Cornerstone Electronics Technology and Robotics II
Programming Review

- **Administration:**
  - Prayer
- **PicBasic Pro Command Review:**
  - **GOTO:**

    Format:
    GOTO *Label*
    Explanation:
    Program execution continues with the statements at *Label*.
    Example:

    ```
        GOTO LED1          'Jump to statement labeled LED1

    LED1:
        PORTB.0 = 1        'Sets pin RB0 to HIGH (+5V)
    ```
  - **HIGH**

    Format:
    HIGH *Pin*
    Explanation:
    Make the specified *Pin* high. *Pin* is automatically made an output. *Pin* may be a constant, 0-15, or a variable that contains a number 0-15 (e.g. B0) or a pin name (e.g. PORTA.0).
    Examples:

    ```
        HIGH 0             ' Make Pin0 an output and set it high
                           (5 volts)

        HIGH PORTA.0       ' Make PORTA.0, (pin 8) an output and
                           sets it high (5 volts)


        led   var   PORTB.0    ' Define LED pin

        HIGH led           ' Make LED pin an output and set it high
                           (5 volts)
    ```
  - **FOR..NEXT:**

    Format:
    FOR *Count* **=** *Start* **TO** *End* {**STEP** {**-**} *Inc*}
    {*Body*}
    NEXT {*Count*}
    Explanation:
    The **FOR**..**NEXT** loop allows programs to execute a number of statements (the *Body*) some number of times using a variable as a counter. Due to its complexity and versatility, **FOR**..**NEXT** is best described step by step:
    1) The value of *Start* is assigned to the index variable, *Count*. *Count* can be a variable of any type.

2) The *Body* is executed. The *Body* is optional and can be omitted (perhaps for a delay loop).
3) The value of *Inc* is added to (or subtracted from if "**-**" is specified) *Count*. If no **STEP** clause is defined, *Count* is incremented by one.
4) If *Count* has not passed *End* or overflowed the variable type, execution returns to Step 2.
If the loop needs to *Count* to more than 255, a word-sized variable must be used.
<u>Examples:</u>

```
FOR i = 1 TO 10      ' Count from 1 to 10
N = N+1
NEXT i               ' Go back to and do next count

FOR B2 = 200 TO 100 STEP -1   'Count from 200 to 100 by -1
PULSOUT 2,B2         'Send position signal to servo
PAUSE 20             'Pause 20 msec
NEXT B2              'Go back to and do next count
```

Referring to Pins by a Number 0 – 15:

- PicBasic Pro Compiler commands can also refer to PORT name and bit number by a simple number. See following list for an 18 pin PIC16F88 microcontroller:

PORT Name.Bit#	Number
PORTB.0	0
PORTB.1	1
PORTB.2	2
PORTB.3	3
PORTB.4	4
PORTB.5	5
PORTB.6	6
PORTB.7	7
PORTA.0	8
PORTA.1	9
PORTA.2	10
PORTA.3	11
PORTA.4	12
PORTA.5	13
PORTA.6	14
PORTA.7	15

- **IF…THEN:**

  <u>Format:</u>

IF Comparison(s) **THEN** Label

IF Comparison(s) **THEN** Statement

<u>Explanation:</u>
The **IF…THEN** statement judges the comparison to whether it is true or false. If the comparison is true (any other value than 0), it will execute the **THEN** portion of the statement. If the comparison is false (0), it will execute the statement following the **IF…THEN** command.
<u>Example:</u>

```
IF PORTB.0 = 1 THEN led2   'If the switch on PORTB.0 is
                           'pushed, PORTB.0 becomes high
                           '(+5V) and the comparison is true,
                           'so the program jumps to label
                           'led2
```

- **GOSUB:**

  <u>Format:</u>

GOSUB *Label*

<u>Explanation:</u>
Jump to the subroutine at *Label* saving its return address on the stack. Unlike **GOTO**, when a **RETURN** statement is reached, execution resumes with the statement following the last executed **GOSUB** statement. An unlimited number of subroutines may be used in a program. Subroutines may also be nested. In other words, it is possible for a subroutine to call another subroutine. Such subroutine nesting must be restricted to no more than four levels deep (12 levels for 7Cxxx and 27 levels for 18Xxxx).
<u>Example:</u>

```
    GOSUB beep          'Execute subroutine named beep

    (More PicBasic Pro program code)

beep:

    HIGH 0              'Turn on LED connected to RB0

    SOUND 1,[80,10]     'Sends tone to RB1

    RETURN              'Go back to the next line in the main
                        'routine after the GOSUB command
```

