' dc_motor.bas (PIC16F88 microcontroller)
'
'Design Example
' Position and Speed Control of a dc Servo Motor.
'
' The user interface includes a keypad for data entry and an LCD for text
' messages. The main menu offers three options: 1 - position control,
' 2 - speed control, and 3 - position control gain and motor PWM control.
' When in position control mode, pressing a button moves to indexed positions
' (1 - 0 degrees, 2 - 45 degrees, 3 - 90 degrees, and 4 - 180 degrees). When
' in speed control mode, pressing 1 decreases the speed, pressing 2 reverses
' the motor direction, pressing 3 increases the speed, and pressing 0 starts
' the motor at the indicated speed and direction. The motor is stopped with
' a separate push-button switch. When in gain and PWM control mode,
' pressing 1/4 increases/decreases the proportional gain factor (kP)
' and pressing 3/6 increases/decreases the number of PWM cycles sent
' to the motor during each control loop update.
'
' Pressing the "#" key from the position, speed, or gain menus returns control
' back to the main menu. E-Lab's EDE1144 keypad encoder is used to detect
' when a key is pressed on the keypad and transmit data (a single byte per
' key press) to the PIC16F88. Acroname's R179-6V-ENC-MOTOR servo motor is
' used with their S17-3A-LV H-bridge for PWM control. A second PIC (16F84),
' running dc_enc.bas, is used to communicate to an Agilent HCTL-2016
' quadrature decoder/counter to track the position of the motor encoder.
' The 16F88 communicates to the 16F84 via handshake (start) and serial
' communication lines.
'
' Configure the internal 8MHz internal oscillator
DEFINE OSC 8
OSCCON.4 = 1 : OSCCON.5 = 1 : OSCCON.6 = 1
'
' Disable comparators (thereby allowing use of pins for I/O)
'CMCON.0 = 1 : CMCON.1 = 1 : CMCON.2 = 1
'CVRCON.6 = 0 : CVRCON.7 = 0
'
' Turn off A/D converters (thereby allowing use of pins for I/O)
ANSEL = 0
'
' Define I/O pin names
key_serial Var PORTB.0     ' keypad serial interface input
key_valid Var PORTB.1      ' keypad serial interface handshake line
motor_dir Var PORTB.7      ' motor H-bridge direction line
motor_pwm Var PORTB.6      ' motor H-bridge pulse-width-modulation line
stop_button Var PORTB.4    ' motor stop button
enc_start Var PORTB.2      ' signal line used to start encoder data transmission
enc_serial Var PORTA.7     ' serial line used to get encoder data from the 16F84
enc_rst Var PORTB.5        ' encoder counter reset signal (active low)
'
' Declare Variables
key_value Var BYTE         ' code byte from the keypad
motor_pos Var Word         ' current motor position in degrees
new_motor_pos Var Word     ' desired motor position (set point) in degrees
error Var Word             ' error magnitude between current and desired positions
motor_speed Var BYTE       ' motor speed as percentage of maximum (0 to 100)
motion_dir Var BIT ' motor direction (1:CW/Forward 0:CCW/Reverse)
on_time Var WORD ' PWM ON pulse width
off_time Var WORD ' PWM OFF pulse width
calc_pos Var WORD ' motor encoder position (highbyte and lowbyte)
i Var Byte ' counter variable for FOR loops
kp Var BYTE ' proportional gain factor position control
pwm_cycles Var BYTE ' # of PWM pulses sent during the position control loop

' Define constants
key_mode Con 0 ' 2400 baud mode for serial connection to keypad
key_1 Con $30 ' hex code for the 1-key on the keypad
key_2 Con $31 ' hex code for the 2-key on the keypad
key_3 Con $32 ' hex code for the 3-key on the keypad
key_4 Con $34 ' hex code for the 4-key on the keypad
key_5 Con $35 ' hex code for the 5-key on the keypad
key_6 Con $36 ' hex code for the 6-key on the keypad
key_7 Con $38 ' hex code for the 7-key on the keypad
key_8 Con $39 ' hex code for the 8-key on the keypad
key_9 Con $41 ' hex code for the 9-key on the keypad
key_star Con $43 ' hex code for the *-key on the keypad
key_0 Con $44 ' hex code for the 0-key on the keypad
key_pound Con $45 ' hex code for the #-key on the keypad
CW Con 1 ' motor clockwise (forward) direction
CCW Con 0 ' motor counterclockwise (reverse) direction
pwm_period Con 50 ' period of each motor PWM signal cycle (in microsec)
enc_mode Con 2 ' 9600 baud mode for serial connection to the encoder IC

