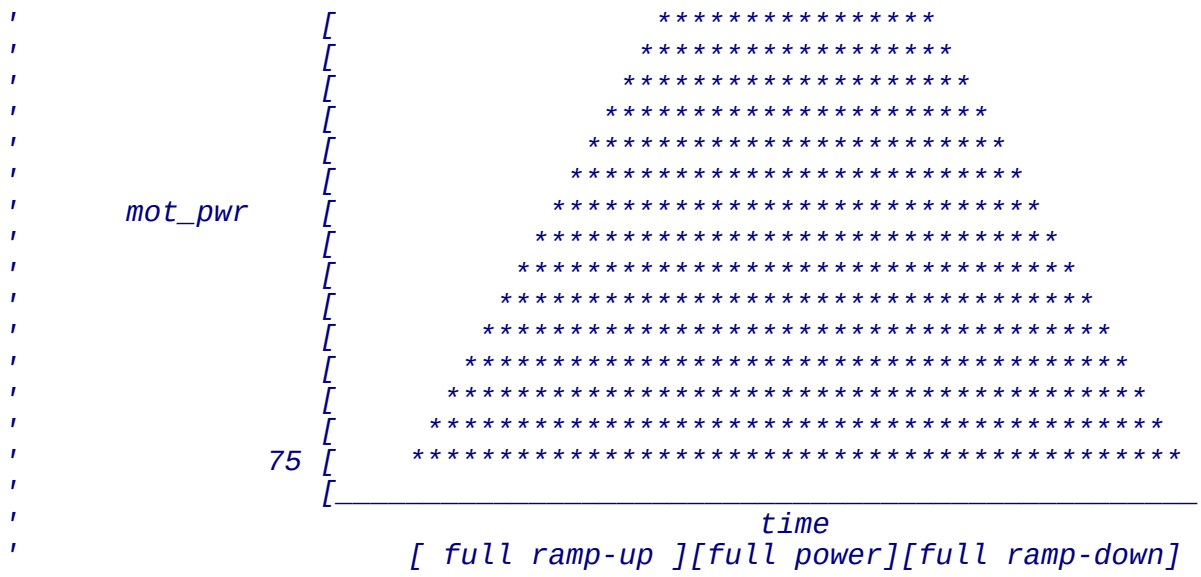
```
'---------------Title--------------

' File......4331_encoder4.pbp
' Started....1/10/10

' Microcontroller Used:  Microchip Technology 18F4331
'   Available at:
'    http://www.microchipdirect.com/ProductDetails.aspx?Category=PIC18F4331
'    or http://www.digikey.com/
' Motor Controller Used: Xavien 2 Motor Driver "XDDCMD-1
'   Available at: http://encodergeek.com/Xavien_Amplifier.html
' Motor and Encoder Used: Small Motor with Quadrature Incremental Encoder
'   Available at: http://encodergeek.com/DCMtr_SMALL.html
'
' PicBasic Pro Code: micro-Engineering Labs, Inc.
'                    melabs.com

'--------Program Desciption--------

' Program ramps up the motor power to full power and
' then slows the motor down as it approaches target position
' (Diff = 0). If the starting position is close to the target,
' the motor will ramp-up then ramp-down power without
' necessarily reaching full power.  See graphs below.
' This program is comment rich, which may help or annoy the user.

'---Review PicBasic Pro Command----

' The PicBasic Pro Compiler Manual is on line at:
' http://www.microengineeringlabs.com/resources/index.htm#Manuals
'
' HPWM Channel,Dutycycle,Frequency
'
' Outputs a PWM signal using the PICs hardware which
' is available on some PICs including the PIC18F4331.
' Channel specifies which PWM channel to use.
' Dutycycle ranges from 0 (0%) to 255 (100%).
' Frequency - lowest frequency depends upon oscillator speed,
' highest frequency at any oscillator speed is 32,767 Hz.
' Look around page 75 in the PicBasic Pro Compiler Manual
' for detailed discussion of the HPWM command.

