

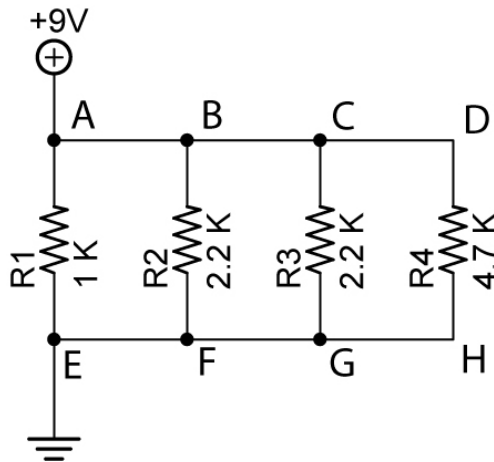
**2011-2012**  
**Cornerstone Academy**  
**Robotics II**  
**Lab Manual**

## Table of Contents

Week 1 – Review 1	3
Week 2 – Review 2	6
Week 3 – Diodes, Power Supplies, Voltage Regulators	13
Week 4 – Op Amps and PAD-234 Analog/Digital Trainer	19
Week 5 – 555 Timers	26
Week 6 – Digital Fundamentals and Binary Numbering	34
Week 7 – Logic Gates Continued	38
Week 8 – Digital Applications	45
Week 9 – Personal Computers	-
Week 10 – Microcontrollers Overview	-
Week 11– PIC Introduction, Programming 1	55
Week 12 – PIC Programming 2	58
Week 13 – PIC Programming 3, Servos	60
Week 14 – LCD 1	66
Week 15 – In Circuit Serial Programming	75
Week 15 – LCD 2, LCD Command Control Codes	76
Week 16 – LCD 3, POT Command and LCD Defines	79
Week 17 – Hacking Servos	85
Week 18 – PicBasic Pro Programming Review	87
Week 19 – Active HIGH, Active LOW	94
Week 20 – Motor Control, H-Bridges	100
Week 21 – Motor Control PWM	109
Week 22 – Sonar Car 1 – Servo Positioning	115
Week 23 – Switch Sensors	119
Week 24 – Resistive Sensors	122
Week 25 – Ultra-sonic Range Finder	128
Week 26 – Sonar Car 2 – Arrays and SRF04 Range Finder	132
Week 27 – Sonar Car 3 – Obstacle Avoidance	134
Week 28 – Sonar Car 4 – Collision Detection	136
Week 29 – DS1620 Digital Thermometer	138
Week 30 – Encoders Using PIC18F4331	147
Week 31 – RS-232 Serial Communications-Hardware	149
Week 32 – RS-232 Serial Communications- Software	151
Week 33 – EAGLE Schematic Software	157
Week 34 – Visual Basic.NET Communications	162
Week 35 – Stepper Motors	168
Week 36 – Read-Modify-Write Problem with Mid-Range PICs	-
Week 37 – Making Small Robot Wheels	-

## Cornerstone Electronics Technology and Robotics II Week 1 Electronics Review 1 Lab 1 – Voltage Drop in a Parallel Circuit

- **Purpose:** The purpose of this lab is to experimentally verify that the voltage drops across parallel resistors are equal.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 V Power Supply
  - 1 – Digital Multimeter
  - 1 – 1 K Ohm Resistor
  - 2 – 2.2 K Ohm Resistors
  - 1 – 4.7 K Ohm Resistor
- **Procedure:**
  - Wire the following circuit
  - Measure and record  $V_{AE}$ ,  $V_{BF}$ ,  $V_{CG}$ , and  $V_{DH}$ .



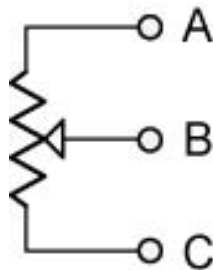
- **Results:**

Points	Voltage Drop
A - E	
B - F	
C - G	
D - H	

- **Conclusions:**
  - How do the voltage drops  $V_{AE}$ ,  $V_{BF}$ ,  $V_{CG}$ , and  $V_{DH}$  mathematically relate to each other?

## Cornerstone Electronics Technology and Robotics II Week 1 Electronics Review 1 Lab 2 – Potentiometers

- **Purpose:** The purpose of this lab is have the student measure tripot values and to help the student understand the function of a potentiometer as a variable resistor.
- **Apparatus and Materials:**
  - 1 – Digital Multimeter
  - 1 – 5 K Ohm Potentiometer
- **Procedure:**
  - Testing potentiometers:
    - Test for maximum resistance of the potentiometer with a DMM, and compare with value printed on the side of the potentiometer.
    - Turn the potentiometer shaft and then flip the DMM leads. How does the maximum resistance value of the potentiometer react?
    - Using the DMM, measure and record the resistance  $R_{AB}$ ,  $R_{BC}$ , and  $R_{AC}$  at three different positions of the potentiometer.



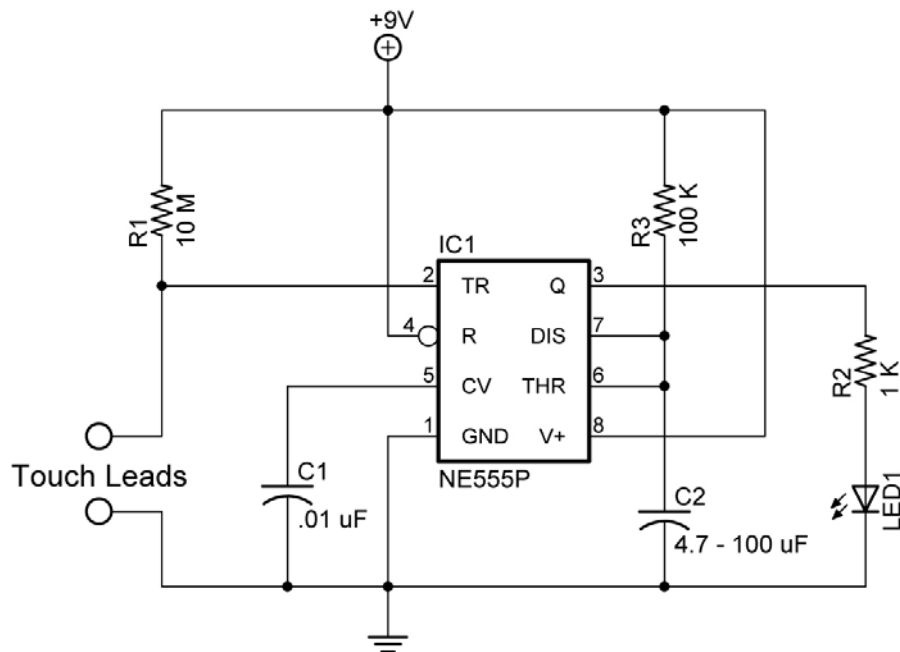
- **Results:**
  - Testing potentiometers:

Potentiometer Test			
	Position 1 (Ohms)	Position 2 (Ohms)	Position 3 (Ohms)
$R_{AB}$			
$R_{BC}$			
$R_{AC}$			

- **Conclusions:**
  - In the potentiometer test, mathematically relate  $R_{AC}$  to  $R_{AB}$  and  $R_{BC}$ .

## Cornerstone Electronics Technology and Robotics II Week 1 Electronics Review 1 Lab 3 – Touch Switch

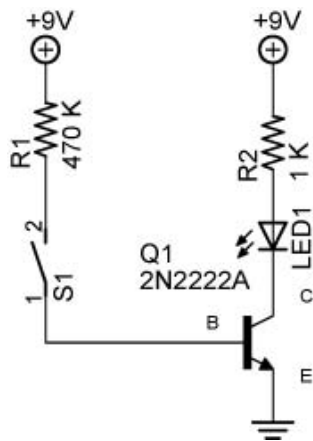
- **Purpose:** The purpose of this lab is to reacquaint the student with wiring a circuit on a breadboard.
- **Apparatus and Materials:**
  - 1 – 555 Timer
  - 1 – 10M  $\Omega$  Resistor
  - 1 – 100K  $\Omega$  Resistor
  - 1 – 1K  $\Omega$  Resistor
  - 1 – 0.01  $\mu$ F Capacitor
  - 1 – 4.7 $\mu$ F, 10 $\mu$ F, 22 $\mu$ F, 47 $\mu$ F, and 100 $\mu$ F Capacitors
  - 1 – LED
- **Procedure:**
  - Wire the touch switch circuit on your breadboard.
  - Use a 4.7  $\mu$ F capacitor for C2 to begin, and then substitute the 10  $\mu$ F, 22  $\mu$ F, 47  $\mu$ F, and 100  $\mu$ F in its place.
  - Use the normal jumpers as your touch leads.



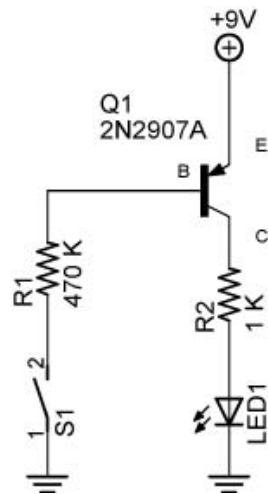
## Electronics Technology and Robotics II Week 2

### Electronics Review 2 Lab 1 – NPN and PNP Transistor Switches

- **Purpose:** The purpose of this lab is to demonstrate that a small base current controls a large collector/emitter current.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 V Power Supply
  - 2 – Digital Multimeters
  - 1 – 2N2907A PNP Transistor
  - 1 – 2N2222A NPN Transistor
  - 1 – SPST Switch
  - 1 – 470 K  $\Omega$  Resistor
  - 1 – 1 K  $\Omega$  Resistor
  - 1 – LED
- **Procedure:**
  - Build these NPN and PNP transistor test circuits.
  - Using the two multimeters, simultaneously measure the collector and base currents. Record the results.
  - Calculate the amplification factor ( $\beta$ ) for each type of resistor.



**NPN Transistor Switch**



**PNP Transistor Switch**

- **Results:**

	Current (mA)	Amplification = $I_C / I_B$
<b>NPN Circuit</b>		
Collector		
Base		
<b>PNP Circuit</b>		
Collector		
Base		

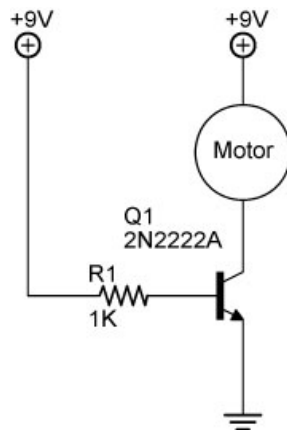
- **Conclusion:**

- Did your results verify that a very small current to the base can control a larger current that flows through the collector/emitter leads?

## Electronics Technology and Robotics II Week 2

### Electronics Review 2 Lab 2 – NPN Transistor Switch Application

- **Purpose:** The purpose of this lab is to demonstrate that a small base current controls a large collector/emitter current.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 V Power Supply
  - 1 – Gearhead Motor
  - 1 – 1K  $\Omega$  Resistor
  - 1 – 2N2222A NPN Transistor
- **Procedure:**
  - Wire the following circuit.
  - Measure the currents in the base and the collector and record.
  - Calculate the amplification.



- **Results:**

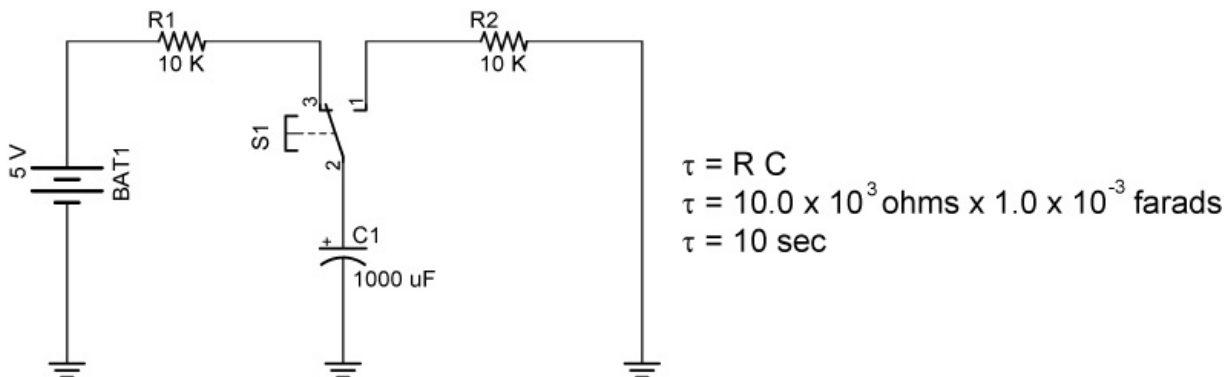
	Current (mA)	Amplification = $I_C / I_B$
<b>NPN Circuit</b>		
Collector		
Base		



## Electronics Technology and Robotics II Week 2

### Review 2 Lab 3 – Charging and Discharging a Capacitor through a Resistor

- **Purpose:** The purpose of this lab is to verify the formula for the time constant,  $\tau$ .
- **Apparatus and Materials:**
  - 1 – Breadboard with a 5 VDC Power Source
  - 2 – Digital Multimeters
  - 1 – Oscilloscope
  - 1 – Stop Watch
  - 2 – 10 K Resistors
  - 2 – 22 K Resistors
  - 1 – SPDT Switch
  - 1 – 1000  $\mu$ F Capacitor
- **Procedure:**
  - Build the following circuit and place a voltmeter across the capacitor C1 and an ammeter between S1 and C1.
  - Qualitative Results:
    - Slide the switch toward the battery to charge the capacitor through resistor R1, and then slide the switch to the other position to discharge the capacitor through resistor R2.
    - Observe the voltage across and the current through the capacitor while switching back and forth. Record your observations.



### Charging and Discharging a Capacitor through a Series Resistor

- Quantitative Results:
  - Measure and record the voltage across the power source.
  - Calculate and record 63.2% of the source voltage.
  - Measure and record the time in seconds it takes the capacitor to charge to 63.2 % of the source voltage (the definition of the time constant,  $\tau$ ).
  - Charge the capacitor until it is fully charged to the source voltage.
  - Subtract 63.2% of the source voltage from the value of the source voltage and record the result.
  - Measure and record the time in seconds it takes the capacitor to lose 63.2 % of its full charge (the definition of the time constant,  $\tau$ ).
- Replace the 2 – 10 K resistors with the 2 – 22 K resistors and repeat the Quantitative Results procedure used for the 10 K resistors.

- **Results:**

- Qualitative Results:

- When charging the capacitor, how does the voltage increase across the capacitor change with time?
    - When charging the capacitor, how does the current decrease through the capacitor change with time?
    - When discharging the capacitor, how does the voltage decrease across the capacitor change with time?
    - When discharging the capacitor, how does the current decrease through the capacitor change with time?

- Quantitative Results:

Measurement	10 K Circuit	22 K Circuit
Power Source Voltage	V	V
63.2% of Power Source Voltage	V	V
Calculated Time to Charge 63.2% of Power Source Voltage	sec	sec
Measured Time to Charge 63.2% of Power Source Voltage	sec	sec
Power Source Voltage - 63.2% of Power Source Voltage	V	V
Calculated Time to Loose 63.2% of the Full Charge	sec	sec
Measured Time to Loose 63.2% of the Full Charge	sec	sec

- **Conclusions:**

- Compare the calculated and measured times for the capacitor to charge to 63.2% of the power source (the time constant,  $\tau$ ). If the two values are not equal, explain the discrepancy.
  - Compare the calculated and measured times for the capacitor to discharge to 63.2% of the power source (the time constant,  $\tau$ ). If the two values are not equal, explain the discrepancy.

## TAP / DRILL SIZES

### American Std. and Unified Form Threads Tap Drill Size is approximately 75% Thread

THREAD NOMINAL SIZE	Pitch Series	DRILL		THREAD NOMINAL SIZE	Pitch Series	DRILL	
		SIZE	DECIMAL			SIZE	DECIMAL
0-80	NF	3/64	.047	9/16-12	NC-UNC	31/64	.484
1-64	NC	53	.060	18	NF-UNF	33/64	.516
72	NF	53	.060	5/8-11	NC-UNC	17/32	.531
2-56	NC	50	.070	18	NF-UNF	37/64	.578
64	NF	50	.070	3/4-10	NC-UNC	21/32	.656
3-48	NC	47	.079	16	NF-UNF	11/16	.688
56	NF	45	.082	7/8-9	NC-UNC	49/64	.766
4-40	NC-UNC	43	.089	14	NF-UNF	13/16	.813
48	NF	42	.094	1-8	NC-UNC	7/8	.875
5-40	NC	38	.102	14	NS	15/16	.938
44	NF	37	.104	1 1/8-7	NC-UNC	63/64	.984
6-32	NC-UNC	36	.107	12	NF-UNF	13/64	1.047
40	NF	33	.113	1 1/4-7	NC-UNC	17/64	1.109
8-32	NC-UNC	29	.136	12	NF-UNF	11 1/64	1.172
36	NF	29	.136	1 3/8-6	NC-UNC	17/32	1.219
10-24	NC-UNC	25	.150	12	NF-UNF	11 9/64	1.297
32	NF-UNF	21	.159	1 1/2-6	NC-UNC	11 1/32	1.344
12-24	NC	16	.177	12	NF-UNF	12 7/64	1.422
28	NF	14	.182	1 3/4-5	NC-UNC	19/16	1.563
1/4-20	NC-UNC	7	.201	2 1/2	NC-UNC	12 5/32	1.781
28	NF-UNF	3	.213	2 1/4-4 1/2	NC-UNC	21/32	2.031
5/16-18	NC-UNC	F	.257	2 1/2-4	NC-UNC	2 1/4	2.250
24	NF-UNF	I	.272	2 3/4-4	NC-UNC	2 1/2	2.500
3/8-16	NC-UNC	5/16	.313	3-4	NC-UNC	2 3/4	2.750
24	NF-UNF	Q	.332	3 1/4-4	NC-UNC	3	3.000
7/16-14	NC-UNC	U	.368	3 1/4-4	NC-UNC	3 1/4	3.250
20	NF-UNF	25/64	.391	3 3/4-4	NC-UNC	3 1/2	3.500
1/2-13	NC-UNC	27/64	.422	4-4	NC-UNC	3 3/4	3.750
20	NF-UNF	29/64	.453				

*TAPER PIPE	
THREAD	DRILL
1/16	D
1/8	R
1/4	7/16
3/8	37/64
1/2	45/64
3/4	59/64
1	1 15/32
1 1/4	1 1/2

*TAPER PIPE	
THREAD	DRILL
1 1/2	1 17/64
2	2 27/32
2 1/2	2 5/8
3	3 1/4
3 1/2	3 3/4
4	4 1/4

\*For tapping without reaming

STRAIGHT PIPE	
THREAD	DRILL
1/16	1/4
1/8	1 1/32
1/4	7/16
3/8	37/64
1/2	23/32
3/4	59/64
1	1 15/32
1 1/4	1 1/2
1 1/2	1 3/4
2	2 27/32
2 1/2	2 21/32

### METRIC THREADS

French and International Standard (D.I.N.)			
TAP SIZE	STD.	DRILL	
		STD.	DC.
2.5-45	French	5/64	.0781
2.6-45	D.I.N.	#45	.082
3-50	D.I.N.	#39	.0995
.60	French	3/32	.0937
.75	Optional	#43	.089
3.5-60	French & D.I.N.	#33	.113
4-70	D.I.N.	#30	.1285
-.75	French	1/8	.125
4.5-75	French & D.I.N.	#26	.147
5-75	Optional	#19	.166
.80	D.I.N.	#19	.166
.90	French	#20	.161
1.00	Optional	5/32*	.156
5.5-75	Optional	3/16	.1875
.90	French & D.I.N.	#14	.182
6-100	French & D.I.N.	#9	.196
1.25	Optional	3/16	.1875
7-100	French & D.I.N.	15/64	.234
1.25	Optional	#1	.228
8-100	French	J	.277
1.25	D.I.N.	17/64	.265
9-100	French	5/16	.3125
1.25	D.I.N.	5/16	.3125
10-100	Optional	23/64	.359
1.25	Optional	11/32	.3437
1.50	French & D.I.N.	R	.339
11-150	D.I.N.	3/8	.375
12-1.25	Optional	7/16	.4375
1.50	French	13/32	.406
1.75	D.I.N.	13/32	.406
13-150	Optional	29/64	.453
1.75	Optional	29/64	.453
2.00	Optional	7/16	.4375
14-1.25	Optional	33/64	.5156
1.75	Optional	1/2	.500
2.00	French & D.I.N.	15/32	.4687
15-1.75	Optional	17/32	.531
2.00	Optional	33/64	.5156
16-2.00	French & D.I.N.	35/64	.5468
17-2.00	Optional	19/32	.5937
18-1.50	Optional	21/32	.656
1/8-28	BSP	21/64	.3281

MAGNA

PROFESSIONAL TOOLS™

MAGNA PROFESSIONAL TOOLS  
ELIZABETHTOWN, KY 42701 U.S.A.

0204059034



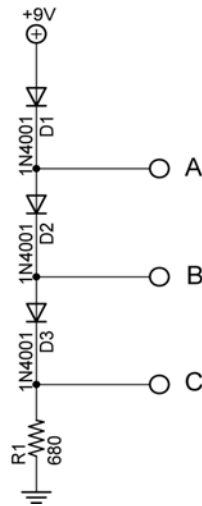
### DECIMAL AND METRIC EQUIVALENTS OF DRILL SIZES

DRILL SIZE	MM	DECIMAL INCHES	DRILL SIZE	MM	DECIMAL INCHES	DRILL SIZE	MM	DECIMAL INCHES
—	0.10	.0039	—	3.00	.1181	R	8.61	.3390
—	0.20	.0079	31	3.05	.1200	11/32	8.73	.3438
—	0.25	.0098	1/8	3.18	.1250	S	8.84	.3480
—	0.30	.0118	30	3.26	.1285	—	9.00	.3543
80	0.34	.0135	29	3.45	.1360	T	9.09	.3580
79	0.37	.0145	28	3.57	.1405	23/64	9.13	.3594
1/64	0.40	.0156	9/64	3.57	.1406	U	9.35	.3680
78	0.41	.0160	27	3.66	.1440	3/8	9.53	.3750
77	0.46	.0180	26	3.73	.1470	V	9.56	.3770
—	.050	.0197	25	3.80	.1495	W	9.80	.3860
76	0.51	.0200	24	3.86	.1520	25/64	9.92	.3906
75	0.53	.0210	23	3.91	.1540	—	10.00	.3937
74	0.57	.0225	5/32	3.97	.1562	X	10.08	.3970
—	0.60	.0236	22	3.99	.1570	Y	10.26	.4040
73	0.61	.0240	—	4.00	.1575	13/32	10.32	.4062
72	0.64	.0250	21	4.04	.1590	Z	10.49	.4130
71	0.66	.0260	20	4.09	.1610	27/64	10.72	.4219
—	0.70	.0276	19	4.22	.1660	—	11.00	.4331
70	0.71	.0280	18	4.31	.1695	7/16	11.11	.4375
69	0.74	.0292	11/64	4.37	.1719	29/64	11.51	.4531
—	0.75	.0295	17	4.39	.1730	15/32	11.91	.4688
68	0.79	.0310	16	4.50	.1770	—	12.00	.4724
1/32	0.79	.0313	15	4.57	.1800	31/64	12.30	.4844
—	0.80	.0315	14	4.62	.1820	1/2	12.70	.5000
67	0.81	.0320	13	4.70	.1850	—	13.00	.5118
66	0.84	.0330	3/16	4.76	.1875	33/64	13.10	.5156
65	0.89	.0350	12	4.80	.1890	17/32	13.49	.5312
—	0.90	.0354	11	4.85	.1910	35/64	13.89	.5469
64	0.91	.0360	10	4.91	.1935	—	14.00	.5512
63	0.94	.0370	9	4.98	.1960	9/16	14.29	.5625
62	0.97	.0380	—	5.00	.1968	37/64	14.68	.5781
61	0.99	.0390	8	5.05	.1990	—	15.00	.5906
—	1.00	.0394	7	5.11	.2010	19/32	15.08	.5938
60	1.02	.0400	13/64	5.16	.2031	39/64	15.48	.6094
59	1.04	.0410	6	5.18	.2040	5/8	15.88	.6250
58	1.07	.0420	5	5.22	.2055	—	16.00	.6299
57	1.09	.0430	4	5.31	.2090	41/64	16.27	.6406
56	1.18	.0465	3	5.41	.2130	21/32	16.67	.6562
3/64	1.19	.0469	7/32	5.56	.2188	—	17.00	.6693
55	1.32	.0520	2	5.61	.2210	43/64	17.07	.6719
54	1.40	.0550	1	5.79	.2280	11/16	17.46	.6875
53	1.51	.0595	A	5.94	.2340	45/64	17.86	.7031
1/16	1.59	.0625	15/64	5.95	.2344	—	18.00	.7087
52	1.61	.0635	—	6.00	.2362	23/32	18.26	.7188
51	1.70	.0670	B	6.05	.2380	47/64	18.65	.7344
50	1.78	.0700	C	6.15	.2420	—	19.00	.7480
49	1.85	.0730	D	6.25	.2460	3/4	19.05	.7500
48	1.93	.0760	1/4	6.35	.2500	49/64	19.45	.7656
5/64	1.98	.0781	E	6.35	.2500	25/32	19.84	.7812
47	1.99	.0785	F	6.53	.2570	—	20.00	.7874
—	2.00	.0787	G	6.63	.2610	51/64	20.24	.7969
46	2.06	.0810	17/64	6.75	.2656	13/16	20.64	.8125
45	2.08	.0820	H	6.76	.2660	—	21.00	.8268
44	2.18	.0860	I	6.91	.2720	53/64	21.03	.8281
43	2.26	.0890	—	7.00	.2756	27/32	21.43	.8438
42	2.37	.0935	J	7.04	.2770	55/64	21.84	.8594
3/32	2.38	.0938	K	7.14	.2810	—	22.00	.8661
41	2.44	.0960	9/32	7.14	.2812	7/8	22.23	.8750
40	2.50	.0980	L	7.37	.2900	57/64	22.62	.8906
39	2.53	.0995	M	7.49	.2950	—	23.00	.9055
38	2.58	.1015	19/64	7.54	.2969	29/32	23.02	.9062
37	2.64	.1040	N	7.67	.3020	59/64	23.42	.9219
36	2.71	.1065	5/16	7.94	.3125	15/16	23.81	.9375
7/64	2.78	.1094	—	8.00	.3150	—	24.00	.9449
35	2.79	.1100	O	8.03	.3160	61/64	24.21	.9531
34	2.82	.1110	P	8.20	.3230	31/32	24.61	.9688
33	2.87	.1130	21/64	8.33	.3281	—	25.00	.9842
32	2.95	.1160	Q	8.43	.3320	63/64	25.00	.9844
						1"	25.40	1.0000

For exact decimal equivalent, multiply mm times .03937. For exact mm equivalent, multiply decimal times 25.4.

**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 1 – Voltage Dropper**

- **Purpose:** The purpose of this lab is to demonstrate how a diode may be used as a voltage dropper.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 V Power Supply
  - 1 – Digital Multimeter
  - 3 – 1N4001 Diodes
  - 1 – 680  $\Omega$  Resistor
- **Procedure:**
  - Since a diode typically has a 0.7 V drop, it may be used to drop a voltage.
  - Build the following circuit:



**Voltage Dropper**

- Measure the voltages at points A, B, and C with respect to ground and record your results.
- Measure the voltage drop across each 1N4001 diode and record your results.
- **Results:**
  - $V_A$ ,  $V_B$ , and  $V_C$ :

Point	Voltage (V)
A	
B	
C	

- Voltage across 1N4001:

Voltage Drop across 1N4001	
----------------------------	--

**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 2 – Half-Wave Rectifier**

- **Purpose:** The purpose of this lab is for the student to see the input and output waveforms when ac is applied to a half-wave rectifier.

- **Apparatus and Materials:**

- 1 – Solderless Breadboard with 9 VAC Power Supply
- 1 – Oscilloscope
- 1 – 1K  $\Omega$  Resistor
- 1 – 1N4001 Diode

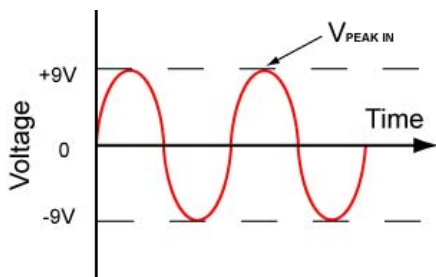
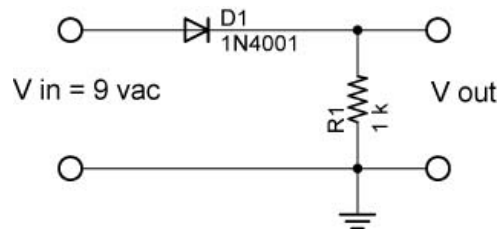
- **Procedure:**

- Build the half-wave rectifier circuit below:
  - Connect Channel 1 of the oscilloscope across the input and Channel 2 across the output load resistor, R1.
  - Use a 9 V, 500 mA transformer as your ac source. Set the oscilloscope to dc operation.
  - Observe:
    - Input sinusoid wave form on Channel 1
    - The output half-wave rectified waveform on Channel 2
    - Notice that the peak of the half-wave output is about 0.7 V lower than the peak of the input sinusoid.

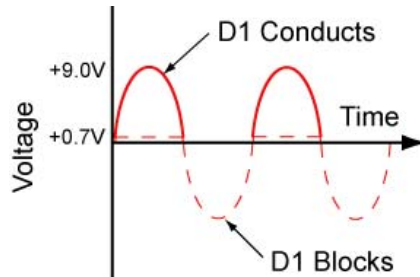
$$V_{\text{peak out}} = V_{\text{peak in}} - 0.7 \text{ V}$$

$$V_{\text{peak out}} = 9 \text{ v} - 0.7 \text{ V}$$

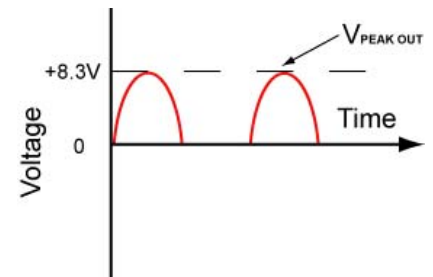
$$V_{\text{peak out}} = 8.3 \text{ V}$$



**Input Waveform**



**Effect of Diode D1**

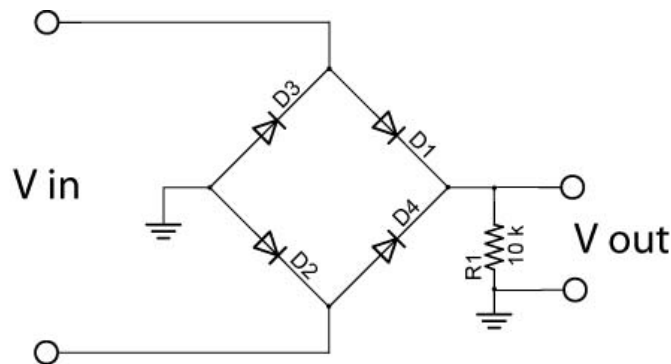


**Output Waveform**

**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 3 – Full-Wave Bridge Rectifier**

- **Purpose:** The purpose of this lab is for the student to observe the input and output waveforms when ac is applied to a full-wave bridge rectifier.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 VAC Power Supply
  - 1 – Oscilloscope
  - 1 – 10K  $\Omega$  Resistor
  - 4 – 1N4001 Diodes
- **Procedure:**
  - Build the full-wave bridge rectifier circuit below.
  - Connect the oscilloscope Channel 1 to the input and Channel 2 to the output:
  - Observe:
    - Input sinusoid waveform on Channel 1
    - The output full-wave rectified waveform on Channel 2
    - Notice that the peak of the full-wave output is about 1.4 V less than the peak of the ac input sinusoid.

$$\begin{aligned}V_{\text{peak out}} &= V_{\text{peak in}} - 1.4 \text{ V} \\V_{\text{peak out}} &= 9 \text{ V} - 1.4 \text{ V} \\V_{\text{peak out}} &= 7.6 \text{ V}\end{aligned}$$

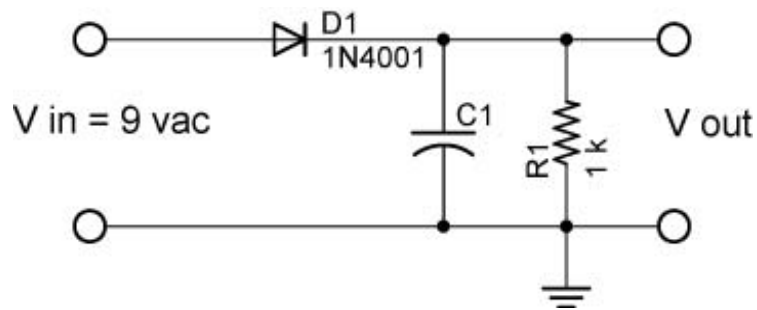


Note: All diodes are 1N4001

**Full-Wave Bridge Rectifier**

**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 4 – Capacitive Filter**

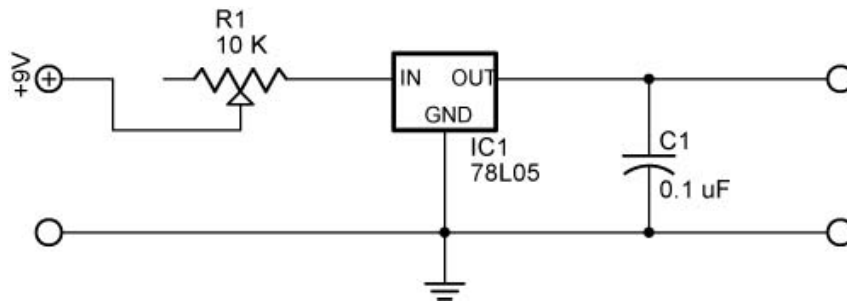
- **Purpose:** The purpose of this lab is for the student to observe the input and output waveforms when ac is applied to a full-wave bridge rectifier.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 VAC Power Supply
  - Safety Goggles
  - 1 – Oscilloscope
  - 1 – 1K  $\Omega$  Resistor
  - 1 – 1N4001 Diode
  - 1 – 10  $\mu$ F Capacitor
  - 1 – 100  $\mu$ F Capacitor
  - 1 – 1000  $\mu$ F Capacitor
- **Procedure:**
  - Build the half-wave rectifier with a capacitive filter circuit below.
  - Be certain that the capacitor is installed with the proper polarity.
  - Connect the oscilloscope Channel 1 to the input and Channel 2 to the output:
  - Use a 10 $\mu$ F, a 100  $\mu$ F, and then a 1000  $\mu$ F capacitor for filter capacitor, C1.
  - Note the changes in the output waveform across the load resistor R1 as the value of C1 increases.





**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 5 – Voltage Regulators**

- **Purpose:** The purpose of this lab is for the student to verify that the input voltage into a voltage regulator must be 2 – 3 volts above its rating.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 VDC Power Supply
  - 2 – Digital Multimeters
  - 1 – 78L05 Voltage Regulator
  - 1 – 0.1 uF Capacitor
  - 1 – 10K Potentiometer
- **Procedure:**
  - Wire the following voltage regulator circuit:
  - The capacitor acts as a surge suppressor. When a heavy load is connected to the voltage source, there may be a mild power spike to your circuit. The voltage regulator will suppress most of this power spike and the capacitor helps suppress it even more.
  - Use two DMMs to measure the input and output voltages of the voltage regulator.
  - Change input voltages and record the output voltages



**Figure 12, Voltage Regulator Circuit**

• **Results:**

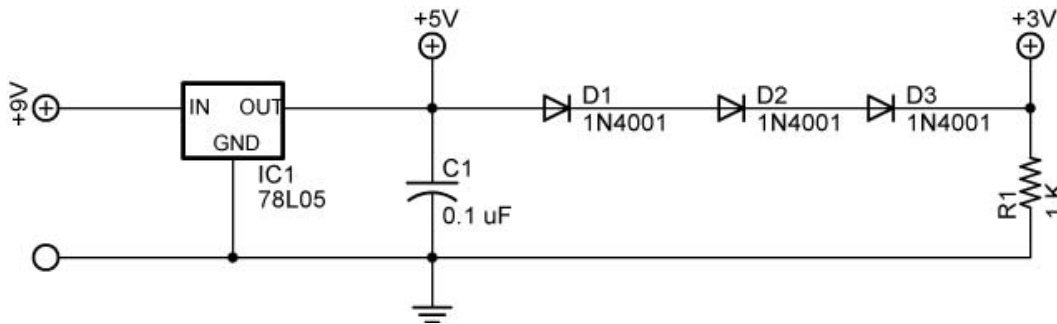
$V_{in}$	$V_{out}$
9.0 V	
8.5 V	
8.0 V	
7.5 V	
7.0 V	
6.5 V	
6.0 V	
5.5 V	
5.0 V	

• **Conclusions:**

- At what input voltage did the voltage regulator stop working properly?

**Cornerstone Electronics Technology and Robotics II**  
**Diodes, Power Supplies, Voltage Regulators**  
**Lab 6– Voltage Regulator with Two Output Voltages**

- **Purpose:** The purpose of this lab is to demonstrate that diodes may be used to drop output voltages in a voltage regulator circuit.
- **Apparatus and Materials:**
  - 1 – Solderless Breadboard with 9 VAC Power Supply
  - 1 – Digital Multimeter
  - 1 – 78L05 Voltage Regulator
  - 1 – 0.1 uF Capacitor
  - 1 – 10K Potentiometer
  - 3 – 1N4001 Diodes
- **Procedure:**
  - Wire the following voltage regulator circuit and verify the +5V and +3V output voltages:
  - Diodes D1, D2, and D3 have a voltage drop of approximately 0.7 V each.



**Voltage Regulator with Two Voltage Outputs**

- **Results:**

Nominal Value	Measured Value
+5V	
+3V	

## Cornerstone Electronics Technology and Robotics II Op Amp LAB 1 – PAD-234 Analog/Digital Trainer

- **Purpose:** The purpose of this lab is to acquaint the student with the PAD-234A Analog/Digital Trainer.
- **Apparatus and Materials:**
  - PAD-234 Analog/Digital Trainer. Available from Electronix Express, [http://www.elexp.com/tst\\_234.htm](http://www.elexp.com/tst_234.htm)
  - 1 – LED
  - 1 – 1K Resistor
- **Procedure:**
  - Read General Operating Procedures, page 1, and Maintenance, page 2.
  - Use of 5V Supply:
    - Connect banana plugs to +5V and GND posts and extend to one + and - breadboard distribution bus.
    - Hookup an LED with a series 1 K resistor on the breadboard and connect to the distribution bus.
    - Turn on the trainer and illuminate the LED.
  - Use of the Function Generator:
    - Read and follow the instructions in the Function Generator section on pages 4 and 5 of the Operator's Manual. Make the load an LED and a series 1 K resistor.
    - Set the frequency to about 2 Hz.
  - Use of + V, Variable DC Supply:
    - See Operator's Manual, page 4
    - Connect DMM to + V and adjust control knob in lower left corner of control panel.

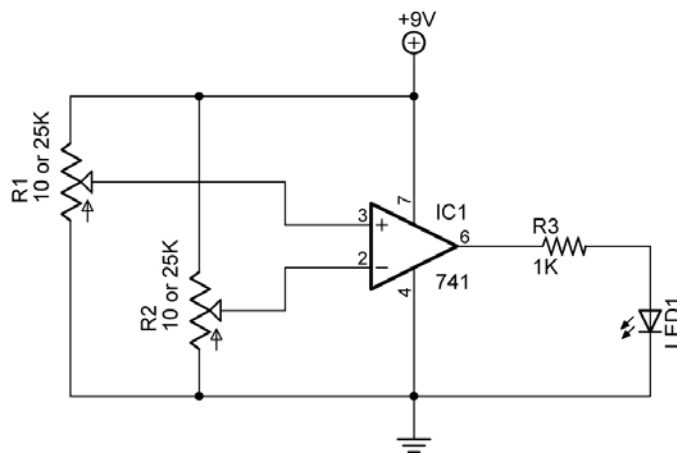


**PAD-234 Analog/Digital Trainer**

## Cornerstone Electronics Technology and Robotics II

### Op Amp LAB 2 – Op Amps as Comparators

- **Purpose:** The purpose of this lab is to review the basic operation of an op amp when used as a comparator. The material was first introduced in Electronics and Robotics I.
- **Apparatus and Materials:**
  - 1 – Breadboard with 9 V Supply
  - 2 – Digital Multi-Meters
  - 2 – 10K or 25 K Cermet Potentiometers
  - 1 – 100K Cermet Potentiometer
  - 1 – 1K Resistor
  - 1 - Photoresistor
  - 1 – 741 Op Amp
  - 1 - LED
- **Procedure:**
  - Build the basic op amp comparator circuit shown below on your breadboard.
  - Set the two DMMs to measure the voltages at Pins 2 and 3.



**Basic Op Amp Comparator Circuit**

- Pin 2 (Invert Input)(-) as the Reference Voltage:
  - Using R2, set the voltage at Pin 2 to approximately 5 V. This is the reference voltage.
  - Adjust R1 so the voltage on Pin 3 is below the voltage on Pin 2.
  - Increase the voltage at Pin 3 until it exceeds the reference voltage on Pin 2.
  - Observations: When the Invert Input (Pin 2) is set as the reference voltage and the other input voltage (Pin 3) is adjusted to exceed the reference voltage, the LED \_\_\_\_\_.
- Pin 3 (Non-Invert Input)(+) as the Reference Voltage:
  - Using R1, set the voltage at Pin 3 to approximately 4 V. This is now the reference voltage.
  - Adjust R2 so the voltage on Pin 2 is below the voltage on Pin 3.
  - Increase the voltage at Pin 2 until it exceeds the reference voltage on Pin 3.
  - Observations: When the Non-Invert Input, (Pin 3), is set as the reference voltage and the other input voltage (Pin 2) exceeds the reference voltage, the LED \_\_\_\_\_.

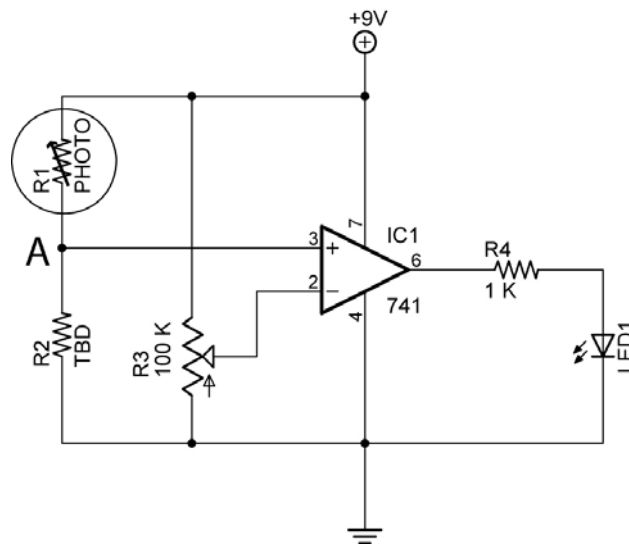
## Cornerstone Electronics Technology and Robotics II Op Amp LAB 2 – Op Amps as Comparators Continued

### ○ Practical Application:

- Build the light-activated circuit below as a practical application of comparators.
- Notice that R1 and R2 form a voltage divider. The resistance in R1 varies relative to the amount of light illuminating the photoresistor. As R1 changes, the voltage at Point A,  $V_A$ , changes, which is an input into the comparator.

$$V_A = 9V (R_2) / (R_1 + R_2)$$

- R3 is also acting as a voltage divider, varying the reference voltage into Pin 2.
- Choose a photoresistor for R1. Measure its resistance,  $R_{DARK}$ , when it is covered by your hand.
- Choose R2 with a value close to value  $R_{DARK}$ .
- Set R3 so the LED just lights on, then back it off so the LED just turns off.
- When a brighter light source illuminates the photoresistor, the LED will light.

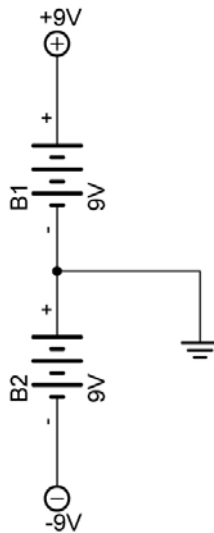


**Light Activated Switch**

## Cornerstone Electronics Technology and Robotics II

### Op Amp LAB 3 – Dual Polarity Power Supply

- **Purpose:** The purpose is to acquaint the student with a power supply that furnishes both positive and negative voltages, a dual polarity power supply.
- **Apparatus and Materials:**
  - 2 - 9 V Batteries
  - 2 – 9 V Battery Snaps
  - 1 - Digital Multi-Meter
  - 1 – Alligator Clip Test Lead
- **Procedure:**
  - Use the battery arrangement below to supply +9V and -9V. Connect batteries together using one alligator clip test lead.
  - Use the DMM to verify voltage values shown below. Attach the black common DMM lead to ground for both measurements.

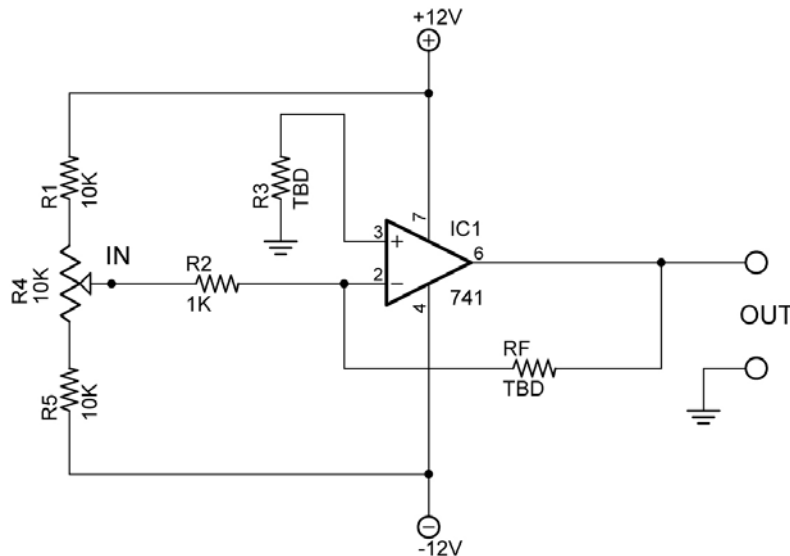


- **Notes:**
  - This power supply circuit could be used in the audio amplifier circuit in LAB 4.

## Cornerstone Electronics Technology and Robotics II

### Op Amp LAB 4 – Inverting Amplifier

- **Purpose:** The purpose of this lab is to have the student design an inverting amplifier utilizing the formulas given in the lesson to determine resistor values. The student will then build their inverting amplifier and test its performance.
- **Apparatus and Materials:**
  - 1 – Breadboard with 9 V Supply or Analog/Digital Trainer
  - 2 – Digital Multi-Meters
  - 1 – 741 Op Amp
  - 2 – 10K Resistors
  - 1 – 1K Resistor
  - 1 – 10K Cermet Potentiometer
  - 2 – Different Resistors TBD (To Be Determined)
- **Procedure:**
  - The +9V and -9V dual polarity power supply in LAB 3 may be substituted for the +12V and -12V shown in the schematic below.
  - Design an inverting operational amplifier with a gain = -10. Let R2 = 1 K ohms.
  - From the formulas in the inverting op amp gain section, determine the values of R3 and RF.
  - Using the values for R3 and RF just calculated; build the amplifier circuit shown below.
  - Measure the input and output voltages using two DMMs. Make sure that input voltage is measured between the point labeled IN (not Pin 2) and ground. The output voltage is measured across the output points. Record several readings below and verify the gain.