' Initialize I/O and variables
TRISB.6 = 0 ' configure H-bridge DIR pin as an output
TRISB.7 = 0 ' configure H-bridge PWM pin as an output
motion_dir = CW ' starting motor direction: CW (forward)
motor_pos = 0 ' define the starting motor position as 0 degrees
motor_speed = 50 ' starting motor speed = 50% duty cycle
kp = 50 ' starting proportional gain for position control
pwm_cycles = 20 ' starting # of PWM pulses sent during the position control loop
Low motor_pwm ' make sure the motor is off to begin with
Low enc_start ' disable encoder reading to begin with
Gosub reset_encoder ' reset the encoder counter

' Wait 1/2 second for everything (e.g., LCD) to power up
Pause 500

' Receive dummy byte from the PIC16F84 to ensure proper communication
SERIN enc_serial, enc_mode, key_value

' Wait for a keypad button to be pressed (i.e., polling loop)
Gosub main_menu ' display the main menu on the LCD
main:
    Serin key_serial, key_mode, key_value
    If (key_value = key_1) Then
        Gosub reset_encoder
        Gosub position
    Else
        If (key_value = key_2) Then
motor_speed = 50    ' initialize to 50% duty cycle
Gosub speed

Else
If (key_value = key_3) Then
Gosub adjust_gains
Endif : Endif : Endif
Goto main   ' continue polling keypad buttons
End ' end of main program

' Subroutine to display the main menu on the LCD
main_menu:
Lcdout $FE, 1, "Main Menu:"
Lcdout $FE, $C0, "1:pos. 2:speed 3:gain"
Return

' Subroutine to reset the motor encoder counter to 0
reset_encoder:
Low enc_rst         ' reset the encoder counter
High enc_rst        ' activate the encoder counter
Return

' Subroutine for position control of the motor
position:
' Display the position control menu on the LCD
Lcdout $FE, 1, "Position Menu:"
Lcdout $FE, $C0, "1:0 2:45 3:90 4:180 #:<"

' Wait for a keypad button to be pressed
Serin key_serial, key_mode, key_value

' Take the appropriate action based on the key pressed
If (key_value == key_1) Then
    new_motor_pos = 0
Else
If (key_value == key_2) Then
    new_motor_pos = 45
Else
If (key_value == key_3) Then
    new_motor_pos = 90
Else
If (key_value == key_4) Then
    new_motor_pos = 180
Else
If (key_value == key_pound) Then
    Gosub main_menu
    Return
Else
    Goto position
Endif : Endif : Endif : Endif : Endif

' Position control loop
While (stop_button == 0)    ' until the stop button is pressed
    ' Get the encoder position (enc_pos)
Gosub get_encoder

' Calculate the error signal magnitude and sign and set the motor
direction
' Convert encoder pulses to degrees. The encoder outputs 1230 pulses
' per 360 degrees of rotation
motor_pos = enc_pos * 36 / 123
If (new_motor_pos >= motor_pos) Then
    error = new_motor_pos - motor_pos
    motor_dir = CW
Else
    error = motor_pos - new_motor_pos
    motor_dir = CCW
Endif

' Set the PWM duty cycle based on the current error
If (error > 20) Then    ' use maximum speed for large errors
    motor_speed = kp
Else
    ' Perform proportional position control for smaller errors
    motor_speed = kp * error / 20
Endif

' Output a series of PWM pulses with the speed-determined duty cycle
Gosub pwm_periods      ' calculate the on and off pulse widths
For I = 1 to pwm_cycles
    Gosub pwm_pulse     ' output a full PWM pulse
Next I

' Display current position and error on the LCD
Lcdout $FE, 1, "pos:", DEC motor_pos, " error:", DEC error
Lcdout $FE, $C0, "exit: stop button"

Wend

Goto position   ' continue the polling loop
Return          ' end of subroutine, not reached (see the #-key If above)

' Subroutine to get the encoder position (enc_pos) from the counter
get_encoder:
    ' Command the PIC16F84 to transmit the low byte
    High enc_start

    ' Receive and display the high byte
    SERIN enc_serial, enc_mode, enc_pos.HighBYTE

    ' Command the PIC16F84 to transmit the high byte
    Low enc_start

    ' Receive and display the low byte
    SERIN enc_serial, enc_mode, enc_pos.LowBYTE
Return

' Subroutine to calculate the PWM on and off pulse widths based on the desired
' motor speed (motor_speed)
pwm_periods:
  ' Be careful to avoid integer arithmetic and
  ' WORD overflow [max=65535] problems
  If (pwm_period >= 655) Then
      on_time = pwm_period/100 * motor_speed
      off_time = pwm_period/100 * (100-motor_speed)
  Else
      on_time = pwm_period*motor_speed / 100
      off_time = pwm_period*(100-motor_speed) / 100
  Endif
Return