'---------PIC Connections----------

'       18F4331 Pin           Wiring
'       ---------           ----------
'         RA3               Signal 1 from Encoder
'         RA4               Signal 2 from Encoder
'         RB5               In Circuit Serial Programming (ICSP) PGM
'                           100K Resistor to GND
'         RB6               ICSP PGC (Clock)
'         RB7               ICSP PGD (Data)
'         RC0               Brake Motor 1 on Xavien XDDCMD-1 (Pin 1)
'         RC1               PWM Motor 1 on Xavien XDDCMD-1 (Pin 2)
'         RC3               Direction Motor 1 on Xavien XDDCMD-1 (Pin 3)
```

```
'         RD4               LCD Data Bit 4
'         RD5               LCD Data Bit 5
'         RD6               LCD Data Bit 6
'         RD7               LCD Data Bit 7
'         RE0               LCD Register Select
'         RE1               LCD Enable
'         MCLR              4.7K Resistor to +5V & ICSP Vpp
'         VDD               +5V
'         VSS               GND
'         OSC1 & OSC2       4 MHz Crystal w/ 2-22 pF Cap. to GND

'----Xavien XDDCMD-1 Connections---

'   Xavien 2x5 Header Pin      Wiring    Pin Layout 2x5 Header
'   ---------------------      ------    ---------------------
'                                             2 4 6 8 10
'   Pin 1 Motor 1 Brake         RC0           o o o o o
'   Pin 2 Motor 1 PWM           RC1           o o o o o
'   Pin 3 Motor 1 Direction     RC3           1 3 5 7 9
'
' See schematic at:
' http://cornerstonerobotics.org/schematics/18f4331_hpwm_motor_encoder.pdf

'--Sample POSCNTH, POSCNTL Values and Corresponding Position Counter--

'   position = 256 * POSCNTH + POSCNTL

'  POSCNTH      POSCNTL     Position Counter
'  -------      -------     ----------------
'     0            0              0
'     0            1              1
'     1            0            255
'     0          128            128
'   128            0          32768
'     0          255            255
'   255            0          65280
'   255          255          65535

'-------------Defines--------------

    DEFINE LCD_DREG PORTD    ' Define LCD Data port as PORTD
    DEFINE LCD_DBIT 4        ' Set starting Data bit as RD4
    DEFINE LCD_BITS 4        ' Set LCD bus size as 4
    DEFINE LCD_RSREG PORTE   ' Set LCD Select Register port as PORTE
    DEFINE LCD_RSBIT 0       ' Select Select Register bit as RE0
    DEFINE LCD_EREG PORTE    ' Set LCD Enable port as PORTE
    DEFINE LCD_EBIT 1        ' Select Select Register bit as RE1
    DEFINE LCD_LINES 2       ' Set number of lines on display as 2
    DEFINE LCD_COMMANDUS 2000 ' Set command delay time in micro seconds
    DEFINE LCD_DATAUS 50     ' Set data delay time in micro seconds
    DEFINE ADC_BITS 8        ' Set number of bits in result as 8
    DEFINE ADC_CLOCK 3       ' Set clock source (rc = 3)
    DEFINE ADC_SAMPLEUS 50   ' Set sampling time in micro seconds
    DEFINE CCP2_REG PORTC    ' Set HPWM Channel 2 port to PORTC
    DEFINE CCP2_BIT 1        ' Set HPWM Channel 2 pin to RC1
```

```
'------------Variables-------------

    target          VAR WORD ' Variable target set up as a WORD
    mot_pwr         VAR WORD
    position        VAR WORD
    diff            VAR WORD
    diff_start      VAR WORD

'----------Initialization----------

    CCP1CON = %00111111         ' See page 153 of the datasheet for the
                                ' CCP1CON CCP1 Control Register
    ANSEL0 = %00000000          ' Set AN0-AN7 to digital
                                ' see datasheet page 250
                                ' for Analog Select Register
    ANSEL1 = %00000000          ' Set AN8 to digital
    TRISA = %00011111           ' Set PORTA RA0-RA4 pins as inputs,
                                ' all other pins as outputs.
    LATA  = %00000000           ' Set PORTA Data Latch register to all LOWs
    TRISB = %00000000           ' Sets all pins in PORTB as outputs
    TRISC = %00000000           ' Sets all pins in PORTC as outputs
    QEICON = %10001000          ' See page 170 of the datasheet for the
                                ' QEICON Quadrature Encoder Interface
                                ' Control Register
    PORTC.0 = 1                 ' Turn on brake
    PORTC.1 = 0                 ' PWM bit for Channel 2 of HPWM

'-------------Main Code------------

    PAUSE 500                   ' Start up LCD
    PORTC.0 = 0                 ' Turn off brake
    target = 32000              ' Set target position (0 - 65535)
                                ' With the motor and encoder used, the
                                ' difference between the target and
                                ' starting position must be at least 2.
