Mechatronics

Topic #2

BS2 Programming, Hardware, & Interfacing

Programming Tutorial—I

- Data commands
  CON
  PIN
  RANDOM
  VAR
- Flow control
  Label
  END
  PAUSE
  GOTO
  IF (condition) THEN Label
  IF…ENDIF
  IF…THEN…ELSE…ENDIF
  BRANCH index, [Lbl_0 {, Lbl_1, …}]
  DO…LOOP
  DO WHILE (condition) …LOOP
  FOR…NEXT
  GOSUB …RETURN
- Analog I/O
  PWM pin, dutycycle, duration
  RCTIME pin, state, result
  FREQOUT pin, dur, freq 1 {, freq 2}
- Digital I/O
  INPUT
  OUTPUT
  HIGH
  LOW
  TOGGLE
  REVERSE
  PULSOUT pin, time
  PULSIN pin, state, result
  COUNT pin, time, result
  BUTTON pin, …, label
- Serial I/O
  SERIN pin, mode, {…} , [input]
  SEROUT pin, baudmode, {…} , [output]
  SHIFTIN datapin, clockpin, mode, {…}, [result]
  SHIFTOUT datapin, clockpin, mode, {…}, [data]
- Tables
  LOOKUP index, [val0, val1, …], result
  LOOKDOWN match, {operator} [val0, val1, …], result

Programming Tutorial—II

- Simple Program

```
statement 1
statement 2
END
```

- Unconditional Branching

```
Label:
        statement 1
        statement 2
        GOTO Label
```

- Conditional Branching

```
Start:
        statement 1
        statement 2
        IF (condition) THEN Start
        statement 3
```

Condition True

Condition False

- If…EndIf

```
Start:
        IF (condition) THEN
        statement 1
        statement 2
        ENDIF
        statement 3
```

Condition False

Condition True

Programming Tutorial—III

- If…Then…Else…EndIf

```
Start:
    IF (condition) THEN
    statement 1
    statement 2
    ELSE
    statement 4
    statement 5
    ENDIF
    statement 6
```

Condition False

Condition True

- Multiple If…Then

```
IF (index = 0) THEN Label_0
IF (index = 1) THEN Label_1
IF (index = 2) THEN Label_2
```

- Replace Multiple If…Then by Branch

```
BRANCH index, [Label_0, Label_1, Label_2]
```

- Unconditional Looping

```
Do
      statement 1
      statement 2
      LOOP
```

- Conditional Looping—I

```
DO WHILE (condition)
      statement 1
      statement 2
      LOOP
```

Condition True

Programming Tutorial—IV

- Conditional Looping—II

```
DO
    statement 1
    statement 2
    LOOP WHILE (condition)
```

- For…Next Looping

```
FOR ctrVar = startctr TO endctr STEP stepSize
    statement 1
    statement 2
    NEXT
```

- For…Next Looping, auto stepsize

```
FOR ctrVar = 1 TO 10
    statement 1
    statement 2
    NEXT
```

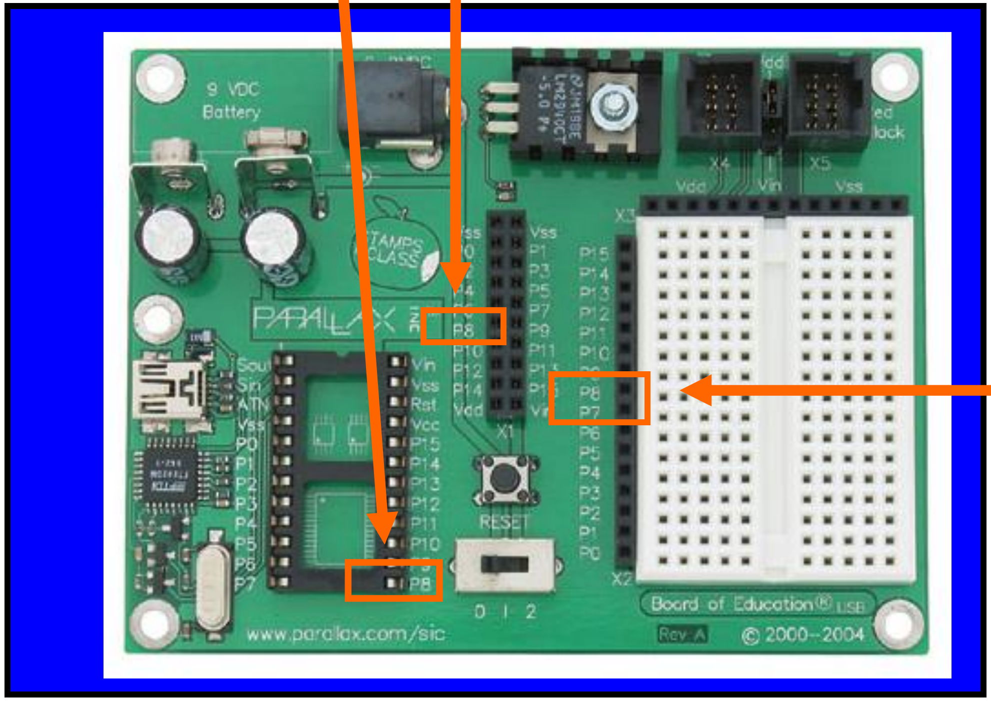
- Subroutines

```
Start:
    DO
    GOSUB My_Sub
    PAUSE 1000
    LOOP
My_Sub:
    statement 1
    statement 2
    RETURN
```

I/O Connections

- Code, such as **HIGH 8** or **LOW 8** will control a digital device connected to digital I/O pin P8 of Basic Stamp.

A connection to I/O pins P0—P15 is also available on the 'App-Mod' header.

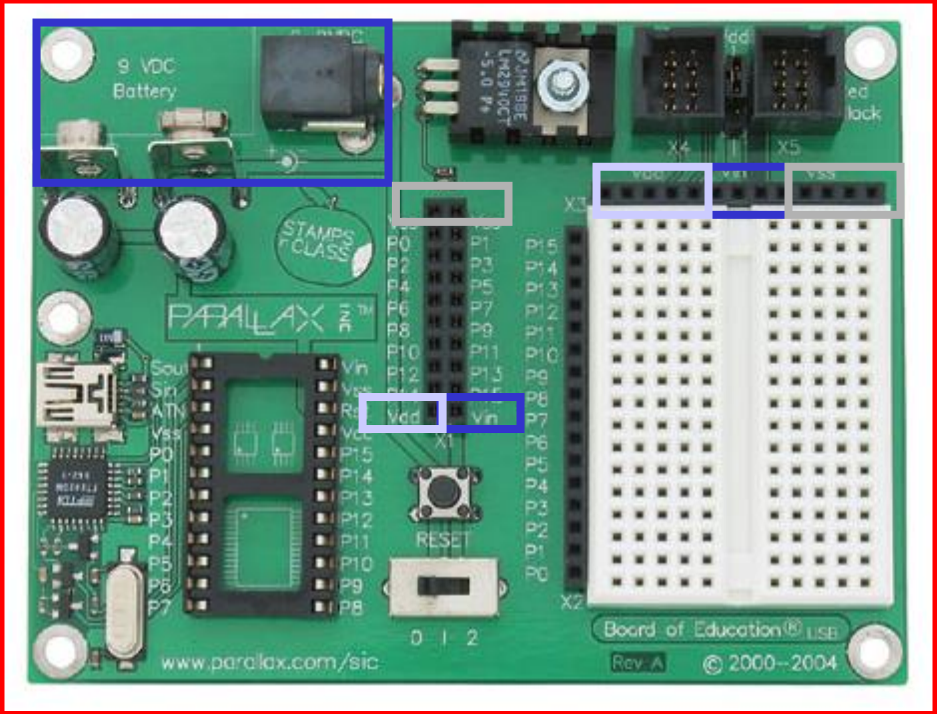


A connection to I/O pins P0—P15 can be made on the header next to the breadboard area.

Component Power Connections

- Power for the components are available on headers also.

+5V (Vdd) | 0V or ground (Vss) | Supply Voltage (Vin)

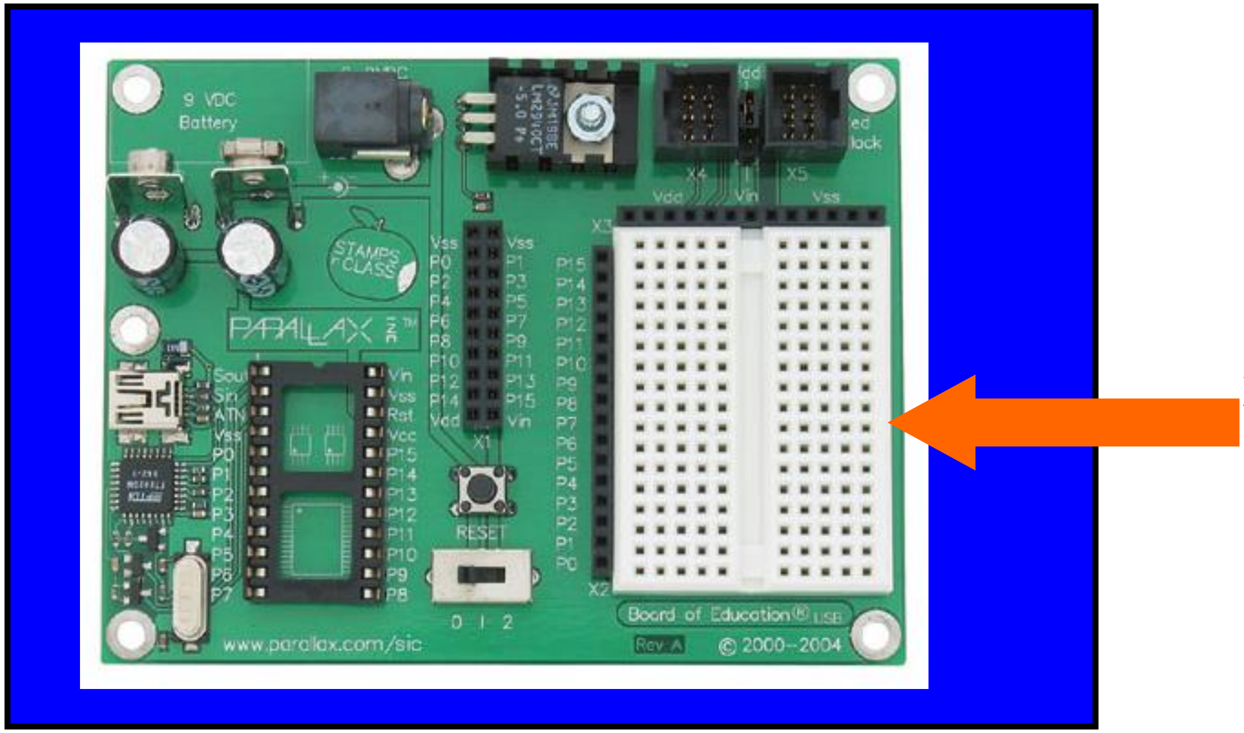


NOTE: Use of Vin may cause damaging voltages to be applied to the BASIC Stamp.

Use only under directed use!

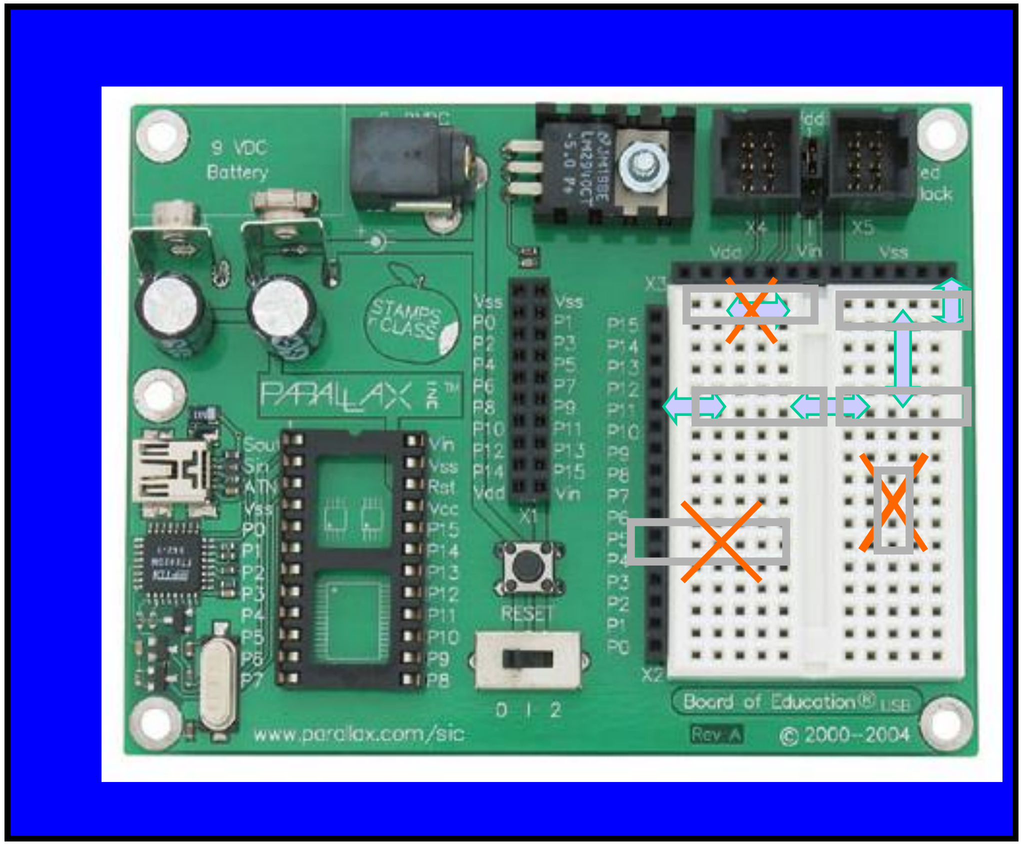
Connecting Components

- An important aspect to any BS2 project are the components that will be connected to the I/O pins of the Stamp.
- The breadboard allows quick connections for the components.