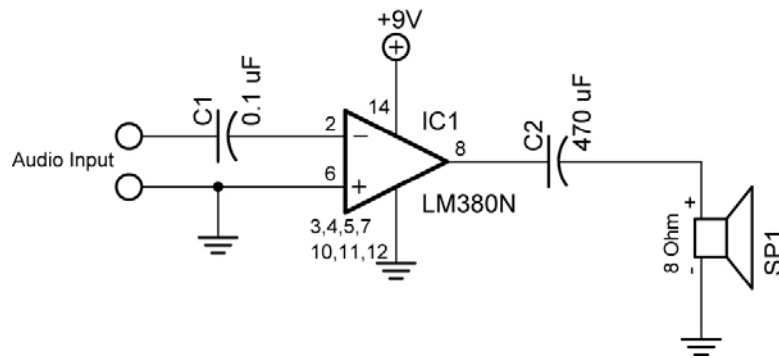
**Inverting Operational Amplifier**

$V_{IN}$	$V_{OUT}$	$Gain = V_{OUT} / V_{IN}$
_____	_____	_____
_____	_____	_____
_____	_____	_____

## Cornerstone Electronics Technology and Robotics II

### Op Amp LAB 5 – Op Amp as an Audio Amplifier

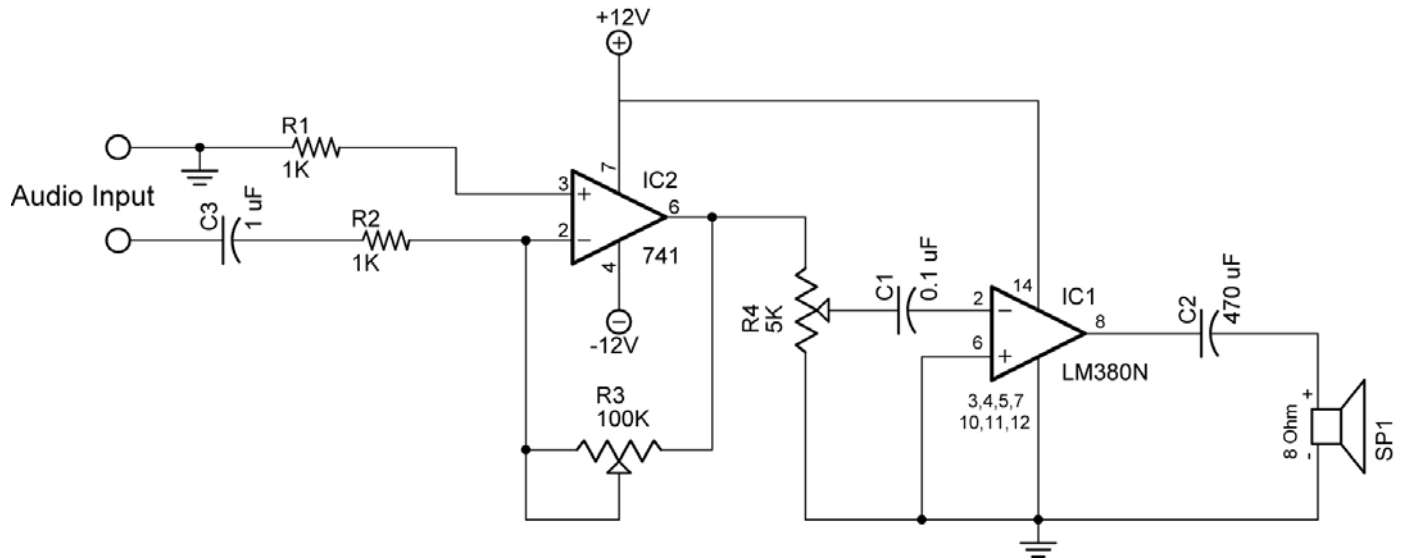
- **Purpose:** The purpose of LAB 4 is to introduce the student to an audio amplifier circuit and its performance.
- **Apparatus and Materials:**
  - 1 – Breadboard with 9 V and -9 V Supplies (See Op Amp LAB 2 – Dual Polarity Power Supply) or Analog/Digital Trainer
  - 1 – LM380 Audio Amplifier IC
  - 1 – 741 Op Amp IC
  - 1 – 5K Potentiometer
  - 1 – 470 uF Capacitor
  - 1 – 1 uF Capacitor
  - 4 – 0.1 uF Capacitors
  - 1 – 100K Cermet Potentiometer
  - 2 – 1K Resistors
  - 1 – 8 Ohm Speaker
  - 1 – Microphone or Modified Piezo Transducer
- **Procedure:**
  - Use the schematic below to build the audio amplifier.
  - If using the analog/digital trainer, +12 vdc can substitute for the +9 v supply.



**LM380 Audio Amplifier**



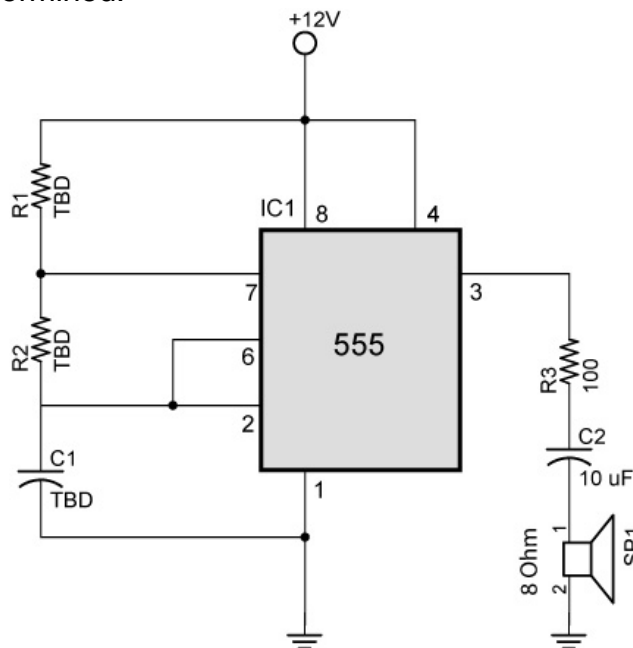
- Now add a 741 preamplifier to the circuit:



**Audio Amplifier with 741 Preamplifier**

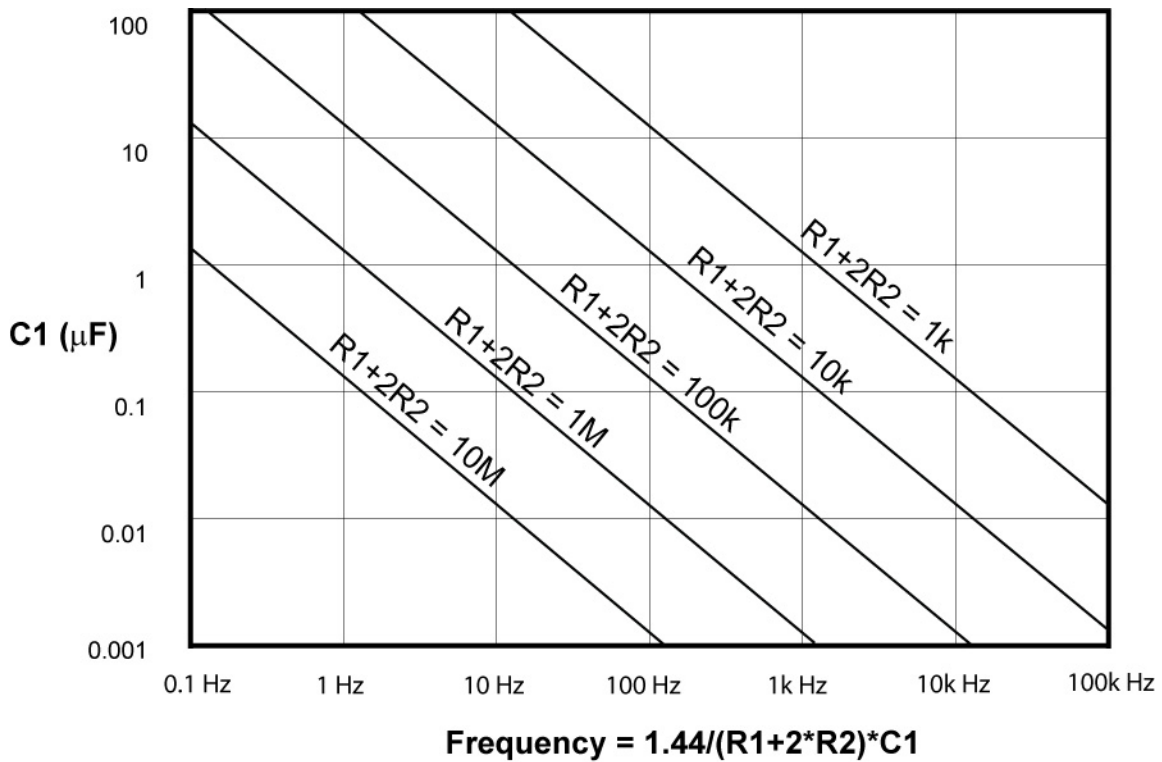
**Cornerstone Electronics Technology and Robotics II**  
**555 Timer Astable LAB 1 – Design Astable Mode Oscillator for Middle C (262 Hz)**

- **Purpose:** The purpose of this lab is for students to generate a tone at 262 Hz by designing the appropriate oscillator. 262 Hz is the frequency of middle C.
- **Apparatus and Materials:**
  - 1 - Breadboard or Analog/Digital Trainer
  - 1 – 555 Timer IC
  - 1 – 100 Ohm Resistor
  - 2 – Resistors, value to be determined
  - 1 – Capacitor, value to be determined
  - 1 – 10  $\mu$ F Capacitor
  - 1 – 8 Ohm Speaker
- **Procedure:**
  - Since we have one equation, ( $f = 1.44 / (R1 + 2R2) C1$ ), and three unknowns, ( $R1$ ,  $R2$ , &  $C1$ ), we will either have to assume the value of two unknowns or use an aid to assist us in solving the equation. We will use an aid. Determine the approximate values of  $R1+2R2$ , and  $C1$  from the 555 Timer Astable Frequency vs.  $R1$ ,  $R2$ , &  $C1$  Graph on the next page.
  - Using these approximate values, calculate the frequency from:  
 $f = 1.44 / (R1 + 2R2) C1$
  - Now adjust the values of either  $R1$  or  $R2$  or both to close in the frequency of middle C, 262 Hz. Keep in mind the resistor and capacitor values available in the shop; they are listed on the following page.
  - Wire the circuit employing the resistors and capacitor values you have determined.



**Astable Circuit to Generate Middle C Tone**

○ **555 Astable (Oscillator) Mode Chart:**

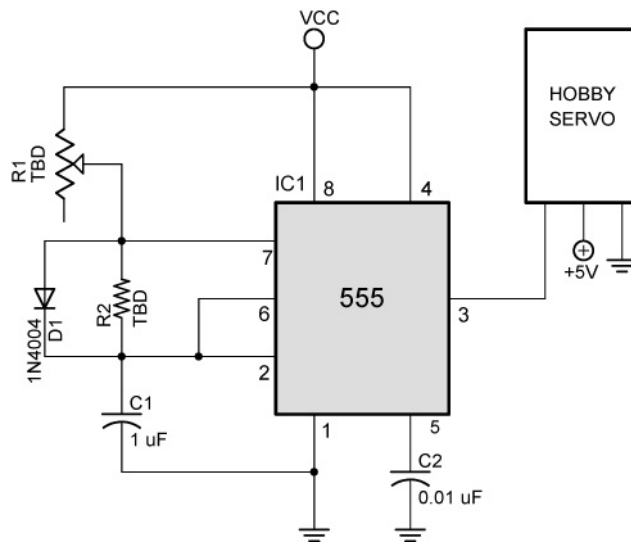


**Figure 123456 – 555 Timer Astable Frequency vs. R1, R2, & C1 Graph**

- Fixed resistor values available in the lab:
  - 10, 47, 150, 270, 330, 390, 470, 680, 1.2K, 1.5K, 2K, 2.2K, 2.7K, 3.3K, 5.1K, 5.6K, 15K, 20K, 22K, 33K, 47K, 470K, 1Meg, 10M ohm resistors
- Capacitor values available in the lab:
  - 22 pf, 0.001 uf, 0.01 uf, 0.022 uf, 0.033 uf, 0.047 uf, 0.1 uf, 0.47 uf, 1uf, 2.2 uf, 3.3 uf, 4.7uf, 10uf, 22 uf, 47 uf, 100uf, 220uf, 470uf, 1000uf, 2200uf, and 3300 uf

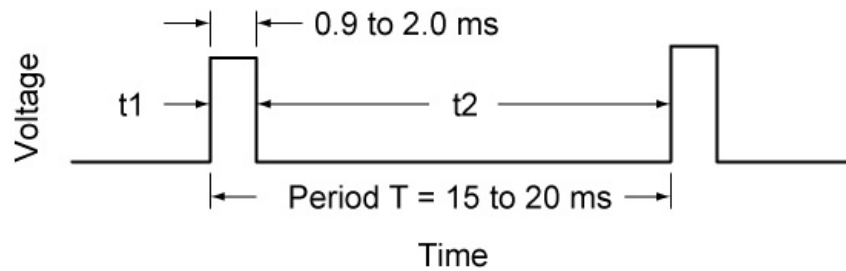
**Cornerstone Electronics Technology and Robotics II**  
**555 Timer Astable LAB 2 – Design a 555 Timer Circuit to Control a Hobby Servo Motor**

- **Purpose:** The purpose of this lab is for the student to design a 555 timer circuit that has a robotic application and also has a duty cycle less than 50%.
- **Apparatus and Materials:**
  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – 555 Timer IC
  - 1– Potentiometer, value to be determined
  - 1 – Resistors, value to be determined
  - 1 – 1 $\mu$ F Capacitor
  - 1 – Hobby Servo Motor
- **Procedure:**



- Assume C1 is 1 $\mu$ F.
- Continued on the next page.

- A hobby servo requires a series of pulses 0.9 – 2.0 ms long with each pulse cycle having a period of approximately 20 ms. See the diagram below.



### Hobby Servo Pulse Waveform Showing Pulse Width and Period T

- Solve for R1:
  - From the pulse waveform above, the pulse to a servo ranges from 0.9 to 2.0 ms. Knowing C1 (1 $\mu$ F) and assuming a pulse (t<sub>1,2.0</sub>) of 2.0 ms, solve for R<sub>1,2.0</sub>. Record your results.

$$\text{Remember, } t_1 = 0.693(R_1) \cdot C_1$$

- Choose a potentiometer R1 which has a maximum value slightly greater than R<sub>1,2.0</sub>. Record your results.
  - Now solve for R<sub>1,0.9</sub> assuming t<sub>1,0.9</sub> = 0.9 ms. Record your results.
  - The servo will only work properly when the potentiometer is within the range of resistances R<sub>1,0.9</sub> to R<sub>1,2.0</sub>. You can tell the servo is responding properly when it locks into a position and resists rotation when you try to turn it.
- Solve for R2:
  - Assume a period of 20 ms.
  - Subtract the pulse time of 2.0 ms from the period to determine the time t<sub>2</sub>.
  - Now solve for R2.

$$t_2 = 0.693(R_2) \cdot C_1$$

- Record your results.
  - Choose a resistor or a combination of resistors that approximate the calculated value of R2. Record your results.
- Test the servo using the 555 timer as the servo controller

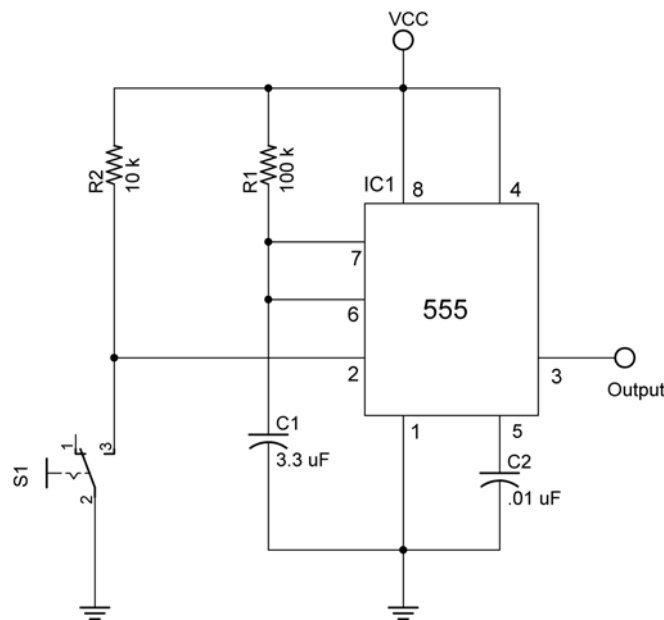
### ○ Results:

- R<sub>1,2.0</sub>, R<sub>1,0.9</sub> and R2:

	Calculated Value	Value Used
R <sub>1,2.0</sub>		
R <sub>1,0.9</sub>		
R2		

## Cornerstone Electronics Technology and Robotics II 555 Timer Monostable LAB 1 – Basic Monostable Circuit

- **Purpose:** The purpose is to acquaint the student with the basic monostable operation of a 555 timer.
- **Apparatus and Materials:**
  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – Stopwatch
  - 1 – 555 Timer IC
  - 1 – NO Momentary Switch
  - 1 – 220 Ohm Resistor
  - 1 – 10 K Resistor
  - 1 – 100 K Resistor
  - 1 – 100 K Resistor
  - 1 – 3.3  $\mu\text{F}$  Capacitor
  - 1 – 10  $\mu\text{F}$  Capacitor
  - 1 – 0.01  $\mu\text{F}$  Capacitor
  - 1 – LED
- **Procedure:**
  - Build the circuit below:
  - Let  $V_{CC} = +5\text{ V}$ . According to the NE555 spec sheet, when  $V_{CC} = +5\text{ V}$ , the typical output voltage is 3.3 V.
  - Use the 220 ohm resistor in series with an LED as the output.



### Basic Monostable Operation

- Calculate the duration of the output pulse  $t$  for  $R1$  and  $C1$  in the circuit above. The pulse duration  $t$  begins when  $S1$  is closed (Pin 2 is driven LOW). ( $t = 1.1 \times R1 \times C1$ )
- Now change  $C1$  to 10  $\mu\text{F}$  and recalculate the pulse duration,  $t$
- Measure the pulse duration  $t$  using a stop watch and record your results.

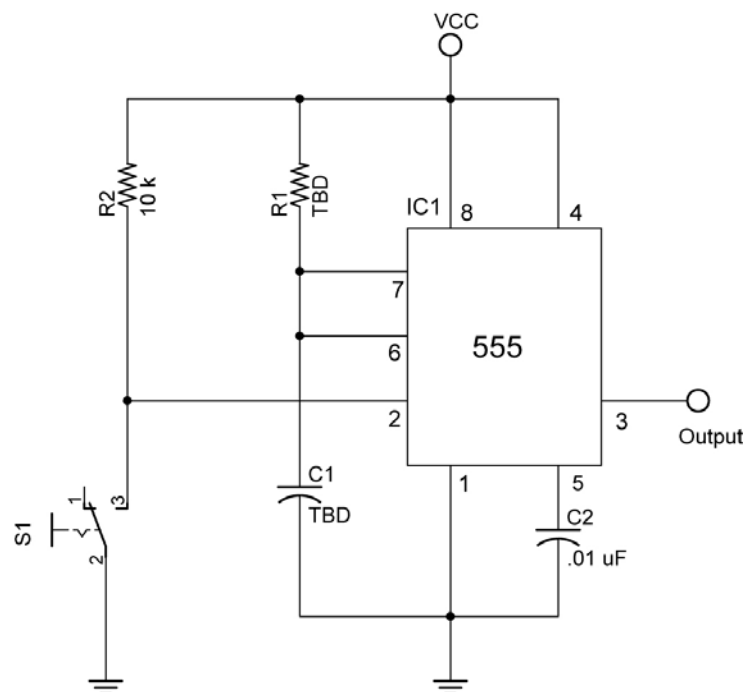
○ **Results:**

	<b>Calculated Pulse Width t (sec)</b>	<b>Meared Pulse Width t (sec)</b>
3.3 uF Capacitor		
10.0 uF Capacitor		

## Cornerstone Electronics Technology and Robotics II

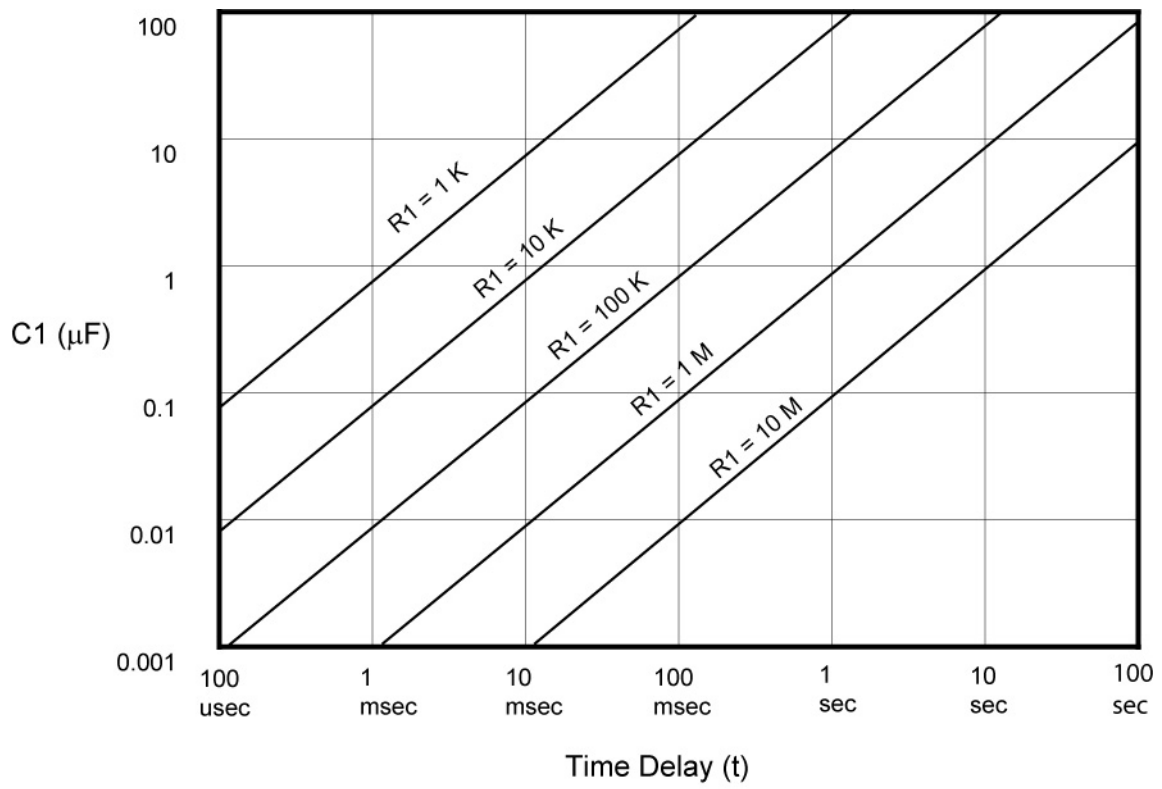
### 555 Timer Monostable LAB 2 – Designing a Monostable Circuit

- **Purpose:** The purpose is to have the student use the appropriate formulas to design a 555 timer circuit for monostable operation.
- **Apparatus and Materials:**
  - 1 - Breadboard or Analog/Digital Trainer
  - 1 – 555 Timer IC
  - 1 – NO (Normally Open) Momentary Switch
  - 1 – 220 Ohm Resistor
  - 1 – 10 K Resistor
  - 1 – Resistor, Value TBD
  - 1 – Capacitor, Value TBD
  - 1 - .01 uF Capacitor
  - 1 - LED
- **Procedure:**
  - Build the monostable circuit below after determining the values of R1 and C1.
  - Let  $V_{CC} = +5\text{ V}$ .
  - Use the 220 ohm resistor in series with an LED as the output.
  - Design a circuit with an output pulse of 5 seconds.
    - Use the 555 Monostable Time Delay Graph on the next page to determine approximate values of R1 and C1.
    - Calculate more precise values using the formula:  $t = 1.1 \times R1 \times C1$ .





- 555 Monostable Time Delay Graph:



**555 Timer Monostable Time Delay Graph**

**Electronics Technology and Robotics II  
Digital Fundamentals LAB 1 – Counting in Decimal**

- **Purpose:** When counting in decimal number system, the digit to the left is incremented as digits are exhausted. The purpose of this lab is to cement this mathematical concept in the mind of the student.
  
- **Materials:**
  - 1 - Pencil
  
- **Procedure:**
  - In Table 1, complete counting from 0 to 20.
  - In Table 2, complete counting from 90 to 110.
  - In Table 3, complete counting from 990 to 1010.

• **Results:**

Tens	Ones
$10^1$	$10^0$
0	0
0	1
2	0

**Table 1**

Hundreds	Tens	Ones
$10^2$	$10^1$	$10^0$
0	9	0
0	9	1
1	1	0

**Table 2**

Thousands	Hundreds	Tens	Ones
$10^3$	$10^2$	$10^1$	$10^0$
0	9	9	0
0	9	9	1
1	0	1	0

**Table 3**

- **Discussion:**
  - Notice that when all of the number combinations to the right have been exhausted, the digit to the left is incremented.

## Electronics Technology and Robotics II Digital Fundamentals LAB 2 – Counting in Binary

- **Purpose:** The purpose of this lab is to develop the student's skill in counting in binary.
- **Materials:**
  - 1 - Pencil
- **Procedure:**
  - In Table 1, fill in the binary equivalent for the decimal given:
  - In Table 2, fill in the next binary number if you are counting:
- **Results:**

Decimal Number	Binary Number
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

**Table 1**

Binary Number	%1001
Next Binary Number	
Binary Number	%0001101
Next Binary Number	
Binary Number	%100100111
Next Binary Number	
Binary Number	%00001111
Next Binary Number	
Binary Number	%10001001010
Next Binary Number	
Binary Number	%11111111111
Next Binary Number	

**Table 2**

**Electronics Technology and Robotics II**  
**Digital Fundamentals LAB 3 – LED Display of Binary Numbers**

- **Purpose:** The purpose of this lab is to show the student a visual representation of binary numbers and to teach several ways of demonstrating a binary state.
- **Materials:**
  - 1 – Analog/Digital Trainer
- **Procedure:**
  - Connect four consecutive HI/LOW toggle switches to four consecutive LEDs on the analog/digital trainer.
  - Remember that 0 is represented by an off or LOW state (0V) and a 1 is represented by an on or HIGH state (+5V).
  - Give an LED display for each of the following binary numbers:
    - %0000
    - %0001
    - %0101
    - %1111
    - %1000

**Electronics Technology and Robotics II**  
**Digital Fundamentals LAB 4 – Converting Binary to Decimal**

- **Purpose:** The purpose of this lab is to develop the student’s skill in converting binary numbers to decimal numbers.
- **Materials:**
  - 1 – Pencil
- **Procedure:**
  - Convert each binary number to a decimal number. Show results below:
    - %111
    - %1001
    - %001001
    - %101010
    - %010111
    - %11100011
    - %11111111
  - Use the Binary Weight Tables as guides:

<b>Binary Weight</b>	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
<b>Decimal Equivalent</b>	128	64	32	16	8	4	2	1
<b>Binary Number</b>								
<b>Decimal Value</b>								

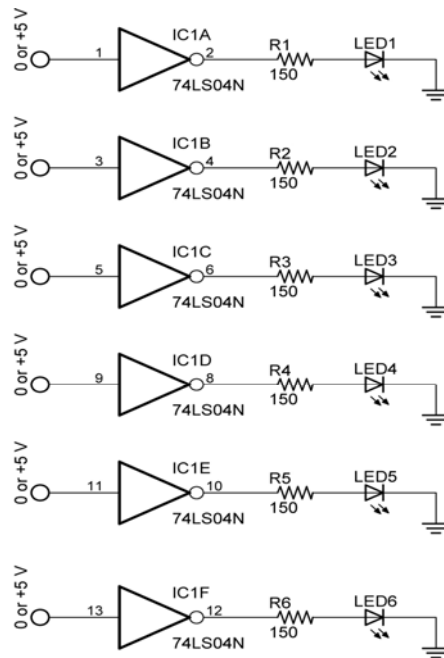
<b>Binary Weight</b>	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
<b>Decimal Equivalent</b>	128	64	32	16	8	4	2	1
<b>Binary Number</b>								
<b>Decimal Value</b>								

<b>Binary Weight</b>	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
<b>Decimal Equivalent</b>	128	64	32	16	8	4	2	1
<b>Binary Number</b>								
<b>Decimal Value</b>								

- **Results:**
  - %111 = \_\_\_\_\_
  - %1001 = \_\_\_\_\_
  - %001001 = \_\_\_\_\_
  - %101010 = \_\_\_\_\_
  - %010111 = \_\_\_\_\_
  - %11100011 = \_\_\_\_\_
  - %11111111 = \_\_\_\_\_

## Electronics Technology and Robotics II Logic Gates LAB 1 – NOT Gates (Inverters)

- **Purpose:** The purpose of this lab is to acquaint the student with a Hex-Inverter
- **Materials:**
  - 1 – Analog/Digital Trainer or Breadboard
  - 1 – 74LS04N Hex-Inverter
  - 1 – 150 Ohm DIP Resistor Package (For Breadboard Only)
  - 6 – LEDs (For Breadboard Only)
- **Procedure:**
  - Wire the circuit below. See the photos on the next page.
  - Connect the six inputs to the HI/LOW toggles on the analog/digital trainer.
  - The LEDs on the analog/digital trainer may be used for LED1-LED6; R1-R6 may be eliminated in this case.



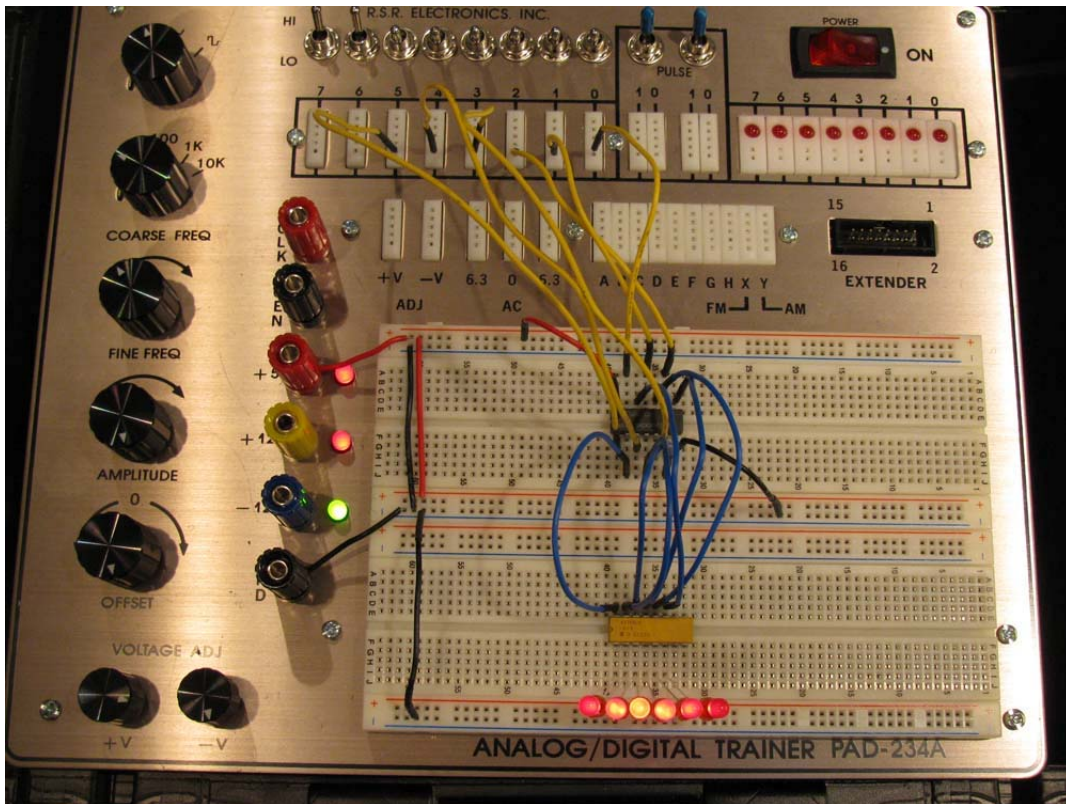
Pin 7 to Ground

Pin 14 to +5 V

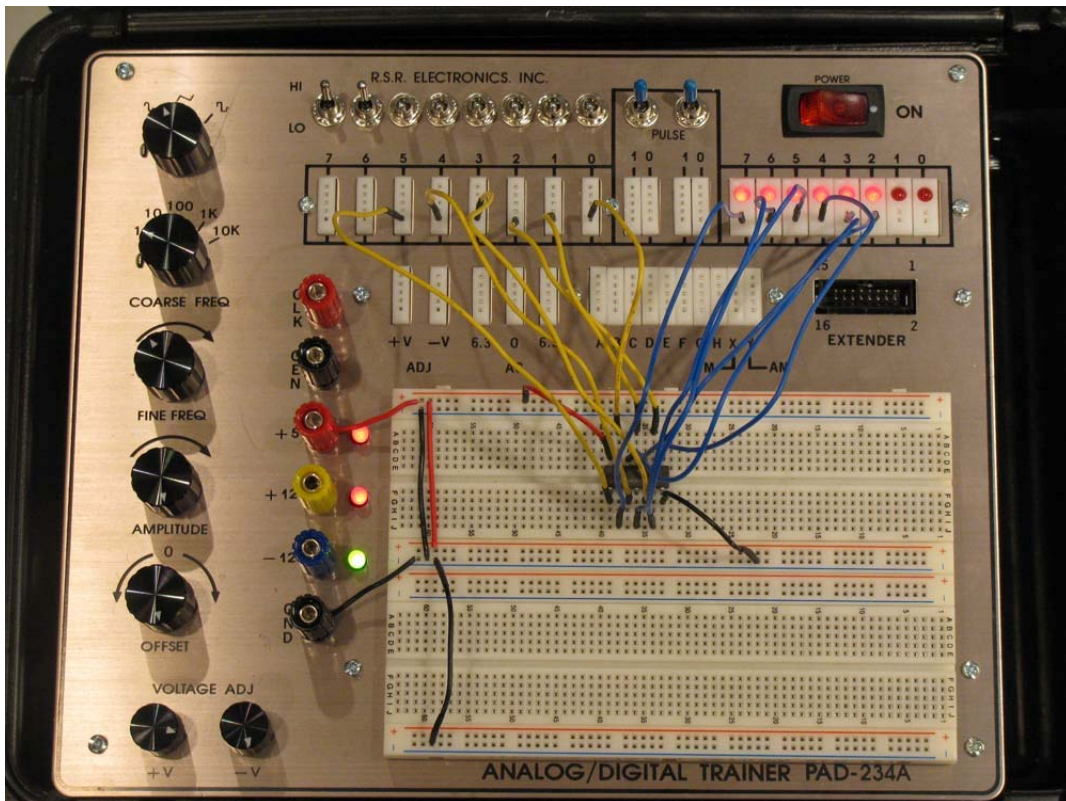
- **Results:**

Input Pin	State	Output Pin	State
1	HIGH	2	
3	LOW	4	
5	LOW	6	
9	HIGH	8	
11	HIGH	10	
13	LOW	12	

- o Photo of layout on the analog/digital trainer using discrete LEDs:



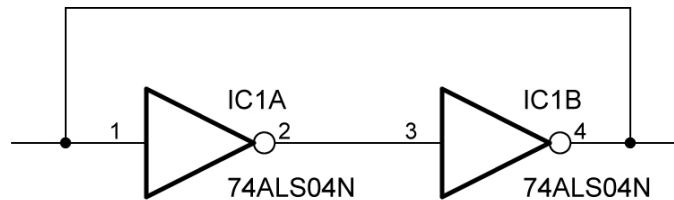
- o Photo of layout on the analog/digital trainer using LEDs on trainer:



## Electronics Technology and Robotics II

### Logic Gates LAB 2 – Inverters as a Bit Storage Unit

- **Purpose:** The purpose of this lab is to acquaint the student with basic bit information storage using inverters.
- **Discussion:**
  - Information (data) in digital systems must be stored for future use.
  - The basic building block for storage is in the form of a single bit (**binary digit**).
  - The circuit below (called a flip-flop) is a simple storage unit that can store one bit of information or data (0 or +5V).
  - With eight of these bits combined, a byte is formed.
- **Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – 74LS04N Hex-Inverter
- **Procedure:**
  - Wire the circuit below.
  - Connect Pin 4 to HI/LOW toggle switch and an LED.
  - Toggle Pin 4 to HIGH then disconnect Pin 4 from the toggle switch. Record the findings below.
  - Now reconnect the toggle to Pin 4 and toggle to LOW. Again disconnect Pin 4 from the toggle switch. Record the findings below.



Pin 7 to Ground, Pin 14 to +5V

#### Bit Storage Using Two Inverters

- **Results:**

Pin 1 & 4 HIGH	LED on/off	Pin 1 & 4 Disconnected	LED on/off
-		-	
Pin 1 & 4 LOW	LED on/off	Pin 1 & 4 Disconnected	LED on/off
-		-	

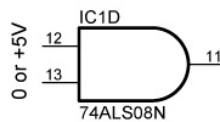
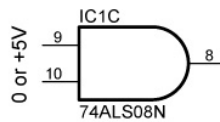
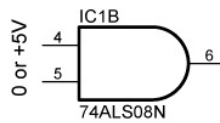
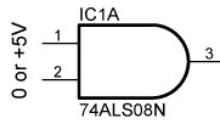
- **Conclusions:**
  - Write your conclusions based upon the results.



## Electronics Technology and Robotics II

### Logic Gates LAB 3 – AND Gates

- **Purpose:** The purpose of this lab is to challenge the student to become acquainted with the basic operation of an AND gate.
- **Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – 74LS08, 2 – Input AND Gate
- **Procedure:**
  - Connect the eight inputs to the HI/LOW toggles on the analog/digital trainer.
  - Use the LEDs on the analog/digital trainer as the outputs.
  - Fill in the table in the result section.



Pin 7 to GND, Pin 14 to +5V

### Quad 2-Input AND Gate

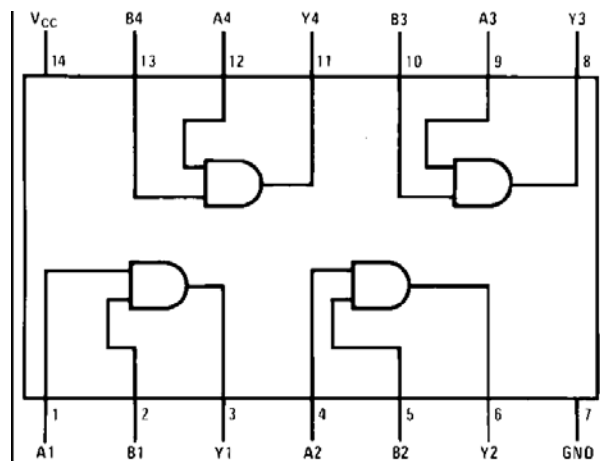
- **Results:**

Pin	HIGH/LOW
1	LOW
2	LOW
3	
4	LOW
5	HIGH
6	
9	HIGH
10	LOW
8	
12	HIGH
13	HIGH

## Electronics Technology and Robotics II

### Logic Gates LAB 4 – AND Gates and NOT Gate

- **Purpose:** The purpose of this lab is to challenge the student to solve a real life design problem using hex-inverter (NOT gate) and an AND gate.
- **Materials:**
  - 1 – Analog/Digital Trainer
  - 2 – SPDT Switches (for ignition and seat belt switches)
  - 1 – 74LS08, 2 – Input AND Gate
  - 1 – 74LS04 Hex-Inverter (NOT Gate)
  - 1 – Piezo Buzzer
- **Procedure:**
  - Design and build a circuit using a hex-inverter and an AND gate to simulate a seat belt alarm. The alarm (piezo buzzer) must turn on only when the following conditions are met:
    - When the ignition switch is on
    - When the seat belt switch is off (the seat belt is unbuckled)
    - Use the HI/LOW toggle switches for the two switches.
    - 74LS08 pin layout:

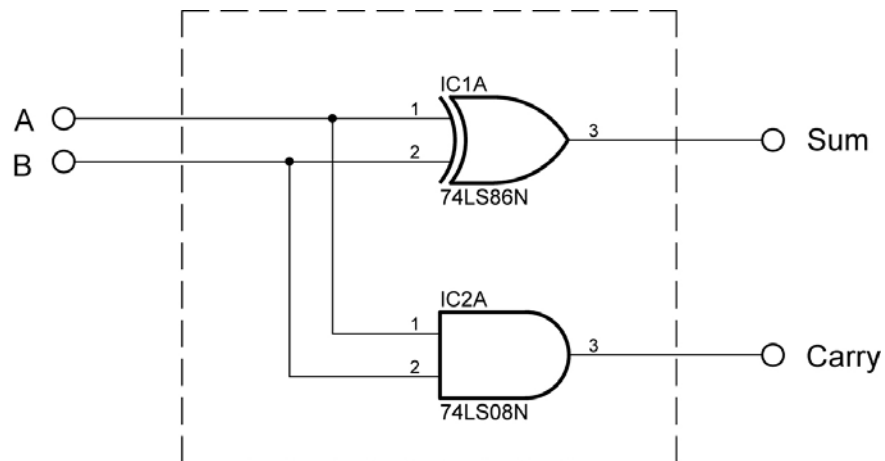


**74LS08 Triple 2-Input AND Gate**

## Electronics Technology and Robotics II

### Logic Gates LAB 5 – Half-Adders

- **Purpose:** The purpose of this lab is to have the student connect together two logic ICs to generate a half-adder and then test their results.
- **Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – 74LS06N AND Gate
  - 1 – 74LS86N XOR Gate
- **Procedure:**
  - Wire the following schematic and complete the half-added truth table.
  - Make sure not to get the chip outputs crisscrossed.
  - Verify your findings with those in the half-adder section.



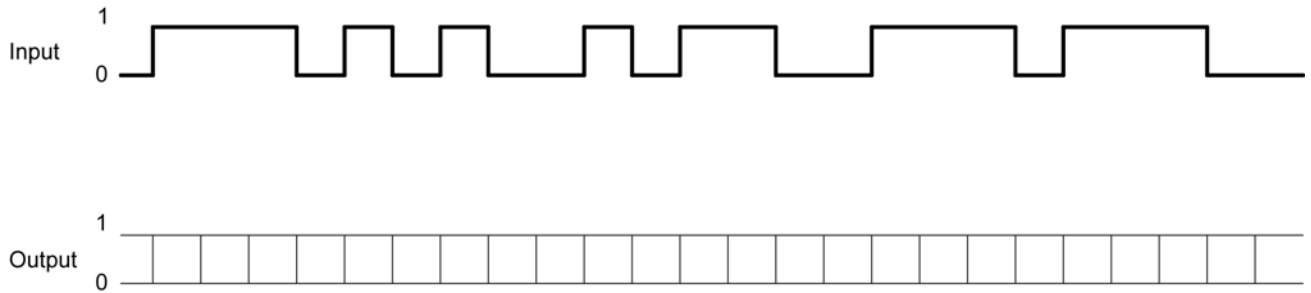
- **Results:**

Binary Inputs		Carry Output (AND Output)	Sum Output (XOR Output)
A	B		
0	0		
0	1		
1	0		
1	1		

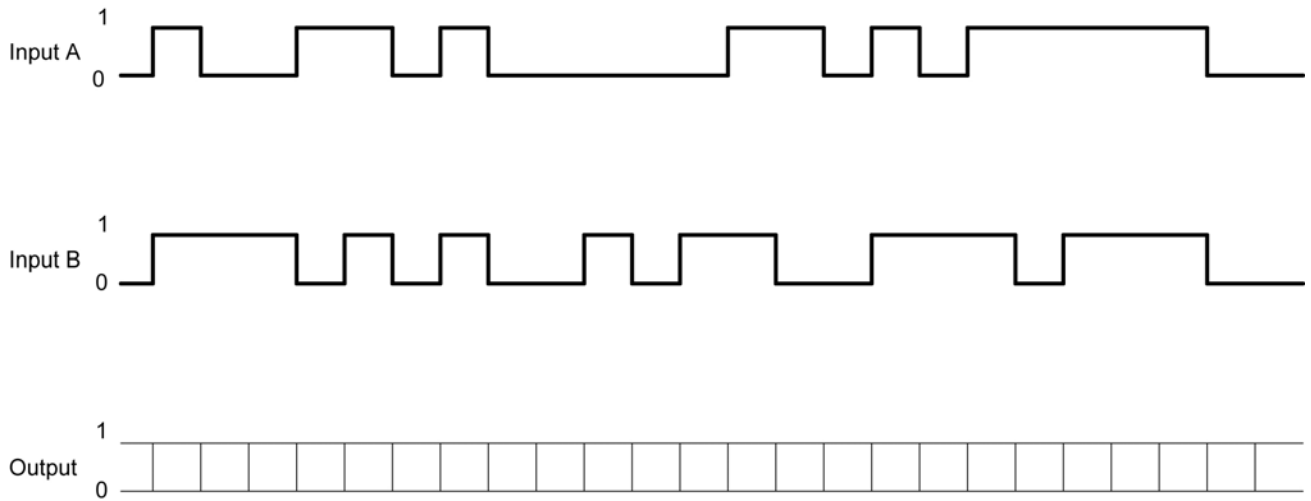
HIGH=1, LOW=0

**Electricity and Electronics II, Chapter 20, Digital Circuits**  
**Logic Gates Continued**  
**Class Interrupts**

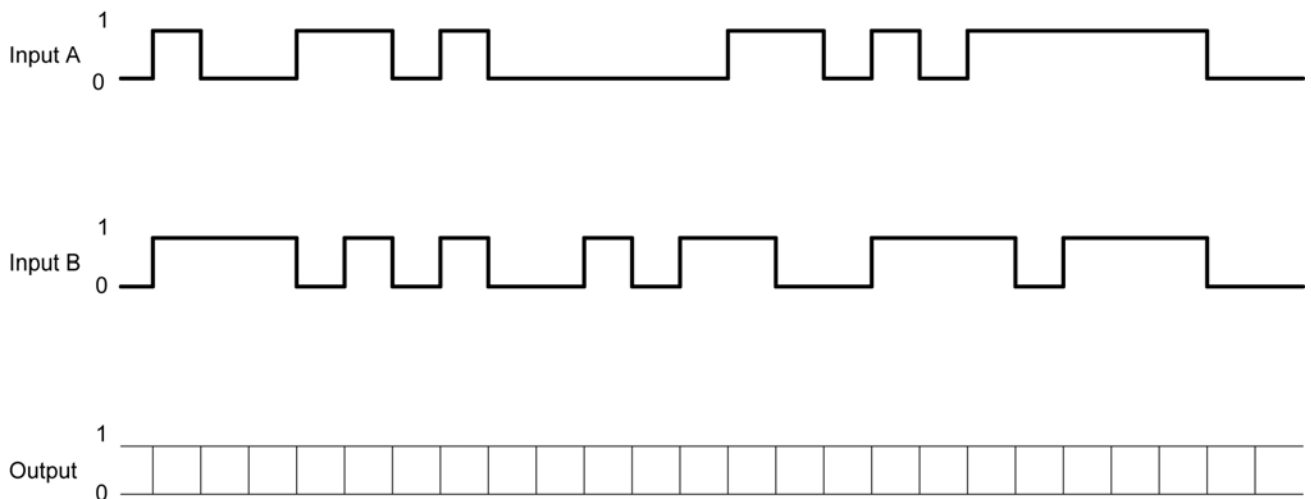
- Draw the output waveform for a NOT gate with the input shown below:



- Draw the output waveform for an AND gate having the inputs shown below:



- Draw the output waveform for an OR gate having the inputs shown below:



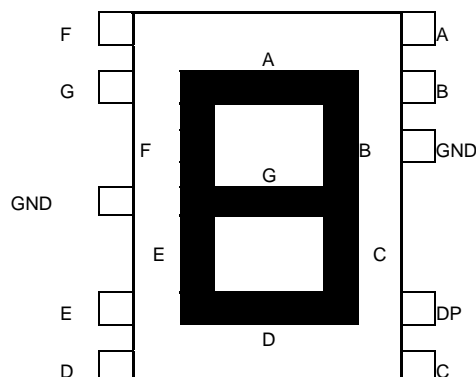
## Cornerstone Electronics Technology and Robotics II Digital Applications LAB 1 – 7-Segment Displays

- **Purpose:** The purpose of this lab is to acquaint the student with a common cathode and common anode 7-segment displays.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – Jameco 24782 Common Cathode 7-Segment Display
    - Maximum voltage rating is 2.8V
  - 1 – Jameco 97172 Common Anode 7-Segment Display
    - Maximum voltage rating is 2.8V
  - 1 – 150 Ohm Resistor
- **Procedure:**
  - Using +5 volts as  $V_{CC}$ , connect the Jameco 24782 common cathode 7-segment display to display the number 3. Connect the 7-segment display pins to the HI/LO switches as shown in the table below.  
**Remember, the maximum voltage rating for this 7-segment display is 2.8V, i.e., remember the 150 ohm resistor.**

7-Segment Pin	HI/LO Switch
A	0
B	1
C	2
D	3
E	4
F	5
G	6

- Use the logic probe to measure each pin and record the results in the first table below.
- Now generate all of the codes for all of the digits 0 – 9 and fill in the table on the next sheet.

### Jameco 24782 Common Cathode Pin Layout



- **Results for Displaying the Number 3:**

Pin	-	G	F	E	D	C	B	A
1 or 0	0							

- **Discussion:** Notice that the line of 1's and 0's in the table can be thought of as a 7-segment code where each number (0-9) has a distinct binary code. Unlike a binary number where each bit is weighted, this binary code is unweighted. This means that the position of each bit does not indicate a magnitude represented for that position.
- **Results Continued:**

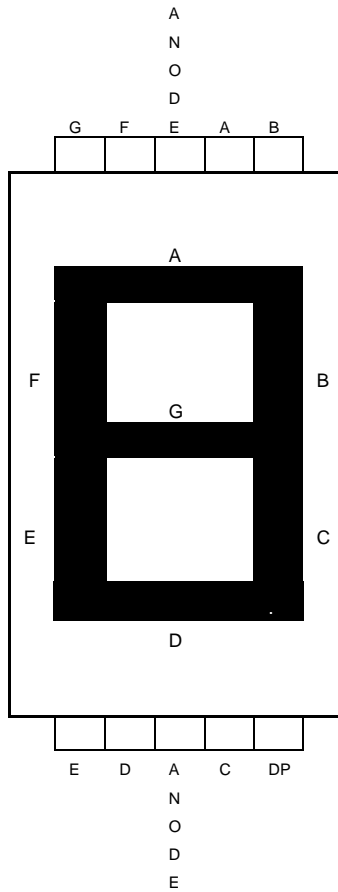
Decimal	-	G	F	E	D	C	B	A
0	0							
1	0							
2	0							
3	0							
4	0							
5	0							
6	0							
7	0							
8	0							
9	0							

**7-Segment Display Look-up Table**

**Cornerstone Electronics Technology and Robotics II  
Digital Applications LAB 1 – 7-Segment Displays Continued**

- Connect the Jameco 97172 common anode 7-segment display to display the number 2. **Remember, the maximum voltage rating for this 7-segment display is 2.8V.**

**Jameco 97172  
Common Anode  
Pin Layout**



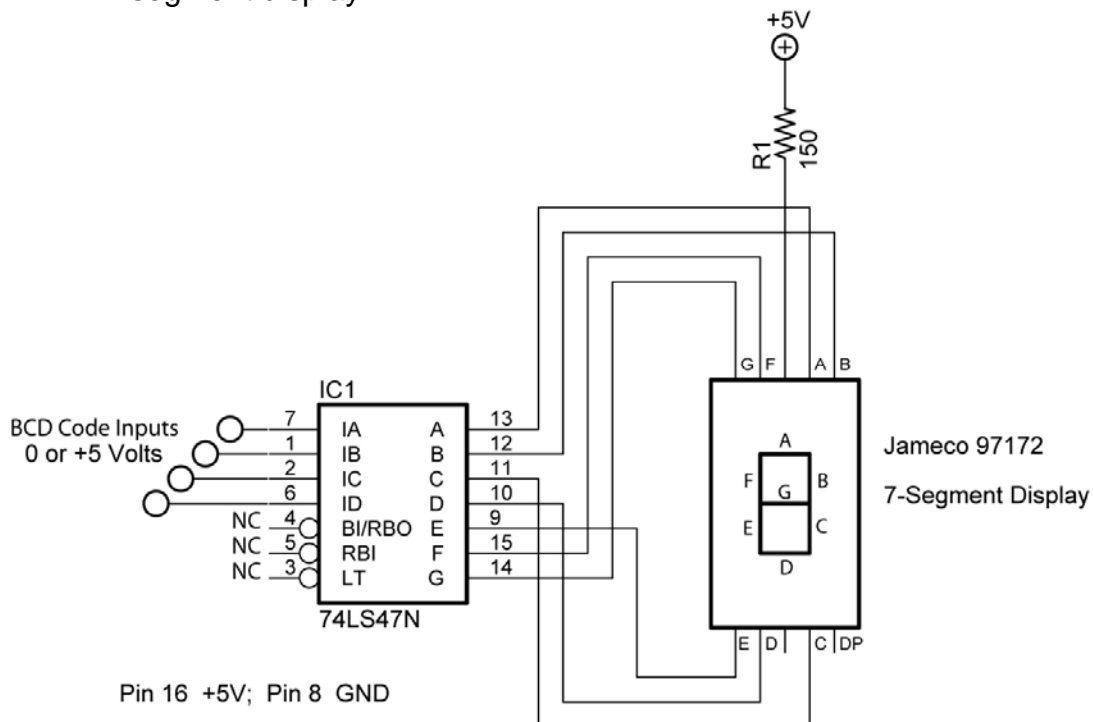
## Cornerstone Electronics Technology and Robotics II Digital Applications LAB 2 – BCD-to-7-Segment Decoder

- **Purpose:** The purpose of this lab is to acquaint the student with the operation of a BCD-to-7-Segment Decoder.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - 1 - 150 Ohm, ½ watt Resistor
  - 1 – Jameco 97172 Common Anode 7-Segment Display
  - 1 - 74LS47 BCD-to-Seven-Segment Decoder
- **Procedure:**
  - Wire the 74LS47 decoder to the 7-segment display as shown. Pin 8 is grounded; Pin 16 is +5V on the 74LS47 chip.
  - Use the HI/LO toggles to create the BCD inputs for Pins 7, 1, 2, and 6. See Table 1 below.