- **PULSOUT:**

  Format:

  PULSOUT *Pin***,** *Period*

  Explanation:

  Generates a pulse on *Pin* of specified *Period*. The pulse is generated by toggling the pin twice, thus the initial state of the pin determines the polarity of the pulse. *Pin* is automatically made an output. *Pin* may be a constant, 0 - 15, or a variable that contains a number 0 - 15 (e.g. B0) or a pin name (e.g. PORTA.0). The resolution of **PULSOUT** is dependent upon the oscillator frequency. If a 4MHz oscillator is used, the *Period* of the generated pulse will be in 10us increments. If a 20MHz oscillator is used, *Period* will have a 2us resolution. Defining an OSC value has no effect on **PULSOUT**. The resolution always changes with the actual oscillator speed.

  Examples:

PULSOUT PORTB.5,100	‘ Send a pulse 1 msec long (at 4MHz) to RB5
PULSOUT 2,200	' Send a pulse 2 msec long to RB2.

- **LCDOUT:**
  - Format:

  LCDOUT *Item*{,Item...}

  Display *Item*s on an intelligent Liquid Crystal Display.
  If a pound sign (#) precedes an *Item*, the ASCII representation for each digit is sent to the LCD.

  - Other:
    - A program should wait for at least half a second before sending the first command to an LCD. It can take quite a while for an LCD to start up.
    - Commands are sent to the LCD by sending a $FE followed by the command. Some useful commands are listed in the following table:

LCD Command Table

Command	Operation
$FE, 1	Clear display
$FE, 2	Return home
$FE, $0C	Cursor off
$FE, $0E	Underline cursor on
$FE, $0F	Blinking cursor on
$FE, $10	Move cursor left one position
$FE, $14	Move cursor right one position
$FE, $18	Display shift left
$FE, $1C	Display shift right
$FE, $80	Move cursor to beginning of first line
$FE, $C0	Move cursor to beginning of second line
$FE, $94	Move cursor to beginning of third line
$FE, $D4	Move cursor to beginning of fourth line

- Examples:

LCDOUT $FE,1,"Hello" ‘ Clear display and show "Hello"

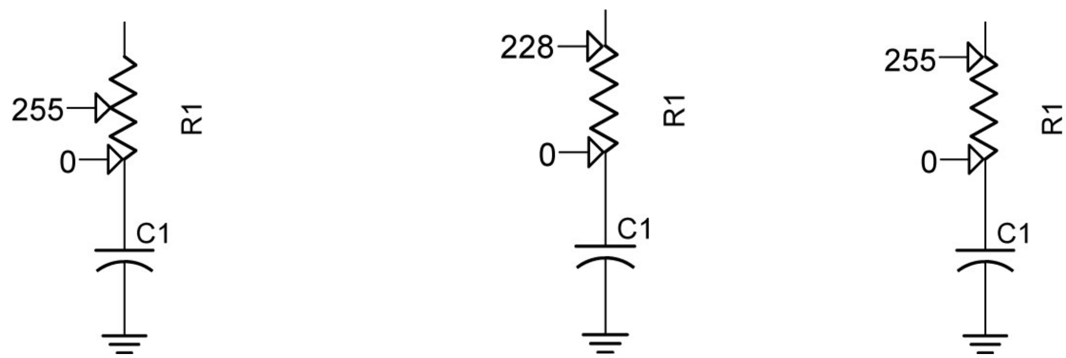
LCDOUT $FE,$C0,"World" ‘ Jump to second line and show "World"

LCDOUT B0,#B1 ‘ Display B0 and decimal ASCII value of B1

- **POT:**
  - Format

  POT Pin,Scale,Var

  Reads a resistive component (5K to 50K) such as a potentiometer on Pin. Pin may be called in the usual format, PORTB.0, or as a constant, 0 – 15, 0 is PORTB.0 and 15 is PORTA.7 (See section 4.11 Pins in the green PBP Compiler manual). To set Scale, see the **POT** command in the green microEngineering Labs PicBasic Pro Compiler manual or the explanation and table below.
  - Explanation of Scale:
    - Scale must be set empirically (with observation and experiments) such that the LCD readout value is 0 with the potentiometer set to one end and 255 when set to the other end. See the illustrations on the next page.



