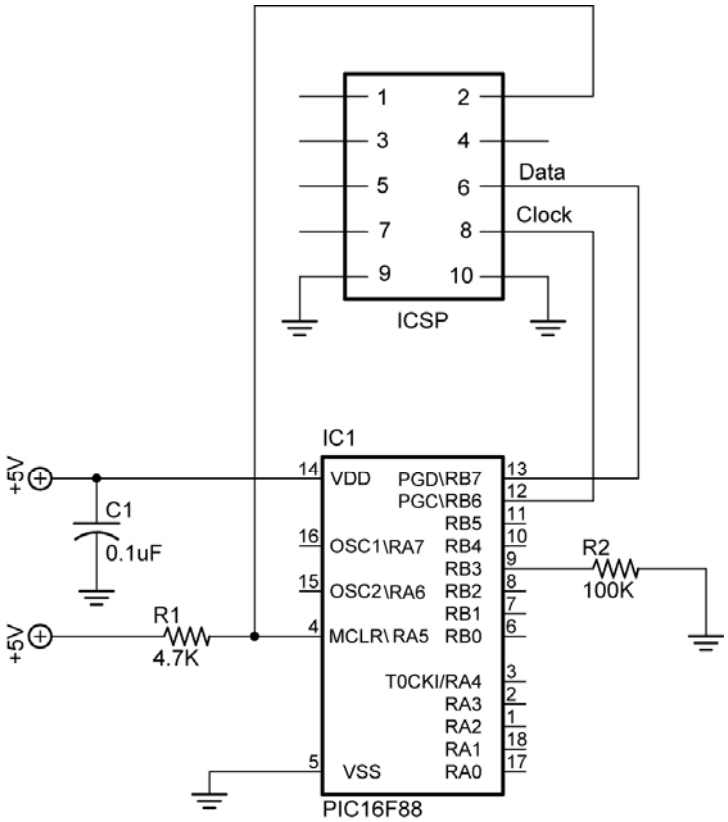
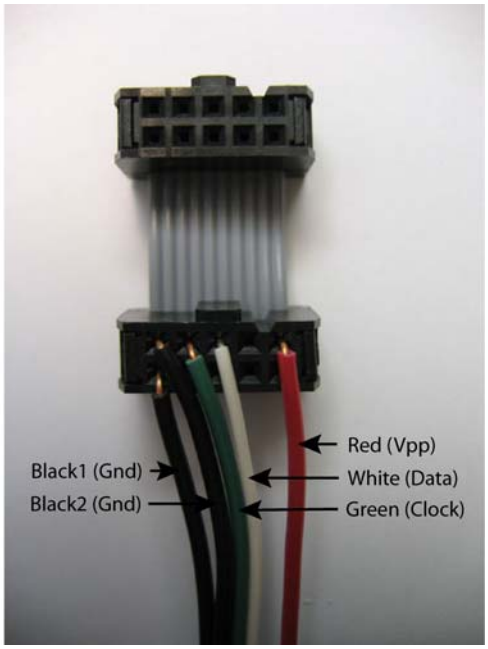
Electronics and Robotics I Week 16
Programming PIC Microcontrollers in PicBasic Pro – Lesson 2

- **Administration:**
  - Prayer
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
  - Lab 1 blink2 as .pdf file: http://www.cornerstonerobotics.org/code/blink2.pdf
  - Lab 1 blink3 as .pdf file: http://www.cornerstonerobotics.org/code/blink3.pdf
- **In-Circuit Serial Programming (ICSP):**
  - Up to this point, when programming a PIC chip, the chip was physically removed from the circuit and placed in the ZIF. By using in-circuit serial programming, the PIC chip remains in the circuit while programming occurs.
  - Schematic for PIC16F88:

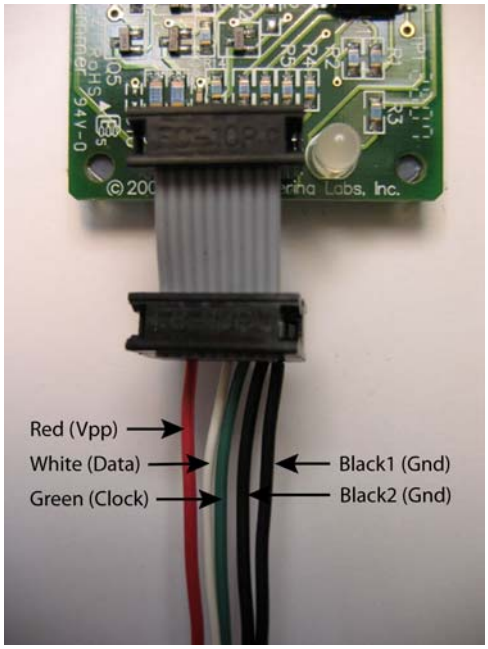


In Circuit Serial Programming (ICSP) Connections

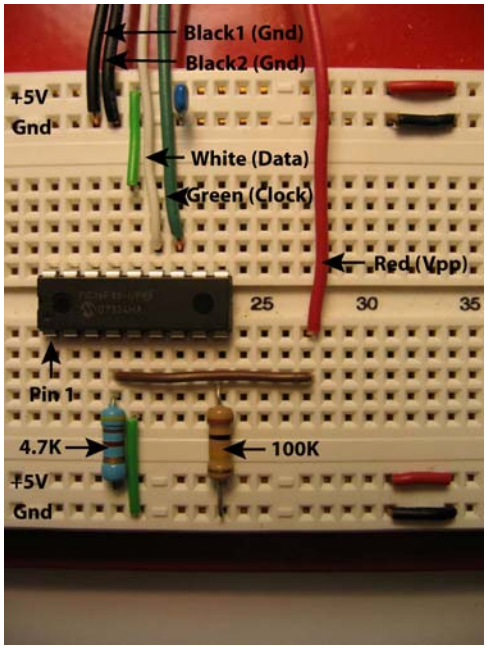
o Photos:



Underside 10-Pin Header Cable

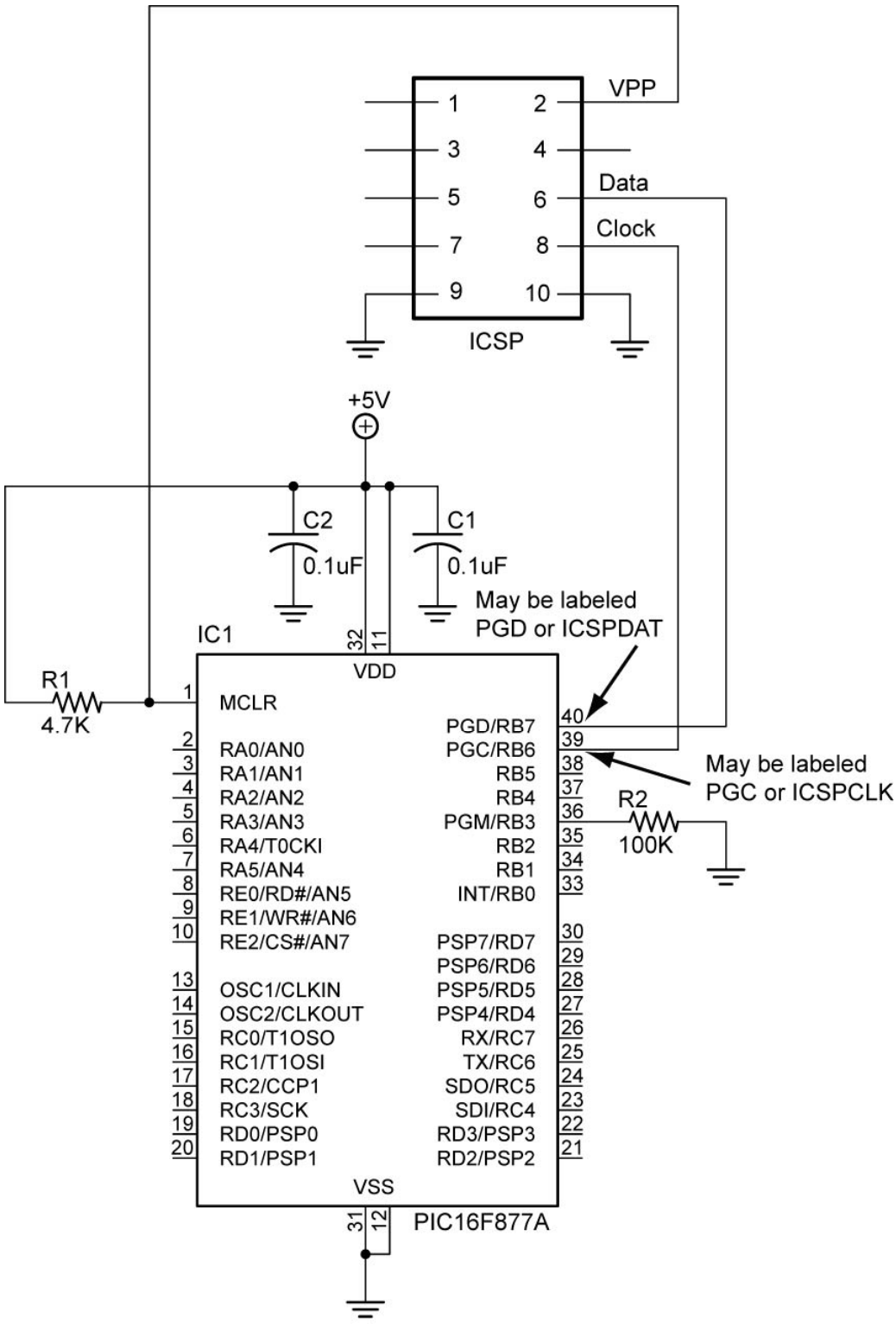


Top View 10-Pin Header Cable in melabs Programmer



Breadboard Connections

o Schematic for PIC16F877A:



In Circuit Serial Programming
(ICSP) Connections for 16F877A

- **Programming PIC Microcontrollers:**
  - **PicBasic Pro Basics:**
    - Comments:
      - A PBP (PicBasic Pro) comment starts with either the **REM** keyword, the single quote (**‘**) or the semi-colon (**;**). All the characters following on this line are ignored.
      - Examples:

REM Comments may be made after REM
PORTB.0 = 1 **‘** All of the characters after the **‘** are an in-line comment.
GOTO start **;** All of the characters after the **;** are an in-line comment.

      - If you want to ignore a line in your program that you may reinsert later, just “comment it out” by placing a single quote (**‘**) at the beginning of the line.
    - Variables:
      - Variables are where temporary data is stored in PicBasic Pro
      - The format for declaring a variable is :

Name of variable **VAR** size {.modifiers}

      - VAR lets PicBasic Pro know that this line is establishing a variable. It may or may not be capitalized.
      - The name of the variable is case insensitive, that is, VALUE and value are treated the same.
      - A variable name can have up to 32 characters.
      - Size is the amount of RAM memory space the variable is going to take. It has to be a BIT, BYTE or WORD. Examples follow.

x	VAR	**BIT**	(1 bit w/ range 0 – 1)
COUNT	VAR	**BYTE**	(8 bits w/ range 0- 255)
large	VAR	**WORD**	(16 bits w/ range 0 – 65535)

      - BIT, BYTE, and WORD variables are always positive numbers.
      - Modifiers will not be covered at this time.
      - The VAR command can be used to add another name to the same variable. In the following example, bmx and BIKE will refer to the same RAM location.

Bmx **VAR BIKE**

- A pin can be named using the VAR command. In this manner, the name can be used in any operation. For example:

```
LED   VAR   PORTB.0      ' Names PORTB.0 (PORTB, pin RB0) as LED
HIGH LED                 ' Set LED (PORTB.0) HIGH
PAUSE 500
LOW LED                  ' Set LED (PORTB.0) LOW
PAUSE 500
END
```

- **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

o **New PicBasic Pro Commands:**

- **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 5                   ' Set PORTB.5 HIGH (+5 volts)

HIGH PORTA.1             ' Set PORTA.1 HIGH (+5 volts)

led   VAR   PORTB.3      'Names PORTB.3 as led
HIGH led                 ' Set led (PORTB.3) HIGH (+5 volts)
```

- **LOW:**
  Format:
  LOW *Pin*
  Explanation:
  Make the specified *Pin* low. *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:

```
LOW 1                ' Set PORTB.1 LOW (0 volts)

LOW PORTA.2          ' Set PORTA.2 LOW (0 volts)

buzzer  VAR  PORTB.5     'Names PORTB.5 as buzzer
LOW buzzer           ' Set buzzer (PORTB.5) LOW (0 volts)
```

- **FOR..NEXT Loop:**
  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.
  Examples:

```
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
```

Nesting:

A **For..Next** loop can be placed inside another **FOR..NEXT** loop; this is called nesting. Nesting is embedding one object (one **FOR..NEXT** loop) in another object of the same type (another **FOR..NEXT** loop).