74LS47 Pin	HI/LO Switch
7	0
1	1
2	2
6	3

Table 1

- Using BCD code as the inputs, display each number,(0 – 9) on the 7-segment display.



**74LS47 BCD to Decimal Decoder**



- Table 2 lists 74LS47 inputs and 7-segment display outputs:

BCD Code	74LS47 Pin Inputs (BCD Code)				7-Segment Display Decimal Output
	Pin 6	Pin 2	Pin 1	Pin 7	
	ID	IC	IB	IA	
0000	0	0	0	0	0
0001	0	0	0	1	1
0010	0	0	1	0	2
0011	0	0	1	1	3
0100	0	1	0	0	4
0101	0	1	0	1	5
0110	0	1	1	0	6
0111	0	1	1	1	7
1000	1	0	0	0	8
1001	1	0	0	1	9

**Table 2**

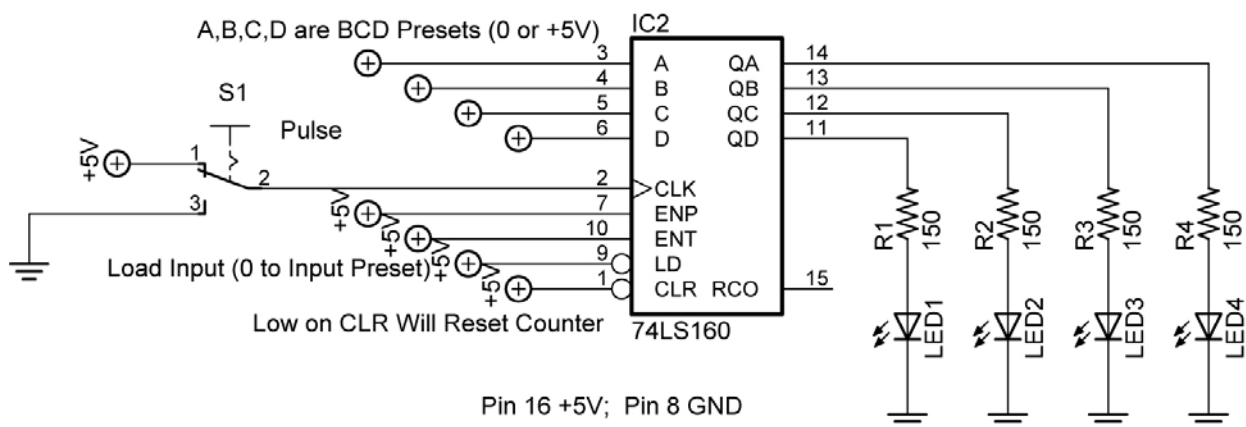
## Cornerstone Electronics Technology and Robotics II Digital Applications LAB 3 – BCD Decade Counter

- **Purpose:** The purpose of this lab is to acquaint the student with the operation of a BCD decade counter.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – 150 Ohm, ½ watt Resistor
  - 1 – Jameco 97172 Common Anode 7-Segment Display
  - 1 – 74LS47 BCD-to-Seven-Segment Decoder
  - 1 – 74LS160 BCD Decade Counter
- **Procedure:**
  - Wire the following BCD decade counter circuit using LEDs as outputs.
  - The LEDs give a binary representation of the counter BCD code output. Toggle the pulse on Pin 2 to advance the BCD count. Pulse (Pin 2) is connected to S1, one of the Pulse toggle switches on the analog/digital trainer.
  - If using a breadboard, S1 must be debounced. See the following links:
    - <http://www.ganssle.com/debouncing.pdf>
    - <http://www.geocities.com/thetonegod/debounce/debounce.html>
    - <http://www.bioinspired.com/users/ajg112/electronics/debounce.shtml>
  - Use the HI/LO toggle switches as the inputs for Pins 3, 4, 5, and 6. See Table 3 below.

74LS160 Pin	HI/LO Switch
3	0
4	1
5	2
6	3

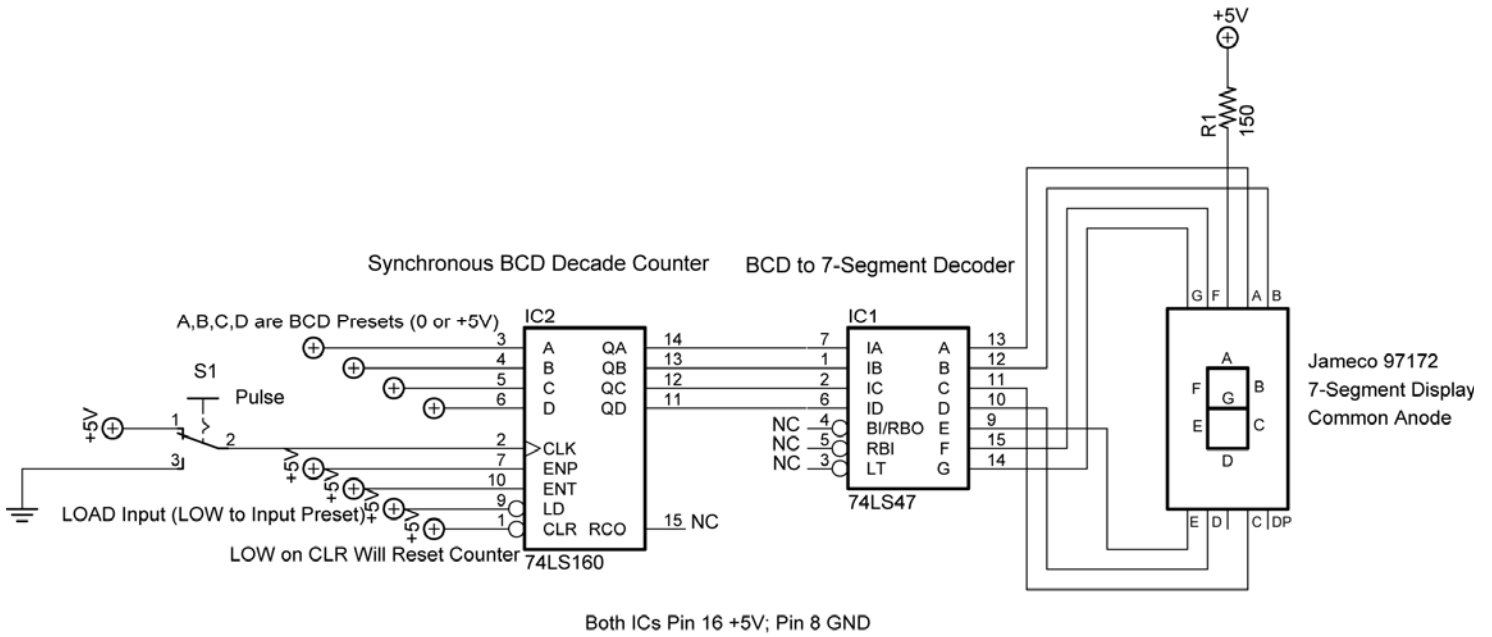
**Table 3**

Synchronous BCD Decade Counter



**Decade Counter Circuit Driving LEDs**

- Connect the counter to the decoder as shown below. **Note:** When connecting the 74LS160 to the 74LS47, just add the 74LS160 BCD decade counter onto the 74LS47 decoder circuit already on the breadboard.



**Decade Counter Circuit Driving BCD to 7-Segment Decoder**

## Cornerstone Electronics Technology and Robotics II

### Digital Applications LAB 3 – BCD Decade Counter Continued

- 74LS160N Features:
  - ENP and ENT, pins 7 and 10 respectively, must be HIGH for the counter to advance. When one or both are LOW, the counter is disabled.
  - LOW on CLR, pin 1, will reset counter to 0.
  - Can preset counter to any BCD count between 0 and 9. Use A, B, C, and D to preset BCD number. A is the LSB, or the Least Significant Bit, and D is the MSB, or the Most Significant Bit. LD (Load), pin 9, must be on LOW to accept the preset number on the next pulse.
  - RCO (Ripple Clock Output), not used here, goes HIGH when the counter reaches the terminal count of 9. This output in conjunction with the enable inputs allows these counters to be cascaded for higher count sequences.
  - See:  
[http://www.datasheetcatalog.com/datasheets\\_pdf/7/4/L/S/74LS160.shtml](http://www.datasheetcatalog.com/datasheets_pdf/7/4/L/S/74LS160.shtml)  
Texas Instruments page 5 for schematic of counter.

**Cornerstone Electronics Technology and Robotics II**  
**Digital Applications LAB 4 – Cascade BCD Decade Counter**

- **Purpose:** The purpose of this lab is to acquaint the student with the method of cascading BCD decade counters.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - 2 – 150 Ohm, ½ watt Resistors
  - 2 – Jameco 97172 Common Anode 7-Segment Displays
  - 2 – 74LS47 BCD-to-Seven-Segment Decoders
  - 2 – 74LS160 BCD Decade Counters
- **Procedure:**
  - Wire the two cascade BCD decade counter circuit on the following page.
  - Use the HI/LO toggle switches as the inputs for Pins 3, 4, 5, and 6 for both BCD decade counters according to Tables 4 and 5.

LSD 74LS160 Pin	HI/LO Switch
3	0
4	1
5	2
6	3

**Table 4**

MSD 74LS160 Pin	HI/LO Switch
3	4
4	5
5	6
6	7

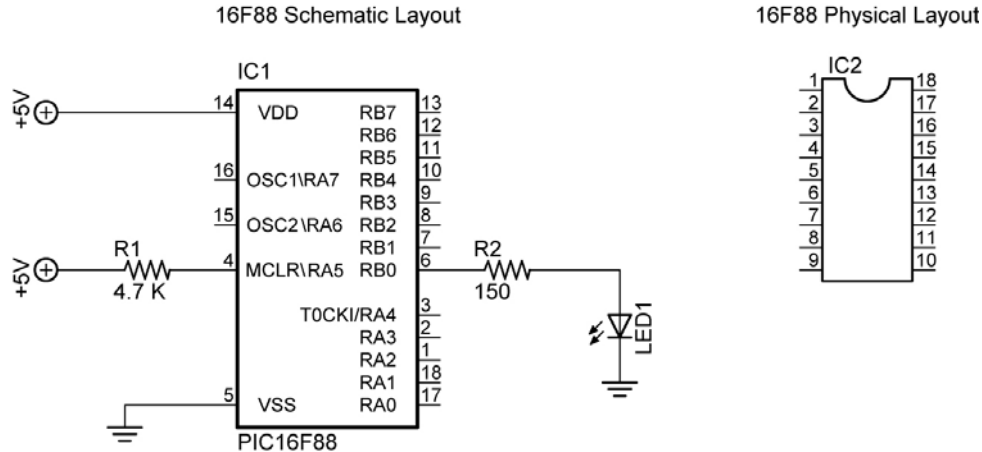
**Table 5**

- LOW on CLR (pin 1) will reset counter to 00 since both CLR's are tied together.
- The counter may be preset to any count between 00 and 99. Use presets A, B, C, and D on each counter to preset BCD number for each counter. A is the LSB, or the Least Significant Bit, and D is the MSB, or the Most Significant Bit for each BCD number. LD (Load), pin 9 on either chip, must be on LOW to accept the preset number on the next pulse. Both of the pin 9s on the counter chips are tied together.
- Use S1 to activate the count.
- **Be careful** not to connect pin 7 to +5V in the MSD counter.
- Additional counters may be cascaded onto this arrangement as shown on page 22 of the Texas Instrument 74LS160 datasheet. See:  
[http://www.datasheetcatalog.com/datasheets\\_pdf/7/4/L/S/74LS160.shtml](http://www.datasheetcatalog.com/datasheets_pdf/7/4/L/S/74LS160.shtml)



## Cornerstone Electronics Technology and Robotics II PIC Microcontrollers Programming 1 LAB 1 - blink1.pbp Program

- **Purpose:** The purpose of this lab is to acquaint the student on how to:
  - Compile a PicBasic program
  - Download a PicBasic program into a PIC16F88 microcontroller
  - Structure a program using three PicBasic commands
  - Make simple modifications to a PicBasic program
  - Make modifications to the TRIS register
  
- **Apparatus and Materials:**
  - 1 – Robotic Car by Student
  - 1 – Breadboard with +5V and +9V Power Supplies
  - 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:**
  - Wire the blink1 circuit below on your robotic car's breadboard.



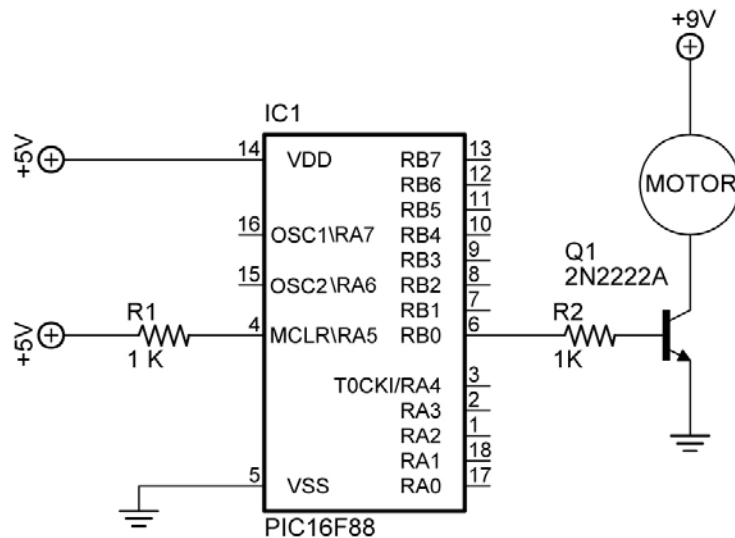
blink1

- Program **blink1.pbp** into the PIC16F88 following the procedure to write, compile and download your program into the PIC chip at the end of the lesson.
- Install the 16F88 and test the circuit and program
- Change the PAUSE values (timing values) and reprogram the chip.

## Cornerstone Electronics Technology and Robotics II PIC Microcontrollers Programming 1 LAB 1, Continued

- **Challenges:**

- Connect the resistor and LED to RB1 and make it blink. Save the program as **blinkrb1**. Remember to change the TRISB register to make RB1 an output.
- **Railroad Crossing:** Wire a circuit using RB0 and RB1 as outputs and program the chip so that two LEDs alternate flashes like a railroad crossing. Let the flash time be 0.75 seconds. Save the program as **railrd1**.
- **Drive a Motor:** Design a circuit and program a PIC16F88 which will drive a motor in one direction only. Name the new program **road1**.
  - Use a 9 vdc power source on a separate breadboard to drive the motor.
  - Tie the **grounds** of the +5 vdc and +9 vdc breadboards together, **but NOT the +5V and +9V power sources**. Use two batteries for the power sources; one for the +5 vdc and the other for the +9 vdc bus rows.
  - Step down the +9 vdc to +5 vdc using a 78L05 voltage regulator circuit. Verify the +5 vdc and +9 vdc bus rows with a DMM.
  - You may use notes from prior classes to review NPN transistor switches.
  - Hints:
    - Keep wires away from the 16F88 chip since it will be removed frequently from the circuit.
    - Place the motor on the on the collector side of the NPN transistor. Place a 1K ohm resistor between the 16F88 drive pin and the base of the NPN transistor. See the circuit below:



**Transistor Switch as a Motor Driver**



- **Drive a Robot:** Now combine this lesson's circuitry and programming to drive your robotic car through the taped course without crossing the inside boundaries of the tape. Revise the program **road1**.
  - You will have to use the process called dead reckoning since the robot is not equipped with any sensors. Wikipedia definition of dead reckoning: Dead reckoning is the process of estimating one's current position based upon a previously determined position, or fix and advancing that position based upon known speed, elapsed time, and course.
  - Hints:
    - Put the following code at the end of the program:

```
PORTB.0 = 0      ' Set PORTB, bit 1 to a LOW (0V)
```

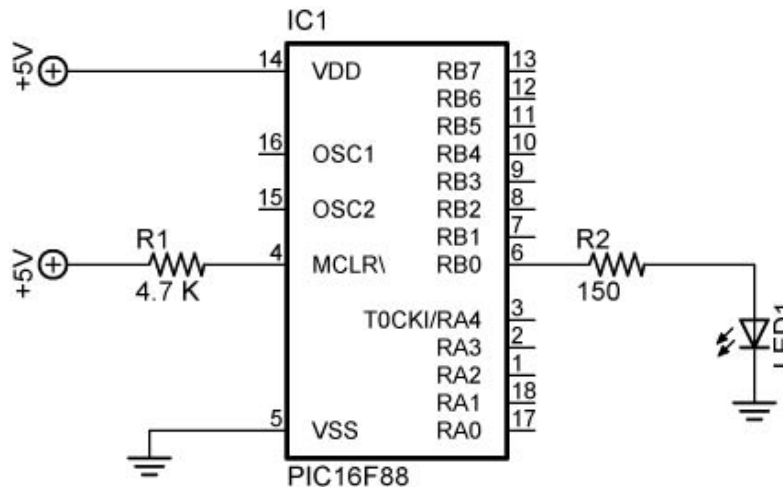
```
PORTB.1 = 0      ' Set PORTB, bit 2 to a LOW (0V)
```

```
PAUSE 1        ' Pause 1 millisecond
```

This code will stop the robotic car.

## Cornerstone Electronics Technology and Robotics II 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.

**Cornerstone Electronics Technology and Robotics II**  
**PIC Microcontrollers Programming 2 LAB 1 – blink2 continued**

- **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**.
  - **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**.

**Cornerstone Electronics Technology and Robotics II**  
**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.

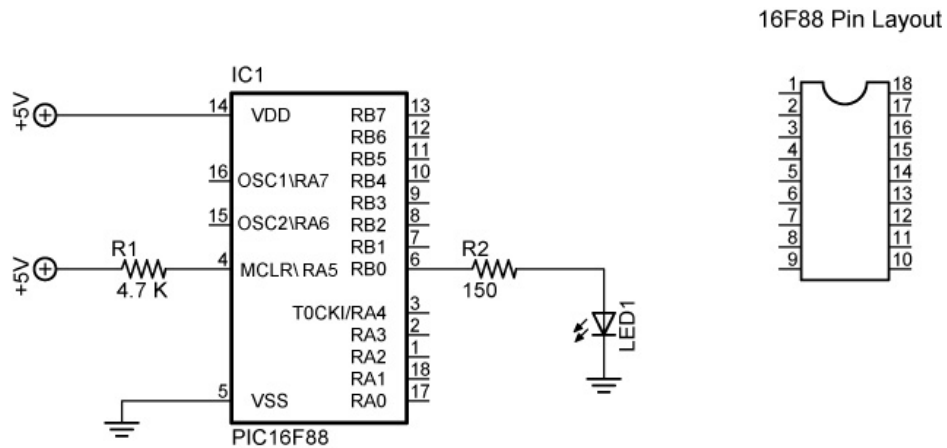
- **Apparatus and Materials:**

- 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



**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:

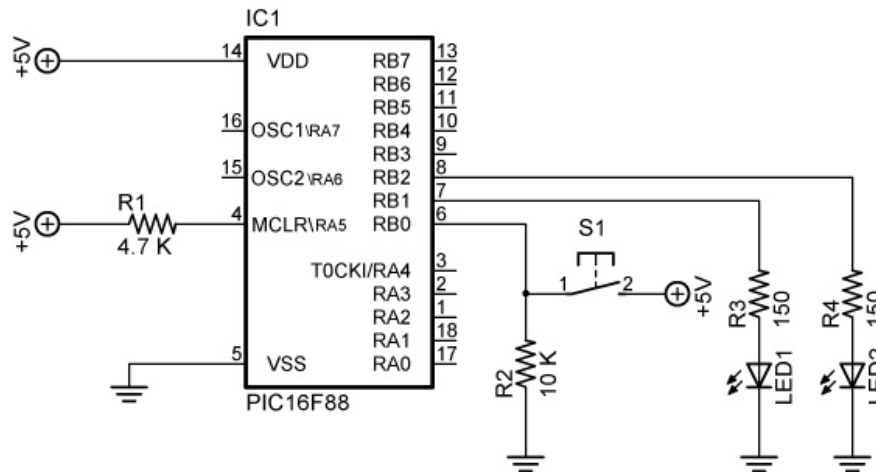
PORTB = x

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.

**Cornerstone Electronics Technology and Robotics II**  
**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** (NOT 16F877A switch1.pbp).
  - 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.

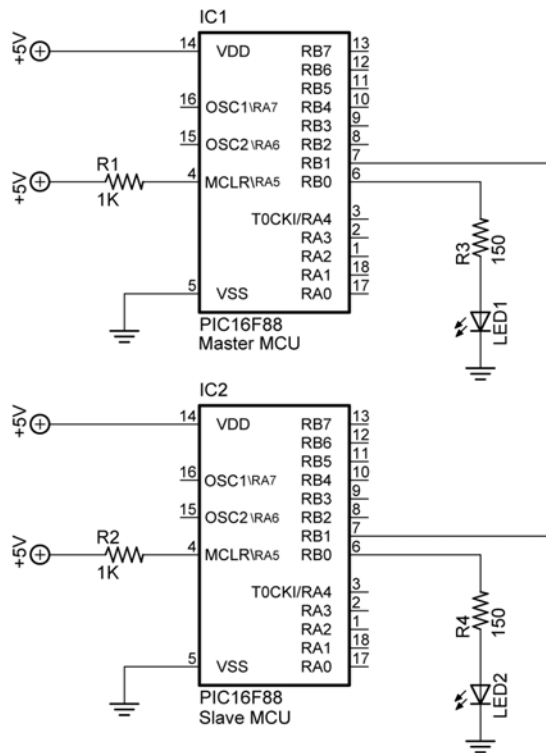


switch1 and switch2

- **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**.

**Cornerstone Electronics Technology and Robotics II**  
**Programming PIC Microcontrollers in PicBasic Pro – Servos**  
**LAB 3 - master\_slave1**

- **Purpose:** The purpose of this lab is to acquaint the student with connecting two PIC microcontrollers such that first MCU controls the timing of the second MCU.
- **Apparatus and Materials:**
  - 1 – Breadboard or Analog/Digital Trainer
  - 2 – PIC16F88
  - 2 – 1K Resistor
  - 2 – 150 Ohm Resistors
  - 2 – LEDs
- **Procedure:**
  - Open master\_slave\_master1.pbp and download to master MCU chip.
  - Open master\_slave\_slave1.pbp and download to slave MCU chip.
  - Wire your breadboard for master\_slave1 as shown below.

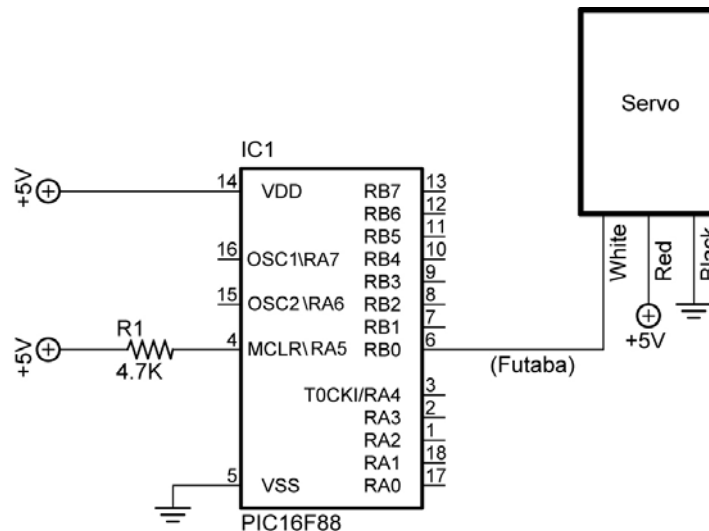


master\_slave1

- **Challenge:**
  - Design a program and circuit using two PIC 16F88s where a switch connected to the first PIC16F88 program turns on/off a dc motor connected to the second PIC16F88. Save the master PIC program as **msm10.pbp** and the slave program as **mss10.pbp**.

**Cornerstone Electronics Technology and Robotics II**  
**Programming PIC Microcontrollers in PicBasic Pro – Servos**  
**LAB 4 - servo1.pbp, servo2.pbp, servo3.pbp, and servo4.pbp**

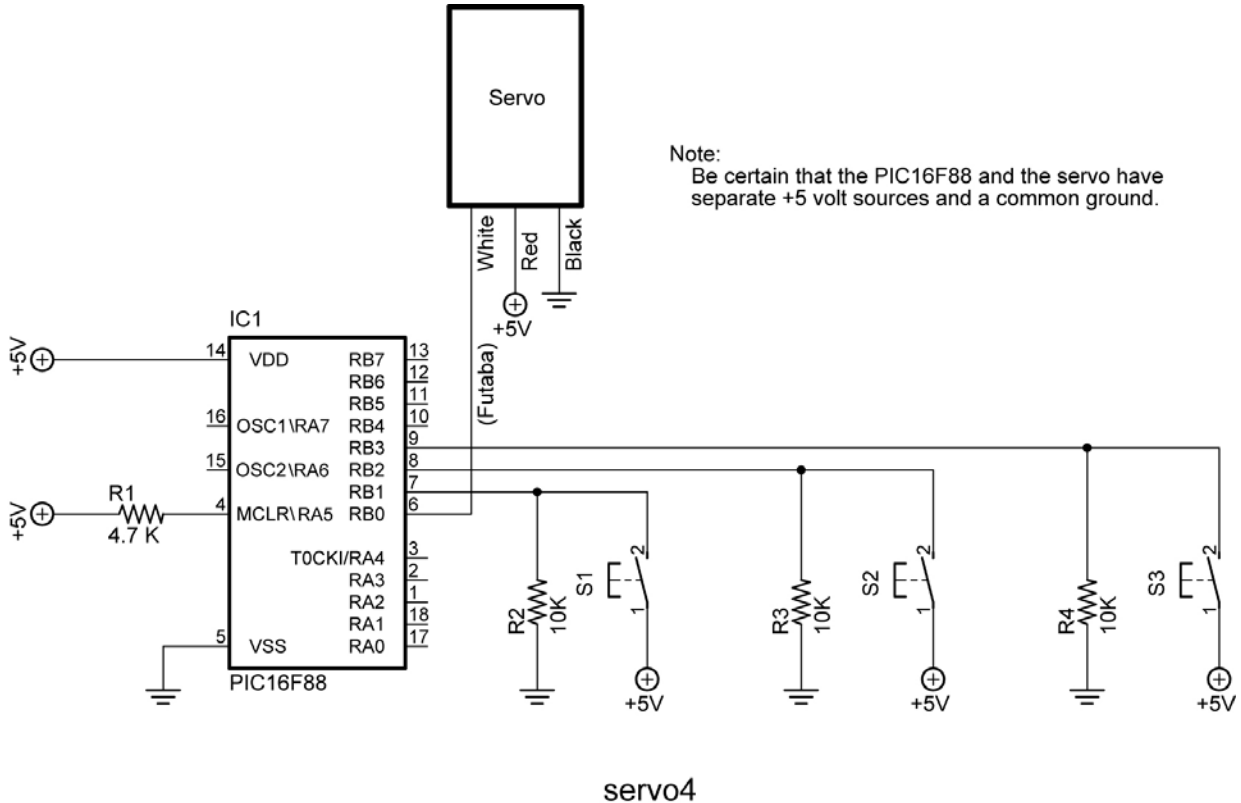
- **Purpose:** The purpose of this lab is to acquaint the student with:
  - PicBasic Pro commands **GOSUB** and **PULSOUT**.
  - The basic operation of a hobby servo.
  
- **Apparatus and Materials:**
  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – Oscilloscope
  - 1 – PIC16F88
  - 1 – 1K Resistor
  - 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.



- **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**  
Statement  
Statement  
Statement  
**ENDIF**

**IF** PORTB.0 = 1**THEN**  
HIGH 2  
LOW 3  
HIGH 6  
**ENDIF**

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

**IF** PORTB.0 = 1**THEN**  
HIGH 2  
LOW 3  
HIGH 6  
**ELSE**  
LOW 2  
LOW 5  
LOW 7  
**ENDIF**

**SELECT CASE** Variable

**SELECT CASE** x

**CASE** Value  
Statement  
Statement

**CASE** 1  
LCDOUT \$FE, 1, "x = ", DEC x  
x = x + 1

**CASE** Another Value  
Statement  
Statement

**CASE** 2  
LCDOUT \$FE, 1, "x = ", DEC x  
x = x + 2

**CASE IS** Comparison  
Statement  
Statement

**CASE IS** > 10  
LCDOUT \$FE, 1, "x > 10 "  
x = 0

**CASE ELSE**  
Statement  
Statement

**CASE ELSE**  
LCDOUT \$FE, 1, "Problem number"  
x = 0

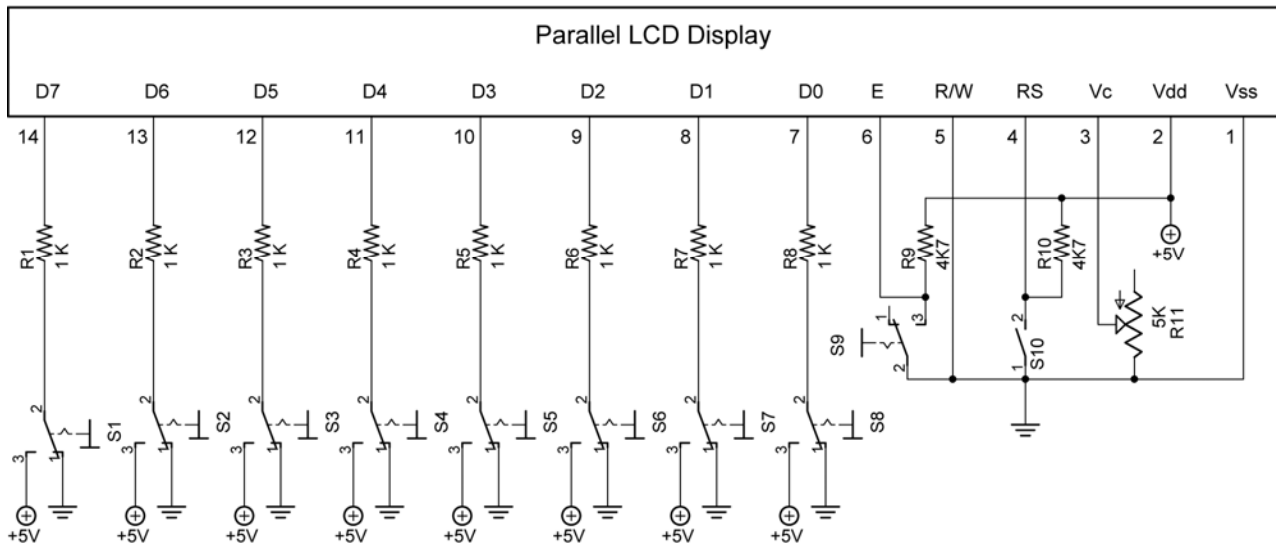
**END SELECT**

**END SELECT**

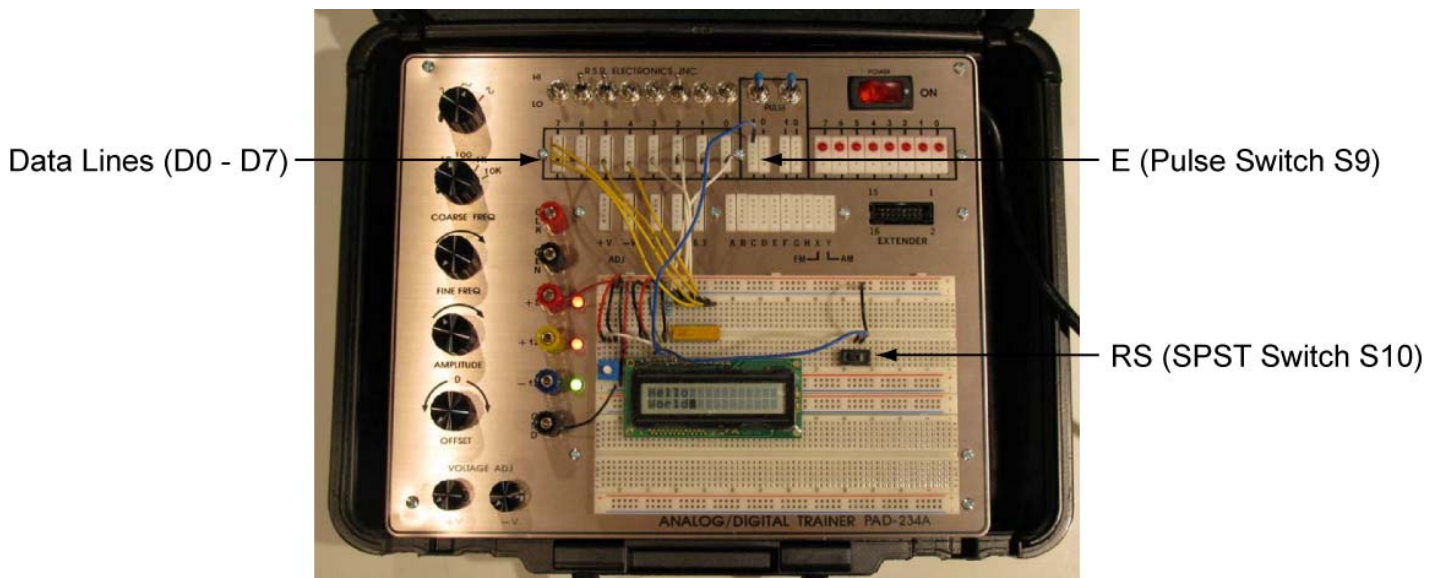
## Cornerstone Electronics Technology and Robotics II LCD1 LAB 1 – Command Control Codes

- **Purpose:** The purpose of this lab is to acquaint the student with the LCD pin functions and sending LCD command control codes to the LCD.
- **Apparatus and Materials for Analog/Digital Trainer Circuit:**
  - 1 – Analog/Digital Trainer
  - 1 – 16x2 Parallel LCD Display, Jameco Part # 618003
  - 1 – 1K DIP Resistor Package
  - 2 – 4K7 Resistors
  - 1 – 5K Ohm Potentiometer
  - 1 – SPST Switch
  - 1 – NO Momentary Switch
- **Discussion and Procedure:**
  - Wire the circuit below.

Jameco Part # 618003 (No Back Light)



### Manual Experiments Circuit for Analog/Digital Trainer



○ **Display On/Off & Cursor Commands:**

Display On/Off & Cursor	0	0	0	0	1	D	U	B
-------------------------	---	---	---	---	---	---	---	---

- Adjust the potentiometer until the squares are slightly visible.
- RS is LOW (RS is LOW whenever using LCD command control codes)
- Set Display On/Off & Cursor commands as follows: S1 – S8 in the schematic to %00001111 (\$0F)
  - D=1, Display On; D=0, Display Off (from LCD Command Control Codes in table above)
  - U=1, Cursor Underline On; U=0, Cursor Underline Off
  - B=1, Cursor Blink On; B=0, Cursor Blink Off
- Press S9 momentarily (E) which enables the chip to accept the command.
- Upper left hand corner cursor should be blinking and underlined.
- Referring to the LCD Command Control Codes Table, turn off the underline mode and leave the blinking mode on. Now turn off the blinking mode and turn the underline mode back on. **Remember, RS is LOW when sending commands to the LCD and that E must be triggered when sending each command.**

○ **Function Set Commands:**

Function Set	0	0	1	8/4	2/1	10/7	x	x
--------------	---	---	---	-----	-----	------	---	---

- RS is LOW
- Set Function Set commands as follows:
  - S1 – S8 to %00111000 (\$38)
  - 8/4: 1=8-bit interface. 0=4-bit interface
  - 2/1: 1=2 line mode, 0=1 line mode
  - 10/7: 0=5x7 dot matrix, 1=5x10 dot matrix
  - This sets the Function Set commands to 8-bit data, two lines, and 5x7 dot matrix format.
- Press S9 momentarily (E) which enables the chip to accept the command.
- Change the 2/1 entry to 1 line mode then back to 2 line mode.
- Increase contrast to increase drive to two lines.
- Return to the lesson.

○ **Clearing Display Command:**

Clear Display	0	0	0	0	0	0	0	1
---------------	---	---	---	---	---	---	---	---

- Set RS back to LOW
- Set the Clear Display command as follows:
  - S1 – S8 to %00000001
- Press S9 momentarily (E) which enables the chip to accept the command.

- **Entering Text:**
  - Set RS to HIGH (Open switch 10).
  - Set S1 – S8 to %01000001
  - Press S9 momentarily (E) which enables the chip to accept the character.
  - The capital letter A should appear in the upper left corner.
  - Enter other text:
    - RS remains on HIGH
    - Set S1 – S8 to the character code found in the Standard LCD Character Table above for the desired character.
    - Press S9 momentarily (E) which enables the chip to accept the characters.
    - Input your name with capital and small letters.
    - The code for a blank space is %11111110.
    - If you want to correct a letter entry, use the Display/Cursor Shift Command below.
- **Display/Cursor Shift Command:**

Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x
----------------------	---	---	---	---	-----	-----	---	---

- The display or the cursor may be shifted to the right or left on the LCD screen.
- RS is LOW
- Set Display/Cursor Shift commands as follows:
  - S1 – S8 to %000100xx where x can be HIGH or LOW – it doesn't matter.
  - D/C: 1=Display Shift, 0=Cursor Move
  - R/L: 1=Right Shift, 0=Left Shift
  - This sets the Display/Cursor Shift commands to move the *cursor* to the left so you are able to reenter a letter.
- To shift the *display* to the left enter %00011000
- Pulse the Enable switch for each shift to the left. Continue pulsing until the string cycles back to the beginning position.
- Move your Enable lead to the clock terminal (CLK) on the analog/digital trainer and change the frequency of your clock.
- Enter the code to shift right.
- **Display & Cursor Home Command:**

Display & Cursor Home	0	0	0	0	0	0	1	X
-----------------------	---	---	---	---	---	---	---	---

- Start by entering some text
- Now set RS to LOW
- Set the Display & Cursor Home command as follows:
  - S1 – S8 to %0000001x where x can be HIGH or LOW, it doesn't matter.
- Press S9 momentarily (E) which enables the chip to accept the command.
- Return to the lesson.

○ **Set Display Address Command:**

Set Display Address	1	A	A	A	A	A	A	A
---------------------	---	---	---	---	---	---	---	---

- RS is LOW
- LCD display address command must be %10000000 greater than the display address. For example, the display address %0000111 must be input as %10000111 and the display address %1000101 must be input as %11000101.
- Press S9 momentarily (E) which enables the chip to accept the command.
- Clear the LCD
- Move cursor to different locations on the LCD
- Enter “Cornerstone Academy” on the first line starting in display address %0000000. Notice that only “Cornerstone Acad” will show up on the display.
- Enter “Gainesville, FL 32605” on the second line indented one space on the display. Only “Gainesville, FL ” will appear on the display.
- Keep “Cornerstone Academy” and “Gainesville, FL 32605” on the display for the next exercise.
- Return to the lesson.

0000000	0000001	0000010	0000011	0000100	0000101	0000110	0000111	0001000	0001001	0001010	0001011	0001100	0001101	0001110	0001111
1000000	1000001	1000010	1000011	1000100	1000101	1000110	1000111	1001000	1001001	1001010	1001011	1001100	1001101	1001110	1001111

**7-Bit Display Addresses for 16 Character 2 Line Display in Binary**

○ **Character Entry Mode Command:**

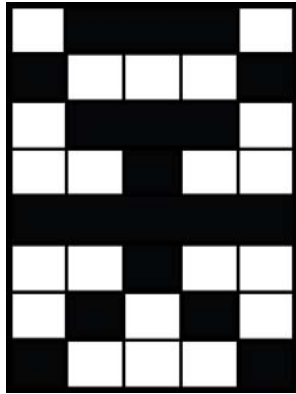
Character Entry Mode	0	0	0	0	0	1	I/D	S
----------------------	---	---	---	---	---	---	-----	---

- Clear the display
- RS is LOW
- Enter %10010000 which shifts the cursor to one location to the right of the right-most top row position, i.e., the cursor is just off the LCD display by one spot.
- Enter the Character Entry Mode command %00000111.
  - I/D: 1=Increment\*, 0=Decrement
  - S: 1=Display Shift On, 0=Display Shift Off
- Set RS HIGH.
- Now enter the characters “195”.

## Cornerstone Electronics Technology and Robotics II

### LCD1 LAB 2 – User-Defined Graphics

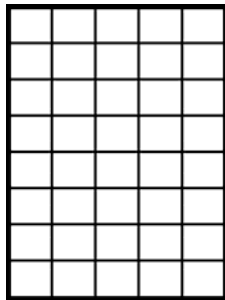
- **Purpose:** The purpose of this lab is to teach the student how to create their own graphical displays on an LCD display.
- **Apparatus and Materials for Analog/Digital Trainer Circuit:**
  - 1 – Analog/Digital Trainer
  - 1 – 16x2 Parallel LCD Display, Jameco Part # 618003
  - 1 – 1K DIP Resistor Package
  - 2 – 4K7 Resistors
  - 1 – 5K Ohm Potentiometer
  - 1 – SPST Switch
  - 1 – NO Momentary Switch
- **Procedure:**
  - **RS is LOW**
  - Set Display On/Off & Cursor commands as 00001111.
  - Set Function Set commands as %00111000.
  - Set Clear Display command as %00000001.
  - Set Set Display Address command to %00000000.
  - **Set RS HIGH**
  - Enter %00000000 to %00000111 in order, i.e., %00000000, %00000001, %00000010, %00000011, %00000100, %00000101, %00000110, %00000111.
  - **Set RS LOW**
  - Set CGRAM Address command %01000000. The cursor should jump to the second line.
  - **Set RS HIGH**
  - Enter the following data (See graphic below for correlation):
    - %00001110,
    - %00010001
    - %00001110
    - %00000100
    - %00011111
    - %00000100
    - %00001010
    - %00010001
    - A stick man should appear in the upper left corner.



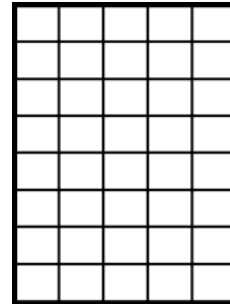
00001110  
 00010001  
 00001110  
 00000100  
 00011111  
 00000100  
 00001010  
 00010001

**Stick Man Layout, Character Code 00000000**

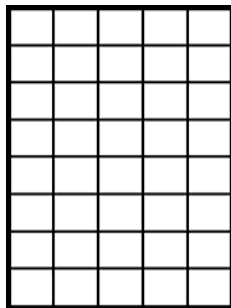
- Use the charts to create up to seven more graphics and continuing entering the data.



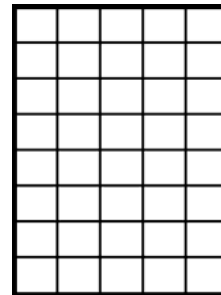
**Char. Code %00000001**



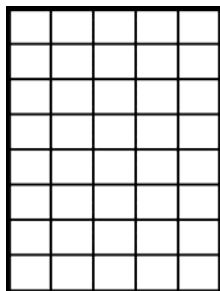
**Char. Code %00000010**



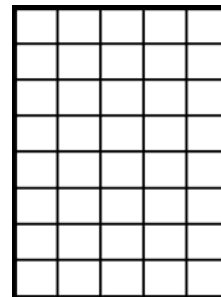
**Char. Code %00000011**



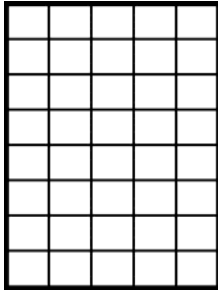
**Char. Code %00000100**



**Char. Code %00000101**



**Char. Code %00000110**



**Char. Code %00000111**

- Set Clear Display command as 00000001.
- Enter the character code for each custom graphic you have entered. Enter the code just as you would for any other character in the Standard LCD Character Table.
- **THE CGRAM MEMORY IS VOLATILE, IF YOU TURN OFF YOUR ANALOG/DIGITAL TRAINER, YOU WILL LOOSE YOUR USER-DEFINED CHARACTERS.**



## Quick Guide to LCD Command Control Codes

See LCD Command Control Codes Table for Details

- RS set to LOW for all command control code inputs
- RS set to HIGH for all character inputs
- Complete all command inputs by momentarily bringing the Enable pin 6 to LOW by toggling Switch 9.
  
- **Clear Display:**                    %00000001
- **Display & Cursor Home:**       %00000010
- **Display On/Off & Cursor:**   %00001111
  - Display On (1)
  - Cursor Underline (1)
  - Cursor Blink On (1)
- **Display/Cursor Shift:**        %00011000        %00011100
  - Display Shift (1)        Left Shift (0)        Right Shift (1)
- **Function Set:**                    %00111000
  - 8-Bit Interface
  - 2 Line Mode (Increase Contrast w/ Potentiometer)
  - 5x7 Dot Matrix Format
- **Set Display Address:**        %1 Plus Seven Position Code

0000000	0000001	0000010	0000011	0000100	0000101	0000110	0000111	0001000	0001001	0001010	0001011	0001100	0001101	0001110	0001111
1000000	1000001	1000010	1000011	1000100	1000101	1000110	1000111	1001000	1001001	1001010	1001011	1001100	1001101	1001110	1001111

### Display Addresses for 16 Character 2 Line Display in Binary

- **Typical Startup Settings:**
  - **Display On/Off & Cursor:**   %00001111
    - Features:
      - Display On
      - Cursor Underline
      - Cursor Blink On
  - **Function Set:**                    %00111000
    - Features:
      - 8-Bit Interface
      - 2 Line Mode
      - 5x7 Dot Matrix

Command	Binary								Hex
	D7	D6	D5	D4	D3	D2	D1	D0	
Clear Display	0	0	0	0	0	0	0	1	01
Display & Cursor Home	0	0	0	0	0	0	1	X	02 or 03
Character Entry Mode	0	0	0	0	0	1	I/D	S	04 to 07
Display On/Off & Cursor	0	0	0	0	1	D	U	B	08 to 0F
Display/Cursor Shift	0	0	0	1	D/C	R/L	x	x	10 to 1F
Function Set	0	0	1	8/4	2/1	10/7	x	x	20 to 3F
Set CGRAM Address	0	1	A	A	A	A	A	A	10 to 7F
Set Display Address	1	A	A	A	A	A	A	A	80 to FF

I/D: 1=Increment*, 0=Decrement	R/L: 1=Right Shift, 0=Left Shift
S: 1=Display Shift On, 0=Display Shift Off	8/4: 1=8-bit Interface*, 0=4-bit Interface
D: 1=Display On, 0=Display Off*	2/1: 1=2 Line Mode, 0=1 Line Mode*
U: 1=Cursor Underline On, 0=Underline Off*	10/7: 1=5x10 Dot Matrix, 0=5x7 Dot Matrix
B: 1=Cursor Blink On, 0=Cursor Blink Off*	
D/C: 1=Display Shift, 0=Cursor Move	X=Don't Care * = Initialization Settings

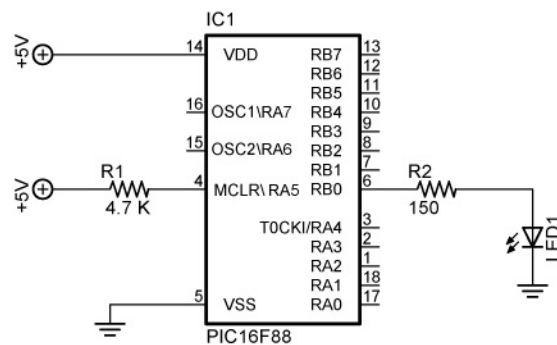
### Char. code

xxxx	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

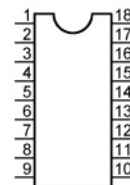
0000000	0000001	0000010	0000011	0000100	0000101	0000110	0000111	0001000	0001001	0001010	0001011	0001100	0001101	0001110	0001111
1000000	1000001	1000010	1000011	1000100	1000101	1000110	1000111	1001000	1001001	1001010	1001011	1001100	1001101	1001110	1001111

## Electronics and Robotics II In-Circuit Serial Programming LAB 1 – blink2

- **Purpose:** The purpose of this lab is to show the student the ease of programming a PIC with in-circuit serial programming.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer or breadboard with 5 V Supply
  - 1 – 150 Ohm, ½ Watt Resistors
  - 1 – 4.7K, ½ Watt Resistor
  - 1 – LED
- **Procedure:**
  - Keep the ICSP connections to the breadboard
  - Open **blink2.pbp** and download to your chip. Wire your breadboard for blink2.
  - Change the pin location and blinking times and download without removing the PIC MCU.