Scale Set Too High **Scale Set Too Low** **Scale Set Properly**

Approximate Scale Values for Resistor	
Resistor	**Approximate Scale Value**
5K	165
10K	91
25K	45
50K	38

- Example:

POT 0,178,x ‘ POT reading on Pin RBO assigned to variable, x. Scale = 178 to give a full range of values over the potentiometer (0 to 255) for the variable, x.

- **PicBasic Pro Program Review:**
  - **Blink1.pbp:** *LED flashes on/off one time per second using **PORTB.0 = 1**. See: http://cornerstonerobotics.org/code/blink1.pbp*
  - **Blink2.pbp:** *LED flashes on/off one time per half second using **HIGH 0**. See: http://cornerstonerobotics.org/code/blink2.pbp*
  - **Blink3.pbp:** *Turns one LED on and off 5 times using **FOR..NEXT loop**. See: http://cornerstonerobotics.org/code/blink3.pbp*
  - **Bounce1:** *Eight LED's scroll on then off from left to right, then right to left. See: http://cornerstonerobotics.org/code/bounce1.pbp*
  - **LCD1:** *Prints “Hello World” to 16 x 2 parallel LCD display using **LCDOUT**. See: http://cornerstonerobotics.org/code/LCD1.pbp*
  - **LCD2:** *Demonstrates several commands to move LCD cursor using **LCDOUT** command. See: http://cornerstonerobotics.org/code/LCD2.pbp*
  - **LCD3:** *Display resistance readings from a potentiometer using **POT** command. See: http://cornerstonerobotics.org/code/LCD3.pbp*
  - **Servo1:** *Servo cycles between counterclockwise and clockwise movements using **PULSOUT** command. See: http://cornerstonerobotics.org/code/servo1.pbp*
  - **Switch1:** *Turn on/off LED's with button switch using **IF..THEN** command. See: http://cornerstonerobotics.org/code/switch1.pbp*
  - **Switch2:** *Switch drives LED and servo using **IF..THEN** command. See: http://cornerstonerobotics.org/code/switch2.pbp*

Cornerstone Electronics Technology and Robotics II
Programming Review LAB 1 – Review PreQuiz

- **Purpose:** The purpose of this lab is to refresh the student's knowledge of PicBasic Pro programming.

- **Apparatus and Materials:**

    - To be determined by student.

- **Challenges:**
    - Create a folder on your desktop labeled "**prequiz**". Put all of your programs into this folder.
    - The challenges may be performed in any order.
    - Set up two +5V voltage regulator circuits first and have the instructor check both circuits before proceeding.
    - Program and wire a PIC to have an LED flash repeatedly for 3 seconds on and then for 0.7 seconds off.
    - Using several **FOR..NEXT** loops, have one red LED flash on for 1 sec. and then off for ½ sec. for 3 times. Then have a green LED flash on for 1/4 sec. and off for 1 sec. 4 times. Repeat this whole sequence only 3 times. Remember, when nesting **FOR..NEXT** loops, use different variable names for each loop.
    - On the first line of an LCD, display only your first name for 1 second, then only your last name at the beginning of the second line for 1 second. Repeat the sequence indefinitely.
    - Display your name on an LCD and have it shift continuously to the left, ½ second for each shift.
    - Have an LCD display a 25K potentiometer reading at the beginning of the second line. Adjust the SCALE value in the POT command to make full ranges active. Do this by empirically (through observation and experimentation) setting the Scale value to its lowest number while the LCDs displays 255.
    - Program a servo to go full clockwise then full counterclockwise and then mid-point.
    - Program and wire a PIC such that when a NO (normally open) momentary switch is pressed, only a red LED turns on and when released, only a green LED illuminates.
    - Program and wire a PIC such that when a NO momentary switch is pressed, a servo will go full clockwise. When the switch is released, the servo goes full counterclockwise.