' Subroutine to output a full PWM pulse based on the data from pwm_periods
pwm_pulse:
  ' Send the ON pulse
  High motor_pwm
  Pauseus on_time

  ' Send the OFF pulse
  Low motor_pwm
  Pauseus off_time
Return

' Subroutine for speed control of the motor
speed:
  ' Display the speed control menu on the LCD
  Gosub speed_menu

  ' Wait for a keypad button to be pressed
  Serin key_serial, key_mode, key_value

  ' Take the appropriate action based on the key pressed
  If (key_value == key_1) Then
      ' Slow the speed by 10%
      If (motor_speed > 0) Then       ' don't let speed go negative
          motor_speed = motor_speed - 10
      Endif
  Else
    If (key_value == key_2) Then
      ' Reverse the motor direction
      motion_dir = ~motion_dir
    Else
      If (key_value == key_3) Then
        ' Increase the speed by 10%
        If (motor_speed < 100) Then     ' don't let speed exceed 100
          motor_speed = motor_speed + 10
        EndIf
      Else
        If (key_value == key_pound) Then
          Gosub main_menu
          Return
        Else
          If (key_value == key_0) Then
            ' Run the motor until the stop button is pressed
            Gosub run_motor
          Endif
        EndIf
  Endif

Else
    ' Wrong key pressed
    Goto speed
Endif : Endif : Endif : Endif : Endif

Goto speed      ' continue the polling loop
Return          ' end of subroutine, not reached (see the #key If above)

'Subroutine to display the speed control menu on the LCD
speed_menu:
    Lcdout $FE, 1, "speed:", DEC motor_speed, " dir:", DEC motion_dir
    Lcdout $FE, $C0, "1:- 2:dir 3:+ 0:start #:<"
Return

'Subroutine to run the motor at the desired speed and direction until the stop button is pressed. The duty cycle of the PWM signal is the motor_speed percentage
run_motor:
    ' Display the current speed and direction
    Lcdout $FE, 1, "speed:", DEC motor_speed, " dir:", DEC motion_dir
    Lcdout $FE, $C0, "exit: stop button"

    ' Set the motor direction
    motor_dir = motion_dir

    ' Output the PWM signal
    Gosub pwm_periods  ' calculate the on and off pulse widths
    While (stop_button == 0)  ' until the stop button is pressed
        Gosub pwm_pulse  ' send out a full PWM pulse
    Wend

    ' Return to the speed menu
    Gosub speed_menu
Return

'Subroutine to wait for the stop button to be pressed and released (used during program debugging)
button_press:
    While (stop_button == 0) : Wend  ' wait for button press
    Pause 50  ' wait for switch bounce to settle
    While (stop_button == 1) : Wend  ' wait for button release
    Pause 50  ' wait for switch bounce to settle
Return

'Subroutine to allow the user to adjust the proportional and derivative gains used in position control mode
adjust_gains:
    ' Display the gain values and menu on the LCD
    Gosub gains_menu

    ' Wait for a keypad button to be pressed
    Serin key_serial, key_mode, key_value
' Take the appropriate action based on the key pressed
If (key_value == key_1) Then
    ' Increase the proportional gain by 10%
    If (kp < 100) Then
        kp = kp + 10
    Endif
Else
If (key_value == key_4) Then
    ' Decrease the proportional gain by 10%
    If (kp > 0) Then    ' don't allow negative gain
        kp = kp - 10
    Endif
Else
If (key_value == key_3) Then
    ' Increase the number of PWM cycles sent each position control loop
    pwm_cycles = pwm_cycles + 5
Else
If (key_value == key_6) Then
    ' Decrease the number of PWM cycles sent each position control loop
    If (kp > 5) Then    ' maintain positive number of pulses
        pwm_cycles = pwm_cycles - 5
    Endif
Else
If (key_value == key_pound) Then
    Gosub main_menu
    Return
Else
    Goto adjust_gains
Endif : Endif : Endif : Endif : Endif
Goto adjust_gains   ' continue the polling loop
Return              ' end of subroutine, not reached (see the #-key If above)

' Subroutine to display the position control gain, the number of PWM cycles/loop, and the adjustment menu on the LCD
subroutine gains_menu:
    Lcdout $FE, 1, "kp:", DEC kp, " PWM:", DEC pwm_cycles
    Lcdout $FE, $C0, "1:+P 4:-P  3:+C 6:-C  #:<"
Return