Electronics and Robotics I Week 17
Programming PIC Microcontrollers in PicBasic Pro – Servos
4 Hour Class

- **Administration:**
  - Prayer
  - Turn in Quiz
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
  - Lab 1 switch1 as .pdf: http://www.cornerstonerobotics.org/code/switch1.pdf
  - Lab 2 master_slave_master1 as pdf: http://www.cornerstonerobotics.org/code/master_slave_master1.pdf
  - Lab 2 master_slave_slave1 as pdf: http://www.cornerstonerobotics.org/code/master_slave_slave1.pdf
  - Lab 3 servo1 as .pdf file: http://www.cornerstonerobotics.org/code/servo1.pdf
  - Lab 3 servo2 as .pdf file: http://www.cornerstonerobotics.org/code/servo2.pdf
  - Lab 3 servo3 as .pdf file: http://www.cornerstonerobotics.org/code/servo3.pdf
  - Lab3 servo4 as .pdf file: http://www.cornerstonerobotics.org/code/servo4.pdf
- **Programming PIC Microcontrollers, PicBasic Pro Basics Continued:**
  - Comments continued:
    - Use lots of comments. Even though it may be perfectly obvious to you what the code is doing as you write it, someone else looking at the program may not have any idea of what you were trying to achieve. While comments take up space (memory) in your Basic source file, they do not take up any additional space in the PIC MCU, so use them freely.
    - Make the comments tell you something useful about what the program is doing. For example, “Turn on the red battery low LED” might be more useful than “Set pin 0 to 1”.
    - A block of comments at the beginning of the program and before each section of code can describe what is about to happen in more detail than just the space remaining after each statement. But don’t include a comment block instead of individual line comments – use both.
    - Specifying what each pin is connected to can be helpful in remembering what hardware this particular program is designed to run on.
  - Constants:
    - Named constants may be created in a similar manner to variables. It can be more convenient to use a constant name instead of a constant number. The standard format for declaring a constant is:

Name of constant **CON** Constant value

        - An example is:

          x **CON** 3

    - Pin, Variable, and Label Names:
        - Give pins names that will help you or someone else decipher the program; judges in competitions are more able to follow the program logic if the pins and variables are coherent. For example, if the right motor is connected to PORTB.1, use the VAR keyword to assign a proper name:

          right_motor **VAR** PORTB.1

        - Also, give variables logical names. For example, if you are counting the number of times a switch1 is pressed, assign a variable name such as the one below:

          count_switch1 **VAR BYTE**

        - Finally, assign coherent names to the program labels. If the section of code is to turn the robotic car to the left, assign the label name of “left_turn”.
        - Identifiers (variable names and labels) may be up to 31 characters long, but must not start with a digit.
    - Ports continued:
        - As we set pins in PORTB for input or output (e.g. TRISB = %11111110), we can also set pins to high or low using PORT. For example:

          PORTB = %00000000 sets all PORTB pins to low (0 volts).
          PORTB = %11111111 sets all PORTB pins to high (+5 volts).

        - Perform PIC Microcontrollers Programming 3 Lab 1 – Blink 3 Ways.
- **New PicBasic Pro Commands:**
    - See the *PicBasic Pro Compiler Manual* by microEngineering Labs, Inc. for a detailed explanation of these commands.
    - **IF…THEN:**
      <u>Formats:</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.

Example:

```
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
```

Summary of Formats for **IF…THEN** and **IF…THEN…ELSE** Conditional Statements:

```
IF Comparison(s) THEN Statement
```

```
IF Comparison(s) THEN
        Statement
        Statement
        Statement
ENDIF
```

```
IF Comparison(s) THEN
        Statement
        Statement
        Statement
ELSE
        Statement
        Statement
        Statement
ENDIF
```

- Perform PIC Microcontrollers Programming 3 Lab 2 – switch1.pbp

- **GOSUB:**

  Format:

  GOSUB *Label*

  Explanation:

  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).