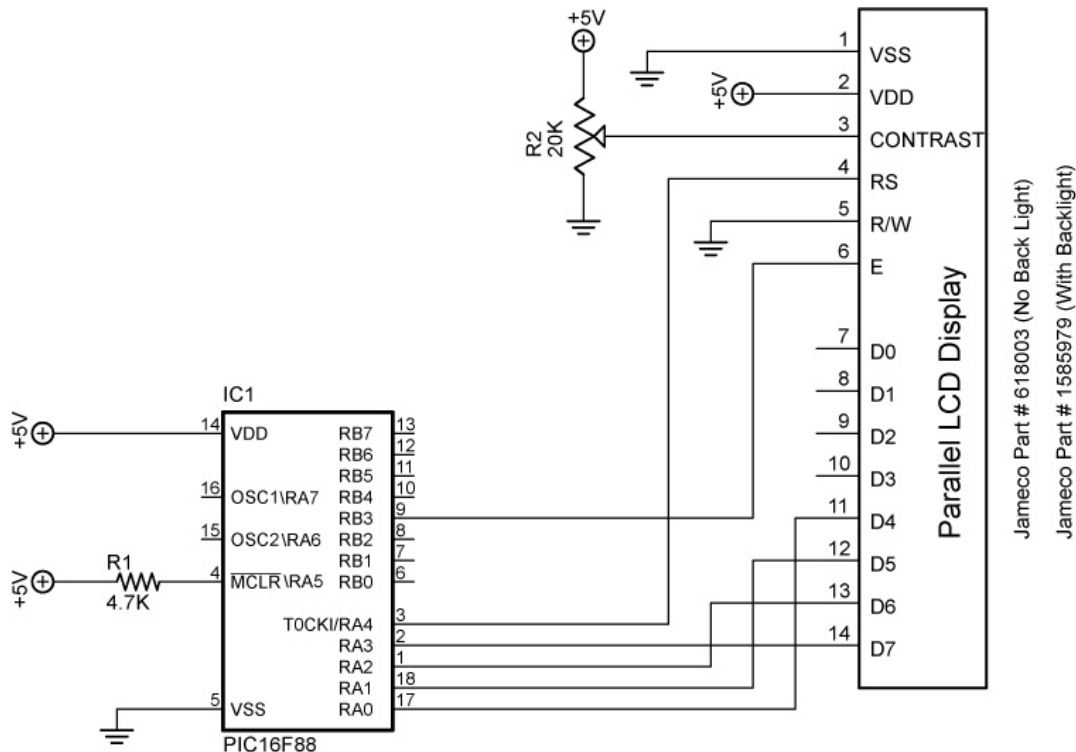
16F88 Pin Layout



**blink1, blink2, blink3**

## Cornerstone Electronics Technology and Robotics II LCD Lesson 2 LAB 1 – lcd1.pbp and lcd2.pbp, LCD Commands

- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro command **LCDOUT** .
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer or Breadboard
  - PIC16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 4.7 K Resistor
- **Procedure:**
  - Wire the following circuit:



lcd1 and lcd2

- Notice that only D4 – D7 data inputs are used; we will use 4-bit data transfer to save pins.
- Open lcd1.pbp from your folder and program your chip. Add your own text. Save your program as **lcd10.pbp**.
- Open lcd2.pbp from your folder and program your chip. Run the program on the breadboard.

Command	Operation	Activity	Completed
\$FE, 1	Clear display	1	
\$FE, 2	Return home	2	
\$FE, \$0C	Cursor off	2	
\$FE, \$0E	Underline cursor on	2	
\$FE, \$0F	Blinking cursor on	2	
\$FE, \$10	Move cursor left one position	5	
\$FE, \$14	Move cursor right one position	5	
\$FE, \$18	Display shift left	4	
\$FE, \$1C	Display shift right	4	
\$FE, \$80	Move cursor to beginning of first line	3	
\$FE, \$C0	Move cursor to beginning of second line	3	
\$FE, \$94	Move cursor to beginning of third line	-	n/a
\$FE, \$D4	Move cursor to beginning of fourth line	-	n/a

- LCD Command Table Experiments: Use the table above to check off each command as it is completed.
  - Activity 1:
    - Open **lcd1.pbp**; save it as **lcd11.pbp**.
    - Revise the program so that “Hello World” remains on the LCD screen for 2 seconds, then make the LCD blank using the \$FE,1 command.
  - Activity 2:
    - Start with **lcd11.pbp** and save the new program as **lcd12.pbp**
    - Revise **lcd12.pbp** such that:
      - “Hello World” remains on the LCD for 2 seconds
      - Clear Display
      - Return the cursor home and blinking for 2 more seconds. Each LCD command may be put on a separate line.
      - Now underline the cursor for 2 seconds
      - Clear display
  - Activity 3:
    - Start again with **lcd11.pbp** and save the new program as **lcd13.pbp**
    - Remember:

**LCDOUT \$FE, \$80 + 4**

sets the display to start writing characters at the forth position of the first line.

**Cornerstone Electronics Technology and Robotics II**  
**LCD Lesson 2 LAB 1 – lcd1.pbp and lcd2.pbp, LCD Commands Continued**

- Activity 3 Continued:
  - Revise **lcd13.pbp** such that:
    - “Hello” begins on the first line, 6 positions in from the right.
    - Pause 1 second
    - “World” begins on the second line, 6 positions in from the right
    - Pause 1 second
    - Clear display for 1 second
    - Repeat the whole sequence 3 times
- Activity 4:
  - Start with **lcd1.pbp** and save the new program as **lcd14.pbp**
  - Revise **lcd14.pbp** such that:
    - “Hello World” shifts to the right one position at a time for 4 positions
    - “Hello World” shifts to the left one position at a time for 4 positions
    - Repeat both shifts 3 times
- Activity 5:
  - Start with **lcd1.pbp** and save the new program as **lcd15.pbp**
  - Delete the main code in **lcd15.pbp**
  - Revise **lcd15.pbp** such that:
    - Display a variable name “Resistor1 = ”
    - Move over to the right one blank space and display the value of the variable r. At this point, you will have to input a value for r, such as, r = 330. To display the value of r in the **LCDOUT** command, place a # sign immediately before r.

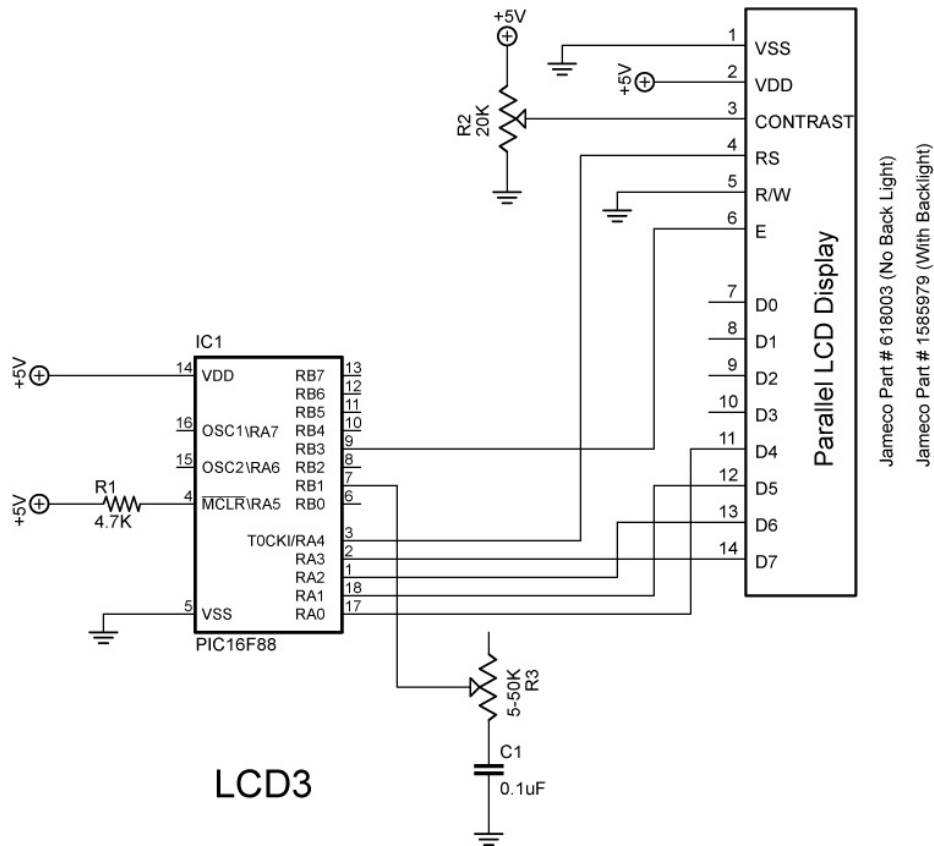
**LCDOUT \$FE,1,“ Resistor1 = ”, #r**

- **Challenge:** Display your first name on the first line and your second name on the second line. The first letters of your first and last names are to appear in the far right positions on the LCD and remain for 2 seconds. Then your first and last names are to scroll across the screen right to left.

## Cornerstone Electronics Technology and Robotics II

### LCD Lesson 3 LAB 1 – lcd3.pbp

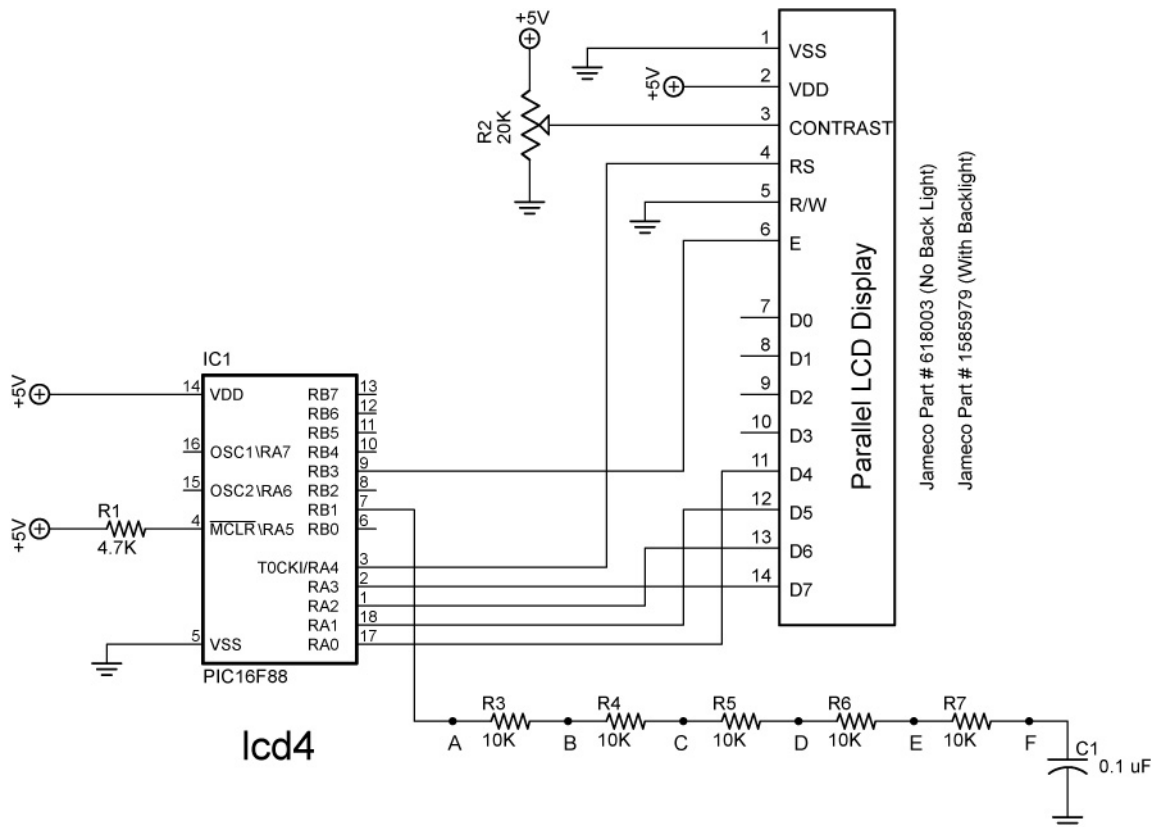
- **Purpose:** The purpose of this lab is to acquaint the student the PicBasic Pro command **POT** and the use of an LCD to monitor variable input values.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - PIC 16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 10K Potentiometer (R3)
  - 4.7 K Resistor
  - 0.1 uF Capacitor
- **Procedure:**
  - Wire the in-circuit serial programming connections before proceeding. See Lesson 15A, In-Circuit Serial Programming for details.
  - Now add the following circuitry. Use a 10K trimpot for R3:



- Open **lcd3.pbp** and download into the PIC16F88.
  - Adjust the 10K potentiometer R3 to see if the full range of the potentiometer is engaged.
  - If the full range is not active, adjust the Scale value in the POT command to make the full range active. Do this by empirically (through observation and experimentation) setting the Scale value to its lowest number while the LCD displays 255.

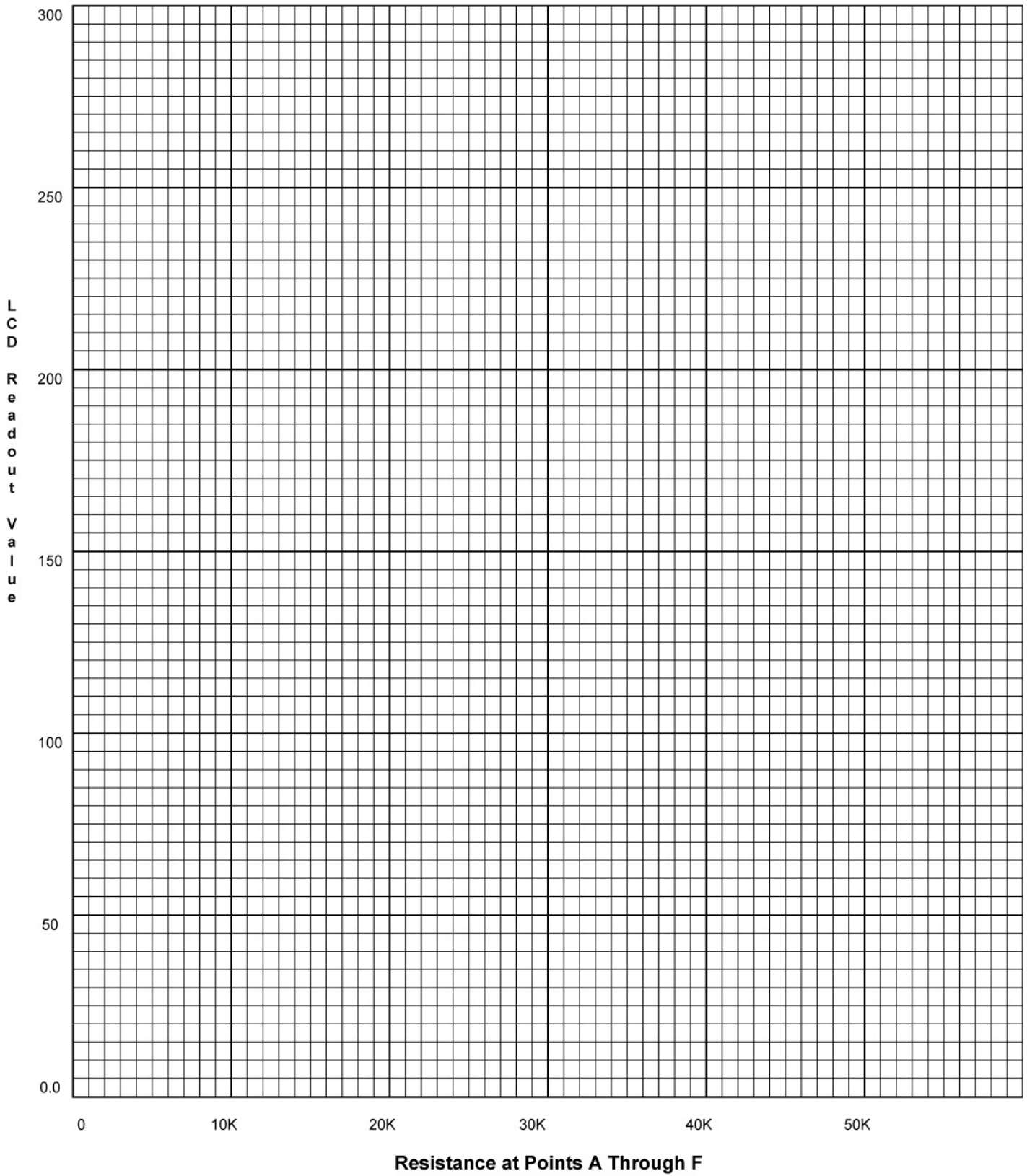
## Cornerstone Electronics Technology and Robotics II LCD Lesson 3 LAB 2 – Five 10K Series Resistors

- **Purpose:** The purpose of this lab is to demonstrate to the student that the PicBasic Pro command **POT** is non-linear in nature.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - PIC 16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 5 - 10K Resistors
  - 4.7 K Resistor
  - 0.1 uF Capacitor
- **Procedure:**
  - Wire the circuit LCD4 below and program the chip with **lcd3.pbp**.
  - Connect pin RB1 to Point A as shown in the schematic below. Set the **POT** command Scale value such that the 255 is the full range for the 5 – 10K resistors. Start by setting the Scale low so the LCD reads a number below 255, and then raise the Scale until you reach a full range reading of 255.
  - Now connect RB1 to Points B through F and plot the results on the accompanying graph. Observe the non-linear nature of the plot.



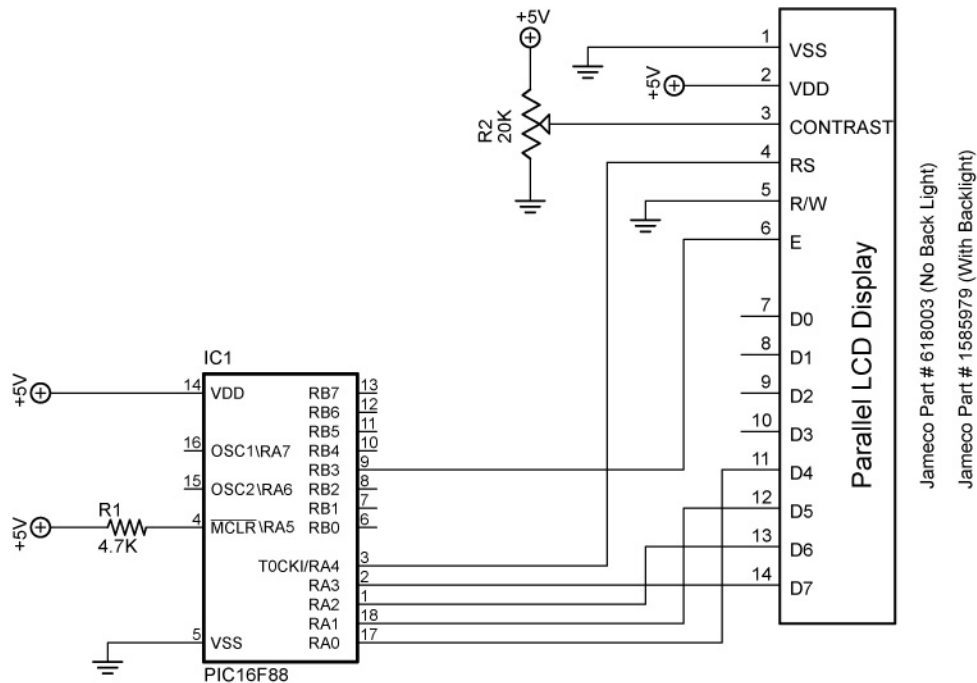


# Resistance vs. LCD Readout for lcd4.pbp



## Cornerstone Electronics Technology and Robotics II LCD Lesson 3 LAB 3 – LED Status on LCD

- **Purpose:** The purpose of this lab is to show the student how to use an LCD to display the state of an output.
- **Apparatus and Materials:**
  - Analog/Digital Trainer or Breadboard w/ 5VDC Supply
  - PIC 16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 1 – 4.7K Resistor
  - 2 – 150 Ohm Resistors
  - 20 K Potentiometer
  - 2 – LEDs
- **Procedure:**
  - Wire the following circuit lcd1.
  - Open **blink1.pbp** from your folder and run the program.



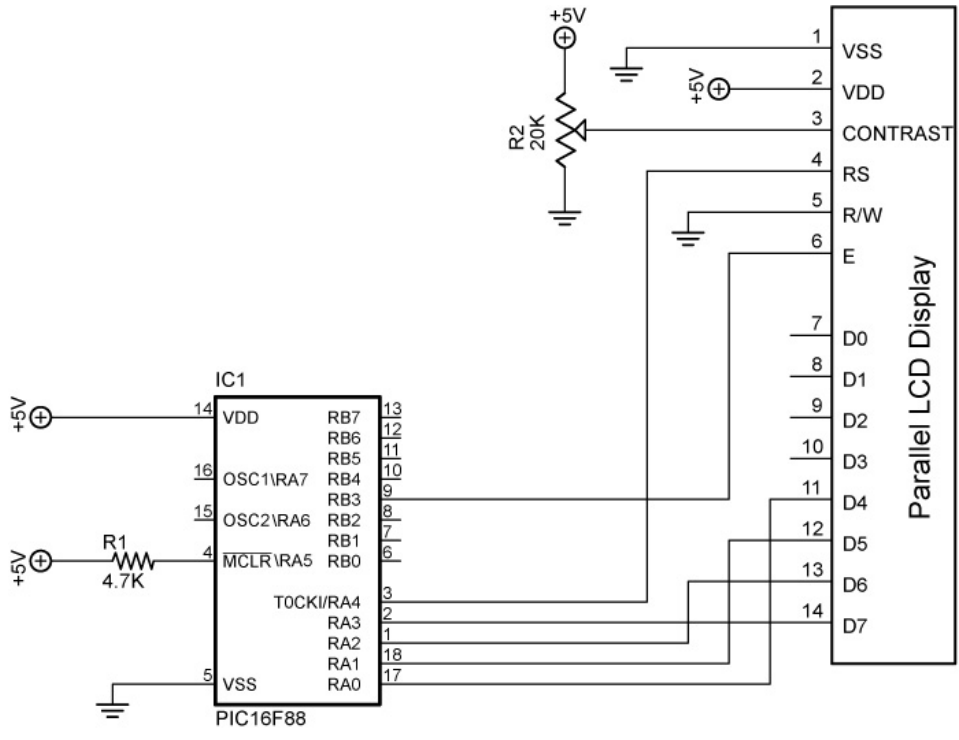
lcd1 and lcd2

- **Challenges:**
  - Connect one LED (LED1) with a 150 ohm resistor to PORTB.1 and program it to blink on and off every second. Display “LED1” and its state (0 or 1) as a variable on the LCD. Save the program as **lcd16.pbp**. Don’t forget to set the proper bits in the TRISB register to outputs as needed.
  - Connect a second LED (LED2) to PORTB.2 and program it to blink opposite LED1. Display “LED1” on the first line and “LED2” on the second line with their respective states as variables. Save the program as **lcd17.pbp**.

## Cornerstone Electronics Technology and Robotics II

### LCD Lesson 3 LAB 4 – Changing LCD Pins on a PIC

- **Purpose:** The purpose of this lab is to acquaint the student with changing the default connections of an LCD to a PIC chip.
  
- **Apparatus and Materials:**
  - Analog/Digital Trainer or Breadboard w/ 5VDC Supply
  - PIC 16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 1 – 4.7K Resistor
  - 1 – 20 K Potentiometer
  
- **Procedure:**
  - Wire the following circuit lcd1.
  - Download **lcd1.pbp** into the PIC16F88
  - Change the LCD pin connections from their default settings to:
    - Data port: PORTB
    - Starting data bit: 0
    - Register Select port: PORTB
    - Register Select bit: 4
    - Enable port: PORTB
    - Enable bit: 5
    - 4 or 8-bit bus: 4-bit bus, the PBP default setting
    - Lines on LCD: 2, the PBP default setting
  - Save the revised **lcd1.pbp** as **lcd18.pbp**
  - Rewire the circuit to reflect the change in LCD pin connections.

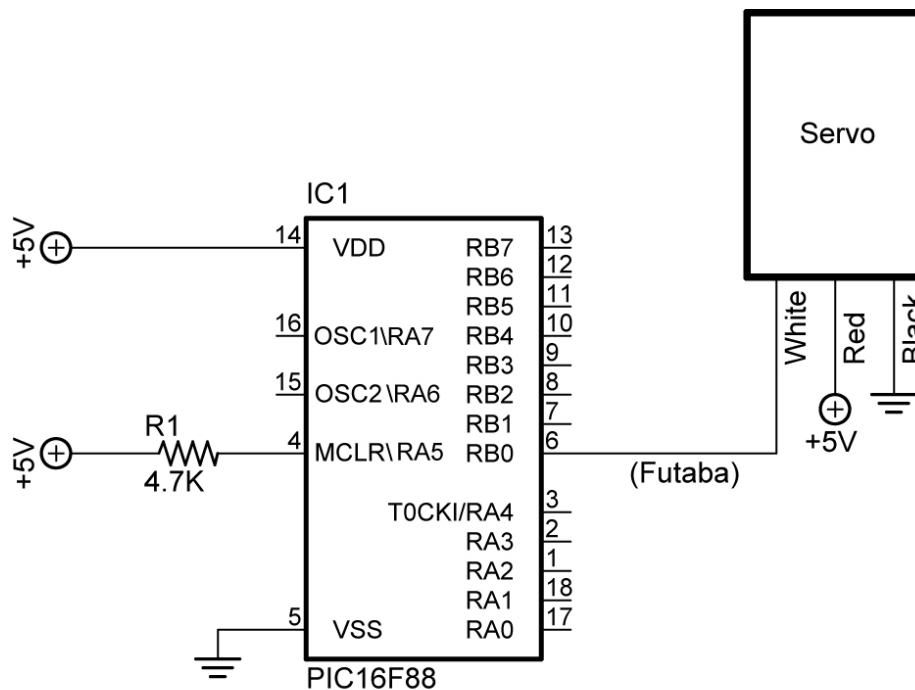


Jameco Part # 618003 (No Back Light)  
 Jameco Part # 1585979 (With Backlight)

lcd1 and lcd2

## Cornerstone Electronics Technology and Robotics II Hacking Servos LAB 1 – servo1.pbp

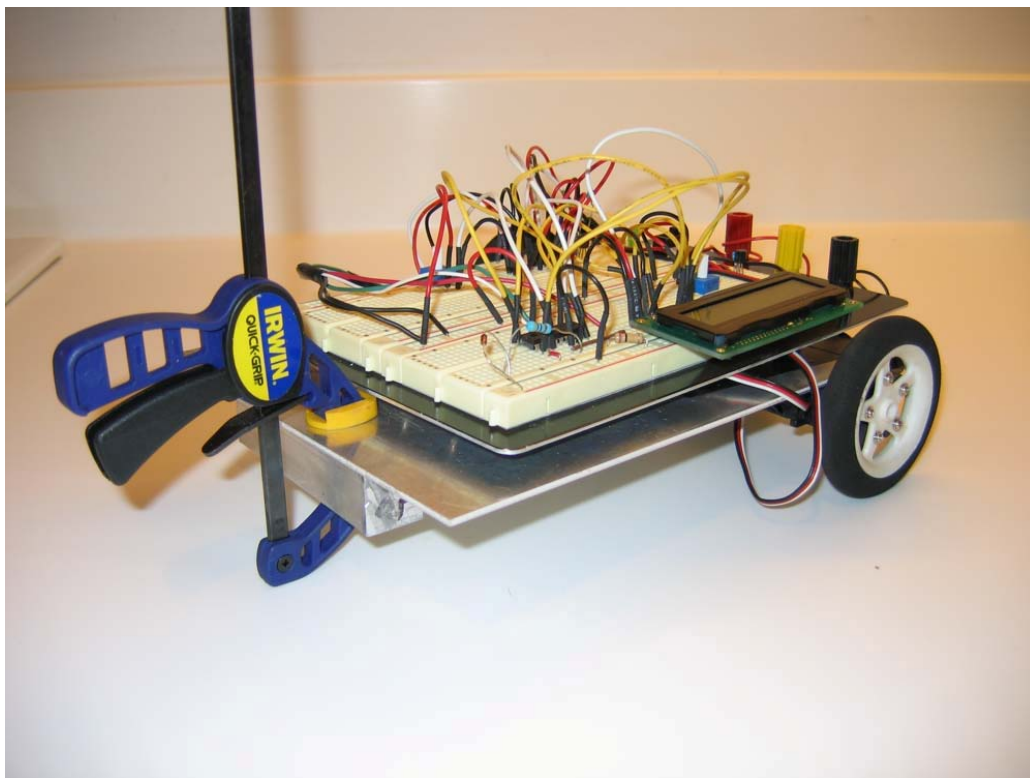
- **Purpose:** The purpose of this lab is to acquaint the student with the operation and programming of a “hacked” hobby servo.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – PIC16F88 Microcontroller
  - 1 – 4.7 K, ½ Watt Resistor
  - 1 – Futaba 3003 Servo
- **Procedure:**
  - Assemble the following circuit on a breadboard. **Make sure that the servo has a separate power supply** (wire another +5V voltage regulator circuit on the breadboard).



### servo1, servo2, and servo3

- Now, open **servo1.pbp** and experiment with the PIC16F88 chip on your breadboard using test values between 100 and 200 for the pulse length. Find the setting around 150 that stops the servo from rotating.

- **Challenge:**
  - Pair up with another student and take two hacked servos to build a car that will navigate through a tape course on the table. Save the program as **servo10.pbp**. Here are the challenge conditions:
    - Dead reckoning may be used to navigate the course.
    - An LCD must display the direction in which the car is traveling, such as forward, right, left, or backward.
    - Create and call up subroutines for each direction of movement including backward. Do not use the word “reverse” since it is a reserved word in PicBasic Pro. See Appendix C in the green PicBasic Pro Compiler Manual by microEngineering Labs, Inc for a listing of all the reserved words in PBP.
    - Electrical tape may be utilized to attach the servos to the car.
    - Use Maxx Products 2.5” light foam wheels (EPW250) mounted on round servo horn as the wheels. Wheels available at Colonial Hobby, Orlando, FL (\$6.69 +/-) or <http://www.maxxprod.com/mpi/mpi-29.html> (\$3.50 +/- plus shipping).
    - Alternate wheels are: Du-Bro 2.5” Micro Sport Wheels – Cat No. 250MS (\$3.70 +/- plus shipping). See: [http://www.shopatron.com/products/productdetail/part\\_number=250MS/101.0.1.1](http://www.shopatron.com/products/productdetail/part_number=250MS/101.0.1.1)
    - Wheel mounting details are available at: [http://cornerstonerobotics.org/wheel\\_making.php](http://cornerstonerobotics.org/wheel_making.php)
    - Use small Irwin Quick Clamp for front pivot.



**Sample Servo Driven Robotic Car**

## Programming Review

### Cornerstone Electronics Technology and Robotics II

- **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.