Breadboard Connections

- Breadboard are rows of connectors used to electrically connect components and wiring.



▭ Each row in each half of the breadboard are electrically the same point.

☒ There exist no connections between the headers and the breadboards or in columns on the breadboard.

↔ Components are connected between rows and to the headers to make electrical connections.

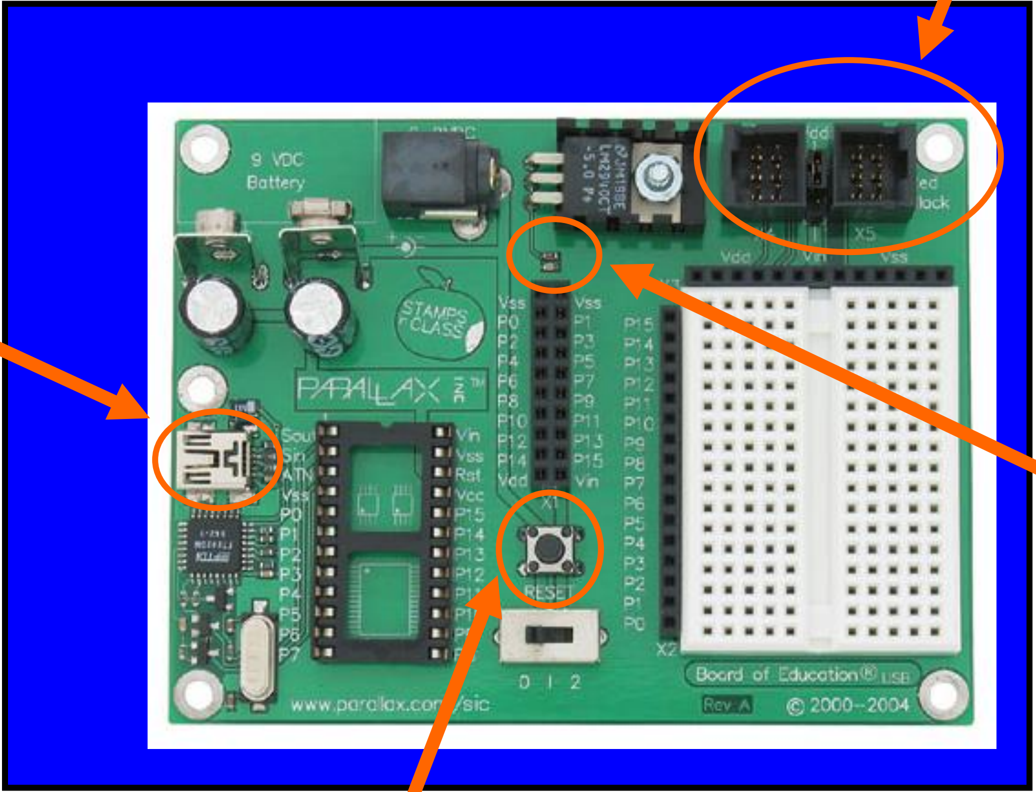
✗ Components should NOT be connected on a single row or they will be shorted out of the circuit.

Other Features

Servo Header Connectors

Caution: Jumper setting

Programming Port

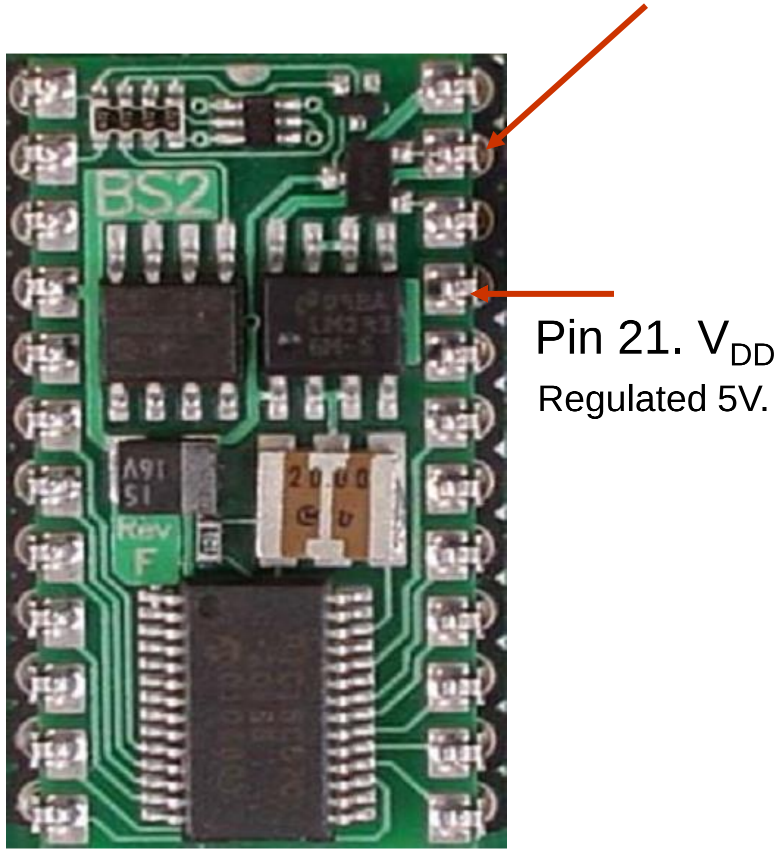


Power-On LED

Reset Button to restart the program on the BASIC Stamp

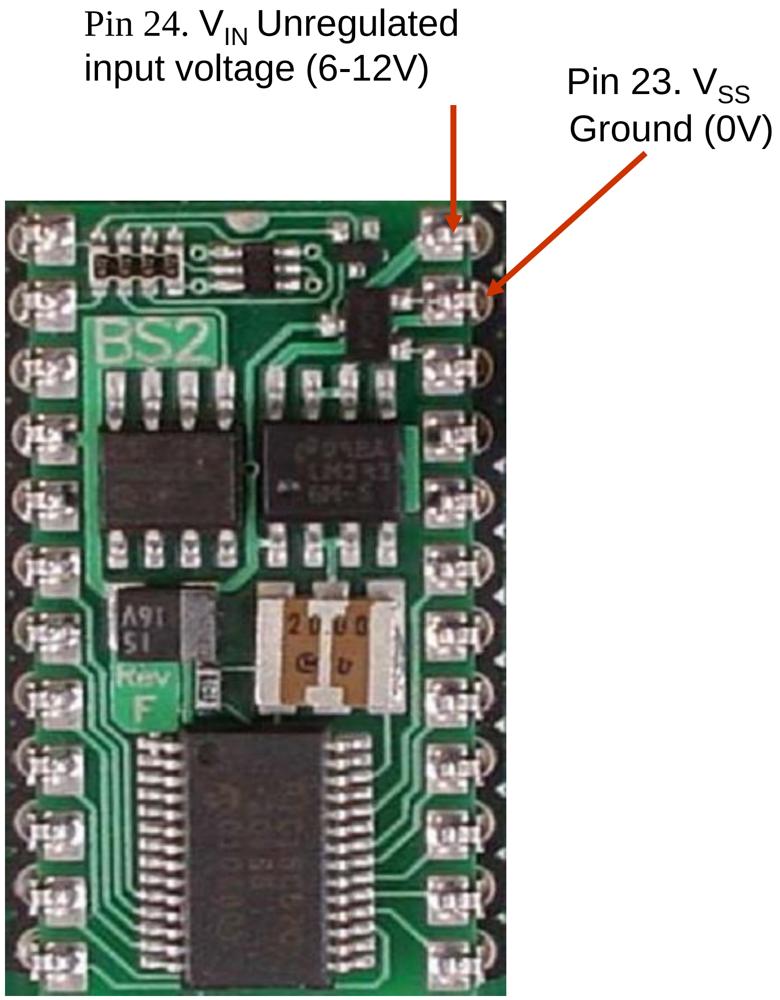
BS2 Hardware Consideration: Power Requirements —I

- BS2 requires 5VDC power supply and approximately 8mA
- When using regulated 5VDC external power supply
  - Connect +5V of supply to pin#21 (V_{dd}) of BS2
  - Connect ground/negative terminal of power supply to pin#23 (V_{ss}) of BS2



BS2 Hardware Consideration: Power Requirements —II

- When using an unregulated 6 — 12 VDC external power supply
  - Connect positive terminal of supply to pin#24 (V_{IN}) of BS2
  - Connect ground/negative terminal of power supply to pin#23 (V_{ss}) of BS2
  - With these connections, on-board voltage regulator of BS2 is powered and it outputs 5VDC for the operation of BS2


Pin 24. V_{IN} Unregulated input voltage (6-12V)
Pin 23. V_{SS} Ground (0V)
BS2

V_{ss}, V_{IN}, and V_{dd} Summary

- Pin#23 (V_{ss}): is connected to ground/negative terminal of power supply irrespective of whether a regulated 5VDC or an unregulated 6 — 12 VDC external supply is used. Vss provides common ground for the prototype circuit.

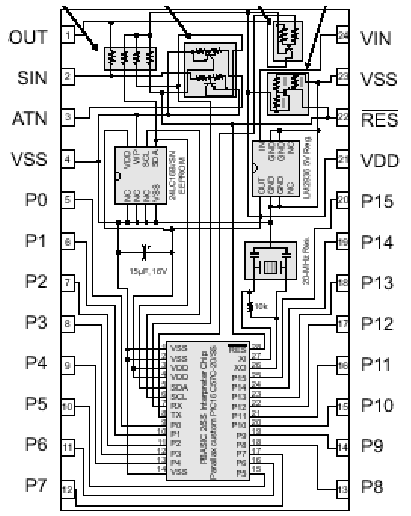
- Pin#24 (V_{IN}): is connected to positive terminal of power supply when 6 — 12 VDC external power supply is used. In this case, circuit elements should not be connected to V_{IN}. When a steady 5VDC external power supply is used, V_{IN} pin is left unconnected.

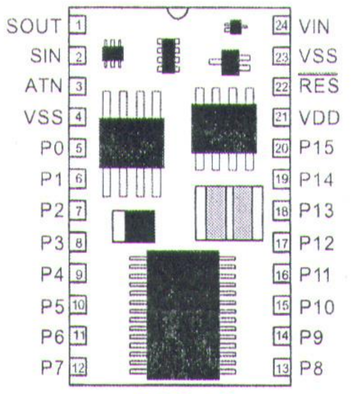
- Pin#21 (V_{dd}): is connected to positive terminal of power supply when regulated 5 VDC external power supply is used. When 6 — 12 VDC external power supply is connected to V_{IN} then V_{dd} can be:
  - left unconnected, or
  - it can be used to power other circuitry.

- When using V_{dd} to power external circuitry note that
  - V_{dd} outputs 5 VDC
  - The current consumption of external circuitry should not exceed capabilities of on-board regulator of BS2

BS2 Pin Descriptions

Pin	Name	Description
1	SOUT	Serial Out: connects to PC serial port RX pin (DB9 pin 2 / DB25 pin 3) for programming.
2	SIN	Serial In: connects to PC serial port TX pin (DB9 pin 3 / DB25 pin 2) for programming.
3	ATN	Attention: connects to PC serial port DTR pin (DB9 pin 4 / DB25 pin 20) for programming.
4	VSS	System ground: (same as pin 23) connects to PC serial port GND pin (DB9 pin 5 / DB25 pin 7) for programming.
5-20	P0-P15	General-purpose I/O pins: each can sink 25 mA and source 20 mA. However, the total of all pins should not exceed 50 mA (sink) and 40 mA (source) if using the internal 5-volt regulator. The total per 8-pin groups (P0 – P7 or P8 – 15) should not exceed 50 mA (sink) and 40 mA (source) if using an external 5-volt regulator.
21	VDD	5-volt DC input/output: if an unregulated voltage is applied to the VIN pin, then this pin will output 5 volts. If no voltage is applied to the VIN pin, then a regulated voltage between 4.5V and 5.5V should be applied to this pin.
22	RES	Reset input/output: goes low when power supply is less than approximately 4.2 volts, causing the BASIC Stamp to reset. Can be driven low to force a reset. This pin is internally pulled high and may be left disconnected if not needed. Do not drive high.
23	VSS	System ground: (same as pin 4) connects to power supply's ground (GND) terminal.
24	VIN	Unregulated power in: accepts 5.5 - 15 VDC (6-40 VDC on BS2-IC Rev. e, f, and g), which is then internally regulated to 5 volts. Must be left unconnected if 5 volts is applied to the VDD (+5V) pin.