  Example:

```
        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**. If you want HIGH pulses, initialize the pin to LOW (For example, PORTB.0 = 0). *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:

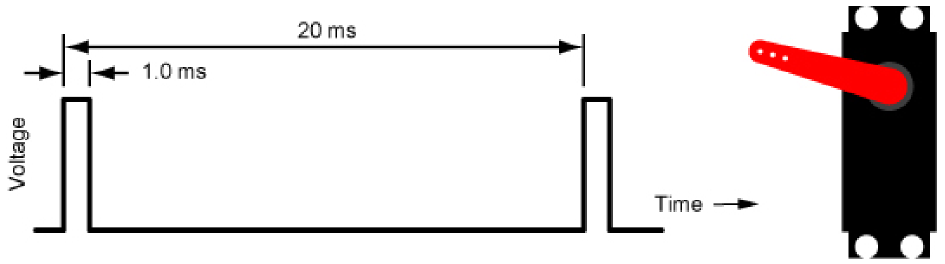
```
PULSOUT PORTB.5,100   ' Send a pulse 1 msec long (at 4MHz) to RB5

PULSOUT 2,200         'Send a pulse 2 msec long to RB2.
```

- **Power Supplies for Digital Circuits:**
  - Digital circuits require power supplies that provide a stable source of power to all circuit components.
  - Even though our circuits are powered from a voltage regulator, motors and servos can disrupt the proper operation of the PIC microcontroller.
  - **Make certain that the servo power supply is separate from the PIC power supply, i.e., have two +5V regulated power supplies.** Otherwise, if the servo spikes the single power line supplying power to both the servo and the PIC, the 16F88 may reset.
- **Introduction to Servos:**
  - A servo is a special motor used to provide control for a desired rotational position through the use of feedback.
  - Most servos have a position range of 180 degrees.
  - A hobby servo is typically used to provide actuation for various mechanical systems such as the steering of a RC car, the flaps on a RC plane, or the rudder of a RC boat.
- **Servo Connections:**

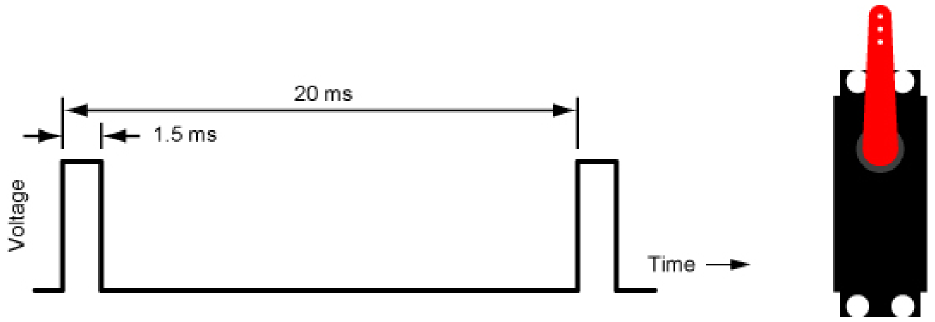
Black = Ground
Red = +5 to +6 dc volts
White/Orange = Control wire (Futaba uses a white wire)

- **Servo Pulse Signal Widths and Period:**
  - Pulse width for counterclockwise position:



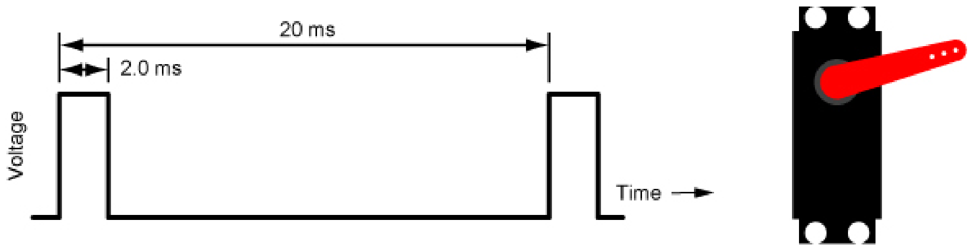
1.0 ms Pulse Width and Corresponding Servo Position

  - Pulse width for center position:



1.5 ms Pulse Width and Corresponding Servo Position

  - Pulse width for clockwise position:



2.0 ms Pulse Width and Corresponding Servo Position

  - Perform LAB 3 – servo1, servo2, servo3, and servo4.

Electronics and Robotics I Week 17
Programming PIC Microcontrollers in PicBasic Pro – Servos
LAB 1 – Blink – 3 Ways

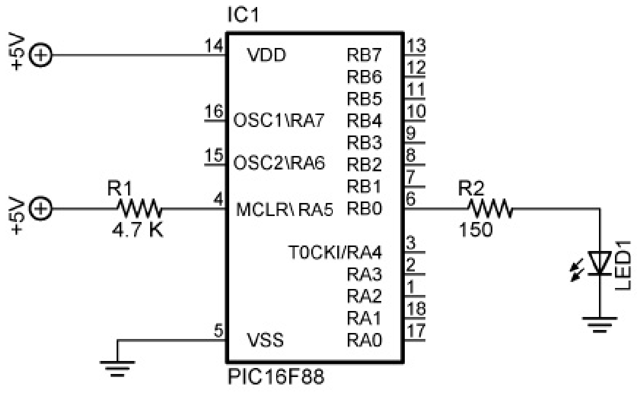
- **Purpose:** The purpose of this lab is to reinforce the three different ways to blink an LED.

- **Apparatus and Materials:**

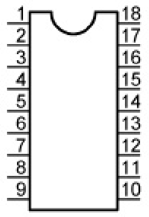
  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – PIC16F88
  - 1 – 4.7K Resistor
  - 1 – 150 Ohm Resistor
  - 1 – LED

- **Procedure:**
  - Wire the circuit blink1 below on a breadboard.
  - Turn on and off the LED using the following three sets of commands:

```
PORTB.0 = 1 & PORTB.0 = 0
HIGH 0 & LOW 0
PORTB = %00000001 & PORTB = %00000000
```



16F88 Pin Layout



blink1

- **Challenge:**
  - Wire an LED and a current limiting resistor to each pin in PORTB.
  - Program the PIC16F88 to display binary counting from 0 to 255 using a FOR…NEXT loop and a variable "x" set up in the following manner:

```
PORTB = x
```

PORTB = %00000000 may be written as PORTB = 0 since %00000000 in binary is equal to 0 in decimal.
PORTB = %11111111 may be written as PORTB = 255 since %11111111 in binary is equal to 255 in decimal.

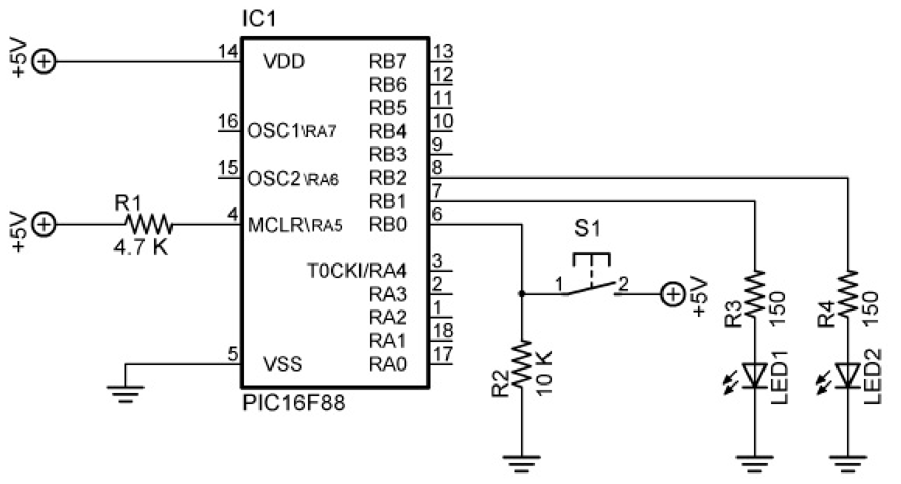
Electronics and Robotics I Week 17
Programming PIC Microcontrollers in PicBasic Pro – Servos
LAB 2 – switch1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro command **IF…THEN** and their first input device into a PIC MCU.

- **Apparatus and Materials:**

    - 1 – Breadboard or Analog/Digital Trainer
    - 1 – PIC16F88
    - 1 – 1K Resistor
    - 1 – 10K Resistor
    - 2 – 150 Ohm Resistors
    - 2 – LEDs
    - 1 – NO Momentary Switch

- **Brief Discussion of Pull-Down Resistor, R2:**
    - Pull-down resistor (R2) is used to hold the input to a zero (low) value when no other component is driving the input, i.e., the switch is open. If nothing is connected to pin RB0, the value of the input is considered to be floating. R2 will allow the pin to keep a steady state at zero until the switch is closed.
- **Procedure:**
    - Wire the circuit switch1 & switch2 below on a breadboard and program the 16F88 with **switch1.pbp** (<u>NOT 16F877A_switch1.pbp</u>).
    - Remember, the switch connected to RB0 is considered an **input device**. Input devices will allow your robotic car to interact with its environment. This is the first input device discussed to date.
    - Demonstrate the program using the circuit switch1.



switch1 and switch2

- **Challenge:**
    - Design a circuit and program such that a momentary switch connected to the PIC turns on and off a dc motor. Save the program as **switch10.pbp**.

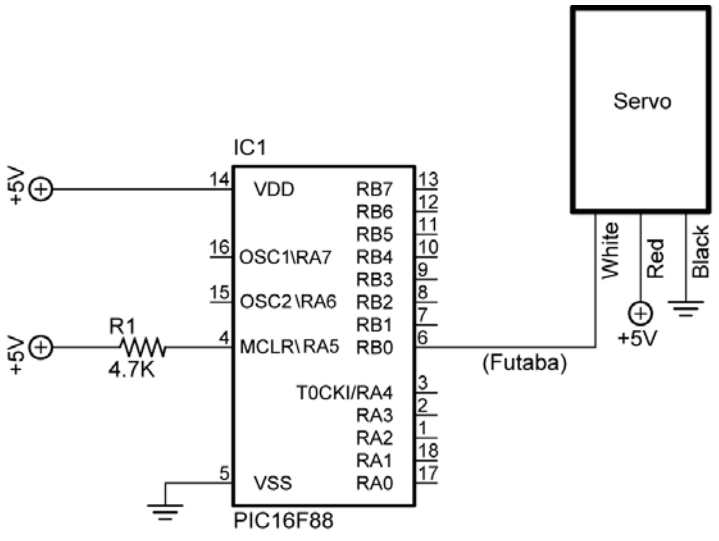
Electronics and Robotics I Week 17
Programming PIC Microcontrollers in PicBasic Pro – Servos