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

```

**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:**

Format:

**IF** Comparison(s) **THEN** Label

**IF** Comparison(s) **THEN** Statement

Explanation:

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  
high 'pushed, PORTB.0 becomes  
comparison is true, '(+5V) and the 'so  
the program jumps to label  
'led2

- **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. *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 *Items* 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”

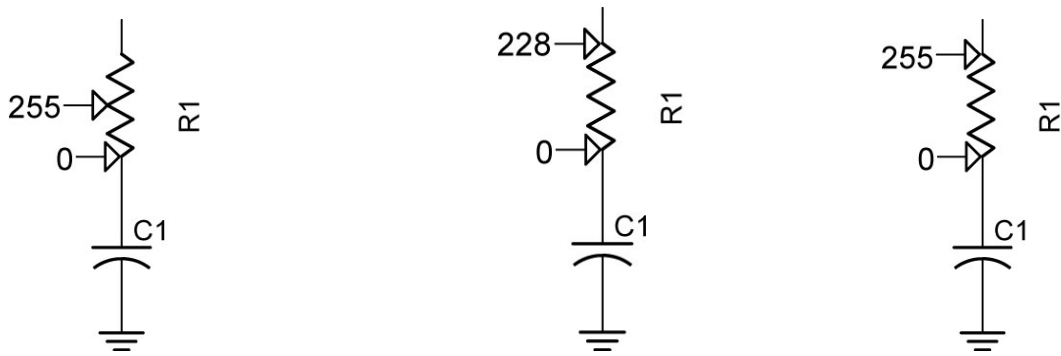
**LCDOUT** B0,#B1                            ‘ Display B0 and decimal ASCII value of B1

- **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:**
  1. Create a folder on your desktop labeled "**prequiz**". Put all of your programs into this folder.
  2. The challenges may be performed in any order.
  3. Set up two +5V voltage regulator circuits first and have the instructor check both circuits before proceeding.
  4. Program and wire a PIC to have an LED flash repeatedly for 3 seconds on and then for 0.7 seconds off.
  5. 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.
  6. 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.
  7. Display your name on an LCD and have it shift continuously to the left, ½ second for each shift.
  8. 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.
  9. Program a servo to go full clockwise then full counterclockwise and then mid-point.
  10. 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.
  11. 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.

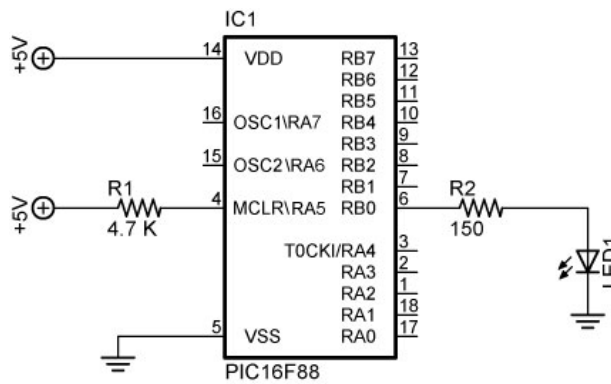
**Cornerstone Electronics Technology and Robotics II**  
**Sink and Source Outputs / Active High and Active Low Inputs**  
**LAB 1 – PIC16F88 as a Source and Sink**

- **Purpose:** The purpose of this lab is to acquaint the student with the PIC microcontroller serving as a source and a sink.
- **Apparatus and Materials:**
  - 1 – Breadboard with 5 V Power Supply
  - 1 – PIC16F88 Microcontroller
  - 1 – 150 Ohm, ½ Watt Resistors
  - 1 – 4.7K Ohm, ½ Watt Resistors
  - 1 – LED

- **Procedure:**

- **PIC as a Source:**

- Wire the circuit below that makes the PIC16F88 function as a source.

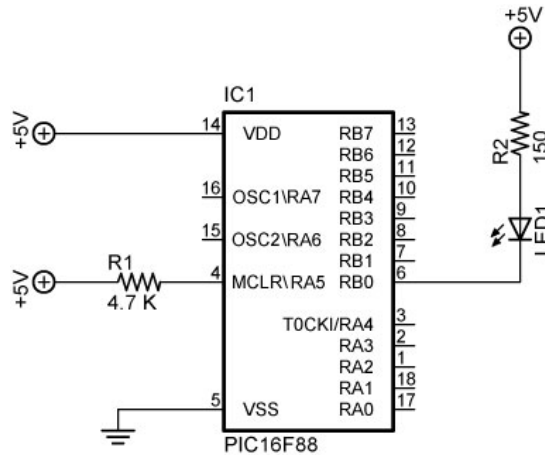


pic\_as\_source

- Program the PIC16F88 with the program **pic\_as\_source.pbp**.
- The PicBasic Pro **HIGH** command on RB0 (**HIGH 0**) supplies +5 volts to pin RB0 so a current of about 13 mA sources from pin RB0 through the resistor R2 and LED to ground. The PIC microcontroller acts as an automatic switch turning the LED on and off; pin RB0 acts as a current source.
- The circuit may turn on and off an LED, a piezo buzzer, or a dc motor driven by a NPN transistor. Program the PIC16F88 (acting as a source) to turn on and off a piezo buzzer from pin RB4. Name the program **pic\_as\_source\_piezo.pbp**
- Do not proceed to the next section until instructed to do so.

- **PIC as a Sink:**

- Wire the circuit below that makes the PIC16F88 function as a sink.

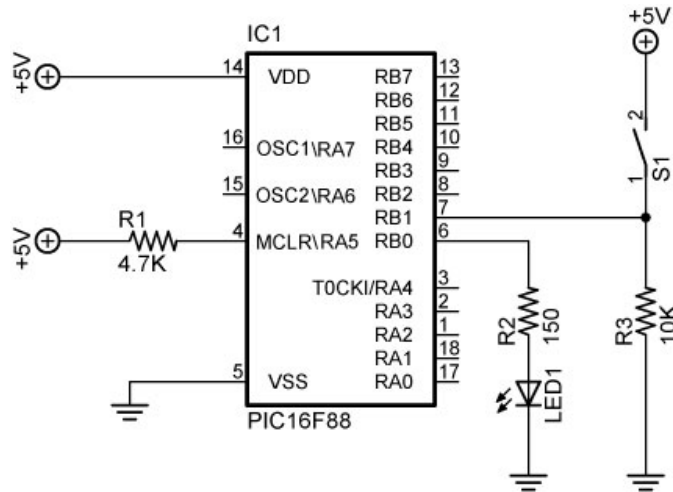


pic\_as\_sink

- Program the PIC16F88 with the program **pic\_as\_sink.pbp**.
- The conventional current goes from the +5 volt bus through the resistor R2 and LED and then sinks into the PIC pin RB0. In this case, when the PIC pin RB0 is programmed to go LOW (**LOW 0**) the LED turns on because the pin “sinks” the current into itself. The PIC microcontroller acts as an automatic switch turning the LED on and off; pin RB0 acts as a current sink.
- Program the PIC16F88 (acting as a source) to turn on and off a piezo buzzer from pin RB1. In the same program, have the PIC act as a sink and turn on and off an LED from pin RB2. Name the program **pic\_as\_source\_and\_sink.pbp**

**Cornerstone Electronics Technology and Robotics II**  
**Sink and Source Outputs / Active High and Active Low Inputs**  
**LAB 2 – Active High and Active Low Inputs**

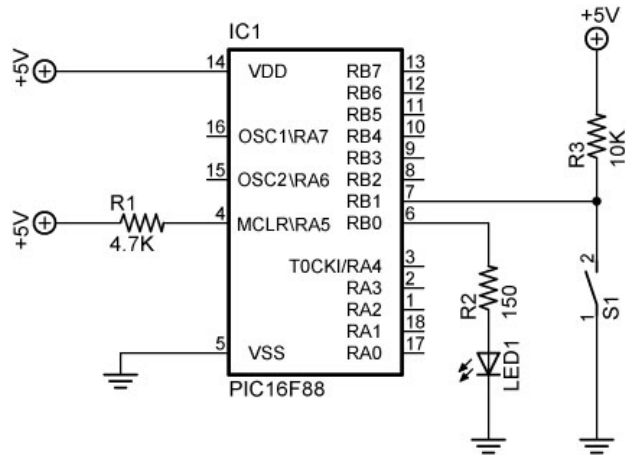
- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro command **IF..THEN** and the principles of active HIGH and active LOW. In addition, the student is challenged to use the logical operator AND.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer
  - 1 – PIC 16F88 Microcontroller
  - 1 – 4.7K Ohm, ½ Watt Resistors
  - 1 – 150 Ohm, ½ Watt Resistors
  - 1 – 10 K Ohm, ½ Watt Resistor
  - 1 – 1K, ½ Watt Resistor
  - 1 – NO Momentary Switch
  - 1 – LEDs
  - This list does not contain the parts needed for the challenges.
- **Procedure:**
  - Open active\_high.pbp and download to your 16F88. Wire your breadboard for the active\_high schematic below. Review the program code and observe the command functions.



active\_high



- Open active\_low.pbp and download to your 16F88. Wire your breadboard for the active\_low schematic below. Review the program code and observe the command functions.



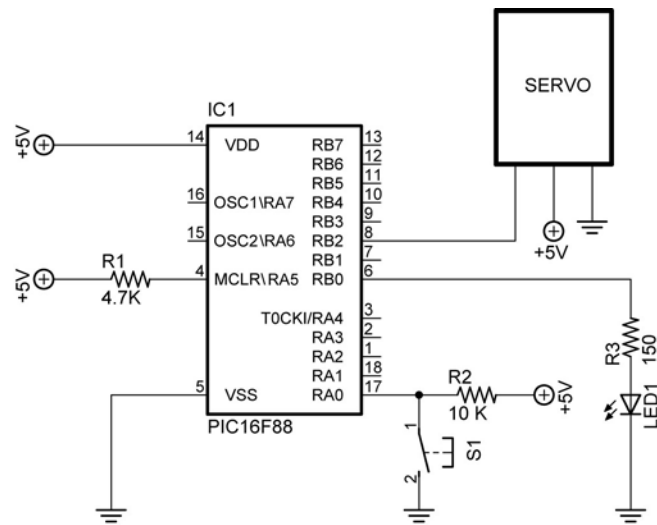
active\_low

- **Challenges:**

- Write a program for the PIC16F88 and wire a circuit such that an LED will light only when two normally open (NO) momentary switches are pressed simultaneously. Use two pins (RB0 and RB1) as the switch inputs. Set one switch input (RB0) to the PIC microcontroller as an active HIGH and the other switch input (RB1) as an active LOW. The output pin (RB2) connected to the LED must be wired as a sink. Save the program as **active\_high\_low\_led1.pbp**.
- Now accomplish the same task using only one input pin (RB0) as an active HIGH to the PIC microcontroller and a 2 – input AND gate (74LS08).

**Cornerstone Electronics Technology and Robotics II**  
**Sink and Source Outputs / Active High and Active Low Inputs**  
**LAB 3 – switch2.pbp**

- **Purpose:** The purpose of this lab is to reinforce the principles of active HIGH and active LOW. In addition, the student is challenged to use the logical operator OR.
- **Apparatus and Materials:**
  - 1 – Breadboard or Analog/Digital Trainer
  - 1 – PIC16F88 Microcontroller
  - 1 – 150 Ohm, ½ Watt Resistors
  - 1 – 10 K Ohm, ½ Watt Resistor
  - 1 – 4.7K Ohm, ½ Watt Resistor
  - 1 – NO Momentary Switch
  - 1 – LED
  - 1 – Servomotor(not “hacked”)
  - This list does not contain the parts needed for the challenges.
- **Procedure:**
  - Open switch2.pbp and download to your chip. Wire switch2 on the digital trainer. Review the program code and observe the command functions.



switch2

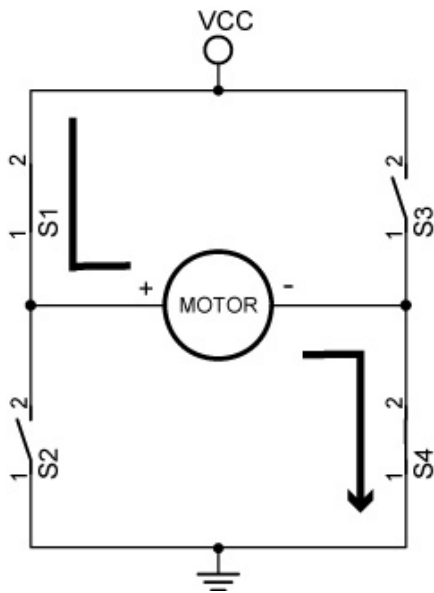
- **Challenges:**
  - Write a program and wire a circuit so that when either one of two normally open (NO) switches is pressed, an LED will light. Use two pins (RB0 and RB1) as the switch inputs. Set one switch input (RB0) to the PIC microcontroller as an active HIGH and the other switch input (RB1) as an active LOW. The output pin (RB2) connected to the LED must be wired as a sink. Save the program as **active\_high\_low\_led2.pbp**.
  - Now accomplish the same task using only one input pin on the PIC microcontroller and a 2 – input OR gate.
  - Write a program and wire the robotic car to make it turn left if the left lever switch is pressed and to turn right if the right lever switch is pressed. If both switches are pressed, make the car travel forward.

## Cornerstone Electronics Technology and Robotics II Motor Control, H-Bridges LAB 1 – Four States of a Motor

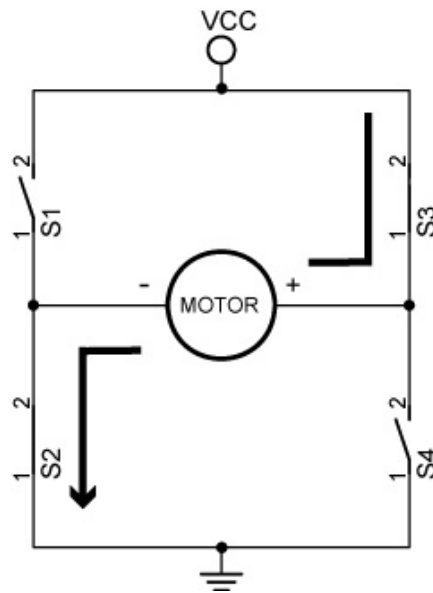
- **Purpose:** The purpose of this lab is to have the student verify the four states of a motor.
- **Apparatus and Materials:**
  - 1 – DC Motor
  - 1 – 9 Volt Battery
- **Procedure:**
  - Connect the motor in the following manner and record the response of the motor:
    - Connect motor terminal A to GND and terminal B to +9 VDC  
Motor Response:\_\_\_\_\_
    - Connect motor terminal A to +9 VDC, and terminal B to GND  
Motor Response:\_\_\_\_\_
    - Disconnect terminals A & B  
Motor Response:\_\_\_\_\_
    - Connect terminal A to terminal B  
Motor Response:\_\_\_\_\_

## Cornerstone Electronics Technology and Robotics II Motor Control, H-Bridges LAB 2 – H-Bridges with SPST Switches

- **Purpose:** The purpose of this lab is to have the student manually verify the basic function of an H-bridge.
- **Apparatus and Materials:**
  - 1 – DC Motor
  - 1 – 9 Volt Battery
  - 4 – SPST Switches
- **Procedure:**
  - Wire the follow circuit.
  - Close the switches in the manner listed in the results table (**Do not close S1 and S2 or S3 and S4 simultaneously – it will create a short circuit**).
  - Document your results.



**Motor Runs Clockwise  
(Switches 1 and 4 Closed)**



**Motor Runs Counter-Clockwise  
(Switches 2 and 3 Closed)**

- **Results:**

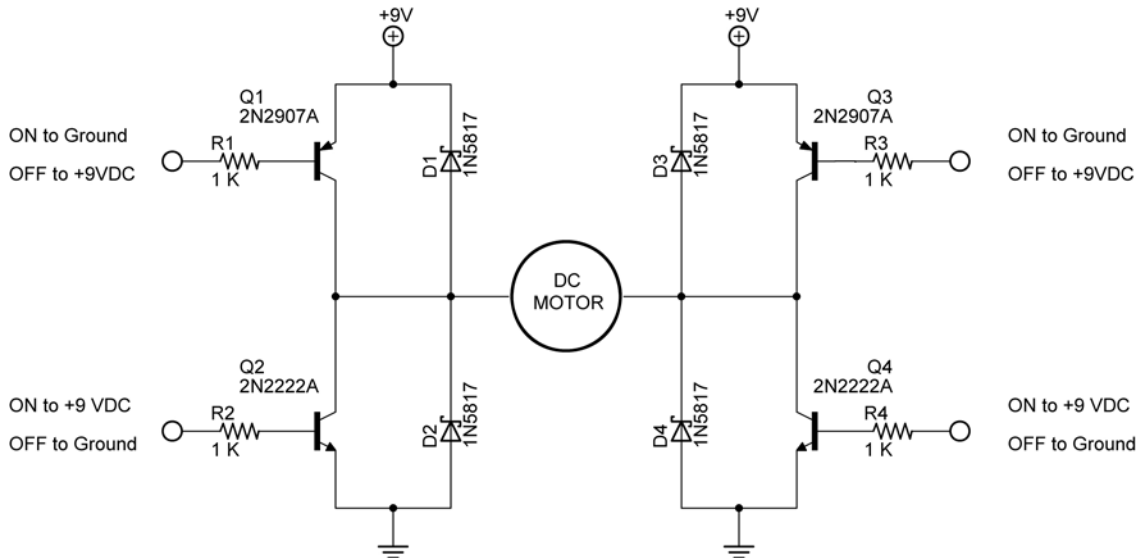
Switch State				Motor Operation
S1	S2	S3	S4	
Closed	Open	Open	Closed	
Open	Closed	Closed	Open	
Closed	Open	Closed	Open	
Open	Closed	Open	Closed	

## Cornerstone Electronics Technology and Robotics II Motor Control, H-Bridges LAB 3 – Bipolar Transistor H-Bridges Motor Driver

- **Purpose:** The purpose of this lab is to have the student setup an electronic H-bridge and to operate it manually.
- **Apparatus and Materials:**
  - 1 – DC Motor
  - 1 – 9 Volt Battery
  - 2 – 2N2907A PNP Transistors
  - 2 – 2N2222A NPN Transistors
  - 4 – 1N5817 Diodes
- **Procedure:**
  - **As with switches, do not short circuit through the transistors (Q1 & Q2 or Q3 & Q4).**
  - Wire the following circuit the robotic car breadboard.
  - Connect the inputs to the transistor bases according to the following table and record the action of the motor:

Transistor Connections				
Q1	Q2	Q3	Q4	Motor Operation
+9V	+9V	GND	GND	
GND	GND	+9V	+9V	
GND	Disconnected	GND	Disconnected	
Disconnected	+9V	Disconnected	+9V	
Disconnected	Disconnected	Disconnected	Disconnected	

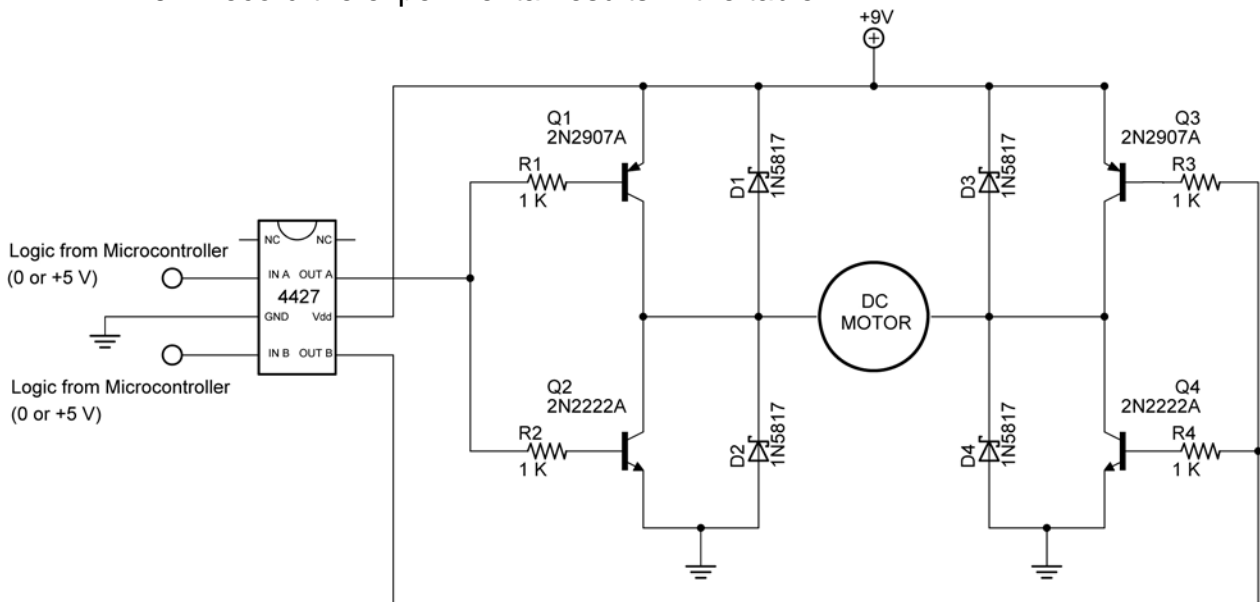
### CHECK LINES 3 & 4 FOR +9 AND GND RESPECTIVELY



- Tie the inputs of Q1 and Q2 together and also connect the inputs of Q3 and Q4 together. What purpose do these connections serve?

## Cornerstone Electronics Technology and Robotics II Motor Control, H-Bridges LAB 4 – 4427 Interface IC

- **Purpose:** The purpose of this lab is to have the student insert a 4427 interface IC to simplify the control of an H-bridge.
- **Apparatus and Materials:**
  - 2 – DC Motors
  - 1 – 4427 Interface IC
  - 2 – 2N2907A PNP Transistors
  - 2 – 2N2222A NPN Transistors
  - 4 – 1N5817 Diodes
- **Procedure:**
  - Refer to the 4427 Interface IC H-Bridge Output Results Table below. Given the inputs for A and B, fill in the states of Q1-Q4 (On or off), then predict the action of the motor (Clockwise, counter-clockwise, braking, or coasting).
  - Now wire the 4427 interface IC circuit below.
  - Apply a HIGH, Low, or Disconnect to the inputs A and B of the 4427 IC.
  - Record the experimental results in the table.



**4427 Interface IC Circuit**

• **Results:**

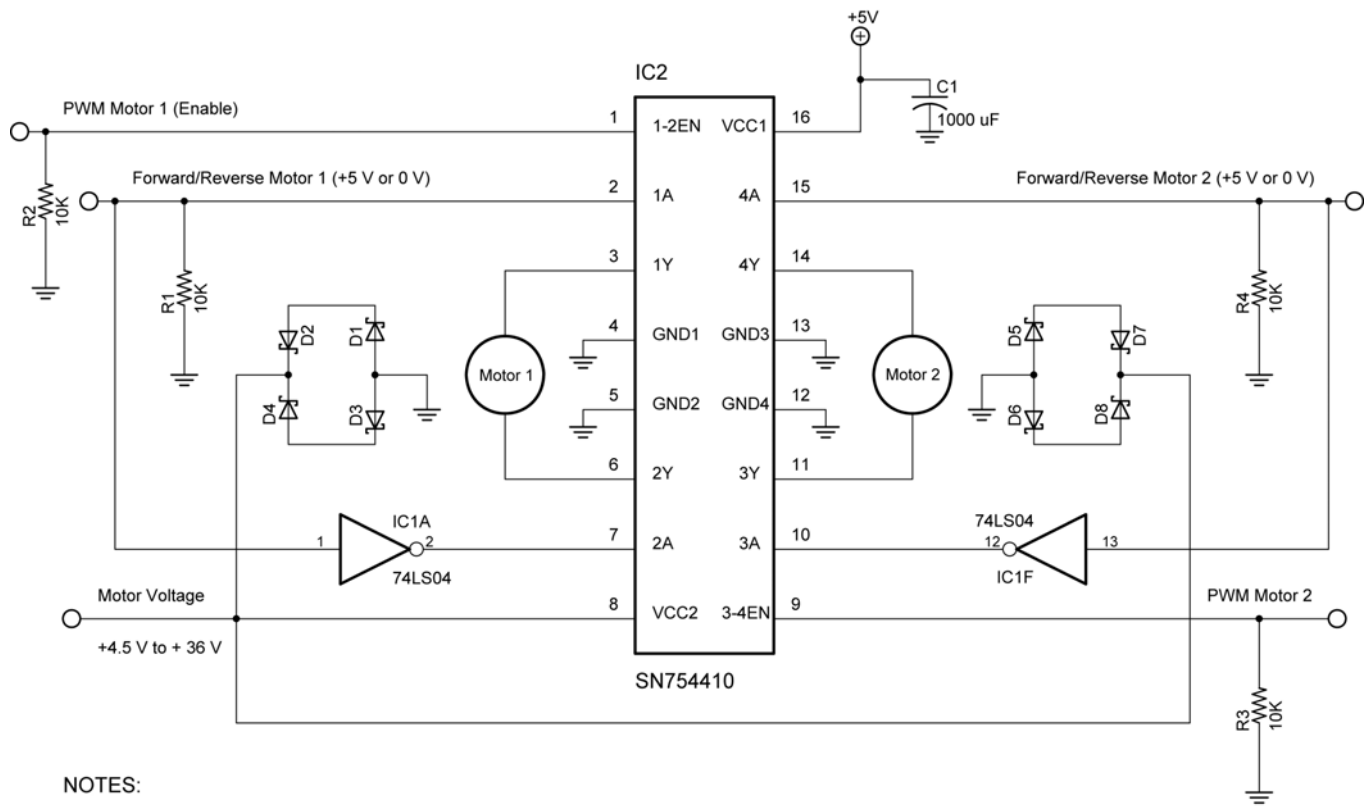
4427 Interface IC H-Bridge Output Results							
Input A	Input B	Q1	Q2	Q3	Q4	Predicted Results	Experimental Results
HIGH	HIGH						
HIGH	LOW						
LOW	HIGH						
LOW	LOW						

## Cornerstone Electronics Technology and Robotics II

### Motor Control, H-Bridges LAB 5 – SN754410 H-Bridges Motor Driver

- **Purpose:** The purpose of this lab is to acquaint the student with the operation of a single chip motor driver – SN754410 by Texas Instrument.
- **Discussion:**
  - PWM has yet to be covered so the PWM ports are either set HIGH (100% duty cycle) or LOW (0% duty cycle) See the lesson on PWM to adjust values between 100% and 0%.
- **Apparatus and Materials:**
  - 2 – Gearhead DC Motors, Jameco #155855
  - 1 – SN754410 Quadruple Half-H Driver, Pololu #0024
  - 1 – 74LS04 Hex-Inverter
  - 8 – 1N5817 Diodes
  - 4 – 10K Resistors
  - 1 – 1000 uF Capacitor
- **Procedure:**
  - Wire the circuit below and make the input connections as follows:
    - Motor + Voltage to +9V
    - PWM Motor 1 and 2 to 0V or +5V
    - Forward/Reverse Motor 1 and 2 to 0V or +5V
    - Complete the table below.

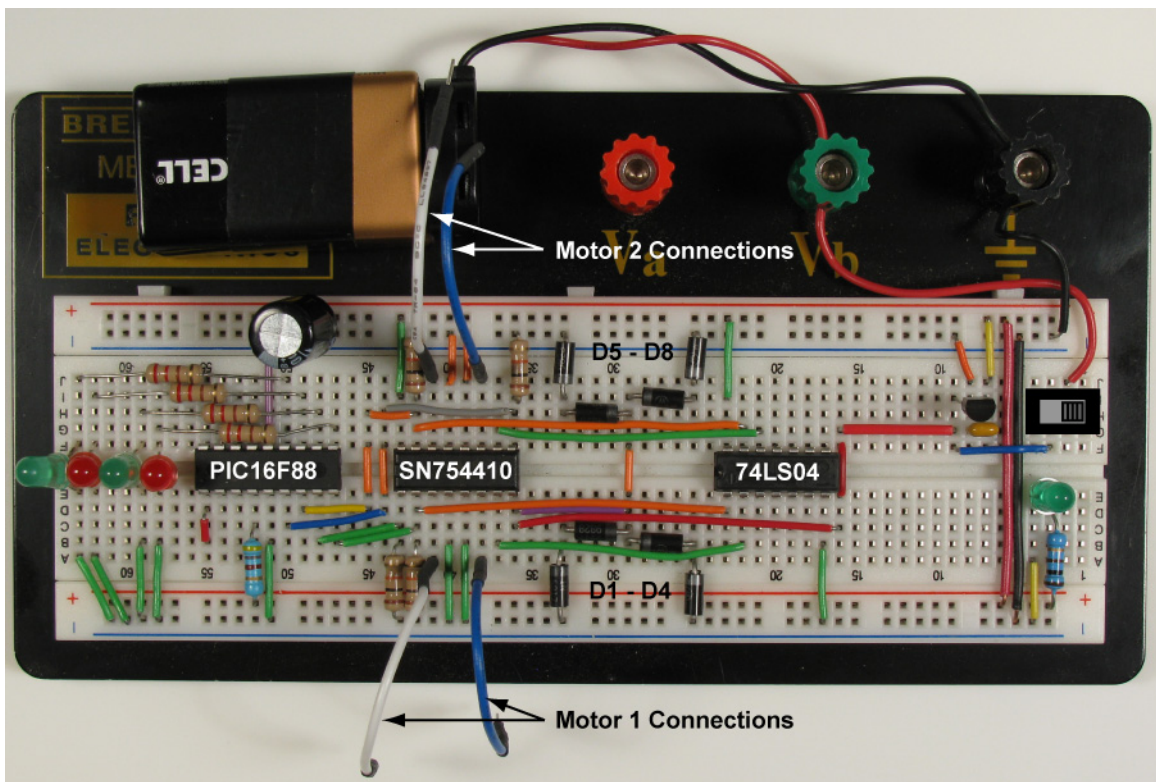




NOTES:

1. D1 - D8 1N5817
2. 74LS04 Pin 7 GND  
74LS04 Pin 14 Vcc +5V
3. SN754410 GNDS Also Act As  
Heatsinks, Ground Separately

Texas Instrument SN754410 H-Bridge Motor Driver



Breadboard Layout Includes PIC16F88 (Lab 6)

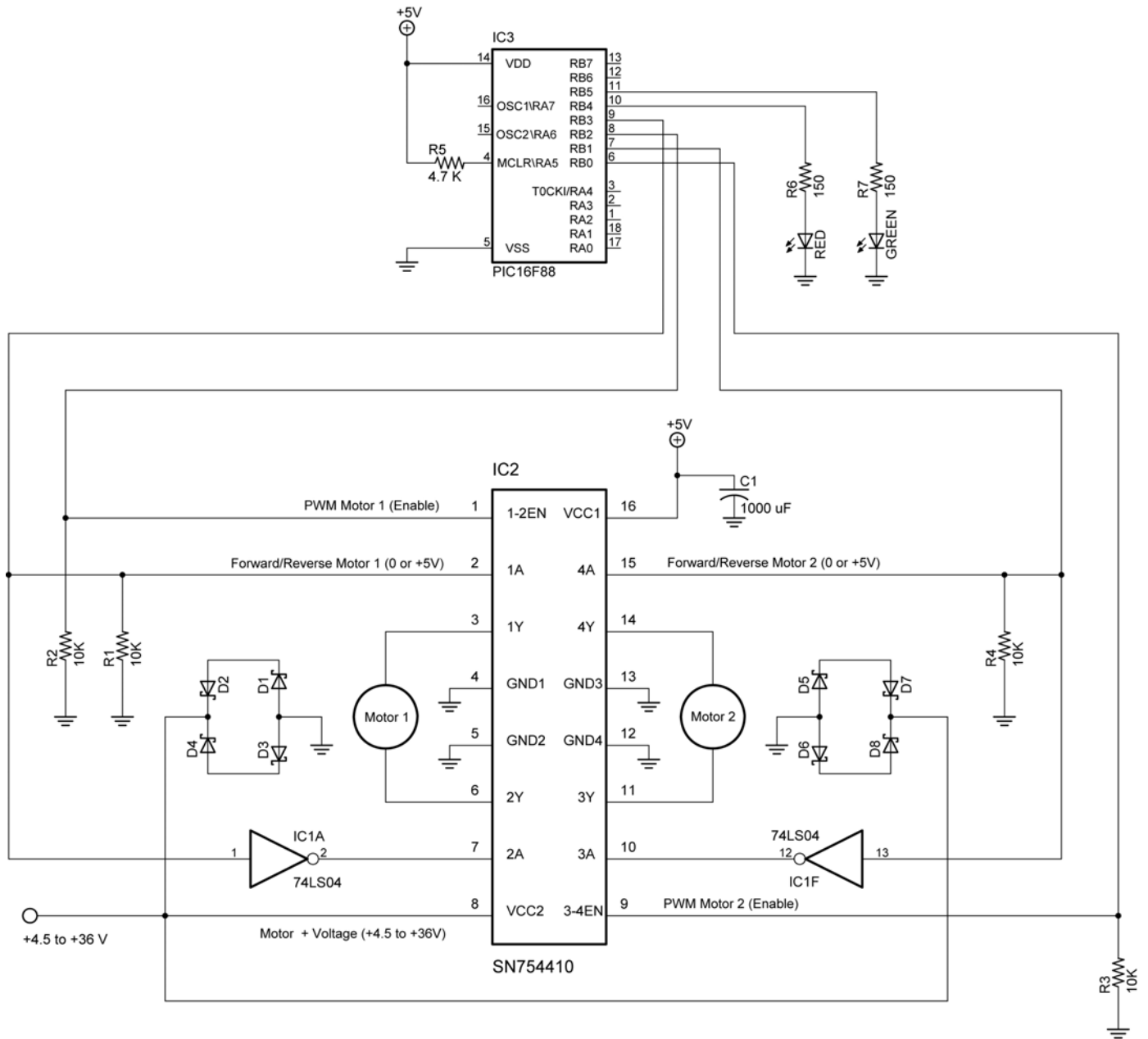
- Results:

<b>SN754410 H-Bridge Motor Driver Results</b>		
<b>PWM Motor 1</b>	<b>F/R Motor 1</b>	<b>Motor Operation</b>
HIGH	HIGH	
HIGH	LOW	
LOW	HIGH	
LOW	LOW	

<b>SN754410 H-Bridge Motor Driver Results</b>		
<b>PWM Motor 2</b>	<b>F/R Motor 2</b>	<b>Motor Operation</b>
HIGH	HIGH	
HIGH	LOW	
LOW	HIGH	
LOW	LOW	

**Cornerstone Electronics Technology and Robotics II**  
**Motor Control, H-Bridges LAB 6 – PIC16F88 Driving the SN754410 H-**  
**Bridges Motor Driver**

- **Purpose:** The purpose of this lab is to acquaint the student with using a PIC microcontroller to drive a single chip motor driver – SN754410 by Texas Instrument.
  
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 2 – Gearhead DC Motors, Jameco #155855
  - 1 – SN754410 Quadruple Half-H Driver, Pololu #0024
  - 1 – 74LS04 Hex-Inverter
  - 1 – PIC16F88
  - 8 – 1N5817 Diodes
  - 1 – 4.7K Resistor
  - 4 – 10K Resistors
  - 2 – 150 Ohm Resistors
  - 2 – LEDs
  - 1 – 1000 uF Capacitor
  
- **Procedure:**
  - Wire the following circuit on the robotic car breadboard. See the photo in Lab 5 for one possible layout.
  - Make sure that the motor is from a power supply separate from the PIC16F88.
  - Program the PIC16F88 with **h\_bridge\_sn754410\_1.pbp**.



NOTES:

1. D1 - D8 1N5817
2. 74LS04 Pin 7 GND  
74LS04 Pin 14 Vcc +5V
3. SN754410 GNDS Also Act As  
Heatsinks, Ground Separately

Texas Instrument SN754410 H-Bridge Motor Driver

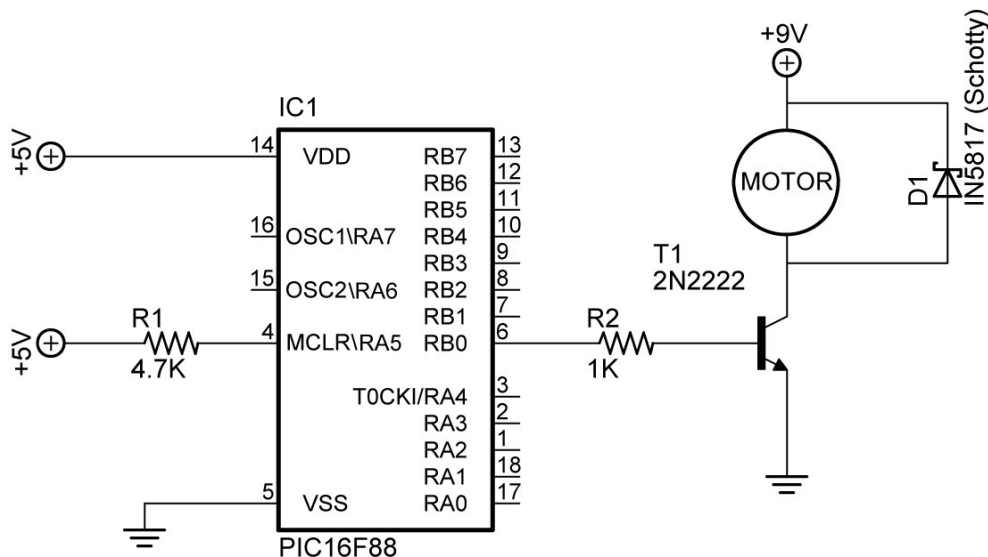
• **Challenge:**

- The robotic car must navigate the given course.
- Dead reckoning may be used to navigate the course.
- An LCD must display which direction the car is traveling, such as, forward, right, left, or backup.
- Create and call up subroutines for each direction of movement including backup. Do not use the word "reverse" since it is a reserved word in PicBasic Pro.
- Save the new program as **sn754410\_navigate.pbp**.

## Cornerstone Electronics Technology and Robotics II

### Motor Control PWM LAB 1 – pwm1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student the PicBasic Pro command **PWM** and how to make basic connections of a motor to a PIC programmed with **PWM**.
- **Apparatus and Materials:**
  - 1 – Breadboard or Robotic Car Platform
  - 1 – PIC16F88
  - 1 – 1K Resistors
  - 1 – 4.7K Resistor
  - 1 – 2N2222 NPN Transistor
  - 1 – 1N5817 Diode
  - 1 – DC Motor
- **Procedure:**
  - Wire the circuit below on your breadboard:
  - Open **pwm1.pbp** from your folder and download to the PIC. The **pwm1.pbp** program changes the motor speed to three levels.
  - Run the program and observe the motor speed changes
  - Now save **pwm1.pbp** as **pwm10.pbp**.
  - Experiment by changing the values of Duty first, and then Cycle in the **PWM** command.

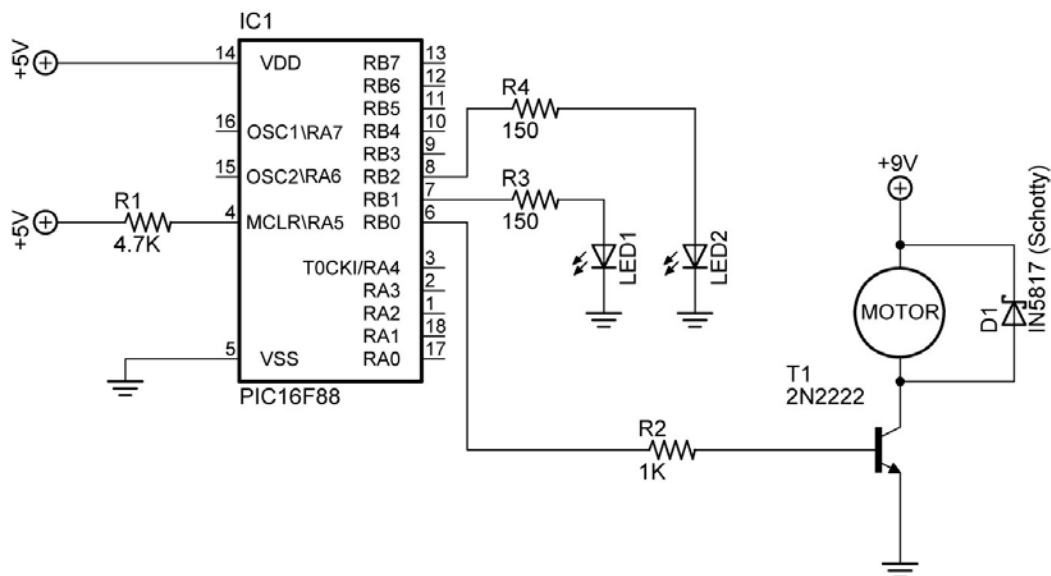


**Circuit for pwm1.pbp**

## Cornerstone Electronics Technology and Robotics II

### Motor Control PWM LAB 2 – hpwm1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the command **HPWM** and how it runs in the background while the program executes other commands.
- **Apparatus and Materials:**
  - 1 – Breadboard or Robotic Car Platform
  - 1 – PIC16F88
  - 1 – 1K Resistors
  - 1 – 4.7K Resistor
  - 1 – 2N2222A NPN Transistor
  - 1 – 1N5817 Diode
  - 1 – DC Motor – The Class Used the Jameco #155855 Gearhead 72 RPM Motor
- **Procedure:**
  - Wire the circuit as shown below.
  - Load **hpwm1.pbp** into the PIC16F88. The **hpwm1.pbp** program runs the HPWM command in the background while executing other PicBasic Pro commands.
  - Take note that the motor continues to run as the program executes the other program commands.

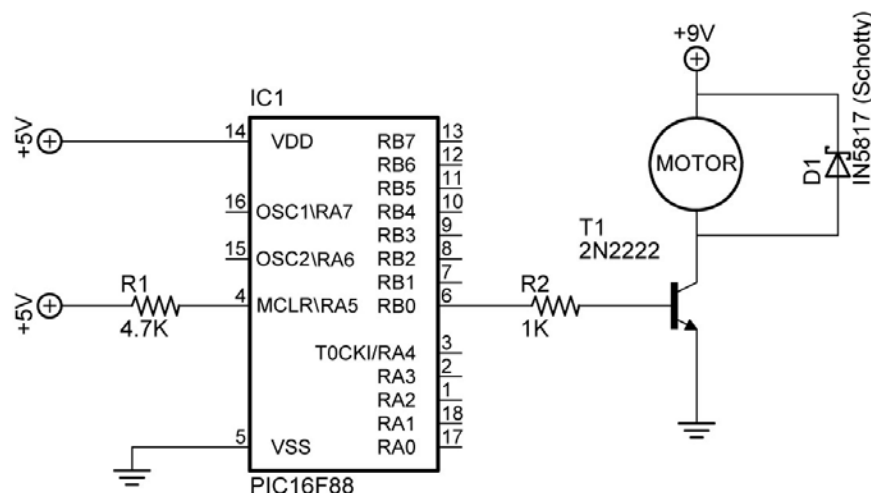


**Circuit for hpwm1.pbp**

- Now save **hpwm10.pbp** as **hpwm10.pbp** and change the Duty cycle values.
- Open **hpwm2.pbp** and load into the PIC16F88. This program changes the value of the Duty cycle from 90 to 255 and then to 0. The motor speed responds accordingly.

## Cornerstone Electronics Technology and Robotics II Motor Control PWM LAB 3 – PWM Calibration

- **Purpose:** The purpose of this lab is to acquaint the student with the process of calibrating PWM-driven motor rotational speeds.
- **Apparatus and Materials:**
  - 1 – RPM Meter
  - 1 – Robotic Car Platform
  - 1 – PIC16F88
  - 1 – 1K Resistors
  - 1 – 4.7K Resistor
  - 1 – 2N2222A NPN Transistor
  - 1 – 1N5817 Diode
  - 2 – DC Gearhead Motors – Jameco #155855
- **Procedure:**
  - Use the same circuit that was used in Lab 1 for Lab 3.



**Circuit for PWM and HPWM Calibration**

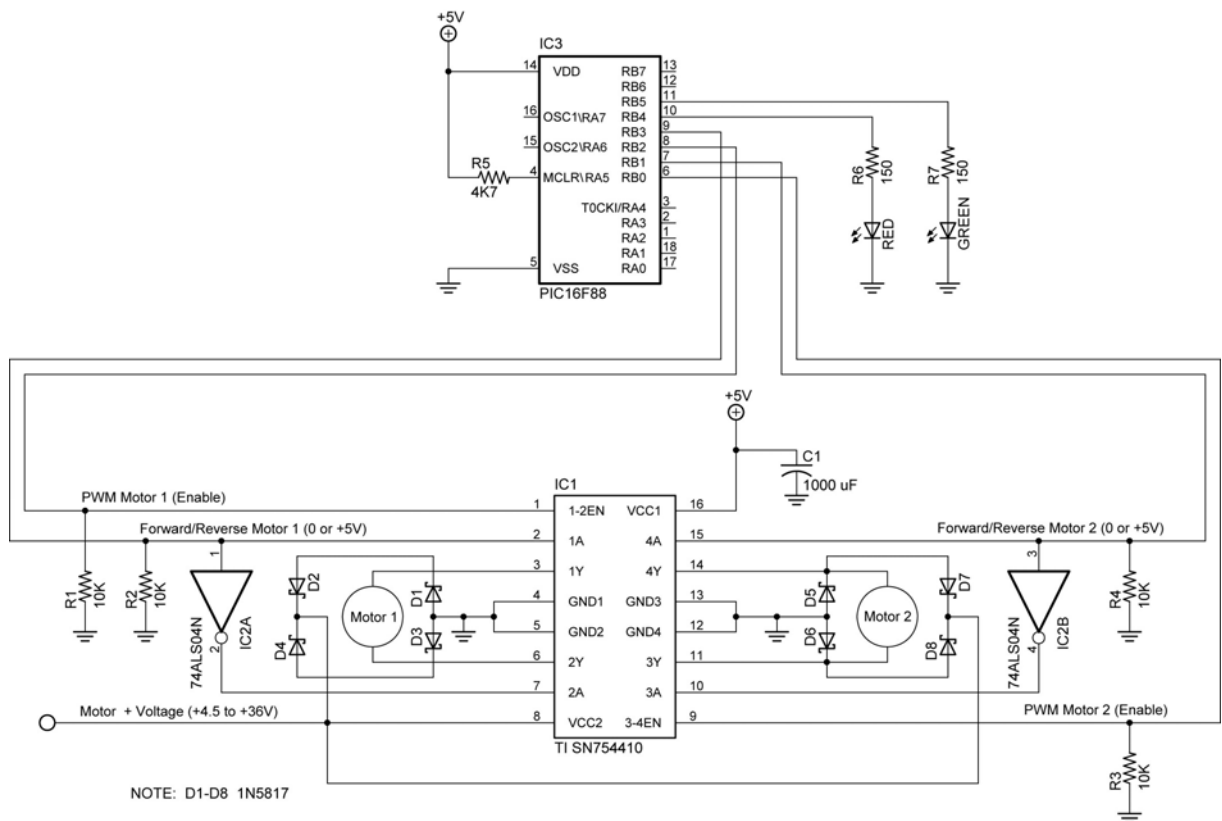
- Motor 1 Calibration with **PWM** Command:
  - Label the motors Motor 1 and 2 and connect the power supply positive lead to the + terminal on Motor 1 and the negative lead to the other terminal on Motor 1. This will be considered forward for Motor 1. Make sure the car does in fact travel forward, not in reverse.
  - Using **pwm10.pbp**, empirically determine the smallest Duty value before Motor 1 stops rotating. This number will be used as the smallest Duty value in the remaining discussion.

- Calculate Input Values:
  - Divide the range from the smallest Duty value to 255 into 9 almost equal parts. For example, if the smallest Duty value is 140, subtract 140 from 255 to yield 115. Divide 115 by 9 to get 12.77, or rounded, 13. Start the input values at 140. Write each of these input values down. Now add 13 to 140 to get the second input value, 153. To calculate the remaining input values, continue adding 13 to the previous total until you arrive at approximately 255. In this example, the input values would be: 140, 153, 166, 179, 192, 205, 218, 231, 244, and 257.
- Setting the Calibration Graph Scales:
  - Set the horizontal scale from your smallest Duty value (at the graph origin) to 255. Adjust the placement of the 255 value so that the lines on the scale make sense when plotting the other input values.
  - Program the PIC16F88 with the Duty value 255 so you can determine the highest rpm that you will measure. Set the vertical scale using this rpm value as the maximum value. Again, adjust this maximum value so the lines on the scale make sense.
- For each input value, measure the rpm of Motor 1 and plot the coordinates on the calibration graph. Label this plot as Motor 1 Forward.
- Switch the polarity of the motor leads and repeat the whole calibration process for Motor 1 in reverse. Plot the results on the same calibration graph as Motor 1 Forward.
- Motor 2 Calibration with **HPWM** Command:
  - Connect the positive lead to the - terminal and the negative lead to the + terminal on Motor 2. This will be considered forward for Motor 2.
  - Repeat the whole calibration process for Motor 2 except use **hpwm3.pbp**. Plot the results on a new calibration graph and label this plot as Motor 2 Forward.
  - Switch the polarity of the motor leads and repeat the whole calibration process for Motor 2 in reverse. Plot the results on the same calibration graph as Motor 2 Forward.
  - Save these plots for motor calibration in the coming weeks.



## Cornerstone Electronics Technology and Robotics II Motor Control PWM LAB 4 – Using PWM for Speed Control with the SN754410

- **Purpose:** The purpose of this lab is to have the student use PWM to change the speed of their robotic car.
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 2 – Gearhead DC Motors, Jameco #155855
  - 1 – SN754410 Quadruple Half-H Driver, Pololu #0024
  - 1 – 74LS04 Hex-Inverter
  - 1 – PIC16F88
  - 8 – 1N5817 Diodes
  - 1 – 4.7K Resistor
  - 4 – 10K Resistors
  - 2 – 150 Ohm Resistors
  - 2 - LEDs
  - 1 – 1000 uF Capacitor
- **Procedure:**
  - Wire the following circuit on the robotic car breadboard. This is the identical circuit that was used in the H-Bridge lesson.
  - Program the PIC16F88 with **pwm\_hpwm\_sn754410.pbp**.

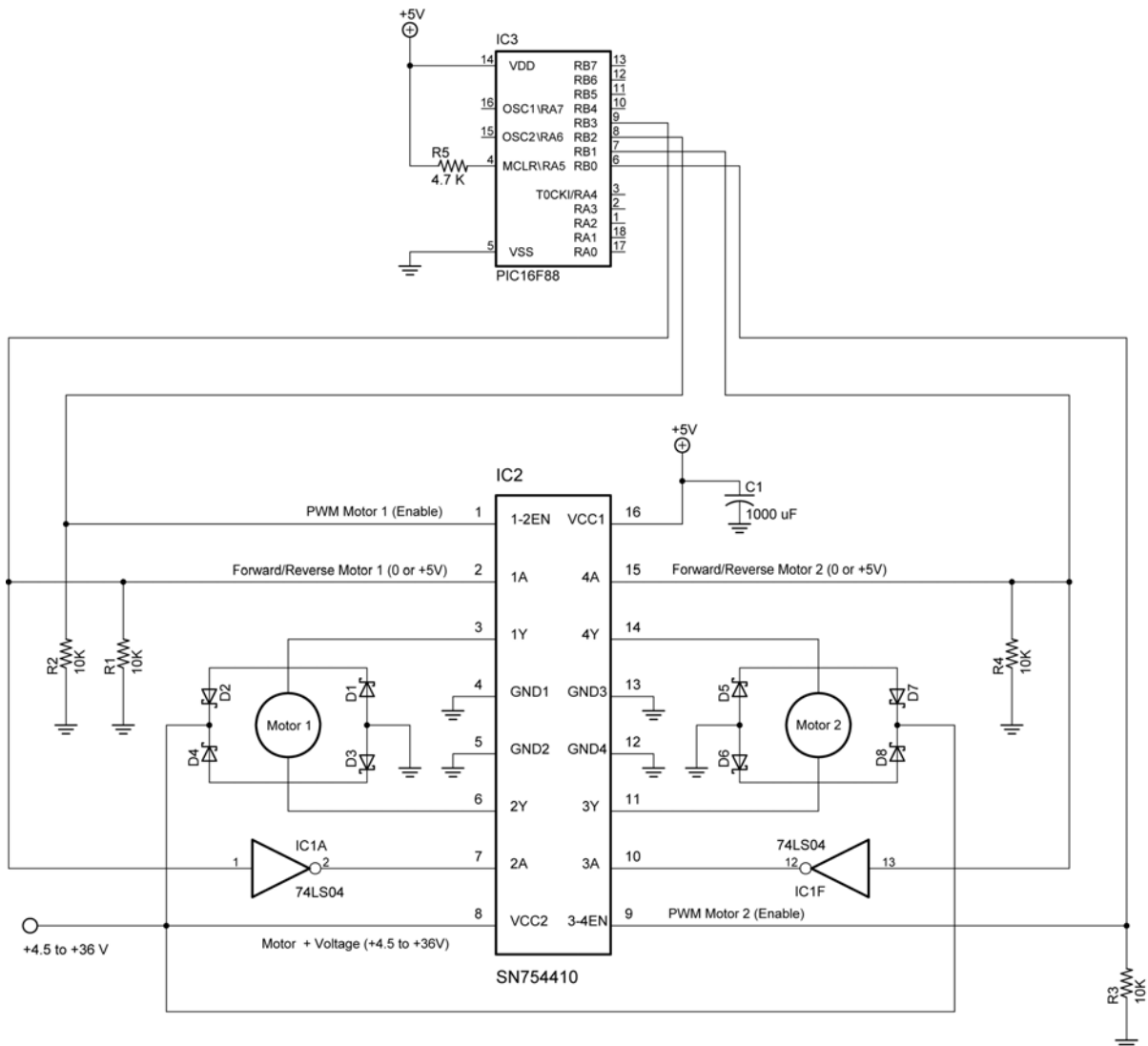


- **Challenge:**
  - Using the calibration data just collected, program the robotic car to travel forward along the straight taped line.
  - Use the calibration graphs to make the robotic car travel at  $\frac{1}{2}$  its maximum speed.
  - Program the robotic car to follow the straight taped line in reverse.

**Sonar Car 1 – Servo Positioning**  
**LAB 1 – Set Servo into Starting Position**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to set the servo on the sonar car into the starting position.
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics below.
- **Procedure:**
  - You can import **servo1.pbp** to provide the basic program code. You will need to make modifications to this program. See: <http://cornerstonerobotics.org/code/servo1.pbp>.
  - Save the imported program as **sonar\_car1.pbp**. You will keep adding to this program in the coming weeks until your sonar car is fully functional.
  - Give the output pin PORTB.6 the name servo\_pin. It can be assigned a name using the **VAR** command. This pin will send the program signal to the servo.
  - Using the **PULSOUT** command, set the Period to 70. Remember the format for **PULSOUT** is: **PULSOUT Pin, Period**
  - Shorten the time to move the servo by reducing the **FOR..NEXT** loop to 20 loops.

# Sonar Car Circuitry 1

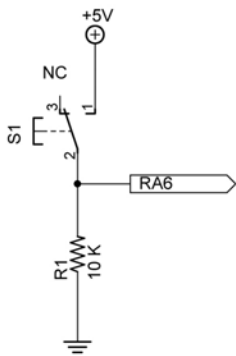


**NOTES:**

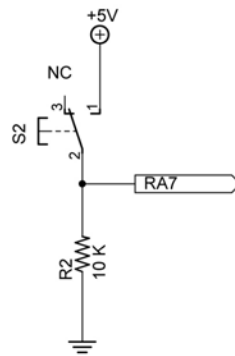
1. D1 - D8 1N5817
2. 74LS04 Pin 7 GND  
74LS04 Pin 14 Vcc +5V
3. SN754410 GNDS Also Act As  
Heatsinks, Ground Separately

Texas Instrument SN754410 H-Bridge Motor Driver

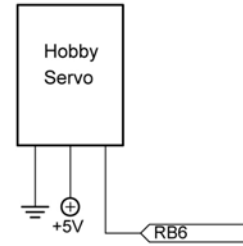
# Sonar Car Circuitry 2



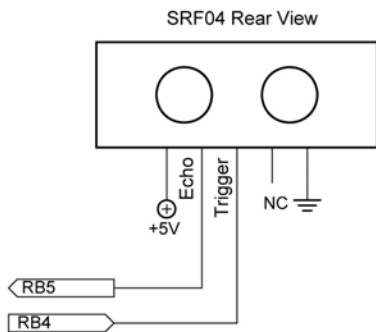
Left Bumper Switch



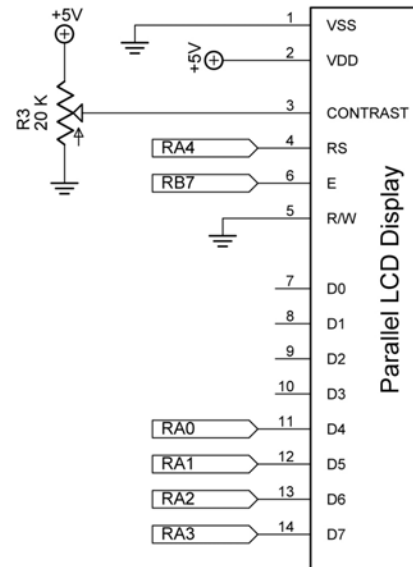
Right Bumper Switch



Servo



Ultrasonic Range Finder



LCD

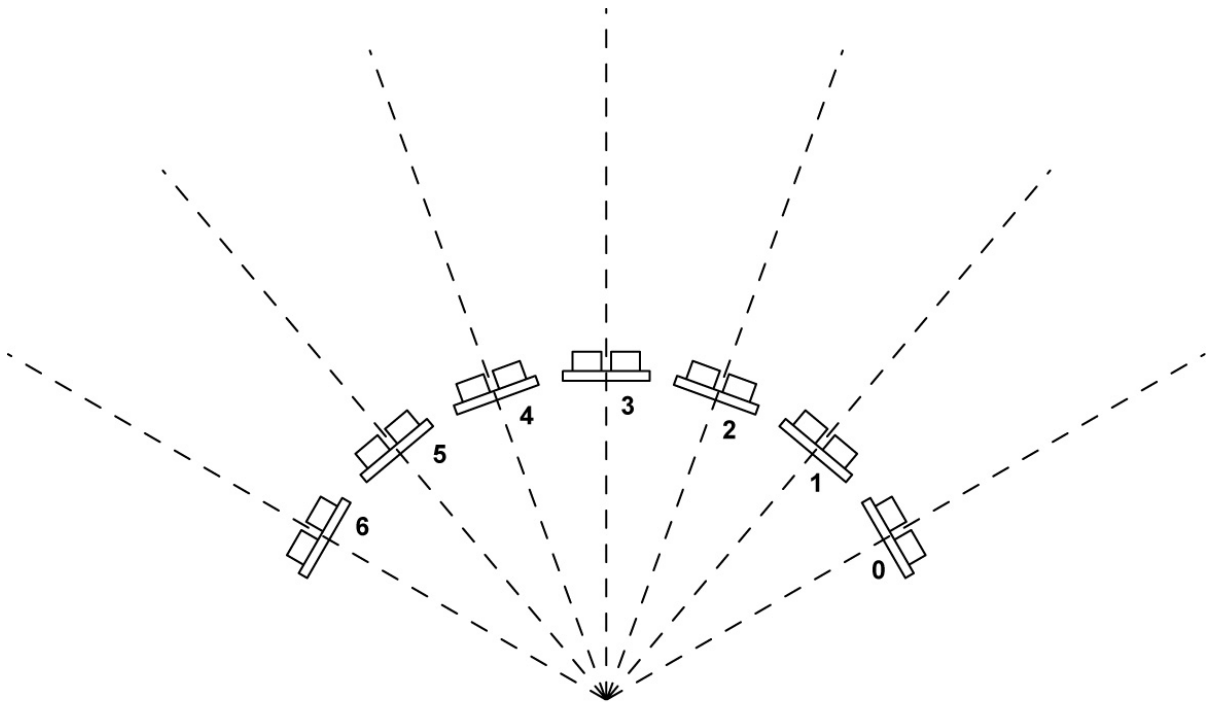
Jameco Part # 618003 (No Back Light)

**Sonar Car 1 – Servo Positioning**  
**LAB 2 – Pan Servo across the Front on the Sonar Car**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to make the servo pan across the front of the sonar car in six discrete steps.
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics in Lab 1.
- **Procedure:**
  - Declare the following variables