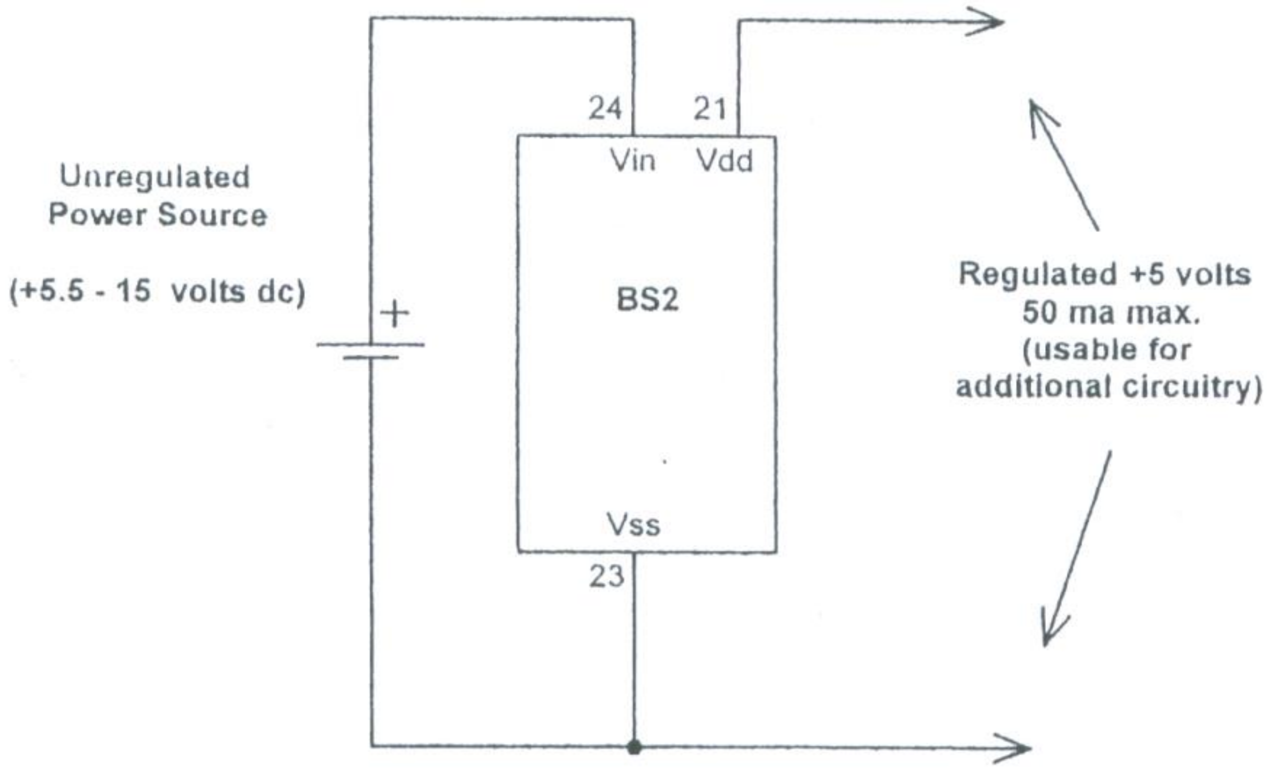
BS2-IC

I/O Pins Current Limitations —Individual Pins

- Each I/O pin of BS2 can
  - source up to 20mA current
  - sink up to 25mA current

I/O Pins Current Limitations —On-board Regulator

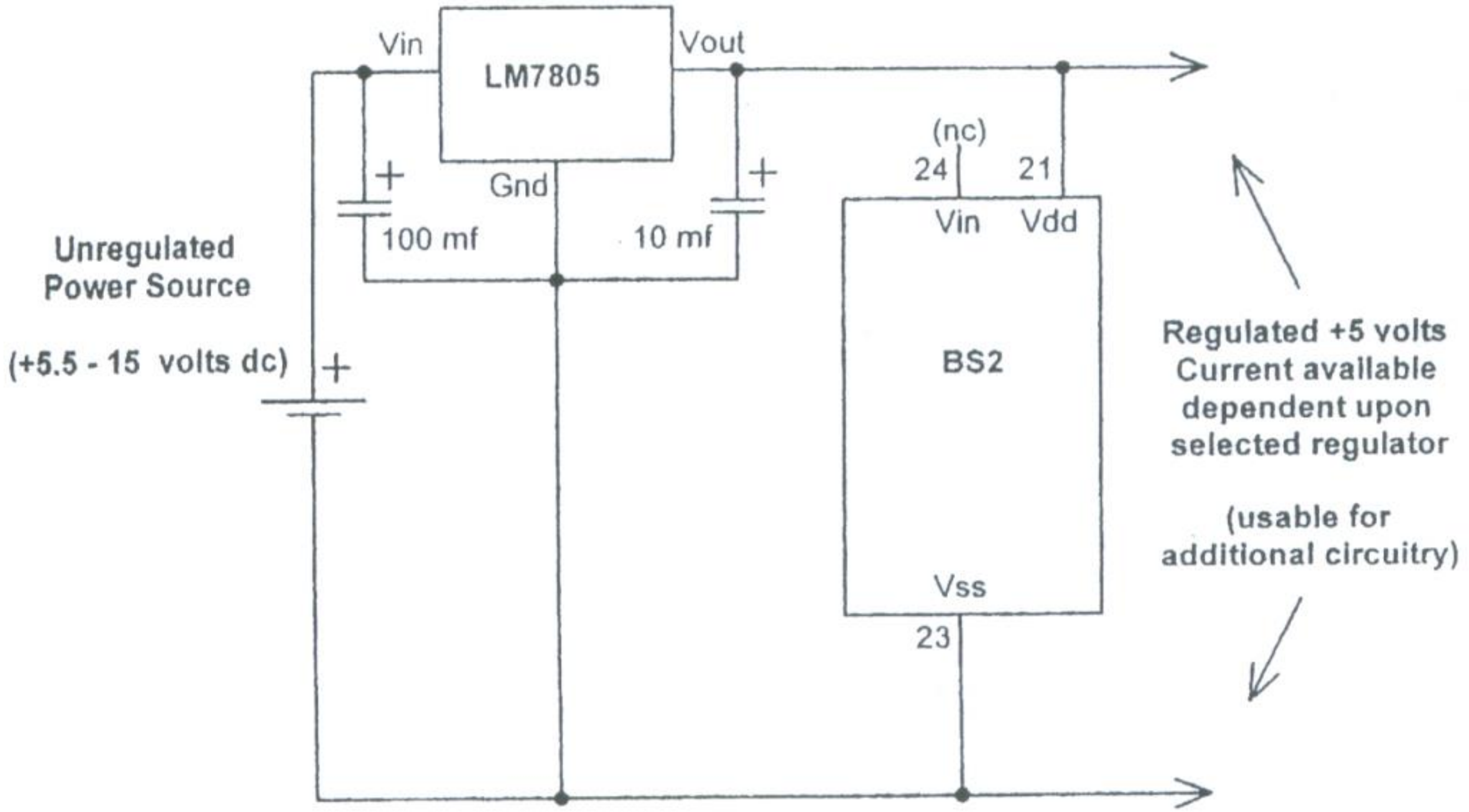
- If BS2’s on-board voltage regulator is used to power external circuits, all I/O pins as a group can
  - source up to 40mA current
  - sink up to 50mA current



Unregulated Power Source
(+5.5 - 15 volts dc)
+
24
21
Vin
Vdd
BS2
Vss
23
Regulated +5 volts 50 ma max. (usable for additional circuitry)

I/O Pins Current Limitations: External Regulator —I

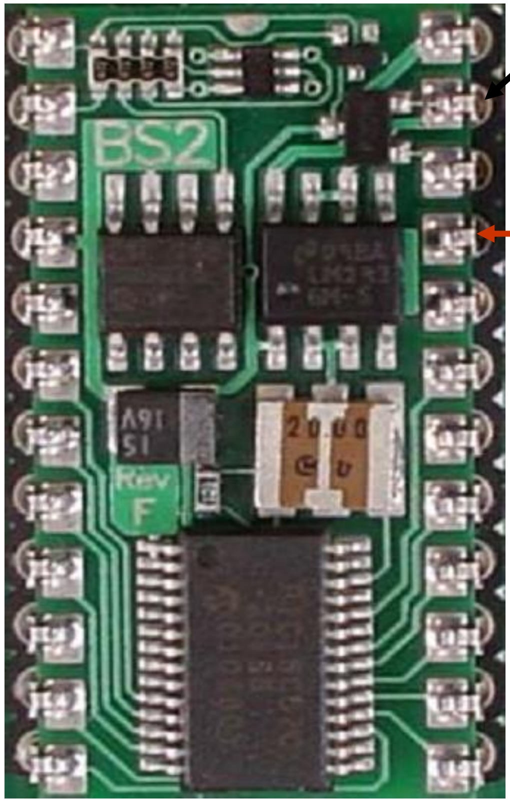
- When an external voltage supply with steady 5VDC is used to power BS2 and this external supply is capable of delivering at least 100mA, then each group of 8 I/O pins (P0-P7 and P8-P15) can
  - source up to 40mA current
  - sink up to 50mA current


Vin
LM7805
Vout
Gnd
+
100 mf
10 mf
+
(nc)
24
21
Vin
Vdd
Unregulated Power Source
(+5.5 - 15 volts dc)
+
BS2
Regulated +5 volts Current available dependent upon selected regulator
(usable for additional circuitry)
Vss
23

I/O Pins Current Limitations:External Regulator —II

Pin 23. V_{SS} Ground (0V)

Pin 21. V_{DD} connected to a 5V regulator with 100mA drive capability



Group limit: 40mA source and 50mA sink

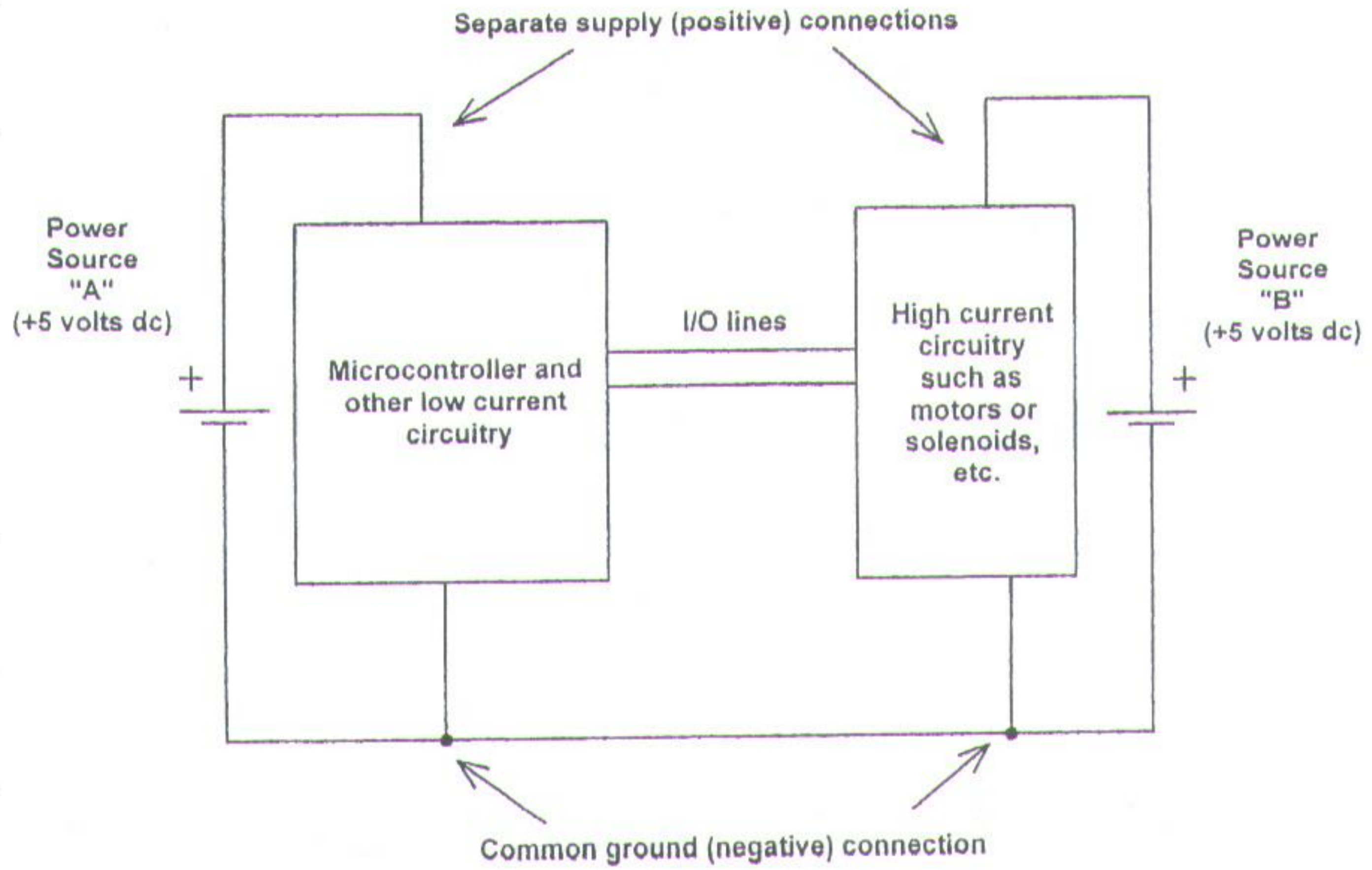
Group limit: 40mA source and 50mA sink

Current Considerations

- Devices such as solenoids, relays, motors, etc., require current beyond BS2’s capability.
  - Solution: Use a separate power supply to power up the device with high current need.
- When using one power supply for BS2 and another for high current load, two supplies must be properly isolated.

