

'-----Title-----

' File.....proportional_1.pbp
' Started....3/15/08
' Microcontroller used: Microchip Technology 16F88
' microchip.com
' PBPro Code, micro-Engineering Labs, Inc.
' melabs.com

'-----Program Description-----

' proportional_1.pbp seeks a specific light level using
' proportional control, that is, the output adjusts
' proportionally to changes in the input. If the light
' sensor value is close to the target light value, the
' robot moves slowly; if the light sensor value differs
' greatly from the target light value, the robot
' moves more quickly.

' The program uses the PicBasic Pro command PAUSEUS
' to generate a PWM signal to control motor speed.
' This simplified the program rather than using the
' PWM and HPWM commands.

'-----New PIC Commands-----

' PAUSEUS Period
' Pauses the program for the period in microseconds

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

16F88 Pin	Wiring
RA0	Center lead CdS voltage divider
RA3	LCD Enable (E)
RA4	LCD Register Select(RS)
RB0	PWM Motor 2
RB1	Direction Motor 2
RB2	PWM Motor 1
RB3	Direction Motor 1
RB4	LCD (DB4)
RB5	LCD (DB5)
RB6	LCD (DB6)
RB7	LCD (DB7)

' See schematic for the other usual PIC connections

'-----LCD Connections-----

LCD Pin	Wiring
1	Ground(Vss)
2	+ 5v(Vdd)
3	Center of 20K Pot(Contrast)
4	RA4(Register Select,RS)
5	Ground(Read/Write,R/W)

```

'          6          RA3(Enable)
'          7          No Connection(DB0)
'          8          No Connection(DB1)
'          9          No Connection(DB2)
'         10          No Connection(DB3)
'         11          RB4(DB4)
'         12          RB5(DB5)
'         13          RB6(DB6)
'         14          RB7(DB7)

'-----Constants/Defines-----

' To free up AN0 and AN1 (Pins RA0 and RA1) for
' an analog input, the default LCD data lines, DB4-DB7,
' function must be removed from RA0 - RA3. In this
' program, we will only be using AN0, pin RA0.
' The LCD data lines are relocated to PORTB.4 - PORTB.7
' (RB4-RB7) using the LCD DEFINE statements below.
' Enable is relocated to PORTA.3. All other default
' LCD pins and functions are left unchanged.

    DEFINE LCD_DREG  PORTB      'Sets PORTB as LCD data port
    DEFINE LCD_DBIT  4          'Start data connections to bit 4
    DEFINE LCD_EREG  PORTA      'Sets PORTA as Enable port
    DEFINE LCD_EBIT  3          'Sets bit 3 as the Enable bit

    DEFINE ADC_BITS  10         'Sets the number of bits in
                                'the result to 10

'-----Variables-----

    cds_read      VAR WORD    'Word for 10-bit cds_read variable
    L0             VAR WORD    'Word for desired or target light value,L0
    diff          VAR WORD    'Word for difference between cds_read
                                'and desired light value, L0, if cds_read
                                'is greater than L0
    diff1         VAR WORD    'Word for difference between cds_read
                                'and desired light value, L0, if cds_read
                                'is less than or equal to L0
    pulse_width   VAR WORD    'Word to store pulse_width

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

    ANSEL = %00000001          'Leaves AN0 & AN1 in analog mode, but
                                'changes other analog bits to digital.
                                'See table below.

'
'   Analog Bit      Analog or Digital      PIC16F88 Pin
'   -----
'   AN0             Analog                 RA0
'   AN1             Digital                RA1
'   AN2             Digital                RA2
'   AN3             Digital                RA3
'   AN4             Digital                RA4

```

```

'      AN5          Digital          RB6
'      AN6          Digital          RB7

ADCON1 = %10000000      'Right justifies 10-bit value of variables
                        'in 16-bit WORD. Adds "0" in the
                        '6 Most Significant bits of the Word,
                        'shifting the 10-bit value of x to
                        'the right.

OSCCON = $60            'Sets the internal oscillator in the
                        '16F88 to 4 MHz

TRISB = %00000000      'Sets all pins in PORTB as outputs

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

      PAUSE 1000          'Pause to allow LCD to setup

loop:

      ADCIN 0, cds_read  'Read analog voltage on AN0, pin RA0,
                        'and convert to 10-bit digital value
                        'and store as cds_read.

      LCDOUT $FE,1,"CdS Read = ", DEC cds_read
                        'Clears LCD screen, displays
                        '"CdS Read = " and the 10-bit
                        'value of cds_read

      L0 = 512           'Establish target light value at 512

      IF cds_read > L0 THEN 'If condition is true, proceed to
                        'next command line, if the condition
                        'is false, that is, cds_read <= L0,
                        'then proceed to ELSE

      LCDOUT $FE,$C0,"diff = ", DEC diff
                        'Cursor moves to beginning of second
                        'line, displays "diff = " and the 10-bit
                        'value of diff

      GOSUB forward      'Jump to subroutine forward

      ELSE               'Program jumps here if the
                        'cds_read > L0 condition in the IF - THEN
                        'statement is false. Continues to the
                        'following command statement

      LCDOUT $FE,$C0,"diff1 = ", DEC diff1
                        'Cursor moves to beginning of second
                        'line, displays "diff1 = " and the 10-bit
                        'value of diff1

      GOSUB backup       'Jump to subroutine backup

```

```
ENDIF

GOTO loop           'Jump to label loop

END

forward:             'Subroutine forward

    diff = cds_read - L0      'Find difference between target or desired
                               'light value and CdS sensor light value
                               'when L0 < cds_read

    pulse_width = diff * 39   'Calculates pulse_width by multiplying
                               'diff value * 39. If maximum diff value
                               '(512) is multiplied by 39,
                               'pulse_width = 19968, giving an almost
                               '100% duty cycle into the H-bridge
                               'As the robot approaches the target light
                               'value, the duty cycle approaches 0%.

    HIGH 1 : HIGH 3      'Set the direction of Motors 2 and 1
                               'to forward

    HIGH 0 : HIGH 2      'Leading edge of pulse into PWM input
                               'pins of SN754410 H-bridge for
                               'Motors 2 and 1.

    PAUSEUS pulse_width   'Length of pulse_width in microseconds
                               'This creates the proportional control

    LOW 0 : LOW 2        'Falling edge of pulse

    PAUSEUS 20000-pulse_width 'LOW for 20 ms period - pulse_width

RETURN

backup:              'Subroutine backup

    diff1 = L0 - cds_read     'Find difference between target or desired
                               'light value and CdS sensor light value
                               'when L0 > cds_read

    pulse_width = diff1 * 39  'Calculates pulse_width by multiplying
                               'diff1 value * 39. If maximum diff1 value
                               '(512) is multiplied by 39,
                               'pulse_width = 19968, giving an almost
                               '100% duty cycle into the H-bridge
                               'As the robot approaches the target light
                               'value, the duty cycle approaches 0%.

    LOW 1 : LOW 3        'Set the direction of Motors 2 and 1
                               'to reverse

    HIGH 0 : HIGH 2      'Leading edge of pulse into PWM input
                               'pins of SN754410 H-bridge for
```

```
                                'Motors 2 and 1.
PAUSEUS pulse_width           'Length of pulse_width in microseconds
                                'This creates the proportional control
LOW 0 : LOW 2                'Falling edge of pulse
PAUSEUS 20000-pulse_width     'LOW for 20 ms period - pulse_width
RETURN
```