LAB 3 - servo1.pbp, servo2.pbp, servo3.pbp, and servo4.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with:
  - PicBasic Pro commands **GOSUB** and **PULSOUT**.
  - The basic operation of a hobby servo.

- **Apparatus and Materials:**

  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – Oscilloscope
  - 1 – PIC16F88
  - 1 – 1K Resistor
  - 3 – 10K Resistors
  - 3 – NO Momentary Switches
  - 1 – Futaba 3003 Servomotor

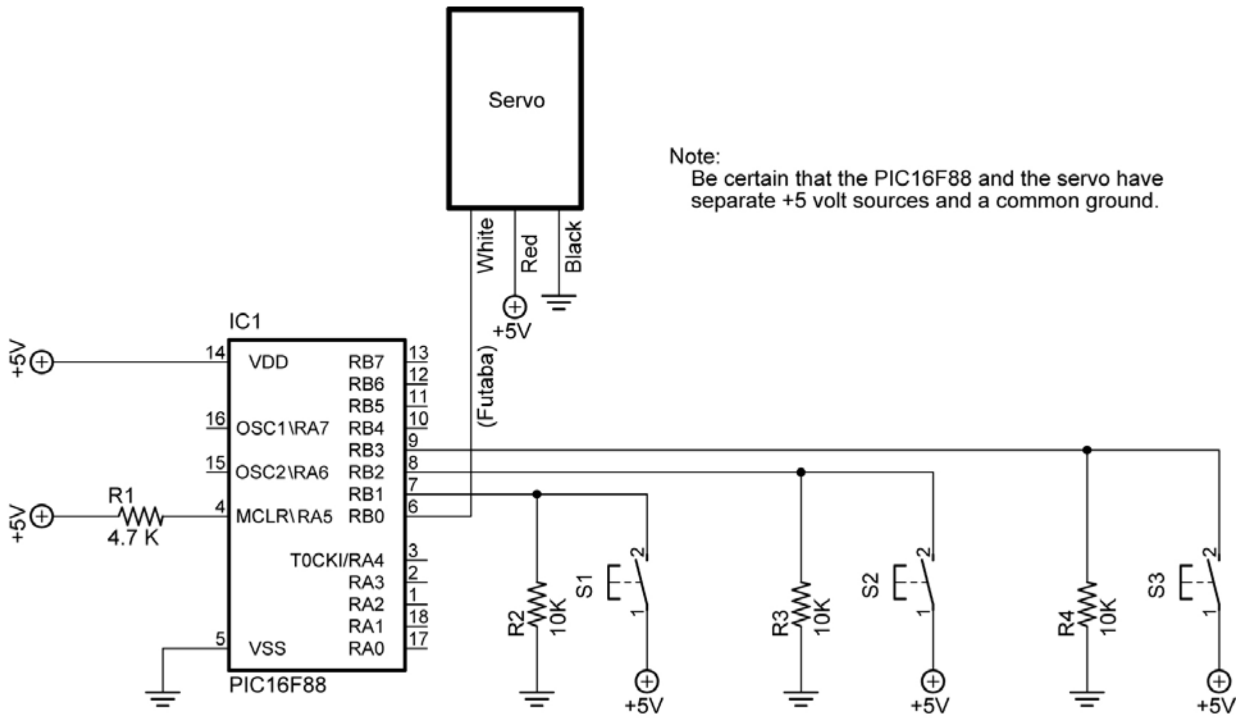
- **Procedure:**
  - Wire your breadboard for servo1 shown below. Program the 16F88 with **servo1.pbp** (NOT 16F877A_servo1.pbp).
  - **Make certain that the servo power supply is separate from the PIC power supply, i.e., have two +5V power supplies.** Otherwise, if the servo spikes the single power line supplying power to both the servo and the PIC, the 16F88 may reset.
  - Relate the program code to the observed servo motions.



servo1, servo2, and servo3

  - Open **servo2.pbp** and download to your chip. Use the same schematic as for servo1.pbp above.

- Observe the servo behavior. This servo action is suitable for panning sensor devices such as sonar sensors.
- Open **servo3.pbp** and download to your chip. Use the same schematic as for servo1.pbp above.
- Observe the waveforms on the oscilloscope. Verify that the waveforms are consistent with the program code.
- Open **servo4.pbp** and download to your chip. Wire your breadboard for servo4 shown below.
- Observe the waveforms on the oscilloscope. Verify that the waveforms are consistent with the program code.



servo4

- **Challenge**:
  - Write a program that slows the panning motion of **servo3.pbp**. Save the program as **pan1.pbp**.
  - Design and build brackets to mount a servo and SRF04 sonar module onto your robotic car. The sonar must be mounted atop the servo horn. Remember when mounting the sonar that it is not accurate at ranges closer than 3 cm.
  - If you mount the SRF04 sonar module lower than 12" above the floor, point it slightly upwards to avoid reflections from the flooring material.

Conditional Statement Summary: Conditional statements allow programs to branch to another part of the program when a conditional comparison is true.