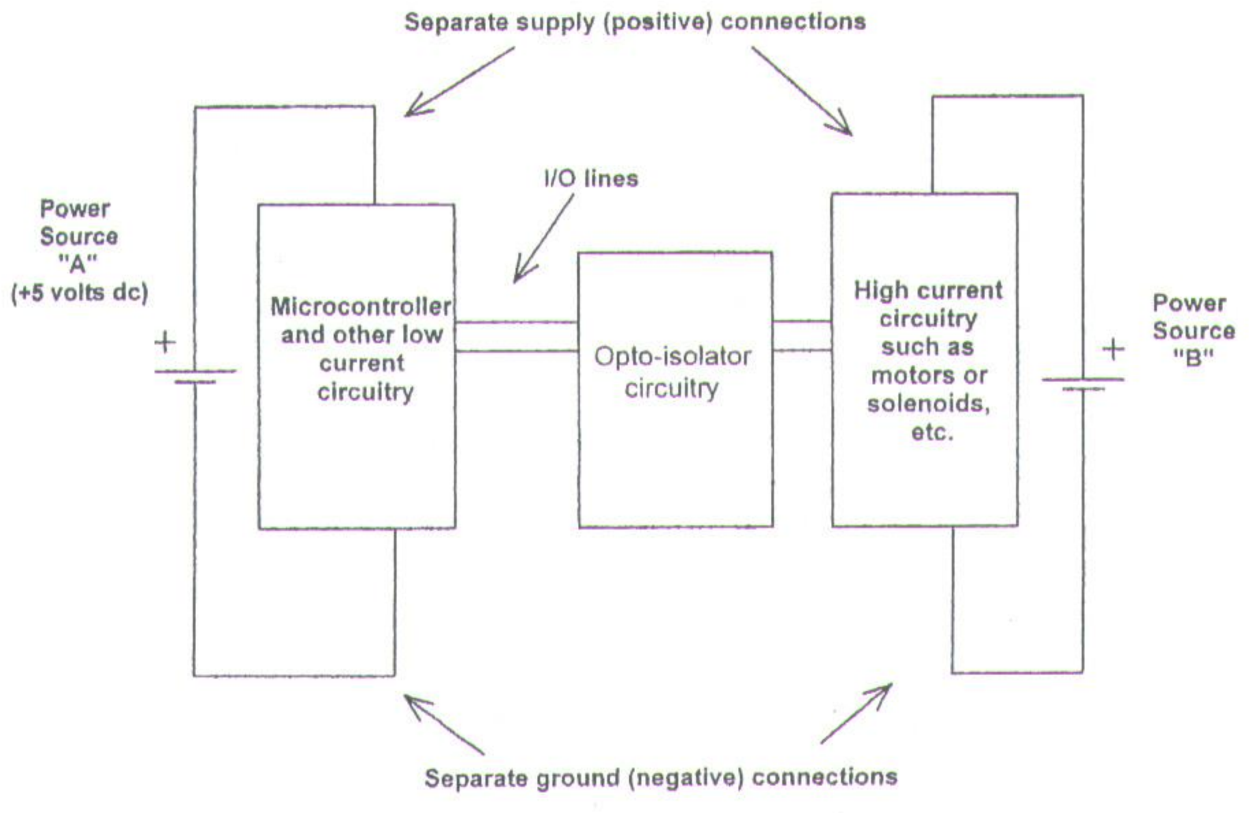
Multiple Power Supplies—Positive Terminal Isolation

- Positive lines of the 2 supplies are kept isolated.
- For proper operation, ground terminals of two supplies are connected to ensure common ground throughout the circuit.


Separate supply (positive) connections
Power Source "A" (+5 volts dc)
+
Microcontroller and other low current circuitry
I/O lines
High current circuitry such as motors or solenoids, etc.
Power Source "B" (+5 volts dc)
+
Common ground (negative) connection

Multiple Power Supplies—Complete Isolation

- Some application may necessitate complete isolation of sensitive BS2 components from high current loads.
  - Both positive and ground terminal are isolated using an opto-isolator.
  - Use of opto-isolator ensures that there are no common electrical connections between the low current and high current portions of circuitry.



Separate supply (positive) connections
I/O lines
Power Source "A" (+5 volts dc)
+
Microcontroller and other low current circuitry
Opto-isolator circuitry
High current circuitry such as motors or solenoids, etc.
+
Power Source "B"
Separate ground (negative) connections

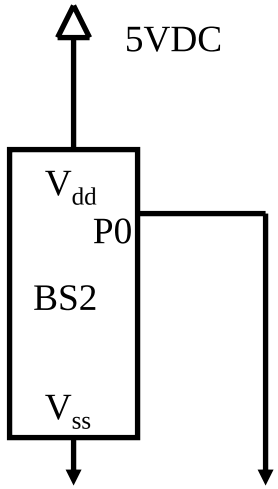
BS2 I/O Pin Protection —I

- Recall that each I/O pin of BS2 (i.e., P0-P15) cannot source (sink) more than 20mA (25mA).

- Suppose P0 is an o/p pin that is mistakenly connected to ground (just using a conductor).
  - When P0 is commanded high, then 5VDC is directly connected to ground via a conductor → P0 is outputting 5VDC while it is "shorted" to ground.
    - → A large current is drawn from P0 to ground, exceeding the 20mA limit on current source capability of P0, thereby destroying the driver circuitry of BS2.

- Recall Ohm's Law V=IR with V=5V, R≅0Ω →

$$I = \frac{V}{R} = \frac{5}{0} \approx \infty$$

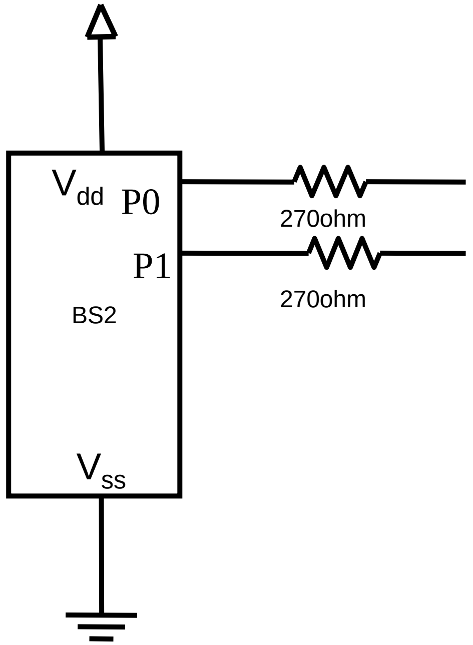


BS2 I/O Pin Protection —II

- To limit I<20mA and prevent accidental damage to BS2, use a resistor R=270Ω to connect P0 to external circuitry.
- Now, suppose o/p pin P0 is mistakenly connected to ground.
  - When P0 is commanded high, then 5VDC is connected to ground via a 270Ω.
- Recall Ohm's Law V=IR with V=5V, R≅270Ω →

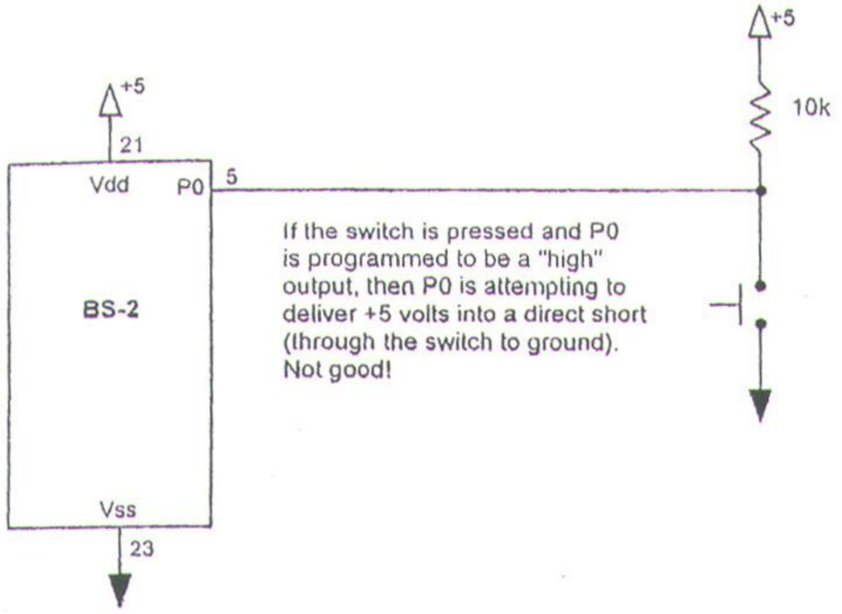
$$I = \frac{V}{R} = \frac{5}{270} \approx 0.019A = 19mA$$

- Moral of the story:
  - To limit source current I<20mA and prevent accidental damage to BS2, always connect external circuitry to BS2 I/O pins, using appropriately sized current limiting resistors.



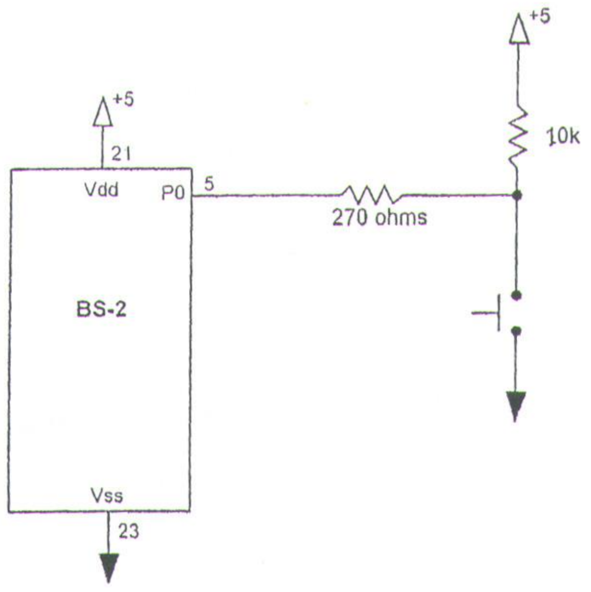
BS2 I/O Pin Protection —III

- Suppose I/O protection resistor is not used since only input is being sensed.
- Input being sensed is state of a switch (open/close).
- When switch is not pressed, 10KΩ resistor pulls P0 high; P0 high → switch is open.
- When switch is pressed, P0 is shortened to ground; P0 low → switch is closed.
- When switch is closed, erroneously instead of reading P0, user outputs high to P0.
  - 5V output from P0 is directly connected to ground, which may damage BS2 driver circuitry.


+5
21
Vdd
P0
5
BS-2
Vss
23
+5
10k
If the switch is pressed and P0 is programmed to be a "high" output, then P0 is attempting to deliver +5 volts into a direct short (through the switch to ground). Not good!

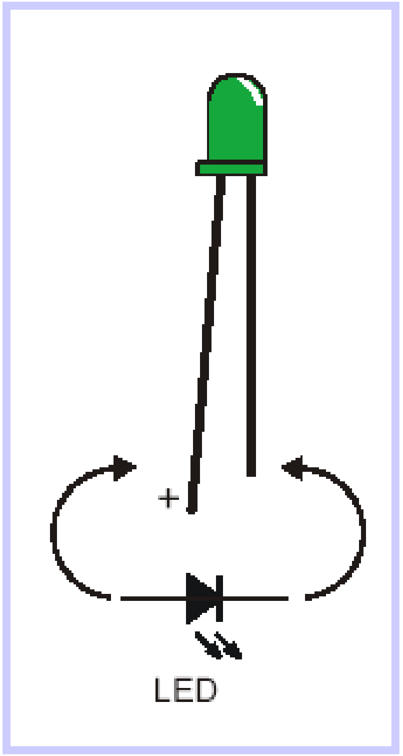
BS2 I/O Pin Protection —IV

- Connect a 270Ω resistor to P0.
- When switch is not pressed, 10270Ω resistor pulls P0 high; P0 high → switch is open.
- When switch is pressed, P0 is connected to ground via 270Ω resistor; P0 low → switch is closed.
- When switch is closed, erroneously instead of reading P0, user outputs high to P0.
  - 5V output from P0 is connected to ground via 270Ω resistor, causing 19mA to flow from P0 to ground. This is within P0's limit.



Driving an LED with BS2 —LED 101

- A Diode is a one-way check valve allowing current flow in only one direction. An LED "Light Emitting Diode" is a diode that emits light as current is passed through it.
- LED must be connected in correct orientation:
  - **Anode:** Connected to + side of voltage.
    - **<u>ID: longer lead.</u>**
  - **Cathode:** Connected to – side of voltage.
    - **<u>ID: shorter lead and a flat portion on the lens.</u>**

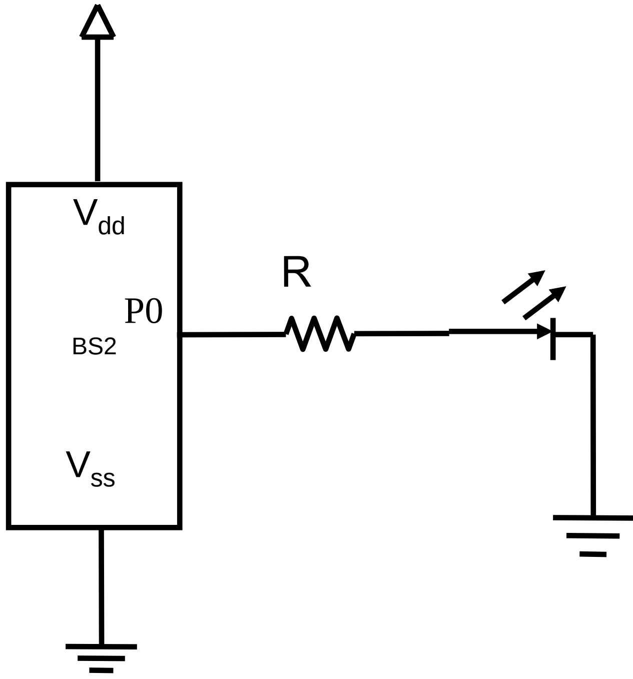


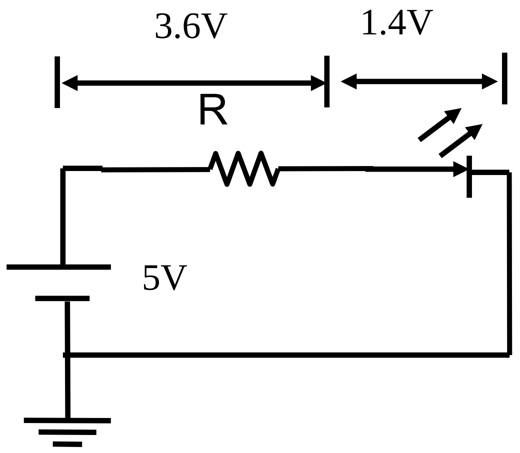
Driving an LED with BS2 —I

- Need to drive an LED using an I/O pin of BS2.
  - LED requires 1.4V and approx. 10mA.
- Consider the given voltage divider circuit of top figure to drive the LED.
- When P0 is driven high → lower figure.
- To limit current through LED to less than 10mA, select R using:

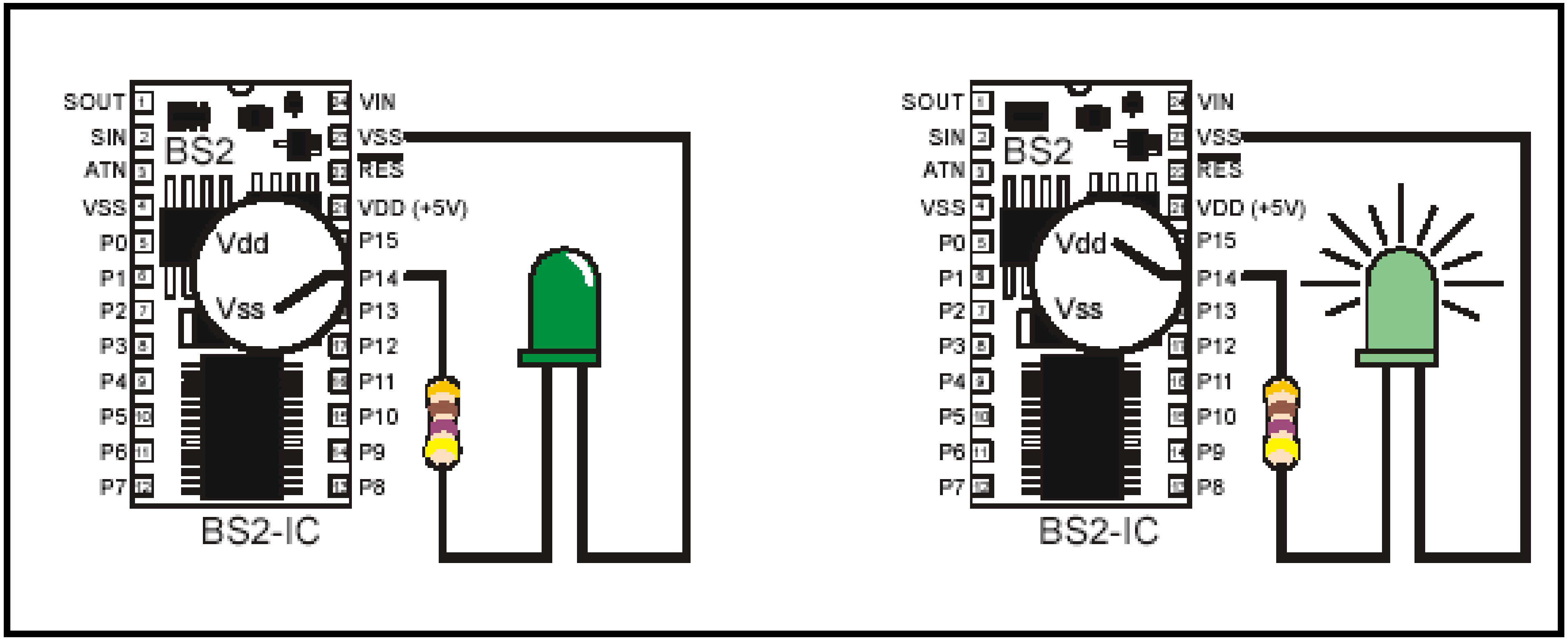
$$R = \frac{V}{I} = \frac{3.6}{10 \times 10^{-3}} = 360\Omega$$

- The 360 Ω resistor protects the LED as well as the I/O pin of the BS2.

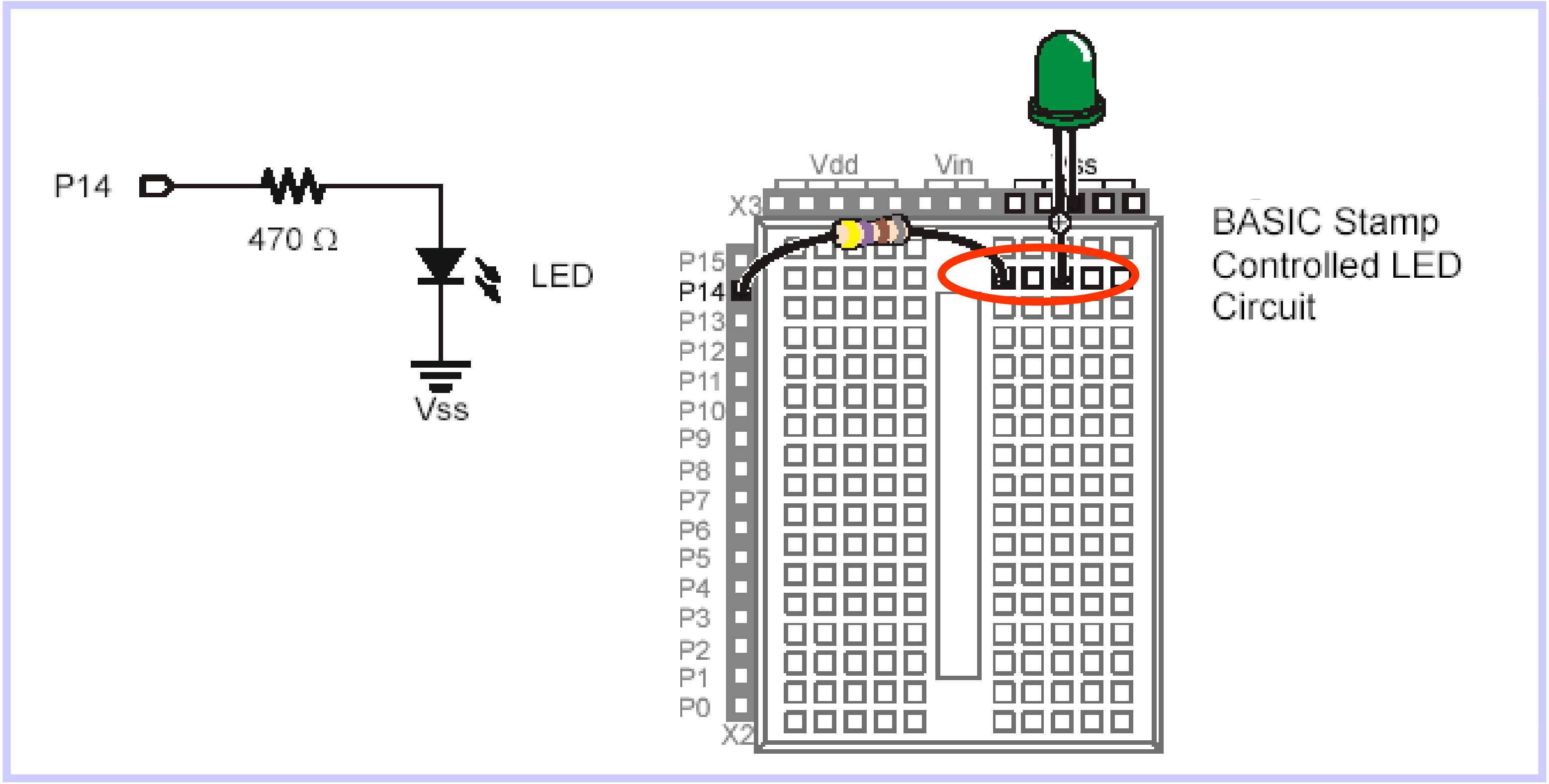




Driving an LED with BS2 —II



Driving an LED with BS2 —III

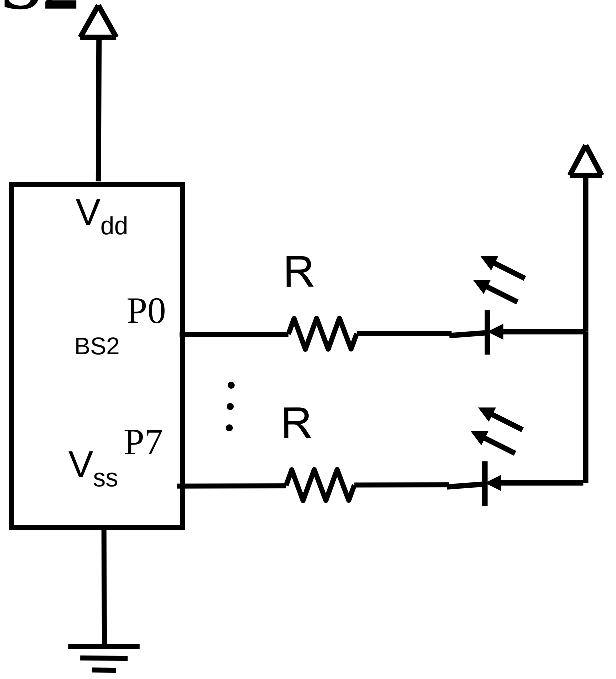


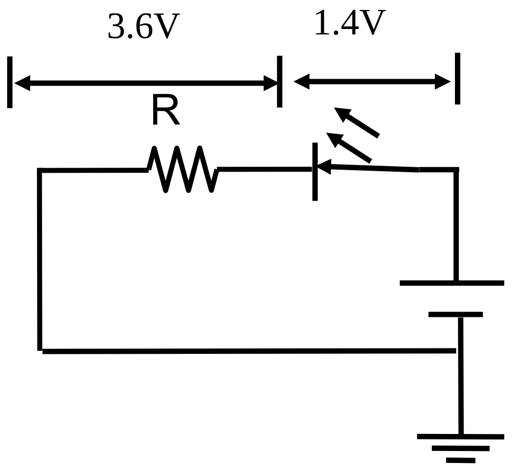
Driving 8 LEDs with BS2

- Need to drive 8 LEDs using 8 I/O pins (P0-P7).
  - If each LED used a 360Ω current protection resistor, then as a group 8 LEDs will attempt to draw 80mA → BS2 can not provide that much current.
- Assume, LEDs are connected so that BS2 sinks current (allowed sink limit 50mA).
  - Each LED is allowed to sink 50/8=6.25mA.
- Find current limiting resistor using

$$R = \frac{V}{I} = \frac{3.6}{6.25 \times 10^{-3}} = 576\Omega$$

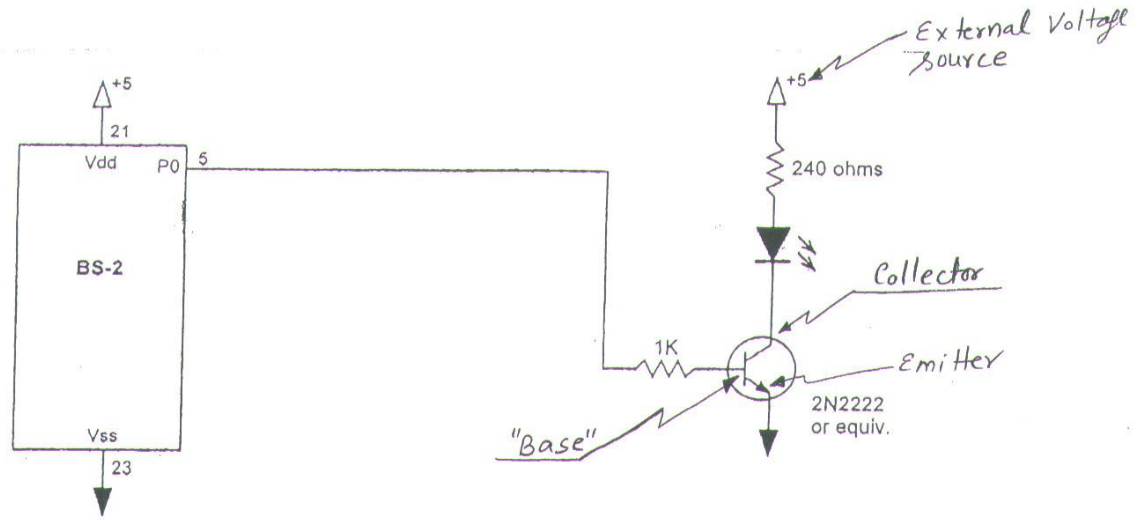
- Since instead of 10mA, only 6.25mA flows through each LED, LEDs will light dimly!
- Here LEDs turns on when P0— P7 are commanded low.