                                ' Also choosing a target at 0 or 65535
                                ' may be a problem if the motor
                                ' overshoots the target and the position
                                ' reading jumps from 0 to the next
                                ' count of 65535 or jumps from 65535
                                ' to the next count of 0.

' Set counter starting position:

    POSCNTH = 127               ' Set counter for encoder, H bit
    POSCNTL = 0                 ' Set counter for encoder, L bit
                                ' With POSCNTH = 127 and POSCNTL = 0,
                                ' position counter will start at 32512.
                                ' (position = 256 * POSCNTH + POSCNTL)
                                ' See table above for more sample values.

' Select ramp-up mode:

    GOSUB choose_ramp_up
```

```
' Take motor to target at full speed until diff <= 180,
' then slow down motor until motor reaches target and stop:

start:

    GOSUB set_motor_direction
    GOSUB full_pwr_and_ramp_down
    GOTO start                    ' Return to loop even after the motor has
                                  ' arrived at the target in case the motor
                                  ' drifts off target.
    END

' Subroutines:

choose_ramp_up:                   ' Select ramp-up mode

    position = 256 * POSCNTH + POSCNTL
                                  ' Read position. Here the starting position
                                  ' has been set to 32512 by setting
                                  ' POSCNTH = 127 and POSCNTL = 0.
                                  ' starting position = 256*POSCNTH + POSCNTL
    IF target >= position THEN    ' Use IF..THEN to get positive value of
                                  ' starting difference.
    diff_start = target - position
    ELSE
    diff_start = position - target
    ENDIF

    IF diff_start >= 360 THEN     ' If the start difference is greater than
                                  ' or equal to the full ramp-up (180) + the
                                  ' full ramp-down (180),then:
        GOSUB full_ramp_up        ' Go to subroutine "full ramp-up mode".

    ELSE                          ' If the start difference is not greater
                                  ' than the full ramp-up (180) + the full
                                  ' ramp-down (180),then:
        GOSUB short_ramp_up       ' Go to subroutine "short ramp-up mode".
    ENDIF
    RETURN                        ' RETURN sends program to loop: label
                                  ' (the next line after "GOSUB
                                  ' choose_ramp_up") The program will then
                                  ' continue at full power until
                                  ' diff <= 180, at which point the
                                  ' mot_pwr is steadily decreased to 76
                                  ' as it approaches the target.

full_ramp_up:                     ' Starts the HPWM mot_pwr at a value of 75
                                  ' (the minimum value to start the motor
                                  ' turning) then increases our mot_pwr to
                                  ' 255 (full power).
' Graph if diff_start >= 360
'
'             255 [                    ************
'                 [                  **************
```




```
    GOSUB set_motor_direction   ' Set motor direction. If motor overshoots
                                ' target, the subroutine
                                ' "set_motor_direction" will reverse the
                                ' motor direction to  make the motor
                                ' rotate back toward the target.
    GOSUB calculate_diff        ' Go to calculate difference subroutine.
    mot_pwr = (diff_start - diff) + 75
                                ' Calculate HPWM motor power. At the
                                ' beginning of the ramp-up,
                                ' diff_start = diff so mot_pwr = 75.
                                ' As diff decreases, mot_pwr
                                ' increases to 255 (full power).

    IF mot_pwr < 255 THEN       ' If mot_pwr is less than full power
                                ' (255),then:
         GOSUB lcd              ' Go to lcd subroutine to change Dutycycle
                                ' to HPWM and display the updated variable
                                ' values.
         GOTO full_ramp_up      ' Also go back to full_ramp_up label until
                                ' mot_pwr = 255.
    ELSE                        ' If mot_pwr is greater than or equal to
                                ' full power (255),then:
         mot_pwr = 255          ' Set mot_pwr to 255.
         GOSUB lcd              ' And go to "lcd" subroutine to change
                                ' Dutycycle of HPWM to 255 and display
                                ' updated variable values.
    ENDIF
    RETURN                      ' RETURN to "choose_ramp_up" routine

short_ramp_up:                 ' Starts HPWM mot_pwr at a value of 75
                                ' (the minimum value to start the motor
                                ' turning) then increases the power for
                                ' half of the start difference
                                ' (diff_start/2).
```

```
                                ' The other half of the start difference
                                ' is used to ramp-down the motor power
                                ' back to a value of 76.

