' User Setup of Time Duration
' ================================
Setup:
GOSUB rotary ' Check the knob.
SEROUT 0,N2400,[I,prnPos] ' Position cursor on the display.
GOSUB showDigs ' Display digits.
BUTTON 5,0,255,0,btn,1,pushed
' Check for knob push on pin 5.
GOTO Setup ' Loop.
' If the knob is pushed in, causing a low on pin 5, the program
' jumps from setup to here. It checks the current printing position
' to determine whether the user has been setting hours, minutes, or
' seconds and determine what to do next.
pushed:
case = prnPos-132/3 ' Convert position to 0-2.
BRANCH case,[setHours,setMins,setSecs] ' Branch based on 0-2
setHours:
hours = cnt ' Put the count into hours.
GOTO continue ' Continue setting timer.
setMins:
minutes = cnt ' Put the count into minutes.
GOTO continue ' Continue setting timer.
setSecs:
seconds = cnt ' Put the count into seconds.
GOTO runTimer ' And start the countdown.
continue:
cnt = 0 ' Continue: clear count for next.
prnPos = prnPos+3 ' Move to next screen position.
GOTO Setup ' Get more input from user.
' ================================
' Timing Countdown
' ================================
runTimer:
old = 0 ' Initialize "old" to track ticks from timer.
LOW 2 ' Turn off the encoder.
' This code counts changes in state from the external timer. Every
' fourth change (transition from 0-1 or 1-0) of the 2-Hz clock means
' that a second has passed. When that happens, the program subtracts
' 1 from the seconds, minutes and hours.
DoTiming:
IF 1 = directn THEN DoTiming' No change? Loop.
old = old + 1 ' Changed: increment old.
new = old & %11 ' Look at bottom two bits of old.
IF new <> 3 THEN DoTiming ' Loop is not 3 (4th count, 0,1,2,3)..
seconds = seconds - 1 ' Fourth count: decrement seconds.
IF seconds <> 255 THEN update ' If not underflow (-1 = 255), update.
seconds = 59 ' Underflow: wrap around to 59 seconds.
minutes = minutes -1 ' Seconds underflowed: borrow 1 from mins.
IF minutes <> 255 THEN update ' If not underflow (-1 = 255), update.
minutes = 59 ' Underflow: wrap to 59 minutes.
hours = hours - 1 ' Minutes underflowed: borrow 1 from hours.
```

```
update:
GOSUB Display ' Display new hours/mins/secs.
check:
IF hours <> 0 THEN DoTiming ' If not 00:00:00, continue timing.
IF minutes <> 0 THEN DoTiming
IF seconds <> 0 THEN DoTiming
HIGH 3 ' Time's up: turn on the output.
GOTO Begin
' hold: GOTO hold ' Endless loop: reset to start again.
' ================================
' Subroutines
' ================================
' Check the rotary encoder. If it has moved, determine direction and
' adjust the value of the variable "count" accordingly.
rotary:
old = new & $C0 ' Make old = top two bits of new.
again:
new = INS & $C0 ' Make new = top two bits of pins.
IF new = old THEN done ' No change? Done.
directn = old ^ new ' Change: determine direction.
IF directn = 1 THEN CW ' Clockwise: goto routine below.
cnt = cnt - 1 ' Counterclockwise: decrement count.
IF cnt <> 255 THEN PASS  ' If count < 0, then count = 59.
cnt = 59
PASS:
RETURN ' Return to main program.
CW:
cnt = cnt + 1 ' Clockwise: increment count.
IF cnt <> 60 THEN done ' If count = 60, wrap around to 0.
cnt = 0
done:
RETURN ' Return to main program.
' Display the hour:minute:second digits on the LCD screen.
Display:
SEROUT 0,N2400,[I,132] ' Start at hours position.
cnt = hours ' Show hours digits.
GOSUB showDigs
SEROUT 0,N2400,[":"] ' Colon.
cnt = minutes ' Now minutes.
GOSUB showDigs
SEROUT 0,N2400,[":"] ' Colon.
cnt = seconds ' Now seconds.
GOSUB showDigs
RETURN ' Return to main program.
' Display the two-digit value stored in count on the LCD.
showDigs:
temp = cnt/10 ' Get the tens-place digit.
SEROUT 0,N2400,[DEC temp] ' Put it on the display.
temp = cnt//10 ' Get the ones-place digit.
SEROUT 0,N2400,[DEC temp] ' Put it on the display.
RETURN ' Return to main program.
```