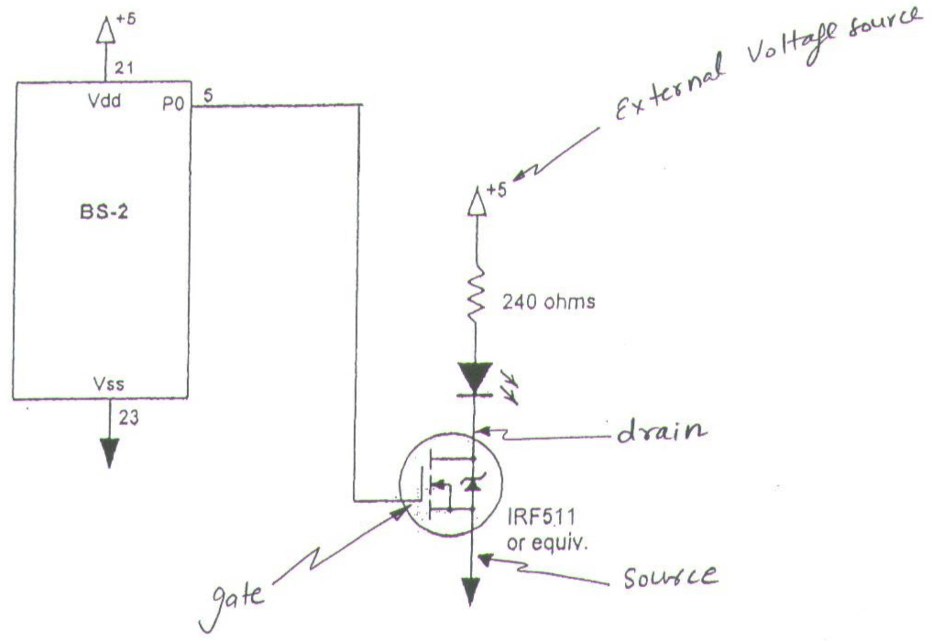
Silicon Steroids for the BS2 —I

- Buffering circuit to impart high current driving capability to BS2
- Bipolar transistor based buffer circuit
  - Use an NPN transistor.
  - Connect NPN base to a BS2 I/O pin, NPN emitter to ground, and NPN collector to positive of an external supply (via the load to be driven).
  - For the collector-emitter pair to conduct, NPN base must be at a potential higher than the potential of emitter and a small current must flow from base to emitter.
  - Driving BS2 pin high causes potential at NPN base to be higher than potential at emitter.
  - Current drawn from BS2 pin is restricted by placing a 1KΩ resistor between the BS2 pin and transistor base.
  - BS2 pin delivers a small signal current (base to emitter), allowing a large current to flow from collector to emitter.

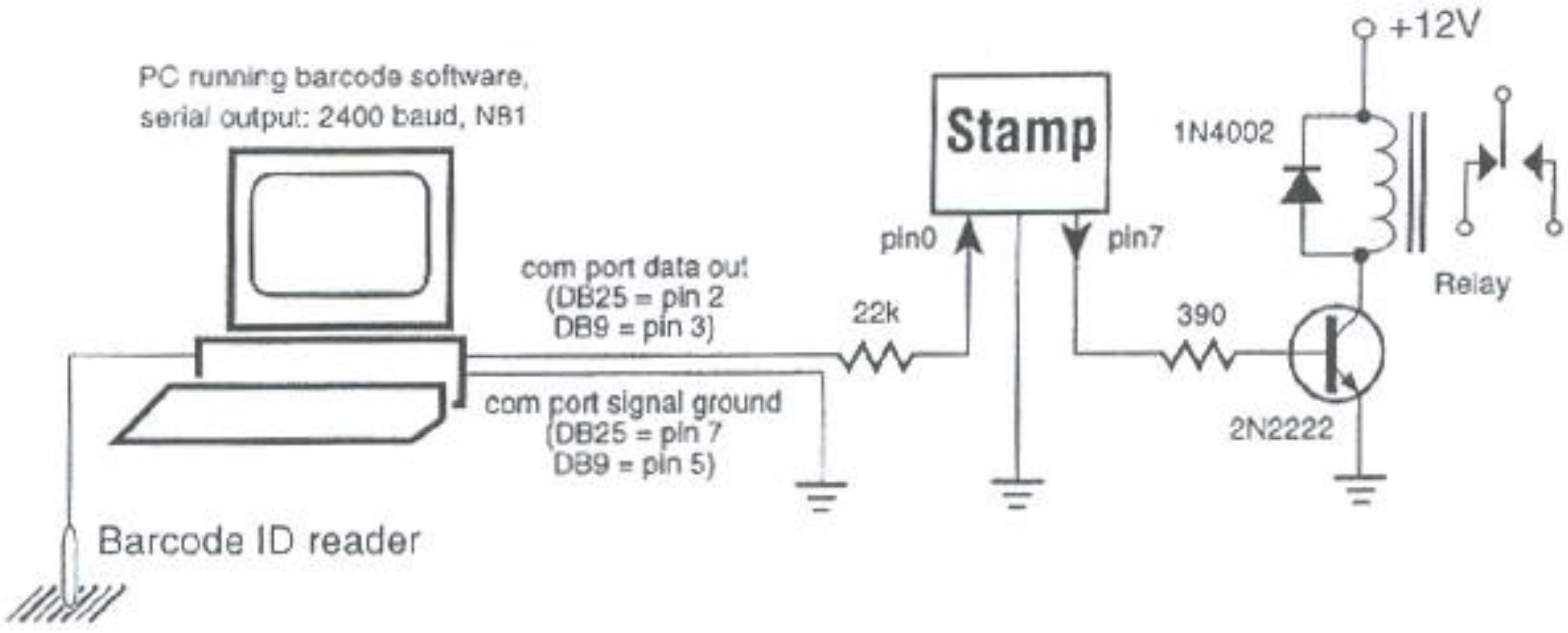


Silicon Steroids for the BS2 —II

- "n-channel" enhancement MOSFET
  - Connect MOSFET gate to a BS2 I/O pin, MOSFET source to ground, and MOSFET drain to positive of an external supply (via the load to be driven).
  - For the drain-source pair to conduct, MOSFET gate must be at a potential higher than the potential of source.
  - Since MOSFET is a voltage controlled device, no significant current is drawn from BS2 pin.
  - Driving BS2 pin high causes potential at MOSFET gate to be higher than potential at MOSFET source
  - Relatively little potential drop across MOSFET (i.e., $V_{source} \approx V_{drain}$), so most power is used by the load

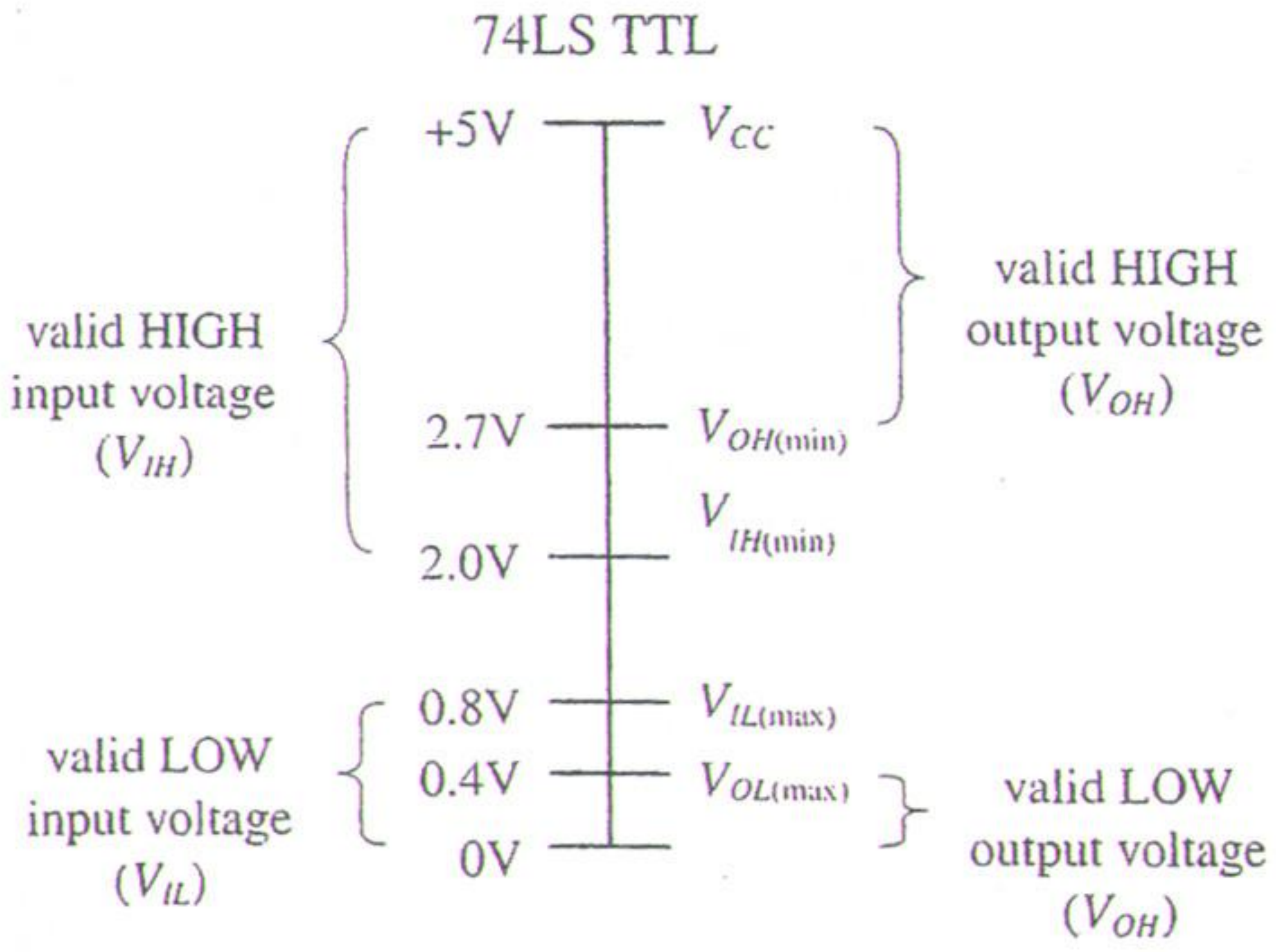


A Real World Example



When the Stamp receives the proper code from the PC bar-code reader, it activates a relay to unlock the door

Valid Input/Output Logic Levels for TTL

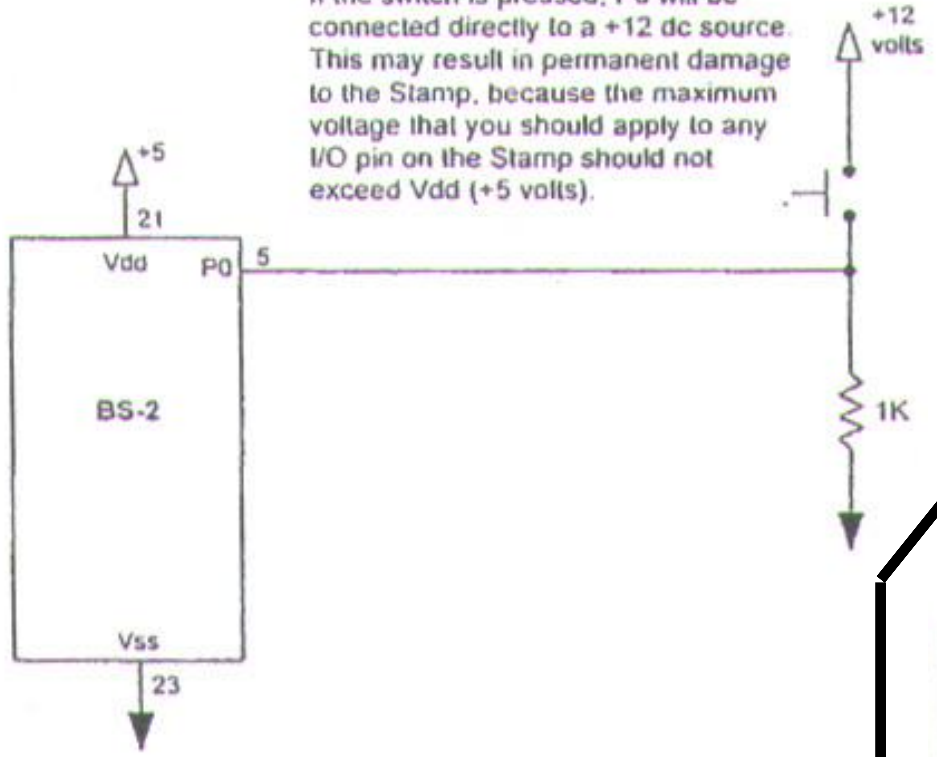

74LS TTL
+5V
V_{CC}
valid HIGH input voltage (V_{IH})
valid HIGH output voltage (V_{OH})
2.7V
$V_{OH(\text{min})}$
$V_{IH(\text{min})}$
2.0V
0.8V
$V_{IL(\text{max})}$
0.4V
$V_{OL(\text{max})}$
0V
valid LOW input voltage (V_{IL})
valid LOW output voltage (V_{OH})

Signal Level Shifting Issue —I

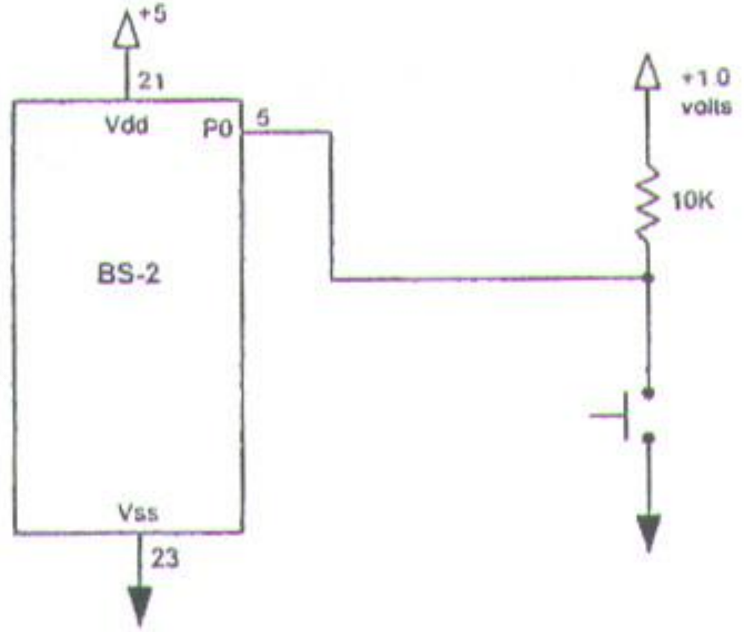
- Input signals applied at any BS2 I/O pins must be:
  - 0 volt to be sensed off.
  - 5 volts to be sensed on.
- Input signals must not sink more than 20mA on any I/O pins.
- Issue: many real-world sensors may present voltage levels that exceed 5V or do not cross 2V threshold to be detected as high!
  - Any input signal larger than 5V cannot be directly connected to BS2 for sensing, because doing so will destroy BS2.
  - If a sensor switches from 0V (off) to 1.0V (on), BS2 cannot detect it!
    - TTL logic requires input to be larger than 2V to be sensed as high (BS2 senses voltage inputs ≥ 1.4V as high).