' Graph if the diff_start < 360,
'
'                     255 [
'                         [
'                         [
'                         [                *   maximum mot_pwr <= 255
'                         [               ***
'                         [              *****
'                         [             *******
'               mot_pwr   [            *********
'                         [           ***********
'                         [          *************
'                         [         ***************
'                         [        *****************
'                         [       *******************
'                         [      *********************
'                      75 [    ***********************
'                         [____________________________________
'                                        time
'                               [ ramp-up ][ ramp-down]


    GOSUB set_motor_direction   ' Set motor direction. If motor overshoots
                                ' target, the subroutine
                                ' "set_motor_direction" will reverse
                                ' motor direction to rotate the motor
                                ' back toward the target.
    GOSUB calculate_diff        ' Go to calculate difference subroutine.
    mot_pwr = (diff_start - diff) + 75
                                ' Calculate HPWM motor power. At the
                                ' beginning of the ramp-up,
                                ' diff_start = diff so mot_pwr = 75.
                                ' As diff decreases, mot_pwr increases.

    IF (diff_start)/2 <= diff THEN
                                ' This IF..THEN cuts off the ramp-up at
                                ' half of the value of the start difference
                                ' so the other half of the start difference
                                ' can be used for the ramp-down mode.
                                ' If the comparison is true, then:
        GOSUB lcd               ' Go to lcd subroutine to change Dutycycle
                                ' of HPWM and display the updated variable
                                ' values.
        GOTO short_ramp_up      ' Also go back to short_ramp_up label until
                                ' the diff < half of the value of the
                                ' start difference.
    ELSE                        ' If the comparison is not true, then
                                ' proceed to ENDIF and RETURN.
    ENDIF
    RETURN                      ' RETURN to "choose_ramp_up" routine
```

```
set_motor_direction:            ' Set direction of motors:

    position = 256 * POSCNTH + POSCNTL  'Read current position
    IF target < position THEN   ' IF..THEN to set correct motor direction

    PORTC.3 = 1                 ' Set motor direction, you may have to flip
                                ' motor directions for position to converge
                                ' on target, PORTC.3 = 0 here.
    ELSE
    PORTC.3 = 0                 ' Set motor direction, you may have to flip
                                ' motor directions for position to converge
                                ' on target, PORTC.3 = 1 here.
    ENDIF
    RETURN

full_pwr_and_ramp_down:

    GOSUB calculate_diff        ' Go to "calculate_difference" subroutine

    SELECT CASE diff            ' Program continues at full power
                                ' until diff <= 180, at which point,
                                ' the mot_pwr is steadily decreased to 76
                                ' as it reaches the target and the brake
                                ' is applied.
        CASE IS = 0             ' Position has arrived at target, diff=0
            PORTC.0 = 1         ' Apply brake
            GOSUB lcd
        CASE IS > 180           ' If diff > 180, then
            PORTC.0 = 0         ' Turn off brake - must include this line
                                ' in case the motor has passed diff = 0
                                ' and the brake has been applied.
            mot_pwr = 255       ' Set mot_pwr to 255
            GOSUB lcd           ' Go to lcd subroutine to change Dutycycle
                                ' of HPWM to 255 and display the updated
                                ' variable values.
        CASE IS <= 180          ' If diff <= 180, then
            PORTC.0 = 0         ' Turn off brake
            mot_pwr = diff + 75 ' Set mot_pwr to diff + 75
            GOSUB lcd           ' Go to lcd subroutine to change Dutycycle
                                ' to motor and display the updated variable
                                ' values.
    END SELECT
    RETURN

calculate_diff:

    position = 256 * POSCNTH + POSCNTL  'Read current position
    IF target >= position THEN  ' Use IF..THEN to get positive value of
                                ' difference.
    diff = target - position
    ELSE
    diff = position - target
    ENDIF
    RETURN
```

```
lcd:

    HPWM 2, mot_pwr, 20000      ' Send a pulse width modulated pulse out
                                ' on PWM channel 2 (RC1, see DEFINES) to
                                ' motor. Dutycycle is set by variable
                                ' mot_pwr with a range of 75 (min power
                                ' to turn motor) - 255 (full power).
                                ' Frequency of PWM signal = 20,000 Hz.
    LCDOUT $FE, $80, "Pwr=",DEC3 mot_pwr," Df=",DEC5 diff
                                ' Display power and difference on line 1.
    LCDOUT $FE, $C0, "T=",DEC5 target," Ps=", DEC5 position
                                ' Display target and position on line 2.

    RETURN
```