```
p0    VAR  BYTE    ' BYTE to store servo pulse period
c0    VAR  WORD    ' WORD for counter
```

- Start the servo at the position generated by Period = 70 (see Lab 1) and then rotate through to Period = 208 in 6 discrete steps. Hint: Nest two **FOR..NEXT** loops.
- Shorten the time to move the servo to each new position by reducing the **FOR..NEXT** loop to 15 loops.
- Set up a loop to make the servo return to the starting position and then pan across the front of the car again. The



**7 Servo Positions with Ultra-sonic Range Finder**

- Compare your program to **sonar\_car\_a.pbp**. See: [http://cornerstonerobotics.org/code/sonar\\_car\\_a.pbp](http://cornerstonerobotics.org/code/sonar_car_a.pbp)

## Cornerstone Electronics Technology and Robotics II Switch Sensor LAB 1 – Wall Alignment

- **Purpose:** The purpose of this lab is to acquaint the student with the ability of robots equipped with switches to align themselves to a wall.
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 2 – Lever Micro-switches
- **Procedure:**
  - For the student to determine
- **Challenge:**
  - Install two lever switches on your robotic car platform and program the PIC for the car to square up with a wall. Keep your PAUSE command periods short, otherwise when a switch is activated, the program may be executing another command with a lengthy PAUSE and not recognize the activated switch.

## Cornerstone Electronics Technology and Robotics II Switch Sensor LAB 2 - Maze Navigation

- **Purpose:** The purpose of this lab is to acquaint the student with the use of a switch as a robotic touch sensor and how to use the information returned to the robot for the robot reaction. In particular, the student uses a switch (or switches) sensor mounted on their robotic car to navigate a maze.
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 1 or 2 – Lever Micro-Switch(es)
- **Procedure:**
  - For the student to determine
- **Challenge:**
  - Install lever switch or switches on your robotic car platform and program the PIC for the car to navigate through the given maze. Again, keep your PAUSE command periods short, otherwise when a switch is activated, the program may be executing another command with a lengthy PAUSE and not recognize the activated switch.

## Cornerstone Electronics Technology and Robotics II Switch Sensor LAB 3 – Collision Avoidance

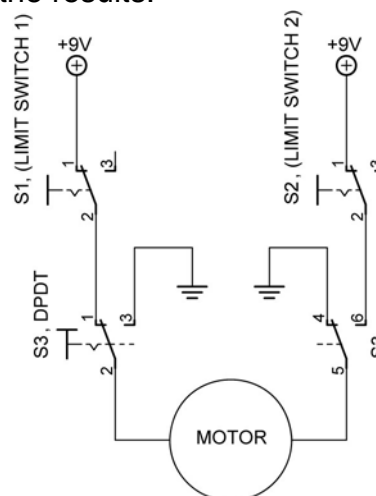
- **Purpose:** The purpose of this lab is to acquaint the student on the use of switches mounted on the robotic car for object avoidance.
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 2 – Lever Micro-switches
- **Procedure:**
  - For the student to determine
- **Challenge:**
  - Install two lever switches on your robotic car platform and program the PIC so the car will avoid objects placed in front of the vehicle. Keep your PAUSE command periods short.

## Cornerstone Electronics Technology and Robotics II Switch Sensor LAB 4 – Limit Switch1

- **Purpose:** The purpose of this lab is to acquaint the student with wiring limit switches to a DPDT switch that is operated manually.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 2 – Lever Micro-Switches
  - 1 – DIP DPDT Switch
- **Procedure:**
  - Wire the circuit below.
  - Operate the circuit and observe the results.

Basic Limit Switch Circuit:

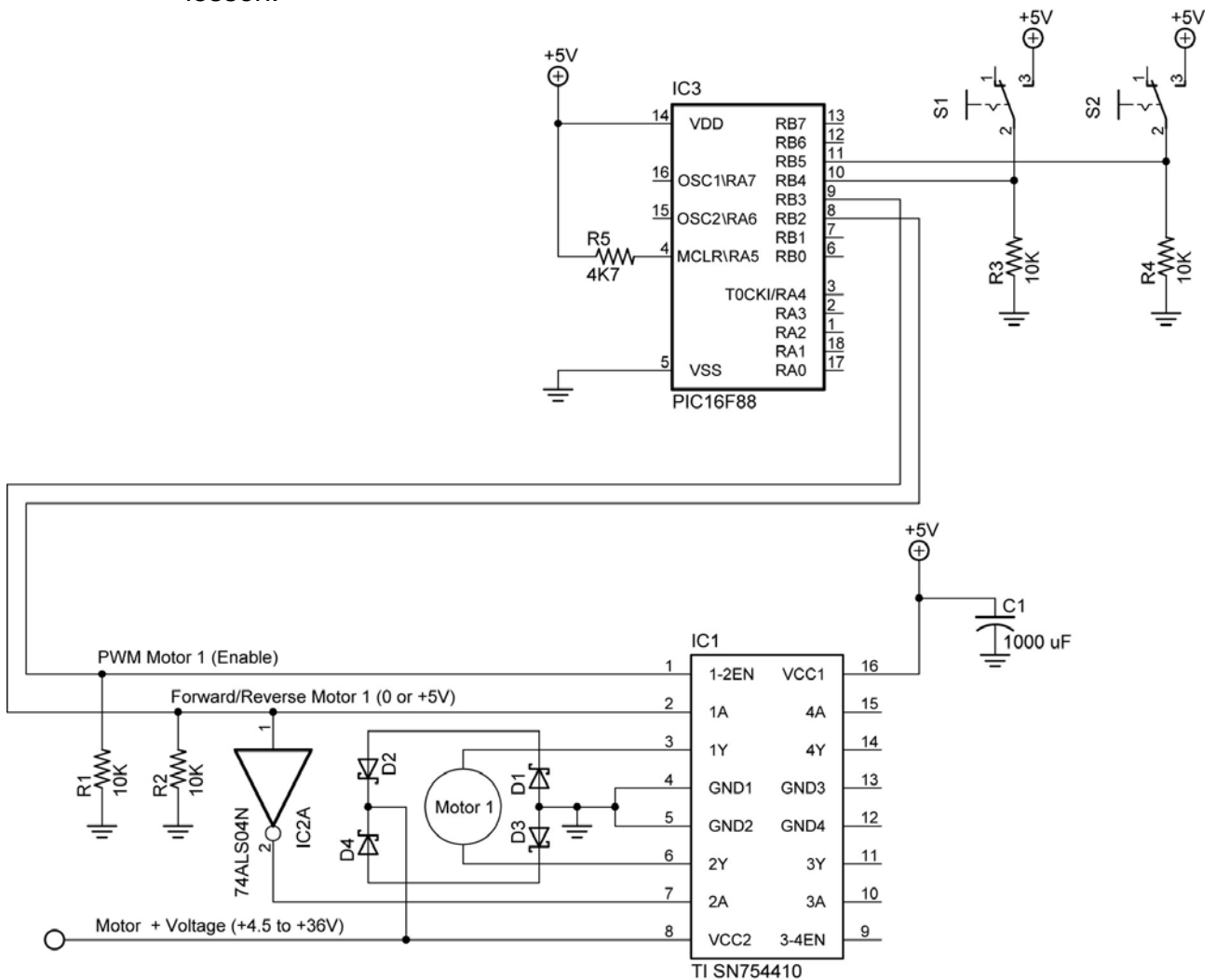
- S1 and S2 serve as limit switches
- S3 changes the direction of the motor.





## Cornerstone Electronics Technology and Robotics II Switch Sensor LAB 5 – Limit Switch2

- **Purpose:** The purpose of this lab is to acquaint the student with programming a PIC microcontroller to control a pair of limit switches.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 2 – Lever Micro-Switches
  - 1 – DIP DPDT Switch
- **Procedure:**
  - Wire the circuit below. Use the SN754410 circuit wired in the PWM lesson.

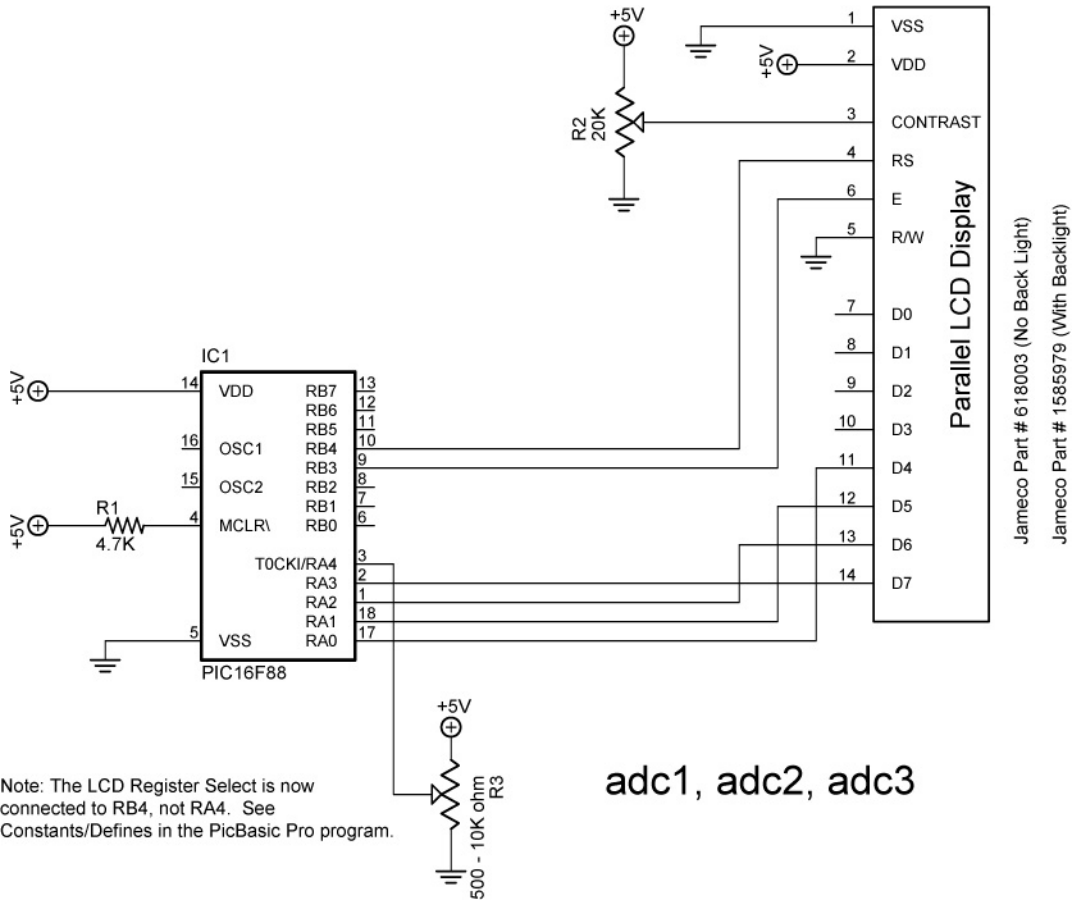


NOTE: D1-D4 1N5817  
S1 and S2 are limit switches

- **Challenge:**
  - Program the PIC16F88 to continuously run the given motor and lever system back and forth between the two limit switches.

## Cornerstone Electronics Technology and Robotics II Resistive Sensors LAB 1 – Reading a Potentiometer

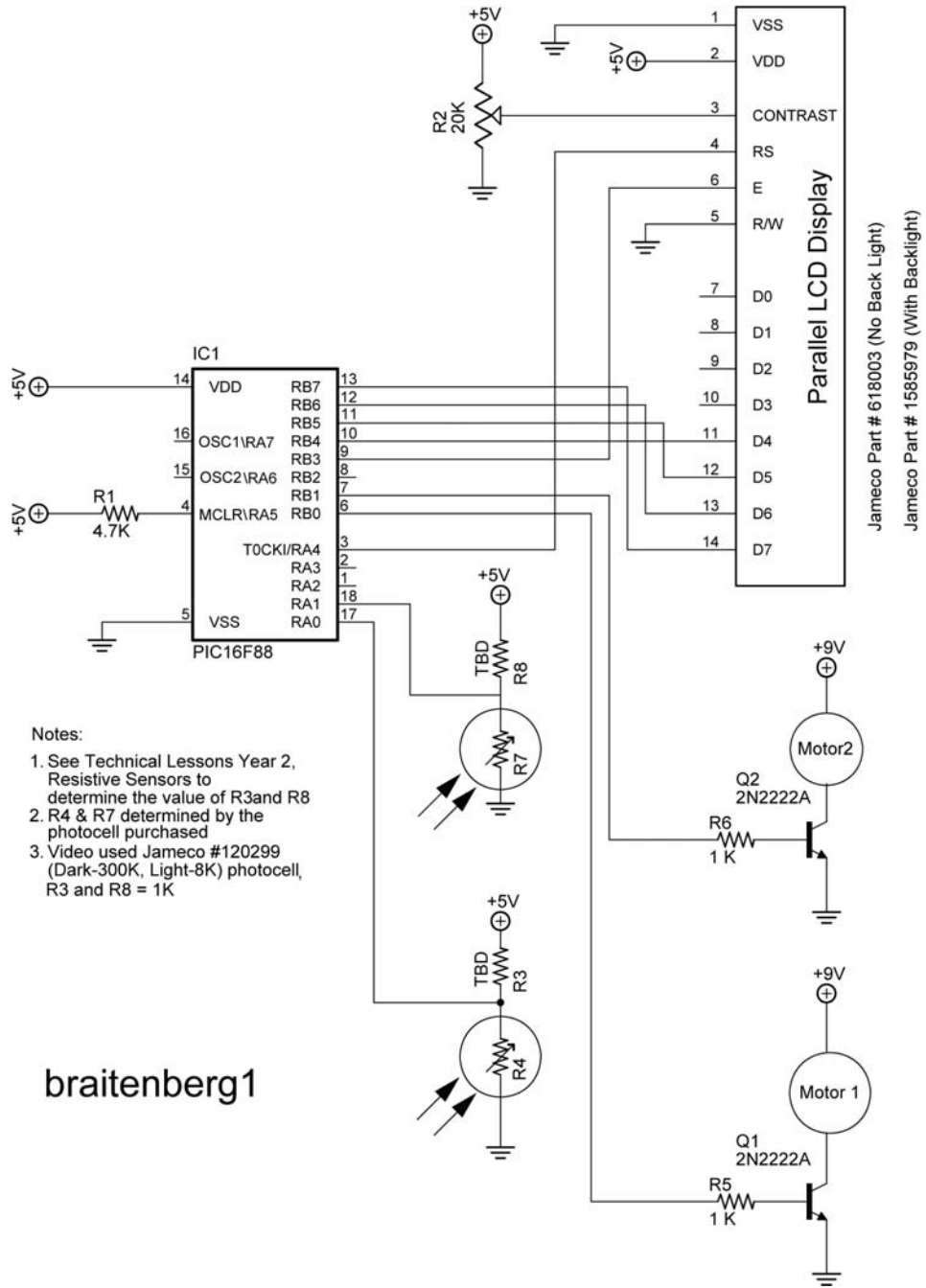
- **Purpose:** The purpose of this lab is to teach the student how to wire a PIC18F88 for analog/digital conversion, to expose the student to the PicBasic Pro command ADCIN, and to introduce the linear slide sensor.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V supply or an Analog/Digital Trainer
  - 1 – PIC 16F88 Microcontroller
  - 1 – 10 K Tripot
  - 1 – Phidgets Slide Sensor, Product # RB-Phi-20 from:  
<http://www.robotshop.ca/home/products/robot-parts/sensors/linear-rotary-resistors/index.html>
  - 2 – 150 Ohm, ½ Watt Resistors
  - 1 – 10 K Ohm, ½ Watt Resistor
  - 1 – 1K, ½ Watt Resistor
  - 1 – NO Momentary Switch
  - 2 – LEDs
- **Procedure:**
  - Wire the circuit below. Use a 10K tripot for R3. Make sure to read the note in the schematic.
  - Program the PIC16F88 with **adc1.pbp**, **adc2.pbp**, and then **adc3.pbp**. Adjust R3 and note the values on the LCD screen for each program.
  - Now substitute the slide sensor for the 10K tripot. Note the changes in the LCD value as the linear position of the slide sensor moves.



## Cornerstone Electronics Technology and Robotics II

### Resistive Sensors LAB 2 – CdS Photoresistors

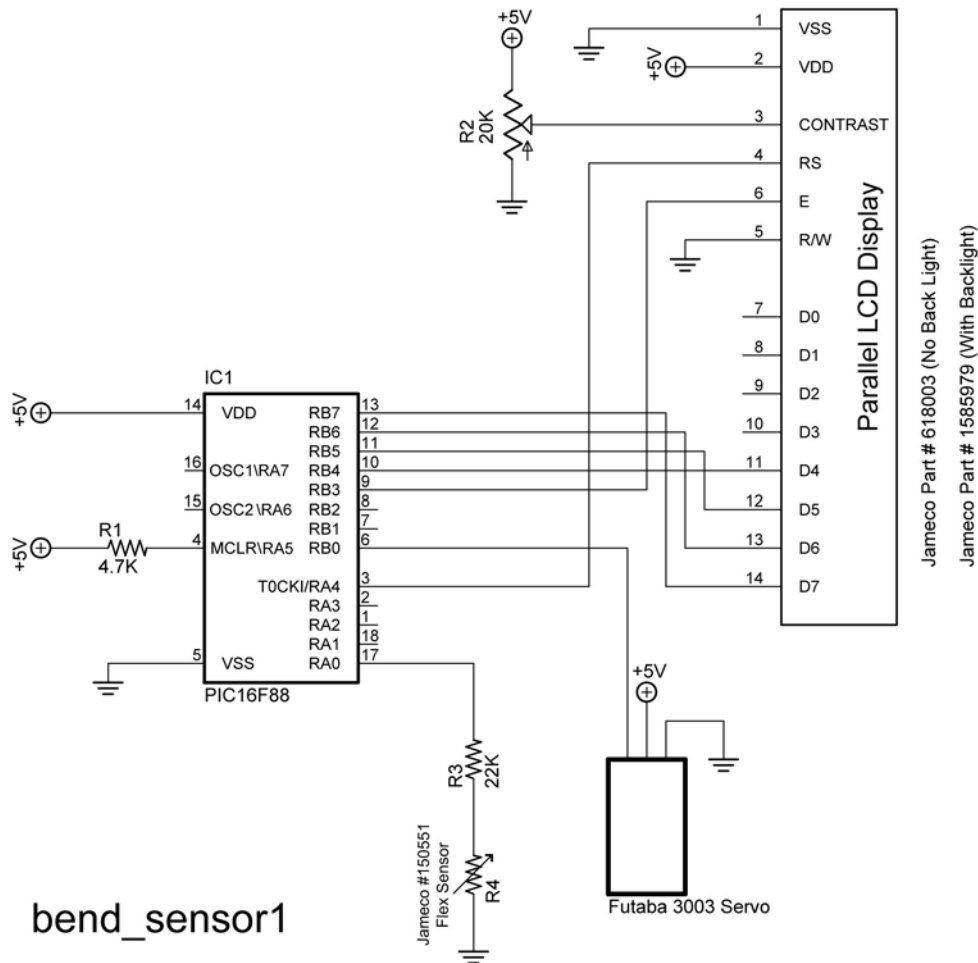
- **Purpose:** The purpose of this lab is to acquaint the student with the use of a CdS photocell as a resistive sensor.
- **Apparatus and Materials:**
  - 1 – Robotic Car Platform
  - 1 – PIC 16F88 – I/P Microcontroller
  - 1 – LCD Screen, Jameco #618003
  - 1 – 20 K Ohm Potentiometer
  - 2 – CdS Photoresistors, Jameco #120299 (300K-Dark, 8K-Light)
  - 1 – 4.7K Resistor
  - 4 – 1K Resistor, Includes R3 and R8
  - 2 – 2N2222A NPN Transistors
- **Procedure:**
  - Wire the following circuit.
  - Program the PIC16F88 with **braitenberg1.pbp**.
  - Adjust the CdS photocells for the car to be attracted to the light.  
See: [http://www.youtube.com/watch?v=N\\_X4\\_VVxOrE](http://www.youtube.com/watch?v=N_X4_VVxOrE)
- **Challenges:**
  - **Line-follower:** Mount CdS sensors on your robotic car for the car to follow a taped line. See: <http://www.youtube.com/watch?v=ut0iTLZykog>
  - **Light-Steering:** Use one photocell to activate steering or forward motion for the robotic car. Use a second photocell to turn the robotic car off. The car must begin in the starting box and come to a complete stop in the finish box.
  - **CdS switch:** Wire a circuit and program a PIC16F88 to serve as a light activated switch which turns on and off an LED.



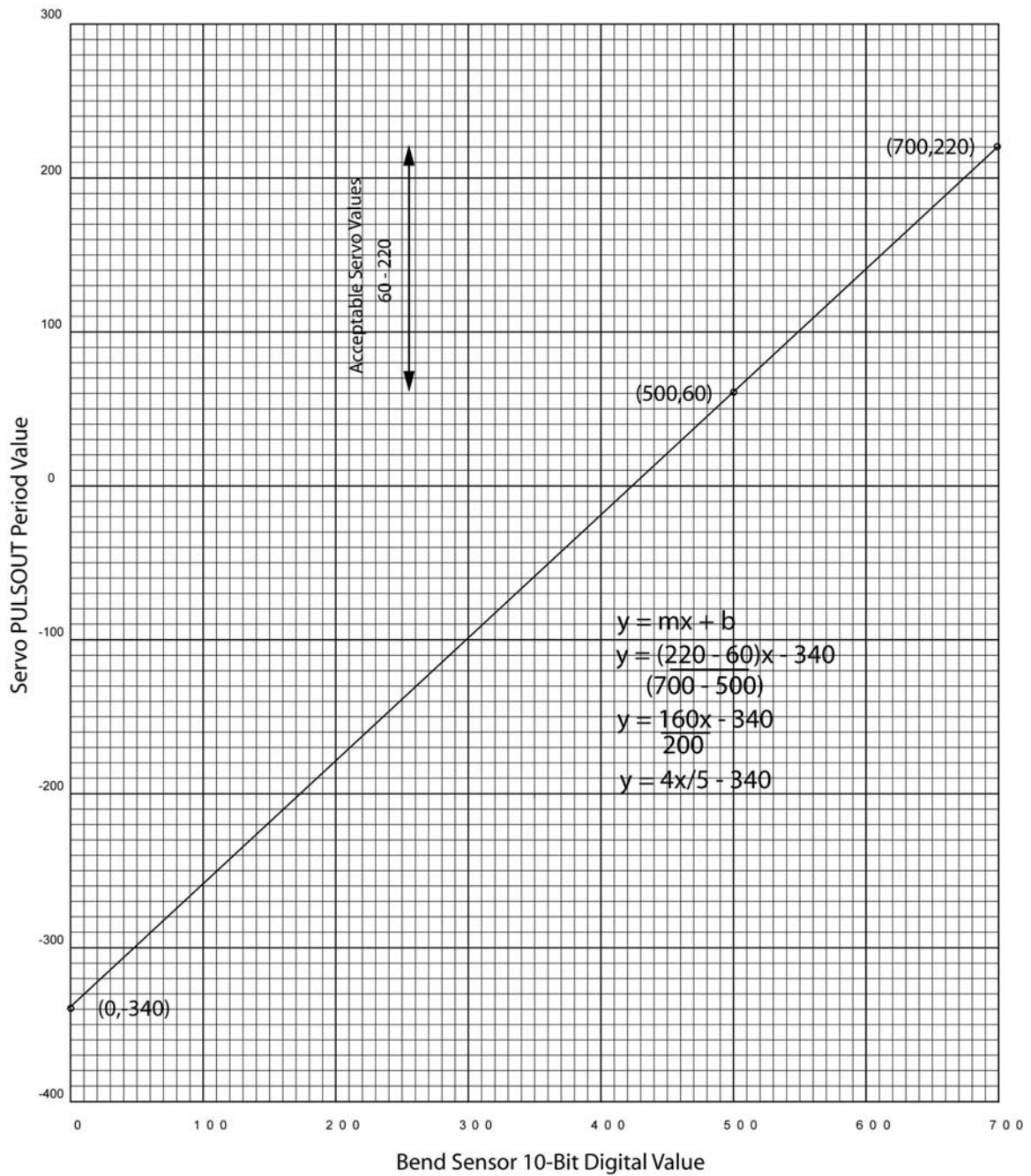
braitenberg1

## Cornerstone Electronics Technology and Robotics II Resistive Sensors LAB 3 – Bend Sensors

- **Purpose:** The purpose of this lab is to acquaint the student with the function of a bend sensor.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – PIC 16F88 Microcontroller
  - 1 – 4.7K Ohm Resistor
  - 1 – 22K Resistor
  - 1 – 20K Tripot
  - 1 – Bend Sensor, Jameco #150551
  - 1 – LCD Screen, Jameco # 618003
  - 1 – Futaba 3003 Hobby Servo
- **Procedure:**
  - Wire the circuit bend\_sensor1.
  - Program the PIC16F88 with **bend\_sensor1.pbp**
  - Bend flex sensor and note servo movement.
  - Review the derivation of the equation in the graph below.

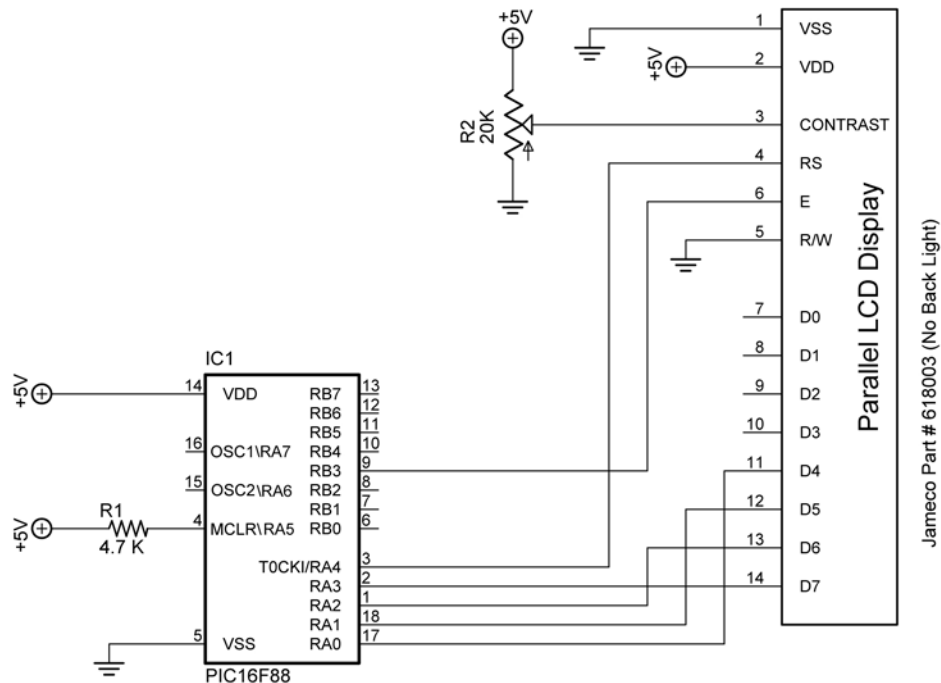


## Derivation of Formula Used in bend\_sensor1.pbp



## Cornerstone Electronics Technology and Robotics II Ultra-Sonic LAB 1 – array1 and array2.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro variable arrays.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – PIC 16F88 Microcontroller
  - 1 – 4.7K Ohm Resistor
  - 1 – 20K Tripot
  - 1 – LCD Screen, Jameco # 618003
- **Procedure:**
  - Wire the circuit array1 as shown below.
  - Program the PIC16F88 with **array1.pbp** and power the chip.
  - Program the PIC16F88 with **array2.pbp** and power the chip.
  - Program the PIC16F88 to three take CdS photoresistor readings and record the sample number and CdS reading in two arrays, sample[3] and cds[3]. Display all of the results of the two arrays at one time on an LCD. Save the program as **array10.pbp**.

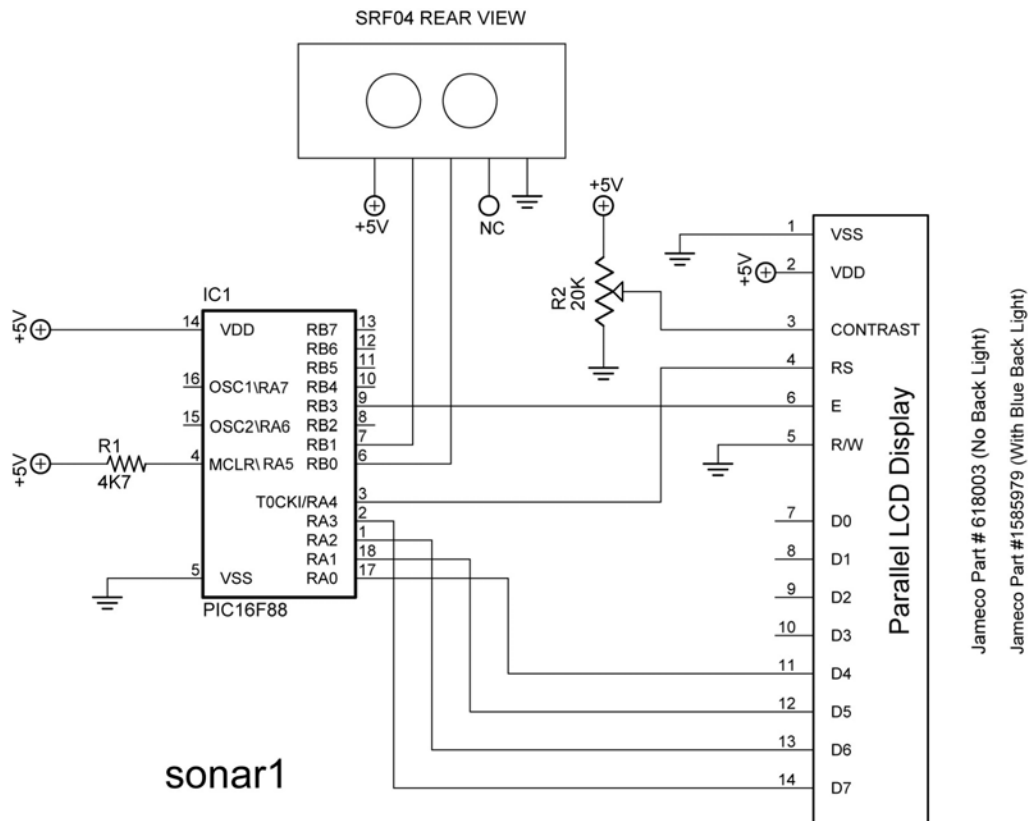


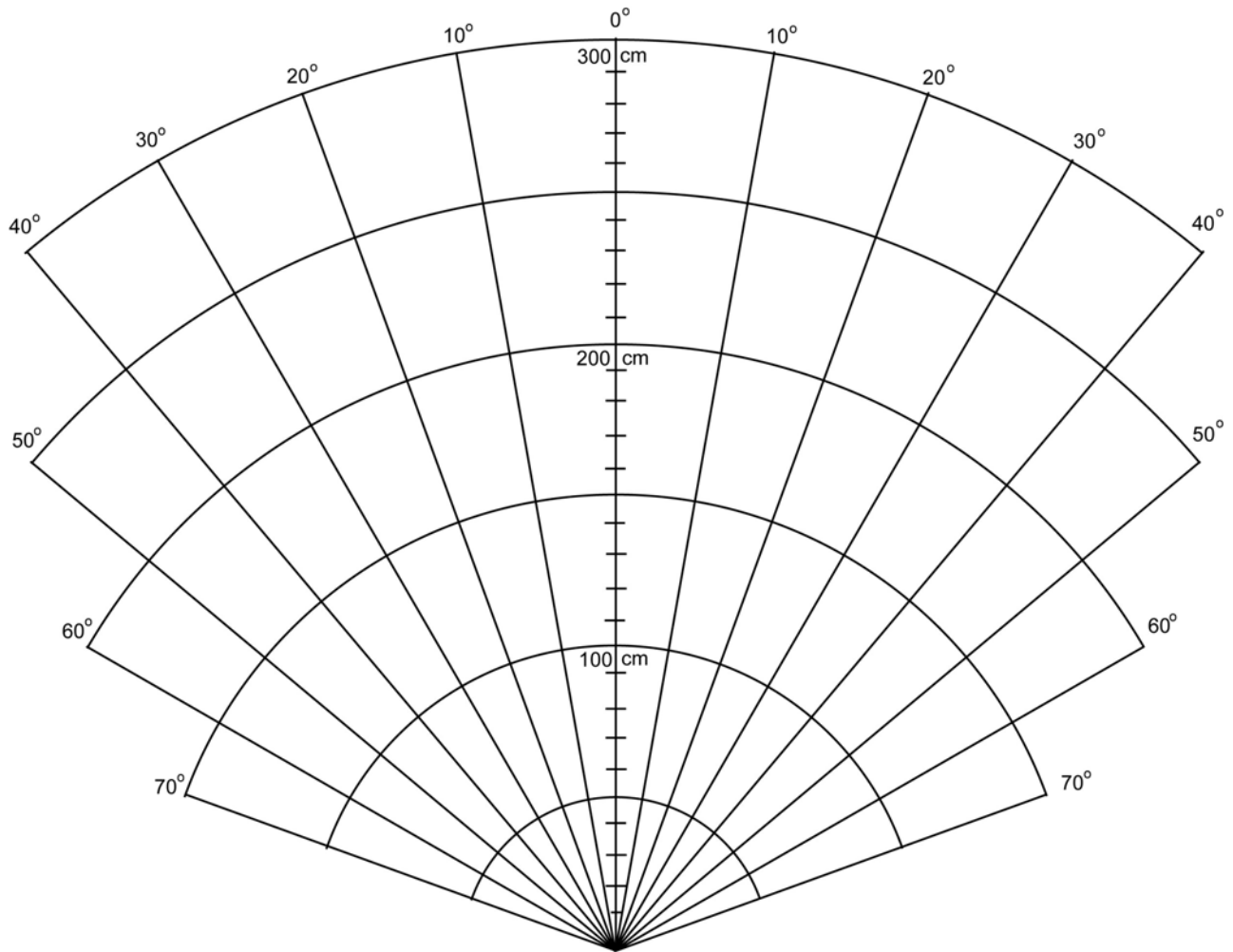
array1 and array2



## Cornerstone Electronics Technology and Robotics II Ultra-Sonic LAB 2 – sonar1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the basic function of the SRF04 ultra-sonic range finder and the PicBasic Pro commands to drive the range finder.
- **Apparatus and Materials:**
  - 1 – Breadboard or Robotic Car
  - 1 – PIC 16F88 Microcontroller
  - 1 – 4.7K Ohm Resistor
  - 1 – 20K Tripot
  - 1 – LCD Screen, Jameco # 618003
- **Procedure:**
  - Wire the circuit sonar1.
  - Open **sonar1.pbp** and download to your chip.
    - Place an object about 10 inches from the sonar and let the sonar take readings over time. Observe any changes in the readings.
    - Use the attached SRF04 Beam Pattern Plot sheet to plot the sensitivity of the SRF04 module detecting a 2"x 4" piece of lumber. Keep the wood perpendicular to the radial lines that converge at the SRF04. Record only readings that are valid and consistent with the range of the sonar.





### SRF04 Beam Pattern Plot

- **Challenge:**
  - Using a servo and a SRF04 for obstacle avoidance:
    - Use a SRF04 mounted on a servo to sweep through 180 degrees and navigate your robot to avoid obstacles. At this point, do not use an interrupt, but rather use pauses in your forward movement.

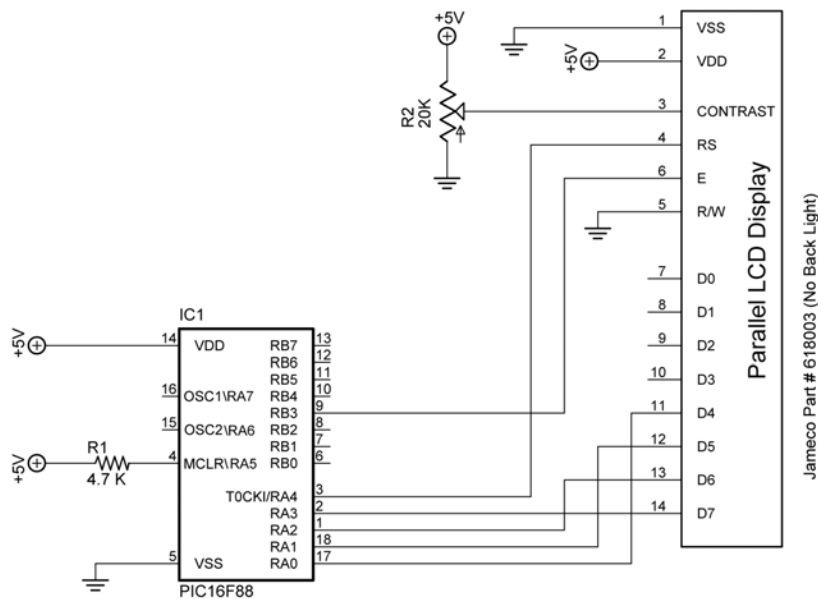
## Cornerstone Electronics Technology and Robotics II

### Ultra-Sonic LAB 3 – Completion of Sonar Car

- **Purpose:** The purpose of this lab is to have the student complete the robotic sonar car project.
- **Apparatus and Materials:**
  - See parts list at:  
[http://www.cornerstonerobotics.org/excel%20doc/robotics\\_2\\_parts\\_list.pdf](http://www.cornerstonerobotics.org/excel%20doc/robotics_2_parts_list.pdf)
- **Procedure:**
  - Complete the mechanical and electronic systems on the car. For the schematics see:
    - Sonar Car Circuitry 1:  
[http://www.cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car1.pdf](http://www.cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf)
    - Sonar Car Circuitry 2:  
[http://www.cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car2.pdf](http://www.cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf)
  - Program the car with sonar\_car1:
    - Format in .pbp:  
[http://www.cornerstonerobotics.org/code/sonar\\_car1.pbp](http://www.cornerstonerobotics.org/code/sonar_car1.pbp)
    - Format in .pdf:  
[http://www.cornerstonerobotics.org/code/sonar\\_car1.pdf](http://www.cornerstonerobotics.org/code/sonar_car1.pdf)
- **Photos:**
  - To see past project solutions see:  
[http://www.cornerstonerobotics.org/sonar\\_car\\_2008.php](http://www.cornerstonerobotics.org/sonar_car_2008.php)

**Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings**  
**LAB 1 – Simple Arrays in PicBasic Pro**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to acquaint the student with the use of arrays in a PicBasic Pro program.
- **Apparatus and Materials:**
  - 1 – Analog/Digital Trainer or Breadboard w/ +5V Power Supply
  - PIC16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 4.7 K Resistor
- **Procedure:**
  - Wire the circuit “array1 and array2” shown below:



array1 and array2

- Import and run **array1.pbp**. See: <http://cornerstonerobotics.org/code/array1.pbp>
- Discuss the operation of the program.
- Import and run **array2.pbp**. See: <http://cornerstonerobotics.org/code/array2.pbp>
- Discuss operation of the program.
- **Challenge:**
  - Using **array2.pbp** and **sonar1.pbp**, write a program that takes 4 ultra-sonic readings one second apart, then display each reading on an LCD for one second. For **sonar1.pbp**, see: <http://cornerstonerobotics.org/code/sonar1.pbp>

**Sonar Car 2 – Arrays and SRF04 Ultra-Sonic Ranger Finder Readings**  
**LAB 2 – Programming the Sonar Car with sonar\_car\_b.pbp**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to review the second in a series of four programs that takes the class through the development of the final program sonar\_car1.pbp. This lab reviews the program that adds taking ultra-sonic readings at each servo position then records data in dx\_in & position arrays.
  
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics at:  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car1.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf)  
and  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car2.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf)
  
- **Procedure:**
  - Open the program as **sonar\_car\_b.pbp**. See:  
[http://cornerstonerobotics.org/code/sonar\\_car\\_b.pbp](http://cornerstonerobotics.org/code/sonar_car_b.pbp)
  - Discuss operation of the program.

**Sonar Car 3 – Select Case Command / Obstacle Avoidance**  
**LAB 1 – SELECT CASE Command**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to acquaint the student with the use of **SELECT CASE** in a PicBasic Pro programming.
  
- **Apparatus and Materials:**
  - 1 – Breadboard w/ +5V Power Supply
  - PIC16F88 Microcontroller
  - 4.7 K Resistor
  - Other materials to be determined by student
  
- **Procedure:**
  - Connect 4 LEDs to pins RB0 – RB3 on a PIC16F88. Don't forget the current limiting resistors.
  - Write a new PicBasic Pro program that accomplishes the following task:
    - Create a **FOR..NEXT** loop to run 100 times.
    - Use the **SELECT CASE** command to:
      - Light the LED connected to RB0 only when the loop number is less than 50.
      - Light the LED tied to RB1 when the loop number equals 50.
      - Light the LED on RB2 when the loop number is greater than 90.
      - Light the LED on RB3 when the LED on RB0, RB1, and RB2 are not lit.
    - Have the program run forever.

**Sonar Car 3 – Select Case Command / Obstacle Avoidance**  
**LAB 2 – Programming the Sonar Car with sonar\_car\_c.pbp**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to review the third in a series of four programs that takes the class through the development of the final program sonar\_car1.pbp. This lab reviews the portion of the program that adds rotating the car to align with the longest sonar reading.
  
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics at:  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car1.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf)  
and  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car2.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf)
  
- **Procedure:**
  - Open the program as **sonar\_car\_c.pbp**. See:  
[http://cornerstonerobotics.org/code/sonar\\_car\\_c.pbp](http://cornerstonerobotics.org/code/sonar_car_c.pbp)
  - Discuss operation of the program.

## Sonar Car 4 – Collision Detection LAB 1 – Backup Routine Cornerstone Electronics Technology and Robotics II

- **Purpose:** The purpose of this lab is to challenge the student to solve a common problem in robotics, collision detection.
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics at:  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car1.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf)  
and  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car2.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf)
- **Procedure:**
  - Program your ultra-sonic car such that when the car travels forward and an object presses one of the switches mounted on the front then the car:
    - Will backup for 1 second,
    - Turn slightly in the direction opposite of the activated switch,
    - Pan through the 7 servo positions again and take new readings.
  - Hint:
    - If the car is programmed to travel forward for a long time using a long **PAUSE** and the front switch is pressed during the long **PAUSE**, the PIC will not recognize the pressed switch event. You must design the program to move forward in short increments so the program can keep monitoring (polling) the front switches. Use an **IF..THEN** loop.



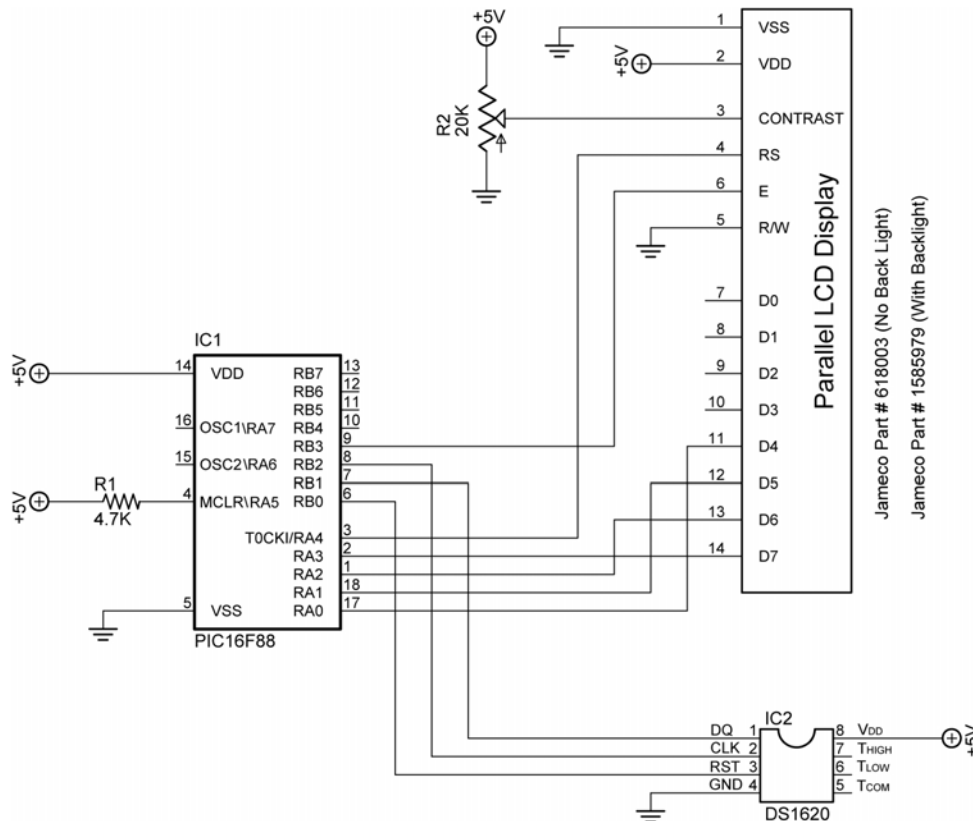
**Sonar Car 4 – Collision Detection**  
**LAB 2 – Programming the Sonar Car with sonar\_car\_d.pbp**  
**Cornerstone Electronics Technology and Robotics II**

- **Purpose:** The purpose of this lab is to review the fourth in a series of four programs that takes the class through the development of the final program sonar\_car1.pbp. This lab reviews the portion of the program that detects collisions with obstacles when the ultra-sonic sensor fails to detect an object.
  
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics at:  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car1.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf)  
and  
[http://cornerstonerobotics.org/schematics/pic\\_programming\\_sonar\\_car2.pdf](http://cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf)
  
- **Procedure:**
  - Open the program as **sonar\_car\_d.pbp** (the same program as sonar\_car1.pbp). See:  
[http://cornerstonerobotics.org/code/sonar\\_car\\_d.pbp](http://cornerstonerobotics.org/code/sonar_car_d.pbp)
  - Discuss operation of the program.

## Electronics and Robotics II

### DS1620 Lab 1 – DD1620 \_1 and DS1620 \_2

- **Purpose:** The purpose of this lab is to have the student understand the basic operation of the Dallas DS1620 Digital Thermometer and be able to write and read temperature settings into the DS1620.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – Dallas DS1620 Digital Thermometer
  - PIC16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 4.7 K Resistor
- **Procedure:**
  - Wire the circuit below.
  - Program the PIC16F88 with **DS1620\_1.pbp**. Place your finger on the DS1620 and observe the change in the LCD readout.
  - Now program the PIC16F88 with **DS1620\_2.pbp**. Change the value of TH and TL using the digital hex values in the appendixes – save as **DS1620\_10.pbp**. Input both positive and negative values.

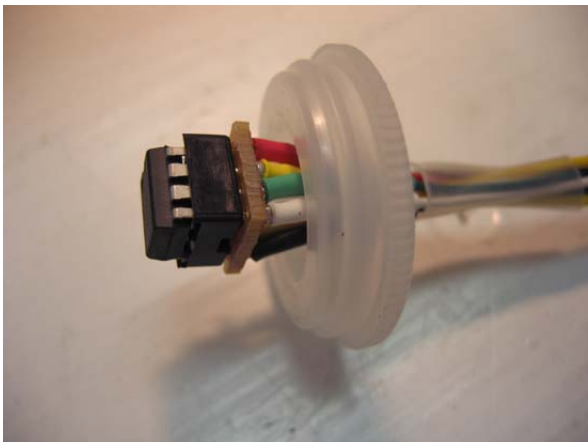


DS1620\_1 and DS1620\_2

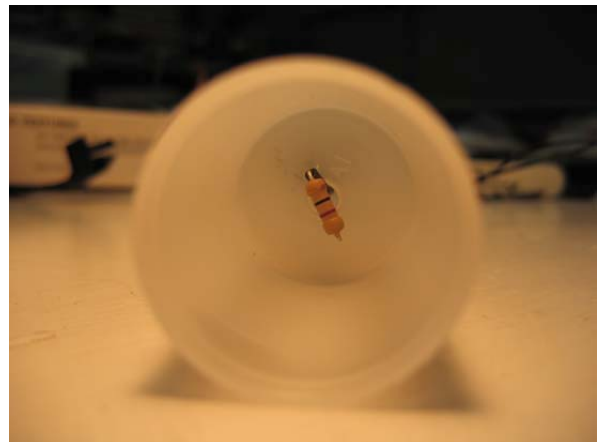
## Electronics and Robotics II

### DS1620 Lab 2 – DD1620\_3\_heater

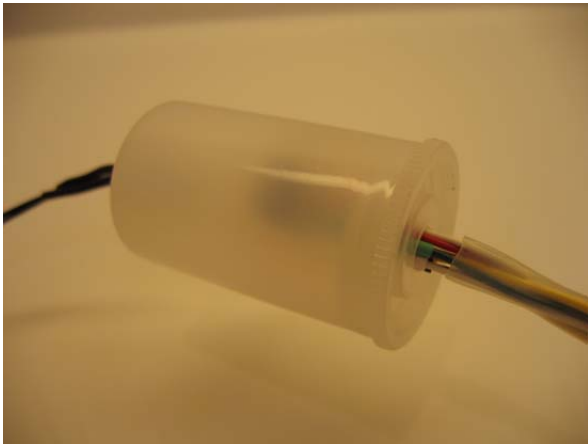
- **Purpose:** The purpose of this lab is to demonstrate the DS1620 function as a thermostat used in conjunction with a heater.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – Dallas DS1620 Digital Thermometer
  - PIC16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 1 – 20 K Potentiometer
  - 1 – 4.7 K Resistor
  - 1 – 1 K Resistor
  - 1 – 150 Ohm resistor
  - 1 – 47 Ohm Resistor, ½ Watt
  - 1 – 2N2222A NPN Transistor
  - 1 – LED
- **Procedure:**
  - Wire the circuit below.
  - Program the PIC16F88 with **DS1620\_3\_heater.pbp**. The high and low temperature limits may need changing depending upon the room temperature. See the Appendixes A and B for hexadecimal number of temperatures permitted for the DS1620. Use DS1620\_2 to input the new TH and TL values into the DS1620 digital thermometer then reprogram with **DS1620\_3\_heater.pbp** and test the circuit's performance.
  - Note that the current through R5 is 0.19 A ( $I_{R5} = +9 \text{ V} / 47 \Omega$ ) and the wattage value for R5 should exceed 1.72 watts ( $P_{R5} = 9 \text{ V} * 0.19 \text{ A}$ ). The 1/2 watt resistor forces the resistor to overheat and act as a heater.
- **Other Resources:**
  - See Parallax version at:  
<http://www.parallax.com/Portals/0/Downloads/docs/books/edu/ic.pdf>
- **Photos:**



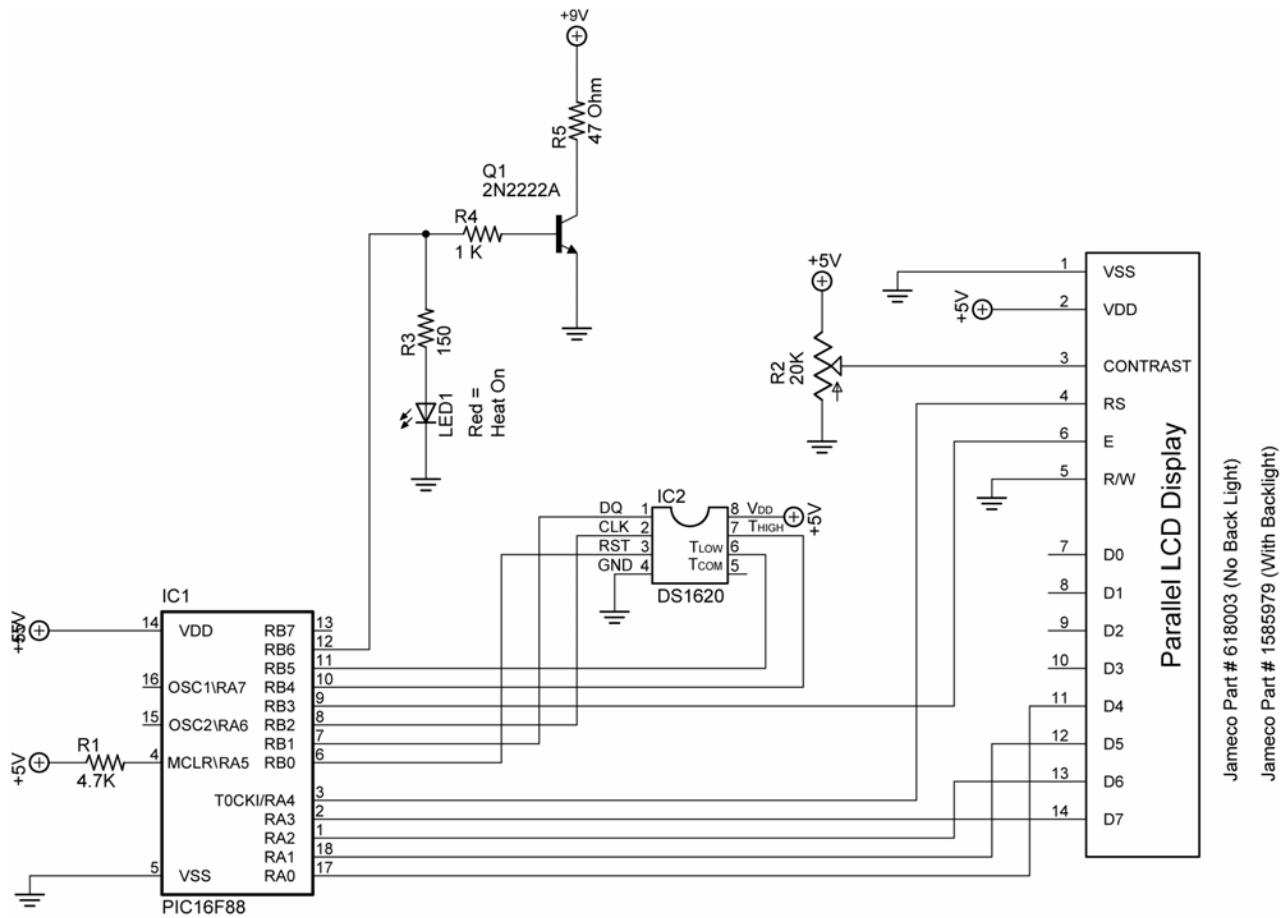
**DS1620 Soldered Assembly**



**47 Ohm Resistor Inside Film Canister**



Film Canister with DS1620 and Resistor



DS1620\_3\_heater

Jameco Part # 618003 (No Back Light)  
 Jameco Part # 1585979 (With Backlight)

## Electronics and Robotics II

### DS1620 Lab 2 – DD1620 \_4\_fan

- **Purpose:** The purpose of this lab is to demonstrate the DS1620 function as a thermostat used in conjunction with a cooling system, which in this case is a fan.
- **Apparatus and Materials:**
  - 1 – Breadboard
  - 1 – Dallas DS1620 Digital Thermometer
  - PIC16F88 Microcontroller
  - Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003
  - 20 K Potentiometer
  - 4.7 K Resistor
  - 1 – 1 K Resistor
  - 1 – 150 Ohm resistor
  - 1 – 12 VDC Fan
  - 1 – 2N2222A NPN Transistor
  - 1 – LED