Signal Level Shifting Issue —II

If the switch is pressed, P0 will be connected directly to a +12 dc source. This may result in permanent damage to the Stamp, because the maximum voltage that you should apply to any I/O pin on the Stamp should not exceed Vdd (+5 volts).

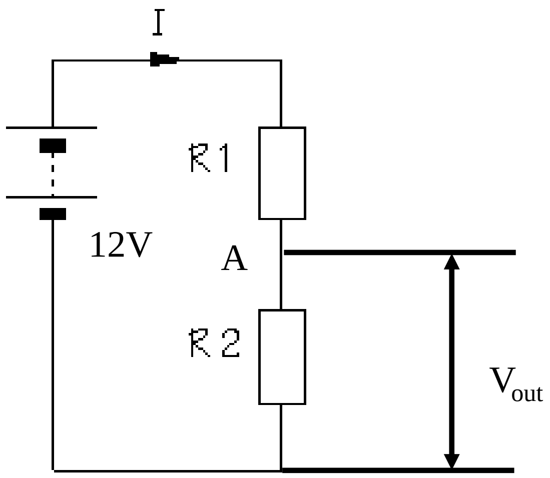


When the switch is pressed, P0 is shorted directly to ground, resulting in a logic "0." When released, P0 is pulled up to 1 0 volts, but since the threshold voltage of a typical IC (including microcontrollers) is usually at least 1.4 volts, the I/O pin will still only "see" a logic level "0."



Downward Shift in Signal Level: Voltage Divider —I

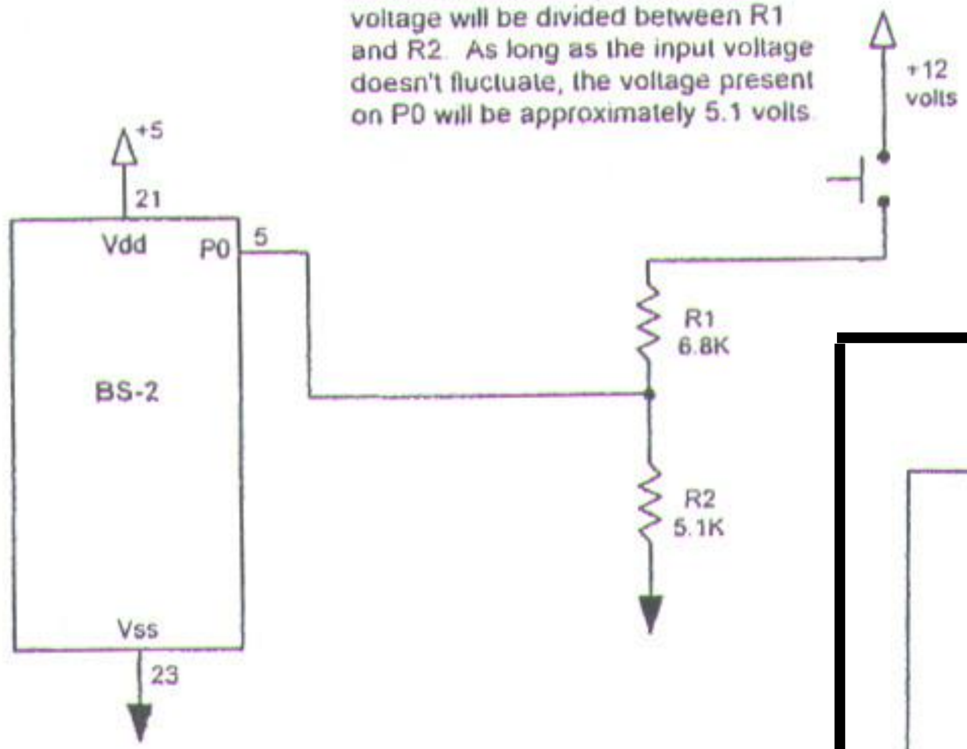
- A sensor outputs 12 volts when on. This signal needs to be scaled down so that it can be interfaced with BS2 for BS2 to register sensor off/on status.
- Solution: Consider voltage divider circuit shown below.

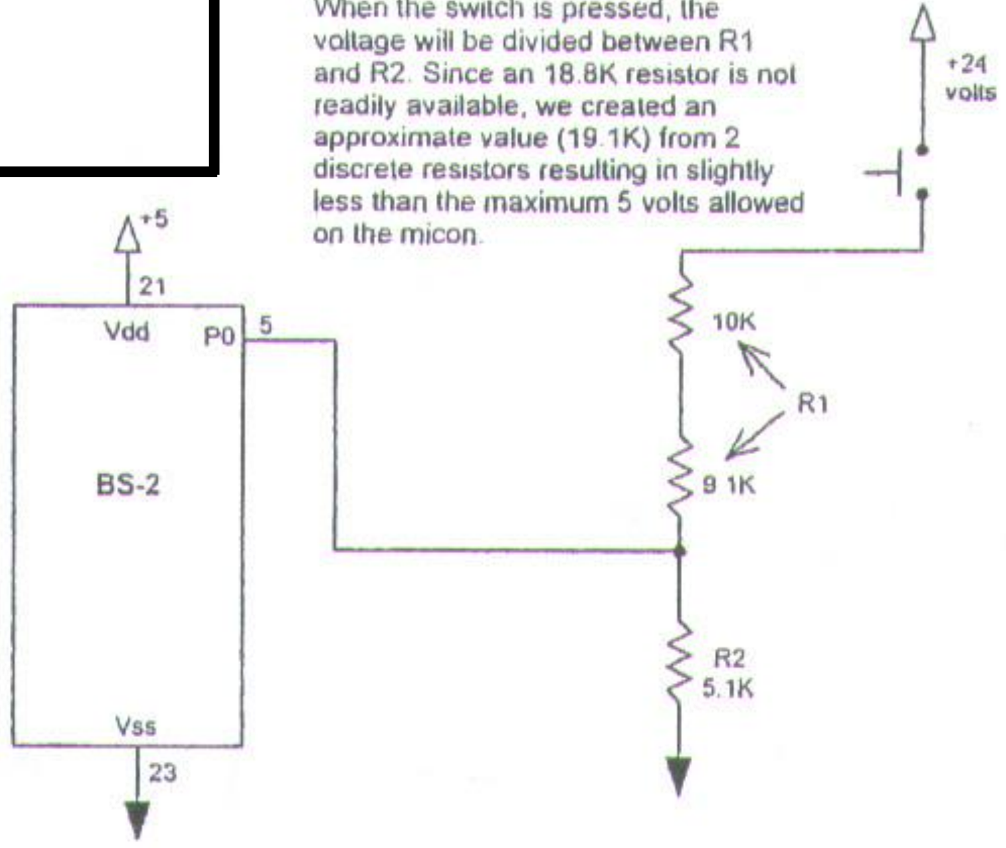


$$V_{out} = \frac{R_2}{R_1 + R_2} V_s$$

- Given V_s=12V and need V_{out}=5V. Let R_2=R and find appropriate value of R_1.
- R_1 =(7/5)R and current I=Vs/(R_1+R_2)=5/R.
- Need to limit current I<20mA. Let R=5KΩ, → I=1mA (acceptable) and R_1=7KΩ
- Now pick R_1 and R_2 from usually available resistor values to be close enough to above design

Downward Shift in Signal Level: Voltage Divider —II

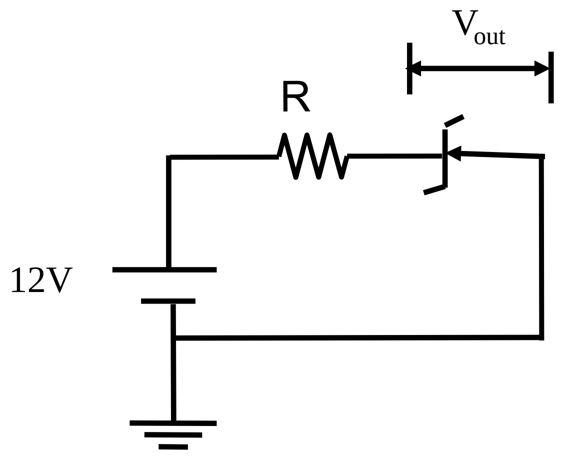

When the switch is pressed, the voltage will be divided between R1 and R2. As long as the input voltage doesn't fluctuate, the voltage present on P0 will be approximately 5.1 volts.


When the switch is pressed, the voltage will be divided between R1 and R2. Since an 18.8K resistor is not readily available, we created an approximate value (19.1K) from 2 discrete resistors resulting in slightly less than the maximum 5 volts allowed on the micon.

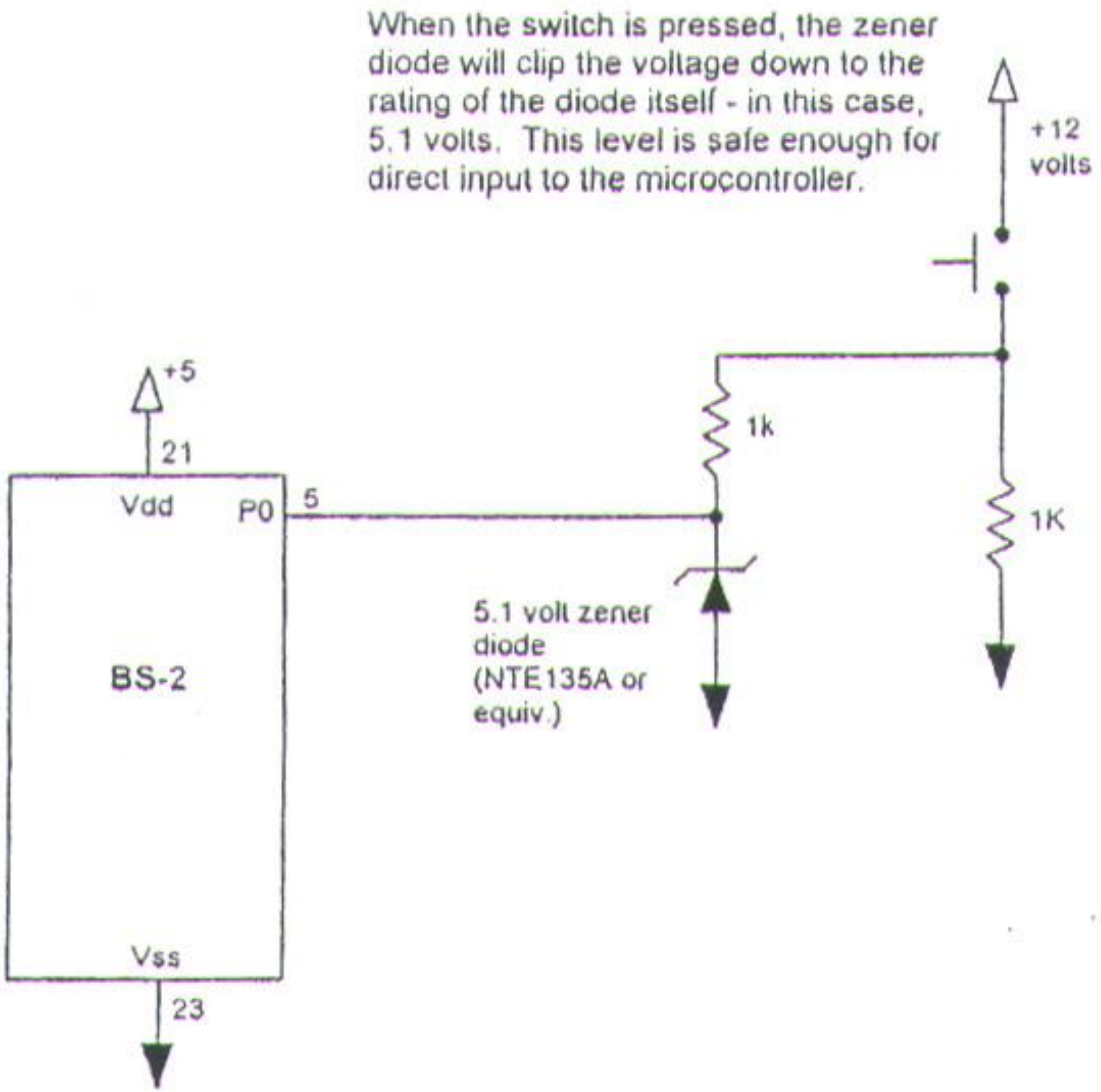
Downward Shift in Signal Level: Zener Circuit —I