```
FOR i = 1 TO 2
        FOR j = 1 TO 10
                HIGH 0
                LOW 1
                PAUSE 500
                LOW 0
                HIGH 1
                PAUSE 500
        NEXT j
                HIGH 2
                PAUSE 500
                LOW 2
NEXT i
```

Note: The FOR…NEXT j loop must be nested entirely within the FOR…NEXT i loop.

- Perform PIC Microcontroller Prpgramming 2 Lab 1 – blink2.

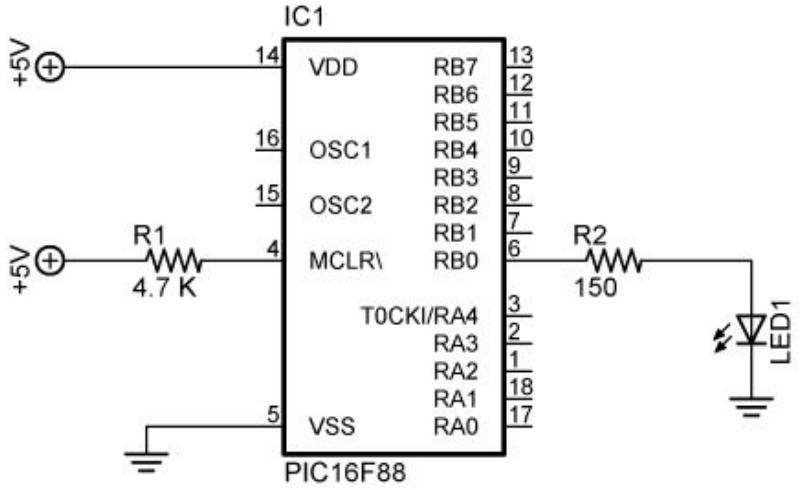
Electronics and Robotics I Week 16
PIC Microcontrollers Programming 2 LAB 1 – blink2

- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro commands: HIGH, LOW, and FOR..NEXT.

- **Apparatus and Materials:**

  - 1 – Analog/Digital Trainer
  - 1 – Breadboard with 9 V Supply
  - 1 – 150 Ohm, ½ Watt Resistors
  - 2 – 470 Ohm, ½ Watt Resistors
  - 1 – 1K, ½ Watt Resistor
  - 1 – LED
  - 2 – DC Motors
  - 2 – 2N2222A NPN Transistors
  - 1 – 78L05 Voltage Regulator
  - 1 – 0.1 uF Capacitor

- **Procedure:**
  - Open **blink2.pbp** and download to your chip. Wire your digital trainer or robotic car for blink2.
  - Change the pin location and blinking times.



blink2 and blink3

  - Open **blink3.pbp** and download to your chip.
  - Change the number of times the LED blinks.

- **Challenges:**
  - **LED Flashing:** Write a program using the **FOR..NEXT** command to make an LED connected to PORTB.7 flash on 4 times and then turn off. Have the LED turn on for 0.2 seconds and off for 0.8 seconds each time. Save the program as **flash1.pbp**.
  - **Piezo Buzzer:** Write a program that makes an LED blink every second (on 500 ms and off 500 ms) and a buzzer to sound on every fifth LED blink. The buzzer is to sound twice then the LED and buzzer are to turn off. Save the program as **buzz1.pbp**.
  - **Knight Rider:** Write a program to make the 8 LED's scroll back and forth. Utilize a 100 ohm DIP resistor package. Save the program as **knight1.pbp**.
  - **Count1:** Write a program to display binary numbers %00000000 to %11111111(0 to 255 in decimal) on the 8 LEDs used in Knight Rider. Save the program as **count1.pbp**. One code line should look like this: PORTB = x, where x is a variable.
  - **Duplicate Laps:** Write a program so that your robotic car will navigate the rectangular track twice. The instructions for the first lap must be identical to the second lap, i.e., use a **FOR…NEXT** command. After the second lap is completed, activate Knight Rider. Save the program as **indy1.pbp**.