- **Procedure:**
  - Wire the circuit below.
  - Program the PIC16F88 with **DS1620\_4\_fan.pbp**. Notice that the cooling system (the fan) is driven directly from the DS1620 and not from the PIC16F88. This is because the TCOM output goes HIGH when the measured temperature meets or exceeds TH, and will remain HIGH until the temperature equals or falls below TL.
- **Other Resources:**
  - See Parallax version at:  
<http://www.parallax.com/Portals/0/Downloads/docs/books/edu/ic.pdf>
- **Photo:**

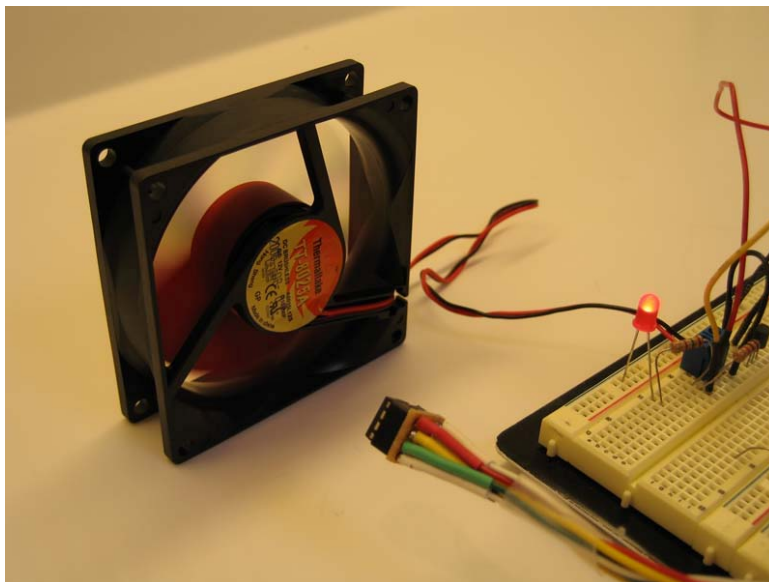
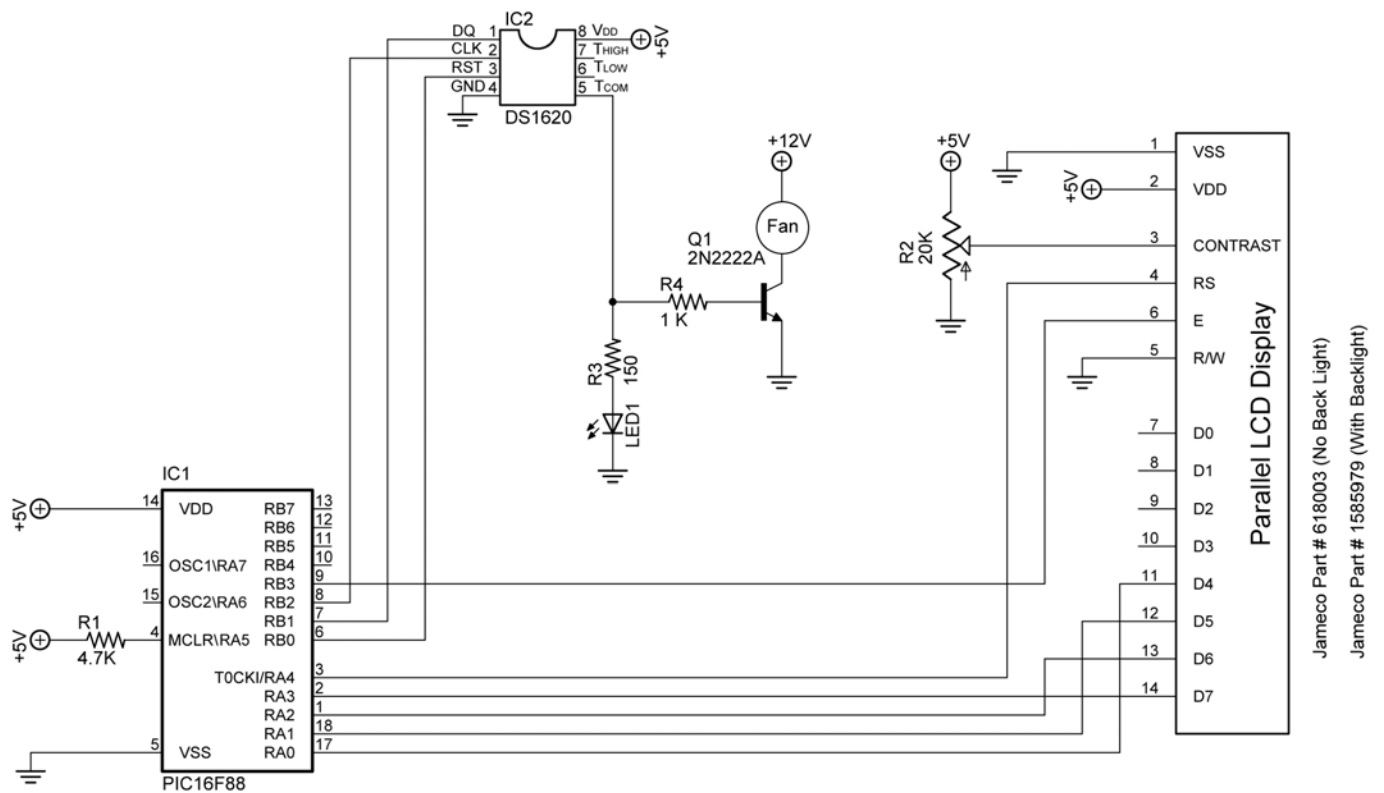


Photo Showing Fan Cooling the DS1620 Digital Thermometer



DS1620\_4\_fan

# DS1620 Positive Temperatures:

DS1620 Positive Temperatures				DS1620 Positive Temperatures			
°C	°F	Binary Digital Output	Hex Digital	°C	°F	Binary Digital Output	Hex Digital
+ 125.0	257.0	%0 11111010	\$00FA	+ 100.0	212.0	%0 11001000	\$00C8
+ 124.5	256.1	%0 11111001	\$00F9	+ 99.5	211.1	%0 11000111	\$00C7
+ 124.0	255.2	%0 11111000	\$00F8	+ 99.0	210.2	%0 11000110	\$00C6
+ 123.5	254.3	%0 11110111	\$00F7	+ 98.5	209.3	%0 11000101	\$00C5
+ 123.0	253.4	%0 11110110	\$00F6	+ 98.0	208.4	%0 11000100	\$00C4
+ 122.5	252.5	%0 11110101	\$00F5	+ 97.5	207.5	%0 11000011	\$00C3
+ 122.0	251.6	%0 11110100	\$00F4	+ 97.0	206.6	%0 11000010	\$00C2
+ 121.5	250.7	%0 11110011	\$00F3	+ 96.5	205.7	%0 11000001	\$00C1
+ 121.0	249.8	%0 11110010	\$00F2	+ 96.0	204.8	%0 11000000	\$00C0
+ 120.5	248.9	%0 11110001	\$00F1	+ 95.5	203.9	%0 10111111	\$00BF
+ 120.0	248.0	%0 11110000	\$00F0	+ 95.0	203.0	%0 10111110	\$00BE
+ 119.5	247.1	%0 11101111	\$00EF	+ 94.5	202.1	%0 101111101	\$00BD
+ 119.0	246.2	%0 11101110	\$00EE	+ 94.0	201.2	%0 101111100	\$00BC
+ 118.5	245.3	%0 11101101	\$00ED	+ 93.5	200.3	%0 10111011	\$00BB
+ 118.0	244.4	%0 11101100	\$00EC	+ 93.0	199.4	%0 10111010	\$00BA
+ 117.5	243.5	%0 11101011	\$00EB	+ 92.5	198.5	%0 10111001	\$00B9
+ 117.0	242.6	%0 11101010	\$00EA	+ 92.0	197.6	%0 10111000	\$00B8
+ 116.5	241.7	%0 11101001	\$00E9	+ 91.5	196.7	%0 10110111	\$00B7
+ 116.0	240.8	%0 11101000	\$00E8	+ 91.0	195.8	%0 10110110	\$00B6
+ 115.5	239.9	%0 11100111	\$00E7	+ 90.5	194.9	%0 10110101	\$00B5
+ 115.0	239.0	%0 11100110	\$00E6	+ 90.0	194.0	%0 10110100	\$00B4
+ 114.5	238.1	%0 11100101	\$00E5	+ 89.5	193.1	%0 10110011	\$00B3
+ 114.0	237.2	%0 11100100	\$00E4	+ 89.0	192.2	%0 10110010	\$00B2
+ 113.5	236.3	%0 11100011	\$00E3	+ 88.5	191.3	%0 10110001	\$00B1
+ 113.0	235.4	%0 11100010	\$00E2	+ 88.0	190.4	%0 10110000	\$00B0
+ 112.5	234.5	%0 11100001	\$00E1	+ 87.5	189.5	%0 10101111	\$00AF
+ 112.0	233.6	%0 11100000	\$00E0	+ 87.0	188.6	%0 10101110	\$00AE
+ 111.5	232.7	%0 11011111	\$00DF	+ 86.5	187.7	%0 101011101	\$00AD
+ 111.0	231.8	%0 11011110	\$00DE	+ 86.0	186.8	%0 101011100	\$00AC
+ 110.5	230.9	%0 11011101	\$00DD	+ 85.5	185.9	%0 10101011	\$00AB
+ 110.0	230.0	%0 11011100	\$00DC	+ 85.0	185.0	%0 10101010	\$00AA
+ 109.5	229.1	%0 11011011	\$00DB	+ 84.5	184.1	%0 10101001	\$00A9
+ 109.0	228.2	%0 11011010	\$00DA	+ 84.0	183.2	%0 10101000	\$00A8
+ 108.5	227.3	%0 11011001	\$00D9	+ 83.5	182.3	%0 10100111	\$00A7
+ 108.0	226.4	%0 11011000	\$00D8	+ 83.0	181.4	%0 10100110	\$00A6
+ 107.5	225.5	%0 11010111	\$00D7	+ 82.5	180.5	%0 10100101	\$00A5
+ 107.0	224.6	%0 11010110	\$00D6	+ 82.0	179.6	%0 10100100	\$00A4
+ 106.5	223.7	%0 11010101	\$00D5	+ 81.5	178.7	%0 10100011	\$00A3
+ 106.0	222.8	%0 11010100	\$00D4	+ 81.0	177.8	%0 10100010	\$00A2
+ 105.5	221.9	%0 11010011	\$00D3	+ 80.5	176.9	%0 10100001	\$00A1
+ 105.0	221.0	%0 11010010	\$00D2	+ 80.0	176.0	%0 10100000	\$00A0
+ 104.5	220.1	%0 11010001	\$00D1	+ 79.5	175.1	%0 10011111	\$009F
+ 104.0	219.2	%0 11010000	\$00D0	+ 79.0	174.2	%0 10011110	\$009E
+ 103.5	218.3	%0 11001111	\$00CF	+ 78.5	173.3	%0 100111101	\$009D
+ 103.0	217.4	%0 11001110	\$00CE	+ 78.0	172.4	%0 100111100	\$009C
+ 102.5	216.5	%0 11001101	\$00CD	+ 77.5	171.5	%0 10011011	\$009B
+ 102.0	215.6	%0 11001100	\$00CC	+ 77.0	170.6	%0 10011010	\$009A
+ 101.5	214.7	%0 11001011	\$00CB	+ 76.5	169.7	%0 10011001	\$0099
+ 101.0	213.8	%0 11001010	\$00CA	+ 76.0	168.8	%0 10011000	\$0098
+ 100.5	212.9	%0 11001001	\$00C9	+ 75.5	167.9	%0 10010111	\$0097

DS1620 Positive Temperatures				DS1620 Positive Temperatures			
°C	°F	Binary Digital Output	Hex Digital Output	°C	°F	Binary Digital Output	Hex Digital Output
+ 75.0	167.0	%0 10010110	\$0096	+ 50.0	122.0	%0 01100100	\$0064
+ 74.5	166.1	%0 10010101	\$0095	+ 49.5	121.1	%0 01100011	\$0063
+ 74.0	165.2	%0 10010100	\$0094	+ 49.0	120.2	%0 01100010	\$0062
+ 73.5	164.3	%0 10010011	\$0093	+ 48.5	119.3	%0 01100001	\$0061
+ 73.0	163.4	%0 10010010	\$0092	+ 48.0	118.4	%0 01100000	\$0060
+ 72.5	162.5	%0 10010001	\$0091	+ 47.5	117.5	%0 01011111	\$005F
+ 72.0	161.6	%0 10010000	\$0090	+ 47.0	116.6	%0 01011110	\$005E
+ 71.5	160.7	%0 10001111	\$008F	+ 46.5	115.7	%0 01011101	\$005D
+ 71.0	159.8	%0 10001110	\$008E	+ 46.0	114.8	%0 01011100	\$005C
+ 70.5	158.9	%0 10001101	\$008D	+ 45.5	113.9	%0 01011011	\$005B
+ 70.0	158.0	%0 10001100	\$008C	+ 45.0	113.0	%0 01011010	\$005A
+ 69.5	157.1	%0 10001011	\$008B	+ 44.5	112.1	%0 01011001	\$0059
+ 69.0	156.2	%0 10001010	\$008A	+ 44.0	111.2	%0 01011000	\$0058
+ 68.5	155.3	%0 10001001	\$0089	+ 43.5	110.3	%0 01010111	\$0057
+ 68.0	154.4	%0 10001000	\$0088	+ 43.0	109.4	%0 01010110	\$0056
+ 67.5	153.5	%0 10000111	\$0087	+ 42.5	108.5	%0 01010101	\$0055
+ 67.0	152.6	%0 10000110	\$0086	+ 42.0	107.6	%0 01010100	\$0054
+ 66.5	151.7	%0 10000101	\$0085	+ 41.5	106.7	%0 01010011	\$0053
+ 66.0	150.8	%0 10000100	\$0084	+ 41.0	105.8	%0 01010010	\$0052
+ 65.5	149.9	%0 10000011	\$0083	+ 40.5	104.9	%0 01010001	\$0051
+ 65.0	149.0	%0 10000010	\$0082	+ 40.0	104.0	%0 01010000	\$0050
+ 64.5	148.1	%0 10000001	\$0081	+ 39.5	103.1	%0 01001111	\$004F
+ 64.0	147.2	%0 10000000	\$0080	+ 39.0	102.2	%0 01001110	\$004E
+ 63.5	146.3	%0 01111111	\$007F	+ 38.5	101.3	%0 01001101	\$004D
+ 63.0	145.4	%0 01111110	\$007E	+ 38.0	100.4	%0 01001100	\$004C
+ 62.5	144.5	%0 01111101	\$007D	+ 37.5	99.5	%0 01001011	\$004B
+ 62.0	143.6	%0 01111100	\$007C	+ 37.0	98.6	%0 01001010	\$004A
+ 61.5	142.7	%0 01111011	\$007B	+ 36.5	97.7	%0 01001001	\$0049
+ 61.0	141.8	%0 01111010	\$007A	+ 36.0	96.8	%0 01001000	\$0048
+ 60.5	140.9	%0 01111001	\$0079	+ 35.5	95.9	%0 01000111	\$0047
+ 60.0	140.0	%0 01111000	\$0078	+ 35.0	95.0	%0 01000110	\$0046
+ 59.5	139.1	%0 01110111	\$0077	+ 34.5	94.1	%0 01000101	\$0045
+ 59.0	138.2	%0 01110110	\$0076	+ 34.0	93.2	%0 01000100	\$0044
+ 58.5	137.3	%0 01110101	\$0075	+ 33.5	92.3	%0 01000011	\$0043
+ 58.0	136.4	%0 01110100	\$0074	+ 33.0	91.4	%0 01000010	\$0042
+ 57.5	135.5	%0 01110011	\$0073	+ 32.5	90.5	%0 01000001	\$0041
+ 57.0	134.6	%0 01110010	\$0072	+ 32.0	89.6	%0 01000000	\$0040
+ 56.5	133.7	%0 01110001	\$0071	+ 31.5	88.7	%0 00111111	\$003F
+ 56.0	132.8	%0 01110000	\$0070	+ 31.0	87.8	%0 00111110	\$003E
+ 55.5	131.9	%0 01101111	\$006F	+ 30.5	86.9	%0 00111101	\$003D
+ 55.0	131.0	%0 01101110	\$006E	+ 30.0	86.0	%0 00111100	\$003C
+ 54.5	130.1	%0 01101101	\$006D	+ 29.5	85.1	%0 00111011	\$003B
+ 54.0	129.2	%0 01101100	\$006C	+ 29.0	84.2	%0 00111010	\$003A
+ 53.5	128.3	%0 01101011	\$006B	+ 28.5	83.3	%0 00111001	\$0039
+ 53.0	127.4	%0 01101010	\$006A	+ 28.0	82.4	%0 00111000	\$0038
+ 52.5	126.5	%0 01101001	\$0069	+ 27.5	81.5	%0 00111011	\$0037
+ 52.0	125.6	%0 01101000	\$0068	+ 27.0	80.6	%0 00111010	\$0036
+ 51.5	124.7	%0 01100111	\$0067	+ 26.5	79.7	%0 001110101	\$0035
+ 51.0	123.8	%0 01100110	\$0066	+ 26.0	78.8	%0 001110100	\$0034
+ 50.5	122.9	%0 01100101	\$0065	+ 25.5	77.9	%0 001110011	\$0033

**DS1620 Positive Temperatures**

°C °F Binary Digital Hex Digital



			Output	Output
+	25.0	77.0	%0 00110010	\$0032
+	24.5	76.1	%0 00110001	\$0031
+	24.0	75.2	%0 00110000	\$0030
+	23.5	74.3	%0 00101111	\$002F
+	23.0	73.4	%0 00101110	\$002E
+	22.5	72.5	%0 00101101	\$002D
+	22.0	71.6	%0 00101100	\$002C
+	21.5	70.7	%0 00101011	\$002B
+	21.0	69.8	%0 00101010	\$002A
+	20.5	68.9	%0 00101001	\$0029
+	20.0	68.0	%0 00101000	\$0028
+	19.5	67.1	%0 00100111	\$0027
+	19.0	66.2	%0 00100110	\$0026
+	18.5	65.3	%0 00100101	\$0025
+	18.0	64.4	%0 00100100	\$0024
+	17.5	63.5	%0 00100011	\$0023
+	17.0	62.6	%0 00100010	\$0022
+	16.5	61.7	%0 00100001	\$0021
+	16.0	60.8	%0 00100000	\$0020
+	15.5	59.9	%0 00011111	\$001F
+	15.0	59.0	%0 00011110	\$001E
+	14.5	58.1	%0 00011101	\$001D
+	14.0	57.2	%0 00011100	\$001C
+	13.5	56.3	%0 00011011	\$001B
+	13.0	55.4	%0 00011010	\$001A
+	12.5	54.5	%0 00011001	\$0019
+	12.0	53.6	%0 00011000	\$0018
+	11.5	52.7	%0 00010111	\$0017
+	11.0	51.8	%0 00010110	\$0016
+	10.5	50.9	%0 00010101	\$0015
+	10.0	50.0	%0 00010100	\$0014
+	9.5	49.1	%0 00010011	\$0013
+	9.0	48.2	%0 00010010	\$0012
+	8.5	47.3	%0 00010001	\$0011
+	8.0	46.4	%0 00010000	\$0010
+	7.5	45.5	%0 00001111	\$000F
+	7.0	44.6	%0 00001110	\$000E
+	6.5	43.7	%0 00001101	\$000D
+	6.0	42.8	%0 00001100	\$000C
+	5.5	41.9	%0 00001011	\$000B
+	5.0	41.0	%0 00001010	\$000A
+	4.5	40.1	%0 00001001	\$0009
+	4.0	39.2	%0 00001000	\$0008
+	3.5	38.3	%0 00000111	\$0007
+	3.0	37.4	%0 00000110	\$0006
+	2.5	36.5	%0 00000101	\$0005
+	2.0	35.6	%0 00000100	\$0004
+	1.5	34.7	%0 00000011	\$0003
+	1.0	33.8	%0 00000010	\$0002
+	0.5	32.9	%0 00000001	\$0001
+	0.0	32.0	%0 00000000	\$0000

## Appendix B DS1620 Negative Temperatures:

DS1620 Negative Temperatures

DS1620 Negative Temperatures

°C	°F	Binary Digital	Hex Digital	°C	°F	Binary Digital	Hex Digital
-0.5	31.1	%1 11111111	\$01FF	-28.0	-18.4	%1 11001000	\$01C8
-1.0	30.2	%1 11111110	\$01FE	-28.5	-19.3	%1 11000111	\$01C7
-1.5	29.3	%1 11111101	\$01FD	-29.0	-20.2	%1 11000110	\$01C6
-2.0	28.4	%1 11111100	\$01FC	-29.5	-21.1	%1 11000101	\$01C5
-2.5	27.5	%1 11111011	\$01FB	-30.0	-22.0	%1 11000100	\$01C4
-3.0	26.6	%1 11111010	\$01FA	-30.5	-22.9	%1 11000011	\$01C3
-3.5	25.7	%1 11111001	\$01F9	-31.0	-23.8	%1 11000010	\$01C2
-4.0	24.8	%1 11111000	\$01F8	-31.5	-24.7	%1 11000001	\$01C1
-4.5	23.9	%1 11110111	\$01F7	-32.0	-25.6	%1 11000000	\$01C0
-5.0	23.0	%1 11110110	\$01F6	-32.5	-26.5	%1 10111111	\$01BF
-5.5	22.1	%1 11110101	\$01F5	-33.0	-27.4	%1 10111110	\$01BE
-6.0	21.2	%1 11110100	\$01F4	-33.5	-28.3	%1 10111101	\$01BD
-6.5	20.3	%1 11110011	\$01F3	-34.0	-29.2	%1 10111100	\$01BC
-7.0	19.4	%1 11110010	\$01F2	-34.5	-30.1	%1 10111011	\$01BB
-7.5	18.5	%1 11110001	\$01F1	-35.0	-31.0	%1 10111010	\$01BA
-8.0	17.6	%1 11110000	\$01F0	-35.5	-31.9	%1 10111001	\$01B9
-8.5	16.7	%1 11101111	\$01EF	-36.0	-32.8	%1 10111000	\$01B8
-9.0	15.8	%1 11101110	\$01EE	-36.5	-33.7	%1 10110111	\$01B7
-9.5	14.9	%1 11101101	\$01ED	-37.0	-34.6	%1 10110110	\$01B6
-10.0	14.0	%1 11101100	\$01EC	-37.5	-35.5	%1 10110101	\$01B5
-10.5	13.1	%1 11101011	\$01EB	-38.0	-36.4	%1 10110100	\$01B4
-11.0	12.2	%1 11101010	\$01EA	-38.5	-37.3	%1 10110011	\$01B3
-11.5	11.3	%1 11101001	\$01E9	-39.0	-38.2	%1 10110010	\$01B2
-12.0	10.4	%1 11101000	\$01E8	-39.5	-39.1	%1 10110001	\$01B1
-12.5	9.5	%1 11100111	\$01E7	-40.0	-40.0	%1 10110000	\$01B0
-13.0	8.6	%1 11100110	\$01E6	-40.5	-40.9	%1 10101111	\$01AF
-13.5	7.7	%1 11100101	\$01E5	-41.0	-41.8	%1 10101110	\$01AE
-14.0	6.8	%1 11100100	\$01E4	-41.5	-42.7	%1 10101101	\$01AD
-14.5	5.9	%1 11100011	\$01E3	-42.0	-43.6	%1 10101100	\$01AC
-15.0	5.0	%1 11100010	\$01E2	-42.5	-44.5	%1 10101011	\$01AB
-15.5	4.1	%1 11100001	\$01E1	-43.0	-45.4	%1 10101010	\$01AA
-16.0	3.2	%1 11100000	\$01E0	-43.5	-46.3	%1 10101001	\$01A9
-16.5	2.3	%1 11011111	\$01DF	-44.0	-47.2	%1 10101000	\$01A8
-17.0	1.4	%1 11011110	\$01DE	-44.5	-48.1	%1 10100111	\$01A7
-17.5	0.5	%1 11011101	\$01DD	-45.0	-49.0	%1 10100110	\$01A6
-18.0	-0.4	%1 11011100	\$01DC	-45.5	-49.9	%1 10100101	\$01A5
-18.5	-1.3	%1 11011011	\$01DB	-46.0	-50.8	%1 10100100	\$01A4
-19.0	-2.2	%1 11011010	\$01DA	-46.5	-51.7	%1 10100011	\$01A3
-19.5	-3.1	%1 11011001	\$01D9	-47.0	-52.6	%1 10100010	\$01A2
-20.0	-4.0	%1 11011000	\$01D8	-47.5	-53.5	%1 10100001	\$01A1
-20.5	-4.9	%1 11010111	\$01D7	-48.0	-54.4	%1 10100000	\$01A0
-21.0	-5.8	%1 11010110	\$01D6	-48.5	-55.3	%1 10011111	\$019F
-21.5	-6.7	%1 11010101	\$01D5	-49.0	-56.2	%1 10011110	\$019E
-22.0	-7.6	%1 11010100	\$01D4	-49.5	-57.1	%1 10011101	\$019D
-22.5	-8.5	%1 11010011	\$01D3	-50.0	-58.0	%1 10011100	\$019C
-23.0	-9.4	%1 11010010	\$01D2	-50.5	-58.9	%1 10011011	\$019B
-23.5	-10.3	%1 11010001	\$01D1	-51.0	-59.8	%1 10011010	\$019A
-24.0	-11.2	%1 11010000	\$01D0	-51.5	-60.7	%1 10011001	\$0199
-24.5	-12.1	%1 11001111	\$01CF	-52.0	-61.6	%1 10011000	\$0198
-25.0	-13.0	%1 11001110	\$01CE	-52.5	-62.5	%1 10010111	\$0197
-25.5	-13.9	%1 11001101	\$01CD	-53.0	-63.4	%1 10010110	\$0196
-26.0	-14.8	%1 11001100	\$01CC	-53.5	-64.3	%1 10010101	\$0195
-26.5	-15.7	%1 11001011	\$01CB	-54.0	-65.2	%1 10010100	\$0194
-27.0	-16.6	%1 11001010	\$01CA	-54.5	-66.1	%1 10010011	\$0193
-27.5	-17.5	%1 11001001	\$01C9	-55.0	-67.0	%1 10010010	\$0192

**Cornerstone Electronics Technology and Robotics II  
Encoders Basics with PIC18F4331 Lab 1 – 4331\_encoder1**

- **Purpose:** The purpose of this lab is to acquaint the student with the basic operation of an encoder hardware and circuit.

- **Apparatus and Materials:**

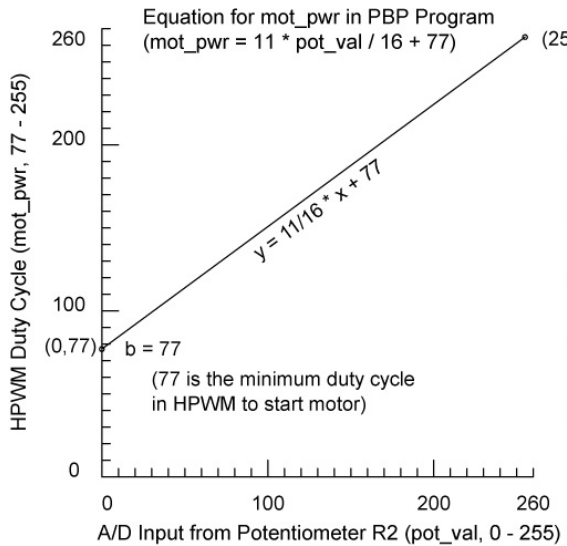
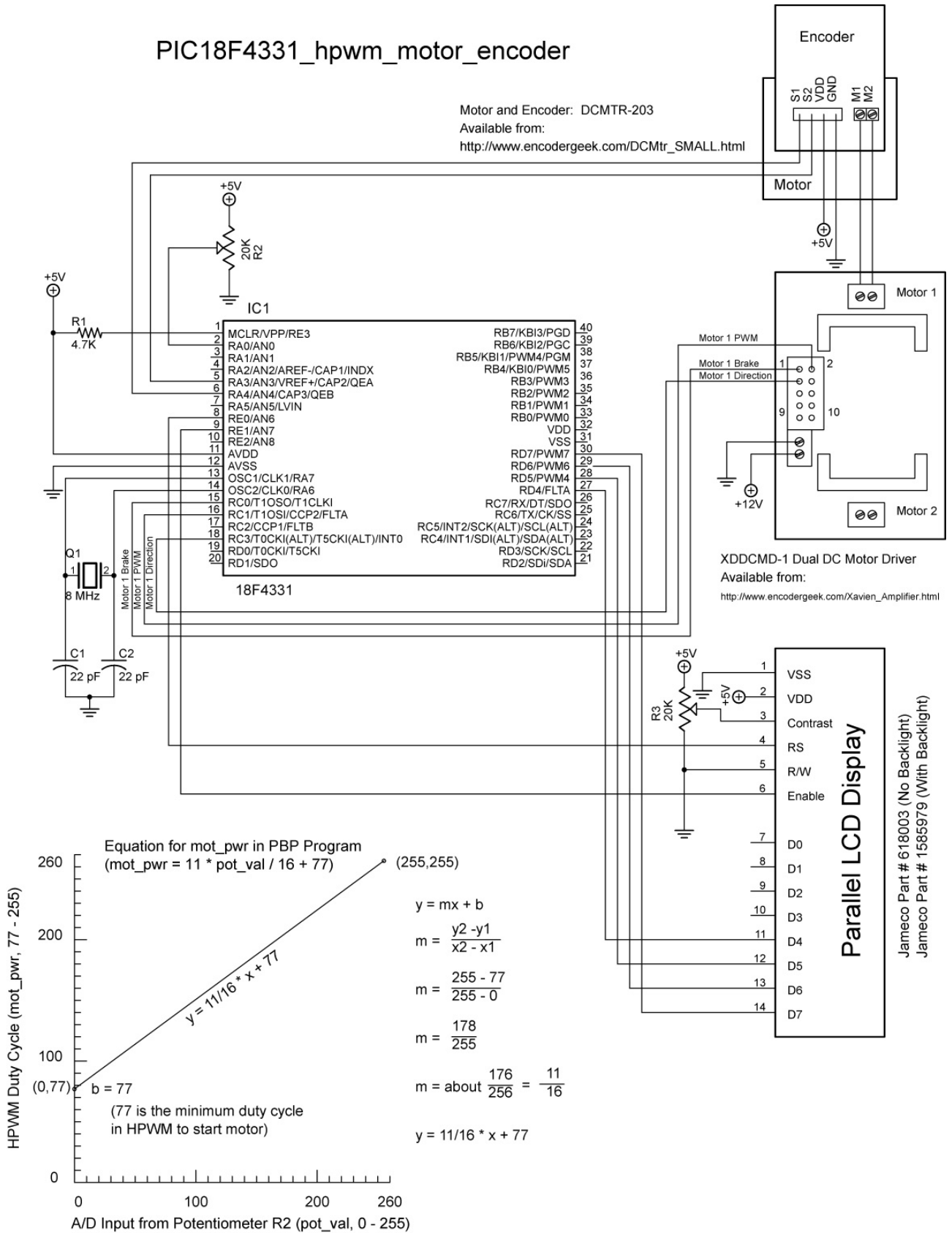
- 1 – Breadboard with +5 V and +12V Power Supplies
- 1 – PIC18F4331 Microcontroller, Datasheet at:  
<http://ww1.microchip.com/downloads/en/DeviceDoc/39616b.pdf>
- 1 – 4 MHz Crystal
- 2 – 22pF Capacitors
- 1 – Xavien 2 Motor Driver “XDDCMD-1”, Available at:  
[http://www.encodergeek.com/Xavien\\_Amplifier.html](http://www.encodergeek.com/Xavien_Amplifier.html)  
The user manual is available at:  
[http://k5systems.com/XDDCMD\\_user\\_manual.pdf](http://k5systems.com/XDDCMD_user_manual.pdf)
- 1 – Small Motor with Quadrature Incremental Encoder:  
The motors are available at:  
[http://www.encodergeek.com/DCMtr\\_SMALL.html](http://www.encodergeek.com/DCMtr_SMALL.html)
- 1 – Hantronix HDM16216H-5-300S 16x2 LCD, Jameco #618003 See:  
[http://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&productId=618003&catalogId=10001&freeText=618003&app.products.maxperpage=15&storeId=10001&search\\_type=jamecoall&ddkey=http:StoreCatalogDrillDownView](http://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&productId=618003&catalogId=10001&freeText=618003&app.products.maxperpage=15&storeId=10001&search_type=jamecoall&ddkey=http:StoreCatalogDrillDownView)
- 2 – 20 K Potentiometer
- 1 – 4.7 K Resistor

- **Procedure:**

- Wire the PIC18F4331\_hpwm\_motor\_encoder circuit as shown on the next page.
- Program the PIC18F4331 with 4331\_encoder1.pbp. See:  
[http://cornerstonerobotics.org/code/4331\\_encoder1.pbp](http://cornerstonerobotics.org/code/4331_encoder1.pbp)
- Before connecting the 12V power source:
  - Rotate the encode wheel in both directions and observe the count on the LCD display.
  - Also note that changes in the motor power potentiometer R2 are reflected on the LCD display.
- Now connect the +12V power supply. If the count decreases in the 65,000s, switch the two encoder leads into the PIC18F4331.

# PIC18F4331\_hpwm\_motor\_encoder

Motor and Encoder: DCMTR-203  
 Available from:  
[http://www.encodergeek.com/DCMtr\\_SMALL.html](http://www.encodergeek.com/DCMtr_SMALL.html)

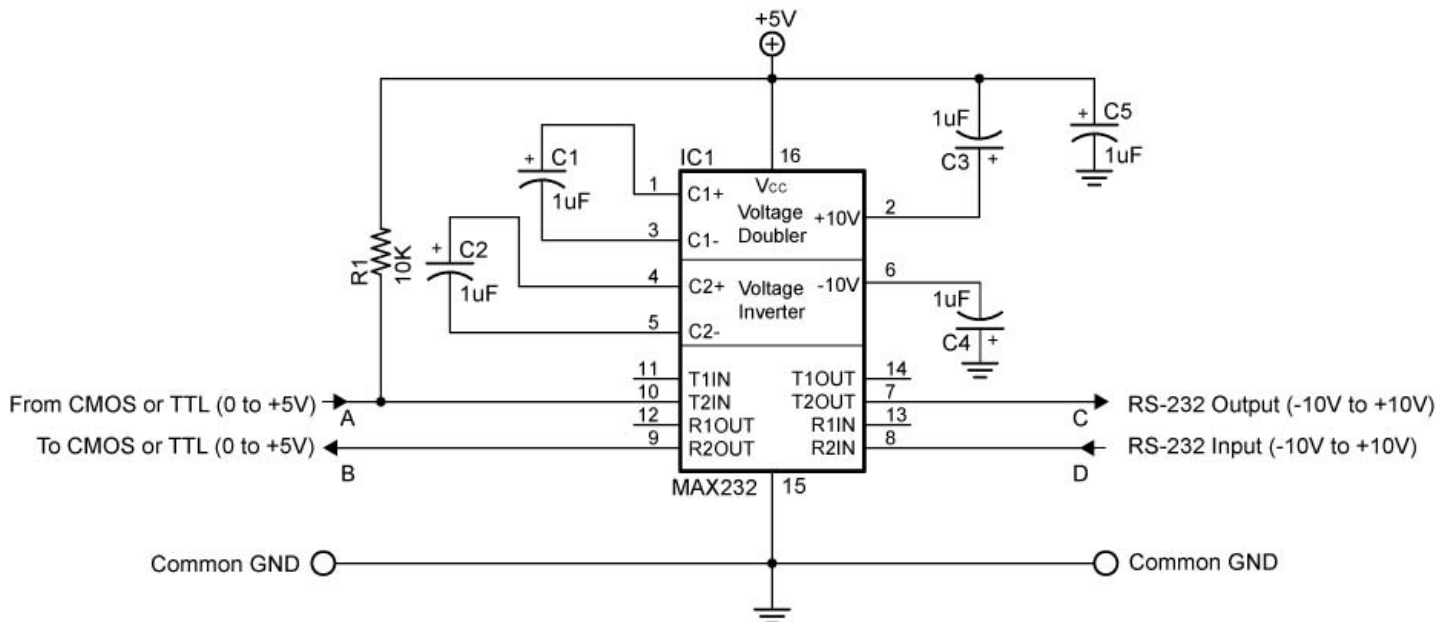


## RS-232 Serial Communication – Hardware – Lab 1, MAX232 Converter Chip

- **Purpose:** The purpose of this lab is to demonstrate the conversion of RS-232 signals to CMOS or TTL signals using the MAX232 converter chip.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – Function Generator
  - 1 - Oscilloscope
  - 1 – MAX232 RS232 Driver/Receiver
    - Different types of MAX232s require different external capacitors. For example, the MAX232A uses 0.1 uF capacitors.
    - MAX233 and MAX233A do not use external capacitors
    - The 10K pull-up resistor R1 helps with the MAX232 noise sensitivity
  - 5 – 1uF Capacitors
  - 1 – 10K Resistor
- **Procedure:**
  - Wire the circuit below. **Be very careful** that you install the capacitors with the polarity placed in the correct manner.
  - **RS-232 to TTL:**
    - Connect the oscilloscope Channel 1 to Point D, (RS-232 input), and the common ground.
      - View only Channel 1 trace.
      - Set the Channel 1 VOLT/DIV to 5V.
      - Set the TIME/DIV to 0.2 ms.
      - Position the trace in the center of the screen.
    - Also connect the function generator to Point D, (RS-232 input), and the common ground.
      - Set the waveform to generate a square wave.
      - Set the frequency to approximately 1 kHz.
      - Viewing Channel 1 on the oscilloscope, adjust the amplitude of the square wave such that it is from +10V to -10V. This square wave signal will simulate a RS-232 signal input into the MAX232.
      - Now position Channel 1 trace to top of the oscilloscope screen.
    - Connect the oscilloscope Channel 2 to Point B, (To CMOS or TTL), and the common ground.
      - View both Channel 1 and Channel 2 traces.
      - Set the Channel 2 VOLT/DIV to 5V.
      - Position the trace on the lower half of the screen.
    - Apply power to the circuit and verify that the MAX232 converts the RS-232 signal (+10V to -10V) in Channel 1 to a CMOS or TTL signal (0V to +5V) in Channel 2. Notice the inversion of the signal.

○ **TTL to RS-232:**

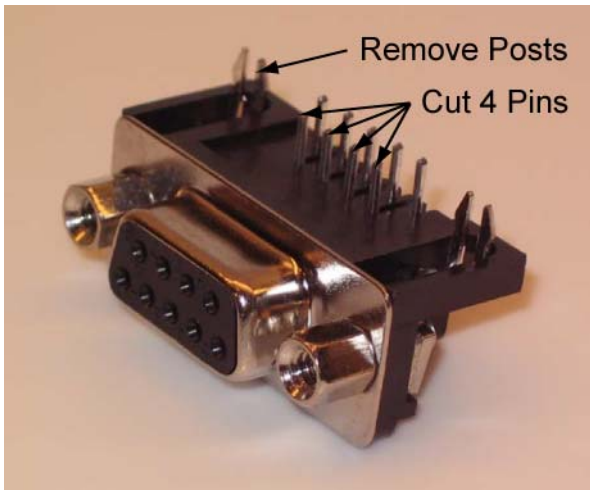
- Connect the oscilloscope Channel 1 to Point A, (From CMOS or TTL), and the common ground.
  - View only Channel 1 trace.
  - Set the Channel 1 VOLT/DIV to 2V.
  - Set the TIME/DIV to 0.2 ms.
  - Position the trace in the center of the screen.
- Also connect the function generator to Point A, (From CMOS or TTL), and the common ground.
  - Set the waveform to generate a square wave.
  - Set the frequency to approximately 1 kHz.
  - Viewing Channel 1 on the oscilloscope, adjust the amplitude of the square wave such that it is from +2.5V to -2.5V.
  - Use the offset control to make this signal shift to 0V to +5V. This square wave will simulate a CMOS or TTL signal input into the MAX232.
  - Now position Channel 1 trace to top of the oscilloscope screen.
- Connect the oscilloscope Channel 2 to Point C, (RS-232 Output), and the common ground.
  - View both Channel 1 and Channel 2 traces.
  - Set the Channel 2 VOLT/DIV to 5V.
  - Position the trace in the lower half of the screen.
- Apply power to the circuit and verify that the MAX232 converts the CMOS or TTL signal (0V to +5V) to a RS-232 signal (+10V to -10V). Notice the inversion of the signal.



**Figure 26-6: MAX232 Dual RS-232 Driver Circuit**

## Cornerstone Electronics Technology and Robotics II RS-232 Serial Communication – Software Lab 1 – Hello

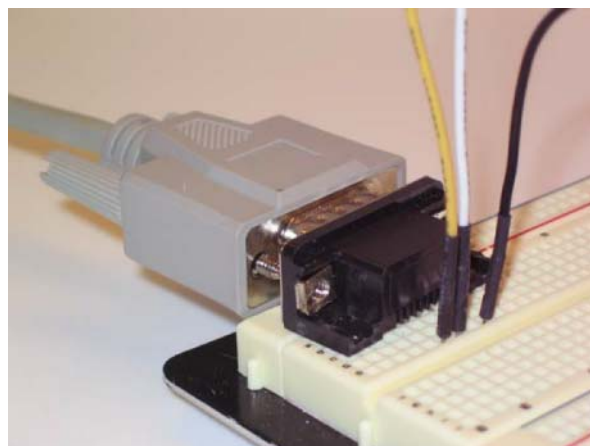
- **Purpose:** The purpose of this lab is to establish basic serial communications between the PIC microcontroller and a PC terminal program.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – MAX232 RS-232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 4.7K Resistor
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
- **Procedure:**
  - Wire the serout2\_hello circuit below. Read the schematic note.
  - To make connection between the DB9 connector and the breadboard, remove the mounting posts and 4 front pins on the connector. See the photos below. A header may be soldered onto the remaining 5 pins.



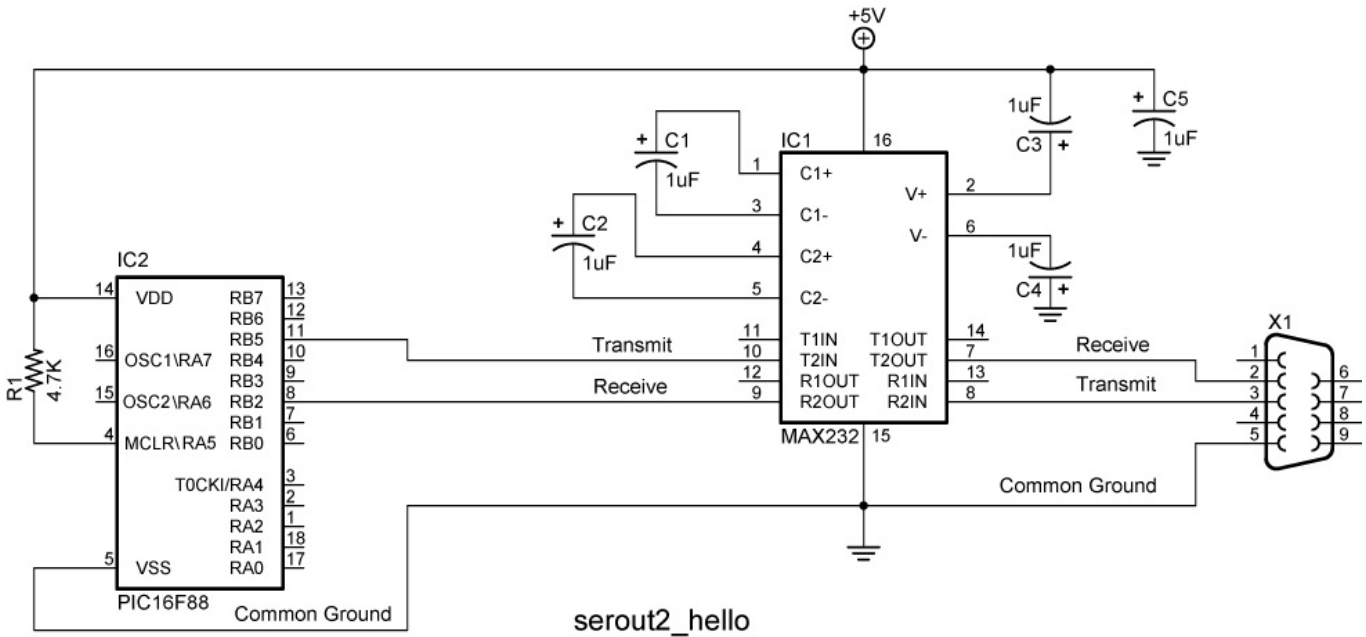
**New DB9 Connector**



**Modified DB9 Connector**



**Modified DB9 Connector Mounted on a Breadboard**



serout2\_hello

NOTE: Make certain that the polarity of each capacitor is wired correctly.

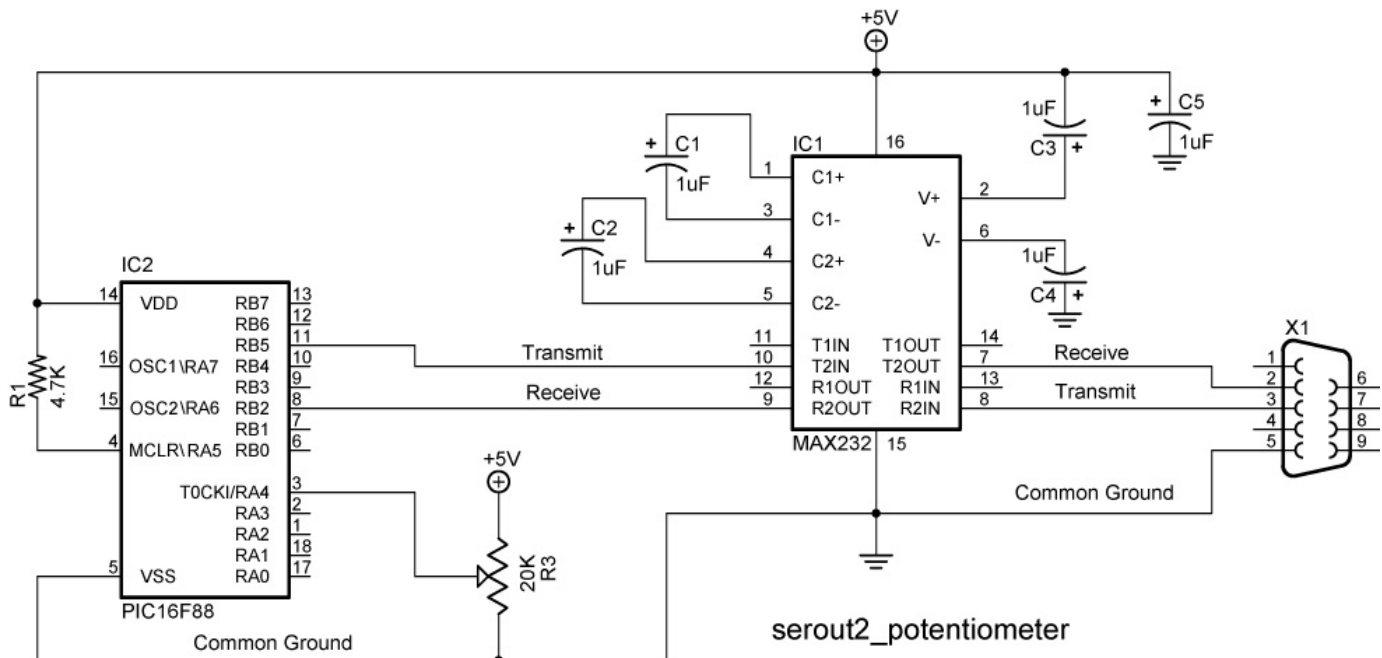
- Program the PIC16F88 with **serout2\_hello**. Copy file from: [http://www.cornerstonerobotics.org/code/serout2\\_hello.pbp](http://www.cornerstonerobotics.org/code/serout2_hello.pbp)
- Use either TeraTerm Pro 3.1.3 or HyperTerminal as the terminal program.
- For the PicBasic Pro program to work, the terminal program window must be the active window.
- Reprogram the PIC16F88 to display additional hellos.
- Save **serout2\_hello** as **serout2\_hello\_modified**.
  - Remove “10” and the “13” from the SEROUT2 command and note the changes in the display. Load the program several times to demonstrate the changes.
  - Reinsert “10” back into the SEROUT2 command. Load the program several times to demonstrate the changes. Note the response.
  - Remove “10” and reinsert “13” back into the SEROUT2 command. Load the program several times to demonstrate the changes. Note the response.



## Cornerstone Electronics Technology and Robotics II

### RS-232 Serial Communication – Software Lab 2 – Potentiometer 8-bit and 10-bit

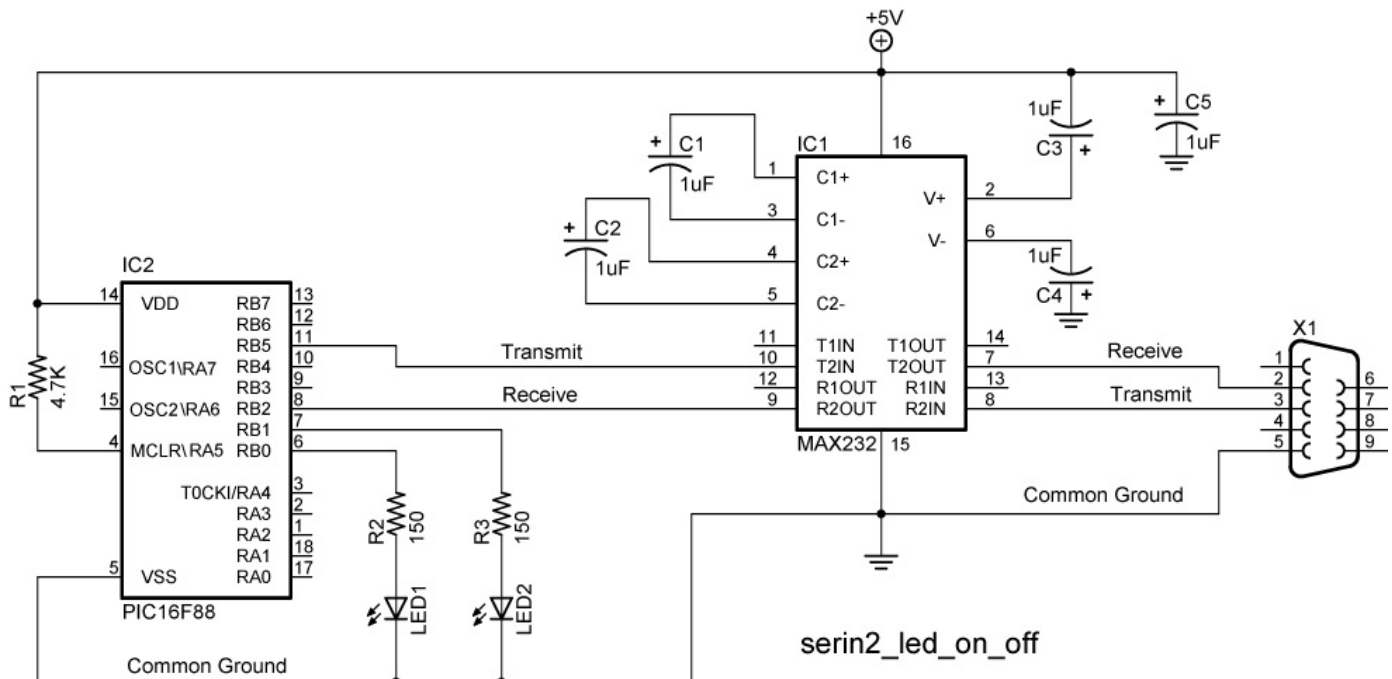