```
Formats:                              Examples:

IF Comparison(s) THEN Statement       IF PORTB.0 = 1 THEN HIGH 2



IF Comparison(s) THEN                 IF PORTB.0 = 1THEN
   Statement                             HIGH 2
   Statement                             LOW 3
   Statement                             HIGH 6
ENDIF                                 ENDIF



IF Comparison(s) THEN                 IF PORTB.0 = 1THEN
   Statement                             HIGH 2
   Statement                             LOW 3
   Statement                             HIGH 6
ELSE                                  ELSE
   Statement                             LOW 2
   Statement                             LOW 5
   Statement                             LOW 7
ENDIF                                 ENDIF



SELECT CASE Variable                  SELECT CASE x

   CASE Value                            CASE 1
      Statement                             LCDOUT $FE, 1, “x = “, DEC x
      Statement                             x = x + 1

   CASE Another Value                    CASE 2
      Statement                             LCDOUT $FE, 1, “x = “, DEC x
      Statement                             x = x + 2

   CASE IS Comparison                    CASE IS > 10
      Statement                             LCDOUT $FE, 1, “x > 10 “
      Statement                             x = 0

   CASE ELSE                             CASE ELSE
      Statement                             LCDOUT $FE, 1, “Problem number”
      Statement                             x = 0

END SELECT                            END SELECT
```