- **Purpose:** The purpose of this lab is to display an 8-bit potentiometer readings (0 – 255) and a 10-bit potentiometer readings (0 – 1023) on a PC terminal program.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – MAX232 RS-232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 4.7K Resistor
  - 1 – 20K Potentiometer
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
- **Procedure:**
  - Wire the serout2\_potentiometer circuit below.
  - Program the PIC16F88 with **serout2\_potentiometer\_8bit**. Copy file from: [http://www.cornerstonerobotics.org/code/serout2\\_potentiometer\\_8bit.pbp](http://www.cornerstonerobotics.org/code/serout2_potentiometer_8bit.pbp)
  - Adjust the potentiometer and note the range of readings on the PC terminal program.
  - Program the PIC16F88 with **serout2\_potentiometer\_10bit**. Copy file from: [http://www.cornerstonerobotics.org/code/serout2\\_potentiometer\\_10bit.pbp](http://www.cornerstonerobotics.org/code/serout2_potentiometer_10bit.pbp)
  - Again, adjust the potentiometer and note the range of readings on the PC terminal program.



NOTE: Make certain that the polarity of each capacitor is wired correctly.

## Cornerstone Electronics Technology and Robotics II RS-232 Serial Communication – Software Lab 3 – LED On/Off

- **Purpose:** The purpose of this lab is to control the output of a PIC from a PC terminal program.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – MAX232 RS232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 4.7K Resistor
  - 3 – 150 Ohm Resistors
  - 3 – LEDs
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
- **Procedure:**
  - Wire the serin2\_led\_on\_off circuit below.
  - Program the PIC16F88 with **serin2\_led\_on\_off**. Copy file from: [http://www.cornerstonerobotics.org/code/serin2\\_led\\_on\\_off.pbp](http://www.cornerstonerobotics.org/code/serin2_led_on_off.pbp)
  - Type in one of the command options, a, b, c, or d and observe the LED response and the output to the PC terminal program.
- **Challenge:**
  - Save **serin2\_led\_on\_off** as **serin2\_led\_on\_off\_modified**.
  - Add another LED to PORTB.4 and add two more options (“e” and “f”) to the command list to turn the new LED on and off from the PC.

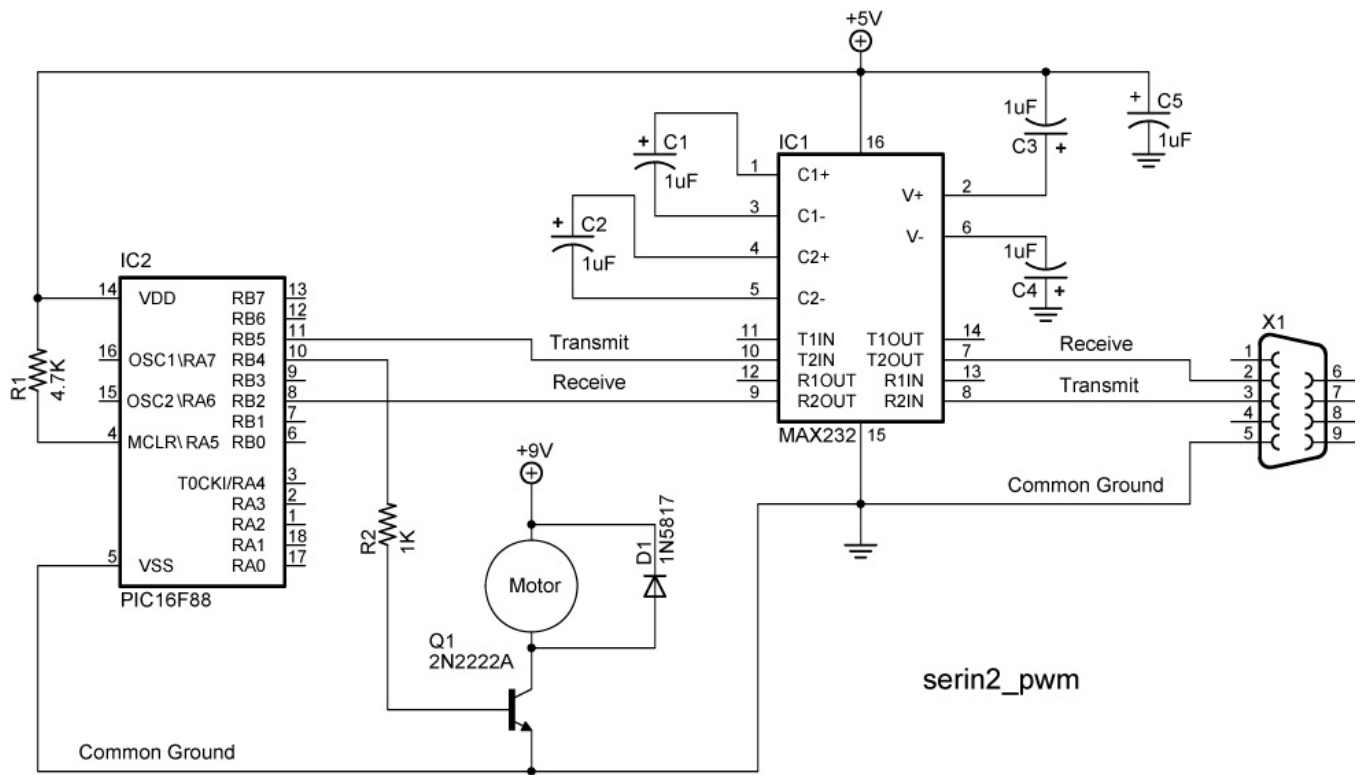


NOTE: Make certain that the polarity of each capacitor is wired correctly.

## Cornerstone Electronics Technology and Robotics II

### RS-232 Serial Communication – Software Lab 4 – PWM Motor Control

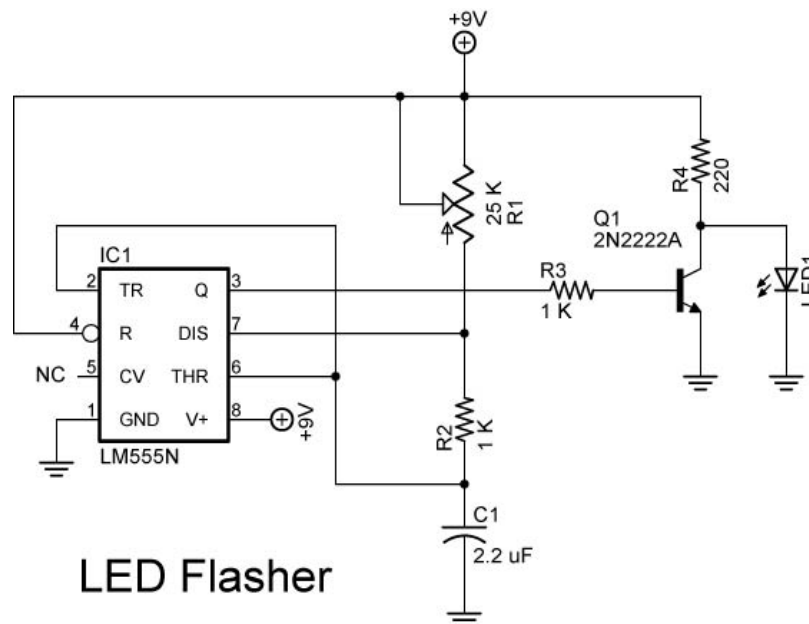
- **Purpose:** The purpose of this lab is to change the Pulse Width Modulation (PWM) to control a motor from a PC terminal program.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V and +9V Power Supplies
  - 1 – MAX232 RS232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 4.7K Resistor
  - 1 – 1K Resistor
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
  - 1 – 1N5817 Diode
  - 1 – 2N2222A NPN Transistor
- **Procedure:**
  - Wire the serin2\_pwm circuit below.
  - Program the PIC16F88 with **serin2\_pwm**. Copy file from: [http://www.cornerstonerobotics.org/code/serin2\\_pwm.pbp](http://www.cornerstonerobotics.org/code/serin2_pwm.pbp)
  - Type in one of the command options, a, b, c, or d and observe the LED response and the output to the PC terminal program.
  - Program the PIC16F88 with **serin2\_pwm\_wait**. Copy file from: [http://www.cornerstonerobotics.org/code/serin2\\_pwm\\_wait.pbp](http://www.cornerstonerobotics.org/code/serin2_pwm_wait.pbp)
  - Follow the program instructions and change the motor speed from the PC.
- **Challenge:**
  - Save **serin2\_pwm\_wait** as **serin2\_servo1**. Copy **servo1.pbp** from <http://www.cornerstonerobotics.org/code/servo1.pbp>. Use **serin2\_servo1** and **servo1.pbp** as a references to control a servo from a PC.




NOTE: Make certain that polarity of each capacitor is wired correctly.

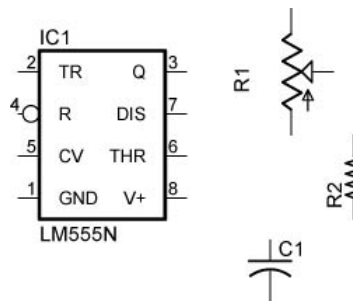
## Cornerstone Electronics Technology and Robotics II EAGLE Tutorial Lab 1 – Making a Schematic

- **Purpose:** The purpose of this lab is to teach the student how to make a simple schematic drawing.
- **Apparatus and Materials:**
  - 1 – PC or Laptop with Internet Connection
- **Procedure:**
  - Start up the EAGLE software.
  - Click the Options/Directories and change the Project default to a location you will use for your files
  - Click on File/New/Project to create a new project.
  - Click on File/New/Schematic to open up a new schematic file within your project.
  - Use File/Save As to assign a name to the schematic.
  - Note on Backup Files: Whenever you save a library, schematic, or PCB design, EAGLE copies the previously-saved design to a backup file (e.g. led\_flasher.s#1 for led\_flasher.sch). Up to ten previously-saved versions are kept, named #1 through #9. These files can help if you make a mistake and need to revert to an earlier version.
  - Right click on the light gray tool bar at the top of the window. In the menu that appears, Actions, Parameters, Command buttons, and Command texts should be checked.
  - This lesson will develop the following LED Flasher schematic:



- **Adding Parts to the Schematic:**

- From the Command buttons on the left side of the screen, click on the Add button  to select a part.
- 555 Timer: Select the LINEAR library and then choose LM555N from the 555 Timer part list library. Left click twice on the part to select the part, and then left click once to place the part. You may rotate the part around by right-clicking.
- Potentiometer R1: From the POT library, select TRIM\_US. We then selected the TRIM\_US-B25U package. Left click the TRIM\_US-B25U twice and place it on the schematic.
- Resistor R2: From the RCL library, select R-US. Double click on R-US\_0204/7 and place the resistor R2 on the schematic.
- Capacitor C1: From the RCL library, select C-US. Double click on C-US025\_025X050 and place the capacitor C1 on the schematic.
- To this point, you should have something that looks like this:




- **Resizing the Schematic:**

- If the drawing becomes too crowded, click on the Out button in the Action tool bar at the top of the window. Add an additional part to the schematic, then click on the Fit button to have the drawing fit the screen.

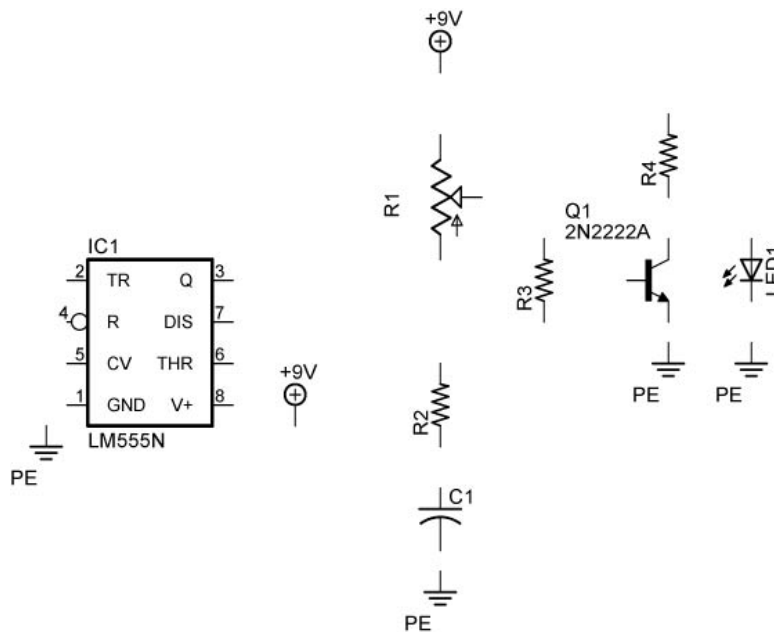
- **Deleting Parts:**

- Select the Delete button and left click on the part to be deleted.

- **Adding the Remaining Parts:**

- Transistor Q1: From TRANSISTOR-NPN library, select 2N2222 and then double click on 2N2222A. Place the transistor in the schematic.
- LED1: From LED library, select LED and then select LEDSFH486. Place the LED1 in the schematic.
- +9V: From SUPPLY2, select +9V and add it to the schematic.
- Ground: From SUPPLY2 select PE and add it to the schematic.
- Click on the Stop Sign  to go back to select mode.

- **Copying Parts:**
  - Resistors R3 and R4: Select the Copy button and left click once on R2. Move the new resistor to the side and left click again making a copy, R3. Click on R3 to make a copy which becomes R4.
  - +9V: Copy +9V symbol and place it close to pin 8 of the 555 timer.
  - Ground: Copy the ground symbol three times and place one by in 1 of the 555 timer, one under Q1 and the other under LED1.
- **Moving Parts:**
  - Your schematic should have all of the following parts: The parts should be in these approximate positions. To move parts around, select the Move button and left click and hold on the part. Drag the part to the new position and release.



- **Rotating Parts:**
  - Rotate R3: Select the Rotate button from the tool menu and then left click on R3. Repeated clicks on R3 will rotate R3 an additional 90 degrees. Rotate R3 into the horizontal position. When using the Rotate tool, place the cursor over the + in the part to be rotated.
  - Rotate +9V: Rotate the +9V adjacent to pin 8 of the 555 timer. The line (wire) should face pin 8.
- **Creating a Mirror Image of Parts:**
  - Mirroring R1: Select the Mirror button and single click on the + in R1. This flips R1 creating the mirror image.

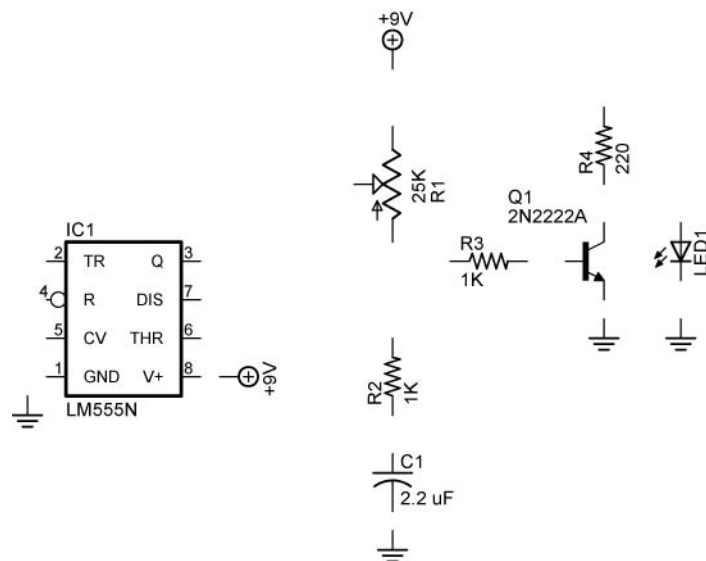
- **Adding Values to Parts:**

- Add values to R12, R2, R3, R4, C1, PE: Select the Value tool button, click on the part and then assign the new values shown below:

R1            25K  
 R2            1K  
 R3            1K  
 R4            220  
 C1            2.2uF

Grounds      When the warning comes up and asks you “Do you want to change it anyway?” answer “Yes”. In the Value window, delete the PE and press “OK”.


- **Final Parts Layout:**



- **Moving Groups of Parts:**

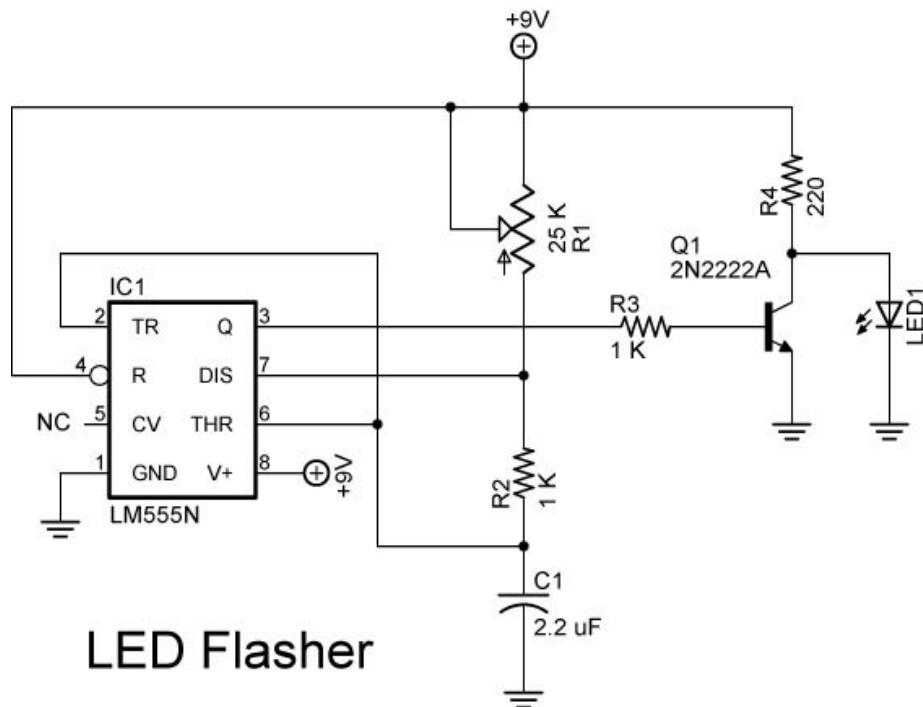
- If you want to move a group of parts, select the Group button and outline the group of parts that must be moved. The parts selected will change color. Now select the Move button and right click on the group of objects to be moved and relocate to the new position.

- **Connecting the Parts:**

- To draw nets (wires) in, select the Net button  (this is the one on the right side of the toolbar). Do not use the wire function – it just draws lines, but does nothing electrically! Left-click to begin drawing; click again when the net is done. If the Net function does not place the junction dots at connections, use the junction button to add the connections. Use the stop sign to end the Net function.



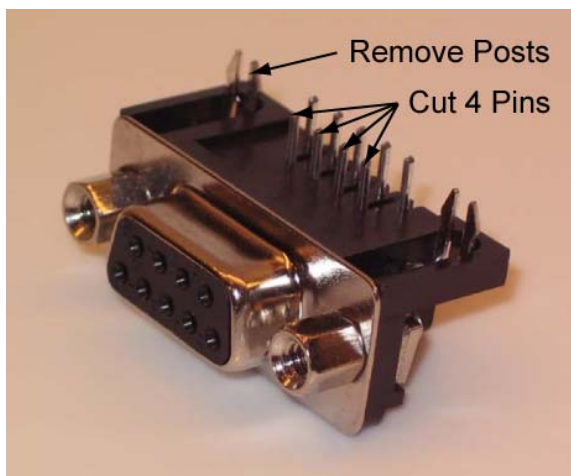
- **Adding Text:**
  - Click on the Text button and the parameters appear on the Parameters tool bar. Type in the desired text in the Text window and press “OK”. At this point you can change the parameters such as size or orientation. Add the text to the schematic. This completes the schematic LED Flasher:



- **Grid System:**
  - Grid is listed in the Command Text menu on the right side of the screen. Click on Grid which activates the Grid window. From the Grid window you can make adjustments to the grid properties such as Display, Style, Size, etc.

## Cornerstone Electronics Technology and Robotics II VB.NET and PicBasic Pro Lab 1 – LED On/Off

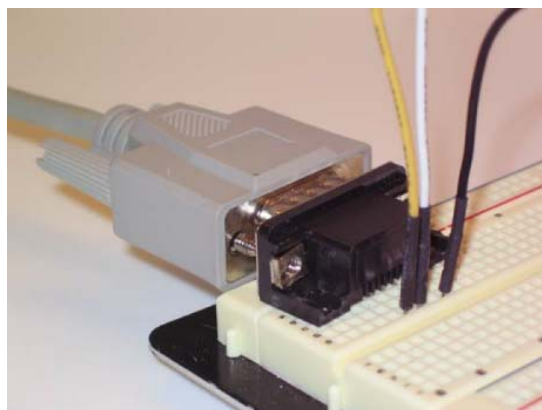
- **Purpose:** The purpose of this lab is to establish basic serial communications between Visual Basic.NET and a PIC microcontroller.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – MAX232 RS-232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 150 Ohm Resistor
  - 1 – 4.7K Resistor
  - 1 – LED
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
- **Procedure:**
  - Wire the vb\_led1 circuit below. Read the schematic note.
  - To make connection between the DB9 connector and the breadboard, remove the mounting posts and 4 front pins on the connector. See the photos below.



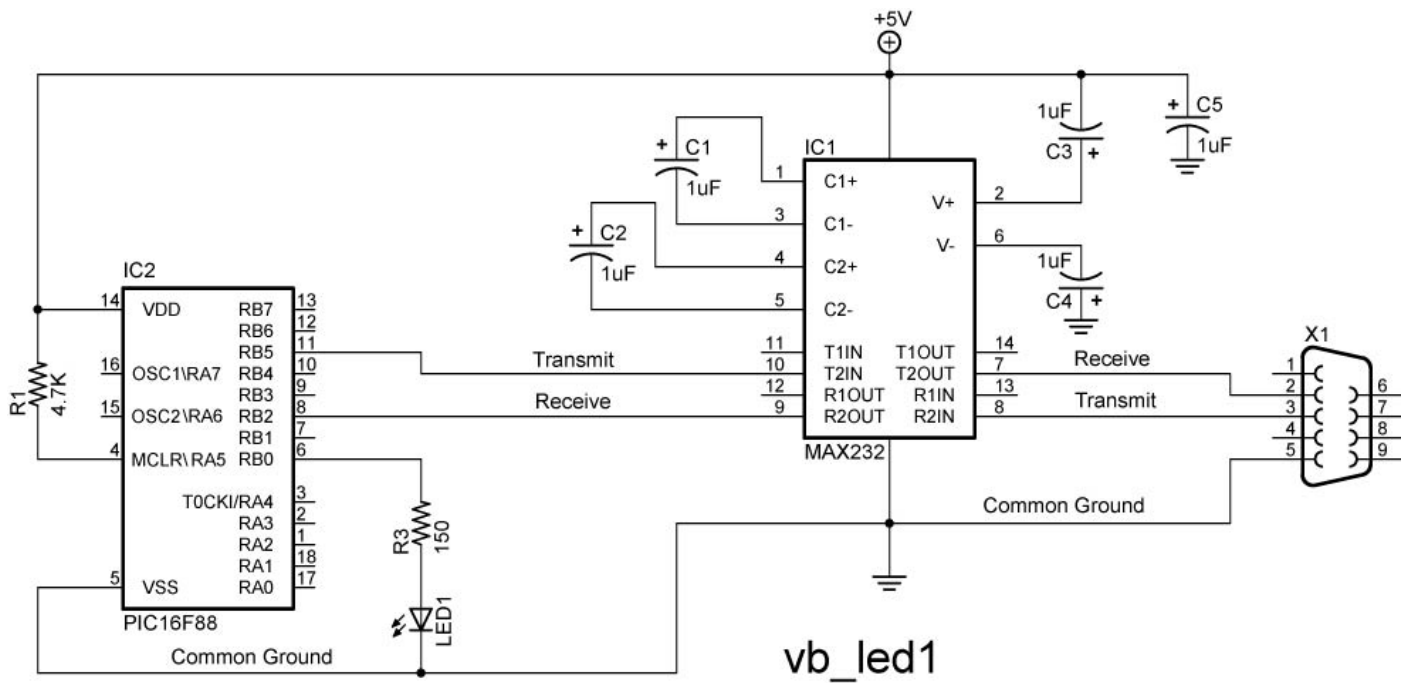
**New DB9 Connector**



**Modified DB9 Connector**



**Modified DB9 Connector Mounted on a Breadboard**



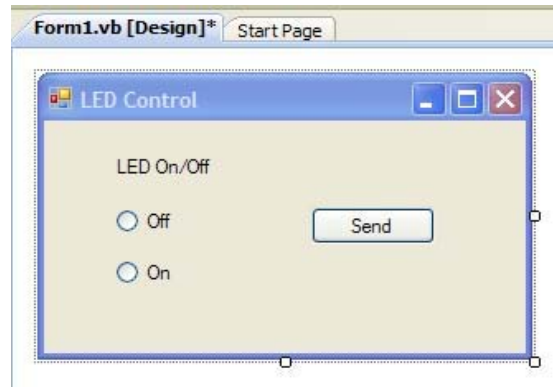
NOTE: Make certain that the polarity of each capacitor is wired correctly.

- Open the Visual Basic 2008 Express Edition
- Create a new project named vb\_led.
- From the Toolbox window on the left side of the IDE, add 1 label, 2 radiobuttons, and 1 button by dragging them over to the form.
- Set the properties to the following:

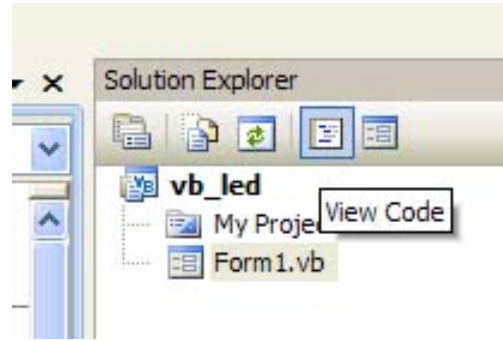
#### Properties for Objects on VB Form

Object	Property	Setting
Label1	Text	"LED On/Off"
Radiobutton1	Text	"Off"
Radiobutton2	Text	"On"
Button1	Name	"btnSend"
Button1	Text	"Send"
Form1	Text	"LED Control"
Form1	Name	"LED_Control"

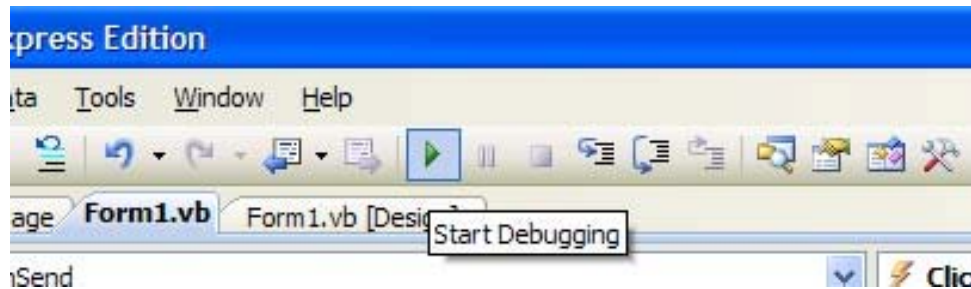
- Click on the form and resize to the general shape below. The form should look like this:



- Click on the View Code button:



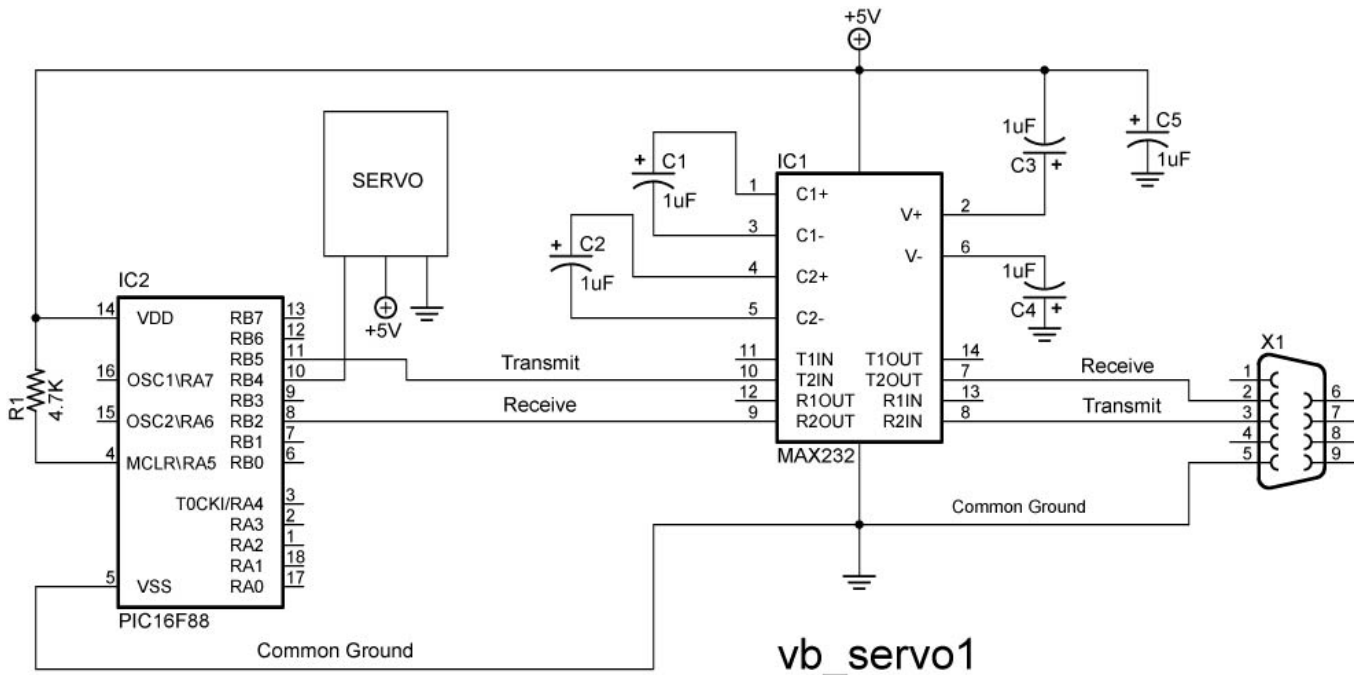
- Copy code from [http://cornerstonerobotics.org/code/vb\\_led1.pdf](http://cornerstonerobotics.org/code/vb_led1.pdf) onto the Code Editor.
- Make sure that “Handles btnSend.Click” is on the same line and follows “Private Sub btnSend\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)”
- Program the PIC16F88 with [http://cornerstonerobotics.org/code/pbp\\_vb\\_led1.pbp](http://cornerstonerobotics.org/code/pbp_vb_led1.pbp)
- Click the Start Debugging button on the Standard toolbar.



- Visual Basic displays the LED Control form in the Visual Studio IDE.
- Choose the HIGH RadioButton and press the Send button. The LED should light up. Choose the LOW RadioButton and press the Send button. The LED should turn off.

## Cornerstone Electronics Technology and Robotics II VB.NET and PicBasic Pro Lab 2 – Servo Position Control

- **Purpose:** The purpose of this lab is to send data in the form of a byte from Visual Basic.NET to a PIC microcontroller.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply
  - 1 – MAX232 RS-232 Driver/Receiver
  - 1 – PIC16F88
  - 5 – 1 uF Capacitors
  - 1 – 150 Ohm Resistor
  - 1 – 4.7K Resistor
  - 1 – Hobby Servo
  - 1 – Modified DB9 Connector - Connector,D-SUB,.318"RT,9P-F, Short Metal Housing, Jameco Part # 104952
- **Procedure:**
  - Wire the vb\_servo1 circuit below. Read the schematic note.



NOTE: Make certain that the polarity of each capacitor is wired correctly.

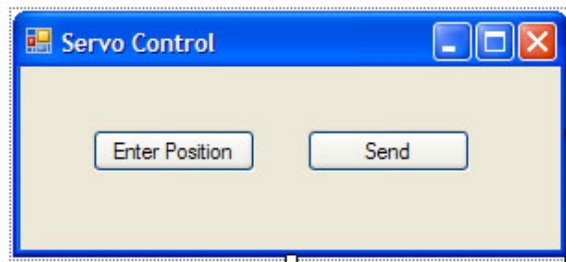
- WITH THE PIC16F88, MAKE SURE TO HAVE SEPARATE POWER SOURCES FOR THE PIC AND THE SERVO. MAKE SURE TO HAVE A COMMON GROUND BETWEEN THE PIC AND SERVO. We use one 9V battery and two 78L05 voltage regulators.
- Create a new project named vb\_servo.
- From the Toolbox, add 2 buttons to the form.

- Set the properties to the following:

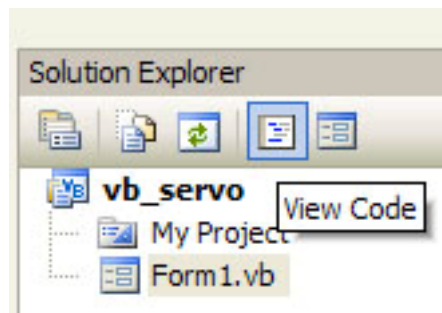
**Properties for Objects on VB Form**

Object	Property	Setting
Button1	Name	"btnEnter"
Button1	Text	"Enter Position"
Button2	Name	"btnSend"
Button2	Text	"Send"
Form1	Text	"Servo Control"

- Click on the form and resize to the general shape below. Now click on each button and expand so that "Enter Position" displays. The form should look like this:

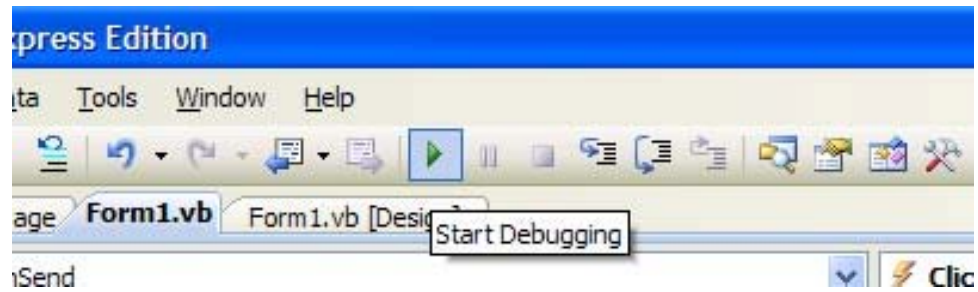


- Click on the View Code button:



- Copy code from [http://www.cornerstonerobotics.org/code/vb\\_servo1.pdf](http://www.cornerstonerobotics.org/code/vb_servo1.pdf) onto the Code Editor.
- As in Lab 1, make sure that both "Handles btnSend.Click"s are moved to the end of the line above.
- Program the PIC16F88 with [http://www.cornerstonerobotics.org/code/pbp\\_vb\\_servo1.pbp](http://www.cornerstonerobotics.org/code/pbp_vb_servo1.pbp)

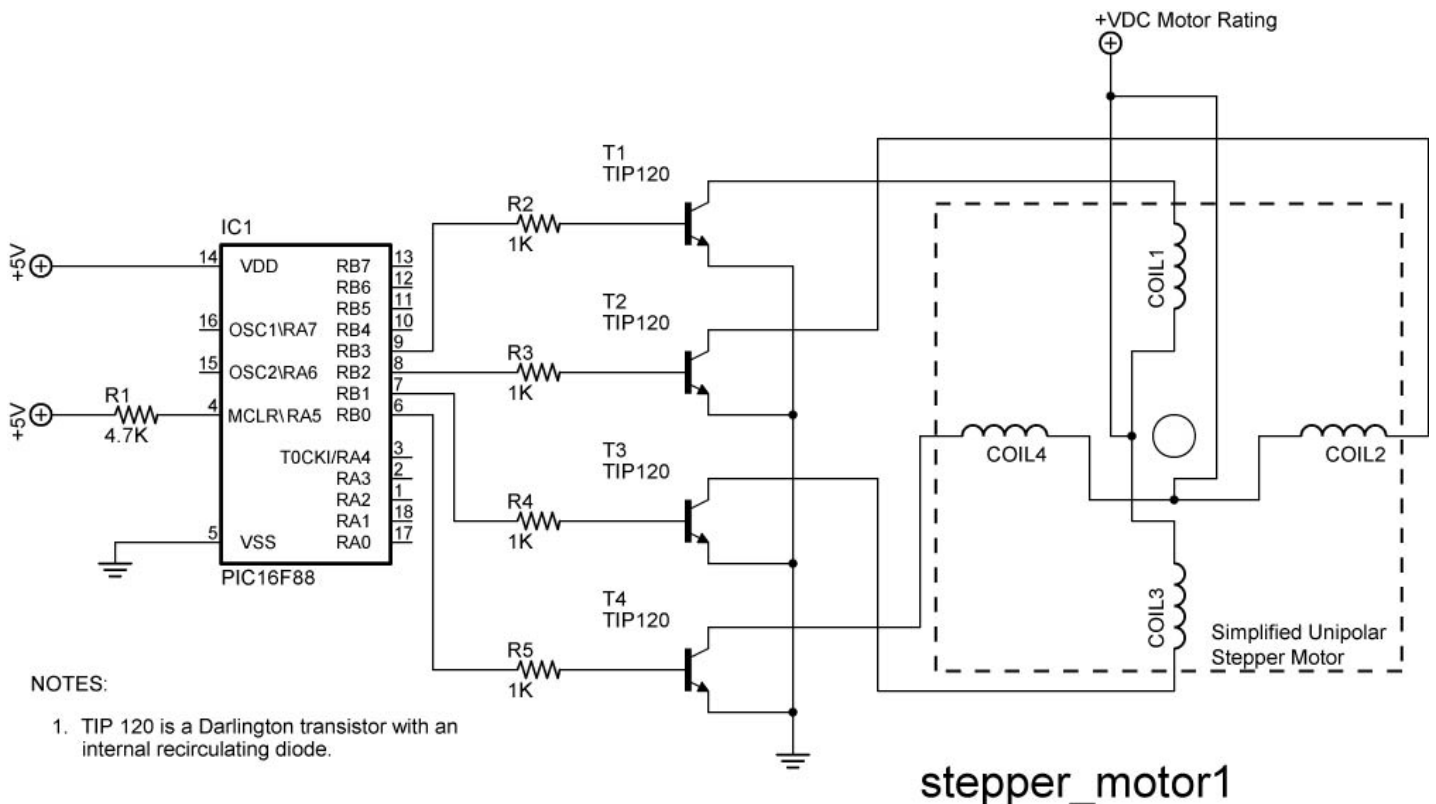
- Click the Start Debugging button on the Standard toolbar.



- Visual Basic displays the Servo Control form in the Visual Studio IDE.
- Click on the Enter Position button; the Servo Position Input window appears. Enter a number between 80 and 220 and press OK. Now press the Send button in the Servo Control window. If the servo does not respond, it may already be in that position so enter 220 and the servo should rotate.

## Cornerstone Electronics Technology and Robotics II Stepper Motor Lab 1 – Making a Stepper Motor Turn

- **Purpose:** The purpose of this lab is to program a PIC microcontroller with PicBasic Pro to drive a stepper motor.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply and a Power Supply Matching the Voltage Rating of the Stepper Motor. **Note:** The stepper motor power supply must be sufficiently robust or the motor will not rotate properly.
  - 1 – Jameco #237623 Unipolar Stepper Motor or Equivalent (#237623 – 4.8V, 1500mA)
  - 1 – Extech # 461895 Combination Contact/Photo Tachometer
  - 1 – PIC16F88
  - 1 – 4.7K Resistor
  - 4 – 1K Resistors
  - 4 – TIP 120 NPN Darlington Transistors
- **Procedure:**
  - Wire the stepper\_motor1 circuit below.
  - Program the PIC16F88 with **step\_mot1.pbp**.
  - Change the values of Delay and note the change in rpm.
  - Set the value of Delay to 250 and count the number of steps the motor takes to complete one revolution. From number of steps/revolution, calculate the angle in degrees of each step. Record your results.
  - Program the PIC16F88 with **step\_mot\_hi\_torque**. Compare the new torque generated with that generated in **step\_mot1.pbp**.



**NOTES:**

1. TIP 120 is a Darlington transistor with an internal recirculating diode.



- **Results:**

- Delay vs RPM:

Delay	RPM

- Number of Steps/Revolution:

Number of steps per revolution = \_\_\_\_\_ steps.

One revolution = \_\_\_\_\_ degrees.

The angle in degrees of each step = \_\_\_\_\_ degrees/step.

- **Challenge:**

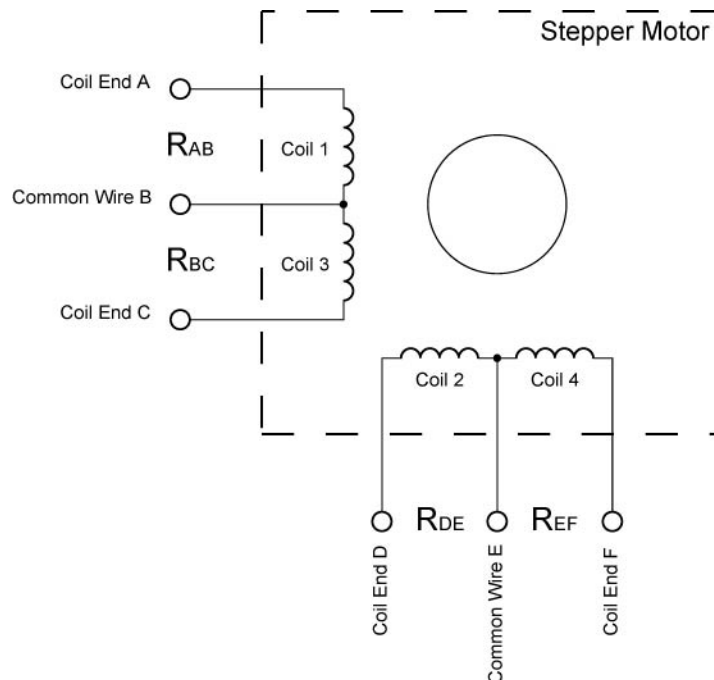
- Create a program called **step\_motor\_1\_sec.pbp** such that the stepper motor makes one revolution per second. First calculate the time for each step.
- Now write a program named **step\_motor\_1\_rev.pbp** so that the stepper will make only one revolution then stop. Knowing the number of steps in one revolution, use a FOR...NEXT loop to solve the challenge.

## Cornerstone Electronics Technology and Robotics II Stepper Motor Lab 2 – Figuring Out Stepper Motor Windings

- **Purpose:** The purpose of this lab is to figure out how to hook up a stepper motor with six leads when a data sheet for the motor is unavailable.
- **Apparatus and Materials:**
  - 1 – Breadboard with +5V Power Supply and a Power Supply Matching the Voltage Rating of the Stepper Motor. **Note:** The stepper motor power supply must be sufficiently robust or the motor will not rotate properly.
  - 1 – Ohmmeter
  - 1 – Jameco #237623 Unipolar Stepper Motor or Equivalent (#237623 – 4.8V, 1500mA, 1.8 Degree Step Angle or 200 Steps/Revolution )
  - 1 – PIC16F88
  - 1 – 4.7K Resistor
  - 4 – 1K Resistors
  - 4 – TIP 120 NPN Darlington Transistors
- **Procedure:**
  - Before hooking up the leads to the stepper motor, we must first determine the order of the lead connections. We do this by measuring resistances of the coils.
  - The resistance between common wire and coil-end wire is always about half of what it is between coil-end and coil-end wires. This is due to the fact that there is actually twice the length of coil between the ends and only half from center (common wire) to the end. From the figure below:

$$R_{AB} = R_{BC} = \frac{1}{2} R_{AC} \text{ and}$$

$$R_{DE} = R_{EF} = \frac{1}{2} R_{DF}$$



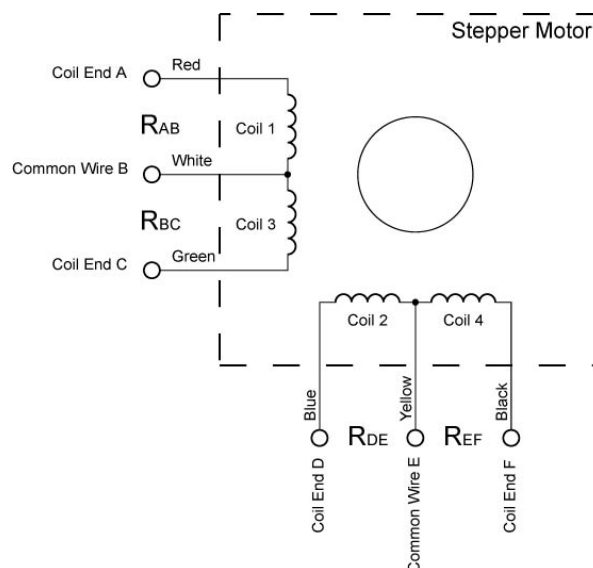
- There are no connections between Coils 1 & 2 and Coils 3 & 4; therefore the resistance between any of their leads is infinite.
- **Example:**
  - Assume we have a stepper motor with its colored leads and no data sheet to tell us how they are connected. The colors were chosen at random.

Test Stepper Motor	
Lead Colors	-
Black	
White	
Red	
Yellow	
Green	
Blue	

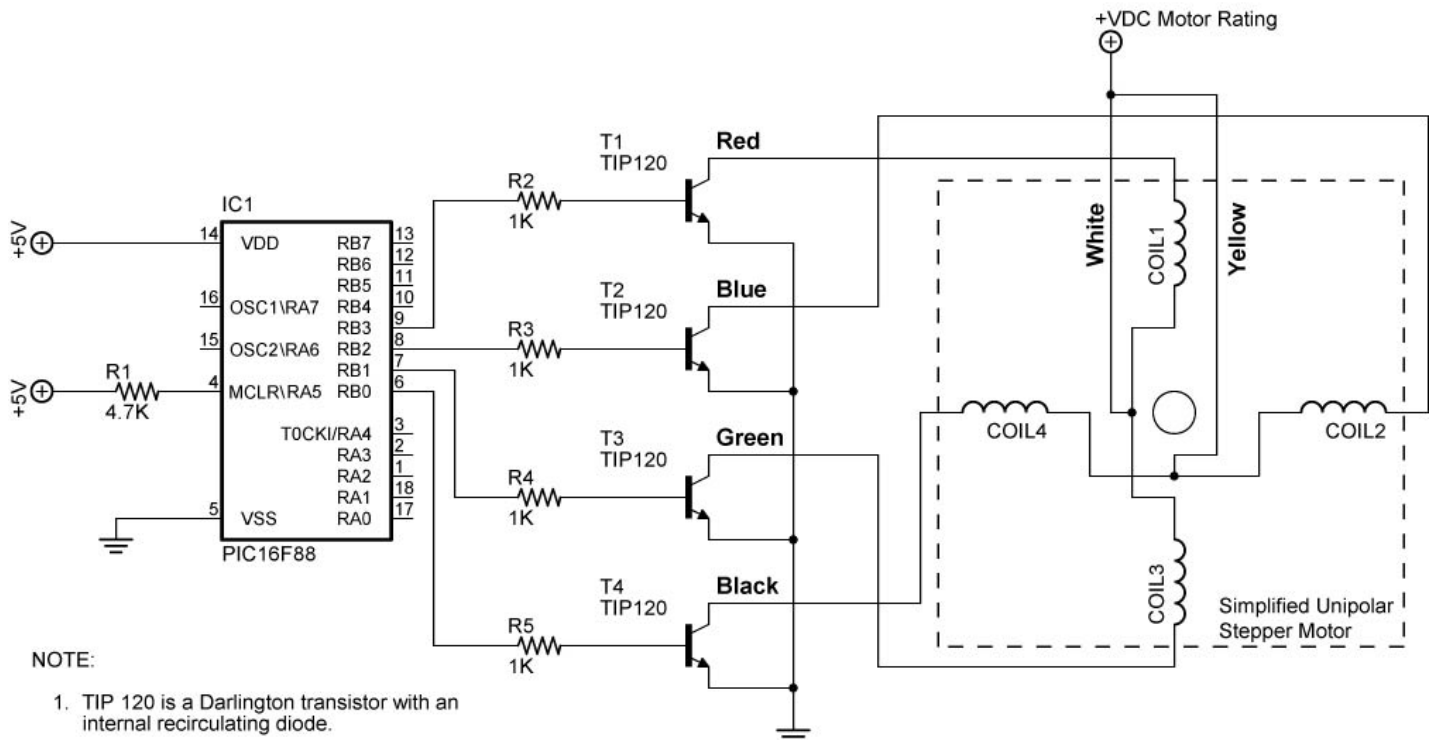
- Measure the resistances of different combinations of leads until you arrive at six lead combinations with measured resistances. See the example table below. Determine which groupings of three leads are interconnected.

Test Stepper Motor	
Lead Combinations	Resistance
Red - White	3.7 $\Omega$
Green - White	3.8 $\Omega$
Red - Green	6.9 $\Omega$
Blue - Yellow	3.6 $\Omega$
Black - Yellow	3.7 $\Omega$
Blue - Black	6.8 $\Omega$
All Other Combinations	Infinite

- From these resistance values, we see that the white lead is the common lead for one coil grouping and yellow lead is the common lead for the other coil grouping. Our stepper motor configuration is as follows:



- Now we can connect the stepper motor leads to the PIC driver circuit.
- First, the two common leads are both connected to a power source equal to the voltage rating of the stepper motor. See the stepper\_motor1\_lab2 schematic below.
- The other four leads are connected in a staggered manner to the TIP 120 transistors. In our example, if the red lead is connected to T1 in the following schematic (stepper\_motor1), then the other coil end lead in that grouping, green, must be connected to T3. If blue is connected to T2, then the other coil end lead in that grouping, black, must be connected to T4. See the stepper\_motor1\_lab2 schematic below.



stepper\_motor1\_lab2

- Now take the stepper motor furnished and determine the connection sequence. Use the tables in the results to organize your measurements.
- Connect to the PIC driver circuit.
- Program the PIC16F88 with **step\_mot1.pbp** and run the program. The stepper motor should rotate in a clockwise or counter-clockwise direction. Record the direction of rotation.
- Switch the leads connected to T2 and T4. Note the direction of rotation.

- **Results:**

- Lead Combination Resistances:

<b>Lab 2 Stepper Motor</b>	
<b>Lead Combinations</b>	<b>Resistance in Ohms</b>
All Other Combinations	

- Direction of Rotation:

- The original direction of rotation: \_\_\_\_\_
- The direction of rotation after switching leads T2 and T4:  
\_\_\_\_\_