- A sensor outputs 12V when on. This signal needs to be scaled down so that it can be interfaced with BS2 for BS2 to register sensor off/on status.
- Solution: Consider zener diode circuit shown below.
- A series circuit of a zener diode (NTE 135A or equivalent) and a resistor R is connected to a 12V battery.
- The zener is rated at 5.1V so, of the 12V input, 5.1V will be presented across zener and the rest (6.9V) across R.
- Select R to limit current sunk into BS2 I/O pin to less than 20mA. Let R= 1KΩ →

$$I = \frac{6.9\,volts}{1000\,\Omega} = 6.9mA << 20mA$$

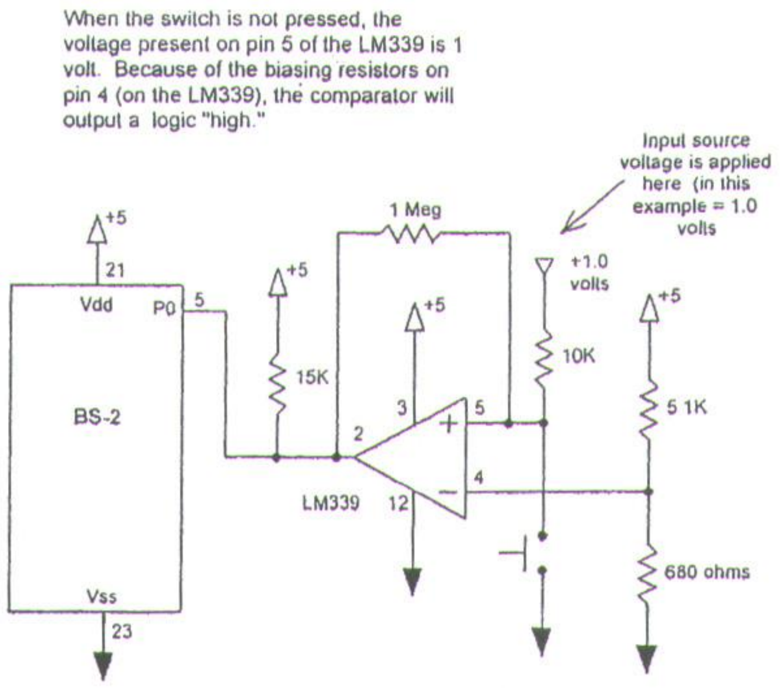


Downward Shift in Signal Level: Zener Circuit —II


When the switch is pressed, the zener diode will clip the voltage down to the rating of the diode itself - in this case, 5.1 volts. This level is safe enough for direct input to the microcontroller.
+12 volts
+5
21
Vdd
P0
5
1k
1K
5.1 volt zener diode (NTE135A or equiv.)
BS-2
Vss
23

Upward Shift in Signal Level—I

- When a sensor outputs 0V (off) and 1V (on), a comparator can be used to interface the sensor to BS2.
- Comparator outputs a digital signal and works as follow.
  - When voltage input at comparator’s noninverting terminal (V_+) is higher than voltage input at comparator’s inverting terminal (V_-), comparator output is equal to its upper saturating voltage.
  - When $V_+ < V_-$, comparator output is equal to its lower saturating voltage.
  - Positive feedback in a comparator introduces a hysteresis behavior in its I/O relationship and eliminates chattering (will study details later).

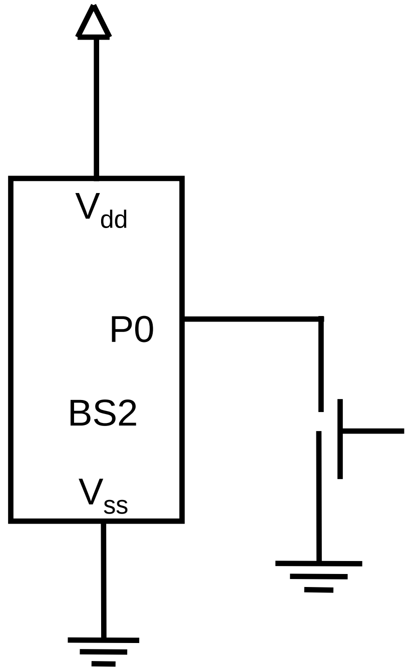


When the switch is not pressed, the voltage present on pin 5 of the LM339 is 1 volt. Because of the biasing resistors on pin 4 (on the LM339), the comparator will output a logic "high."

Input source voltage is applied here (in this example = 1.0 volts)

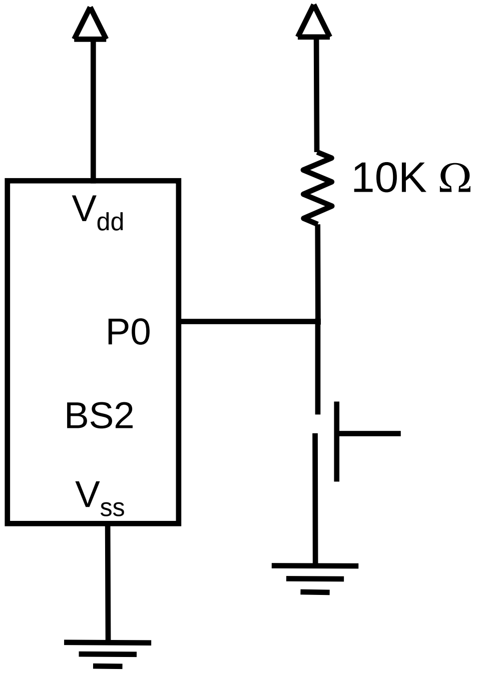
Floating Input Condition

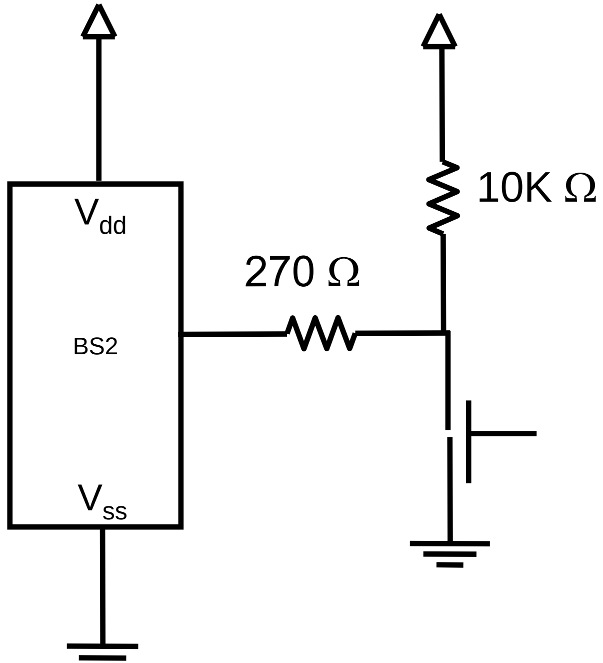
- Suppose a button is connected to P0 as shown.
- When button is pressed, P0 is grounded so P0 status is low.
- When button is not pressed, P0 is not connected to anything
  - What will be P0 input level?
- P0 may indicate low, high, or continuously alter between the two states.
  - Floating input! → cannot distinguish between button pressed/not-pressed.



Avoiding Floating Input Condition

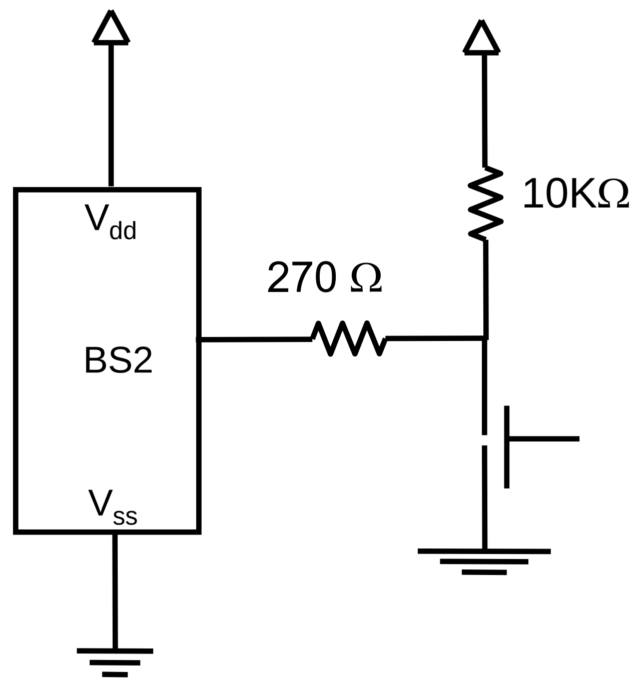
- Use the circuit shown to avoid floating input.
- When button is pressed, P0 is grounded so P0 status is low.
- When button is not pressed, 10KΩ resistor pulls P0 high.
- → P0 low indicates button pressed and P0 high indicates button not pressed.
- A safer circuit will include the 270Ω resistor as shown to ensure that current sourced by BS2 doesn't violate the 20mA limit in case P0 is mistakenly made a High o/p pin.

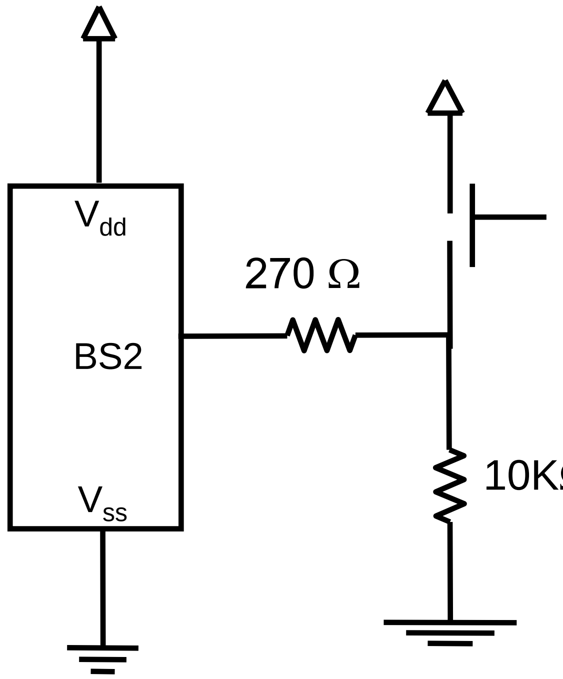




Pull up v/s Pull down Resistors

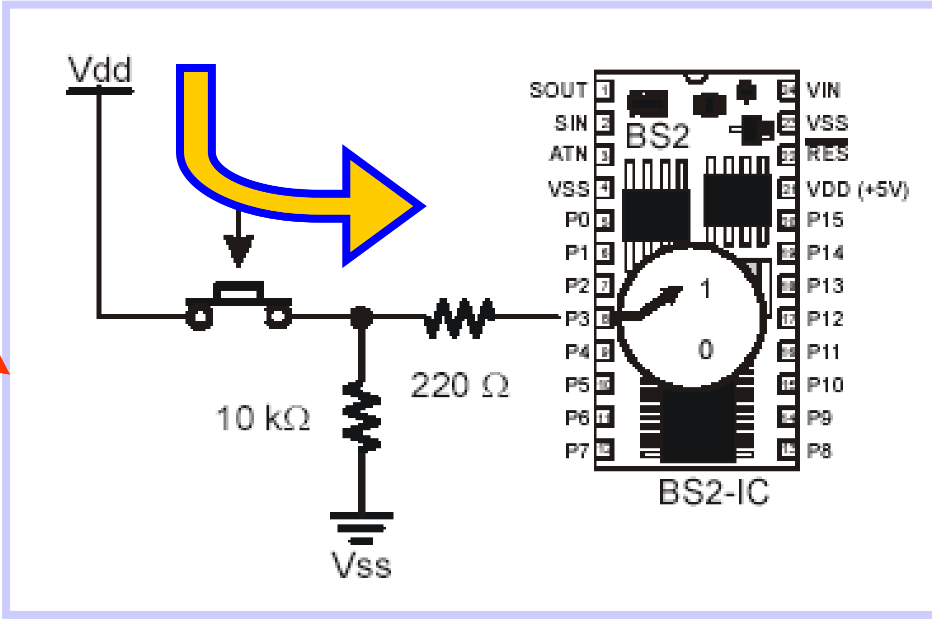
- In the diagram on left, when button is not pressed, 10270Ω resistance pulls P0 high. 10K is pull up resistor.
  - Button not pressed is indicated by P0 high.
- In the diagram on right, when button is not pressed, 10270Ω resistance connects P0 to ground. 10K is pull down resistor.
  - Button not pressed is indicated by P0 low.



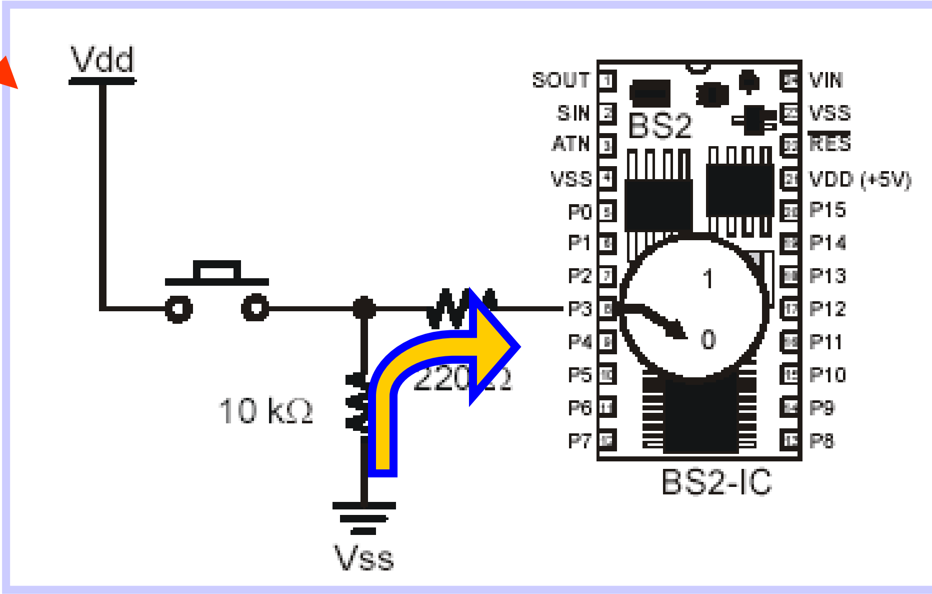


Pull down Resistor: Additional Insight

- When the switch is pressed, V_{dd} (+5V) is sensed at the input of P3.



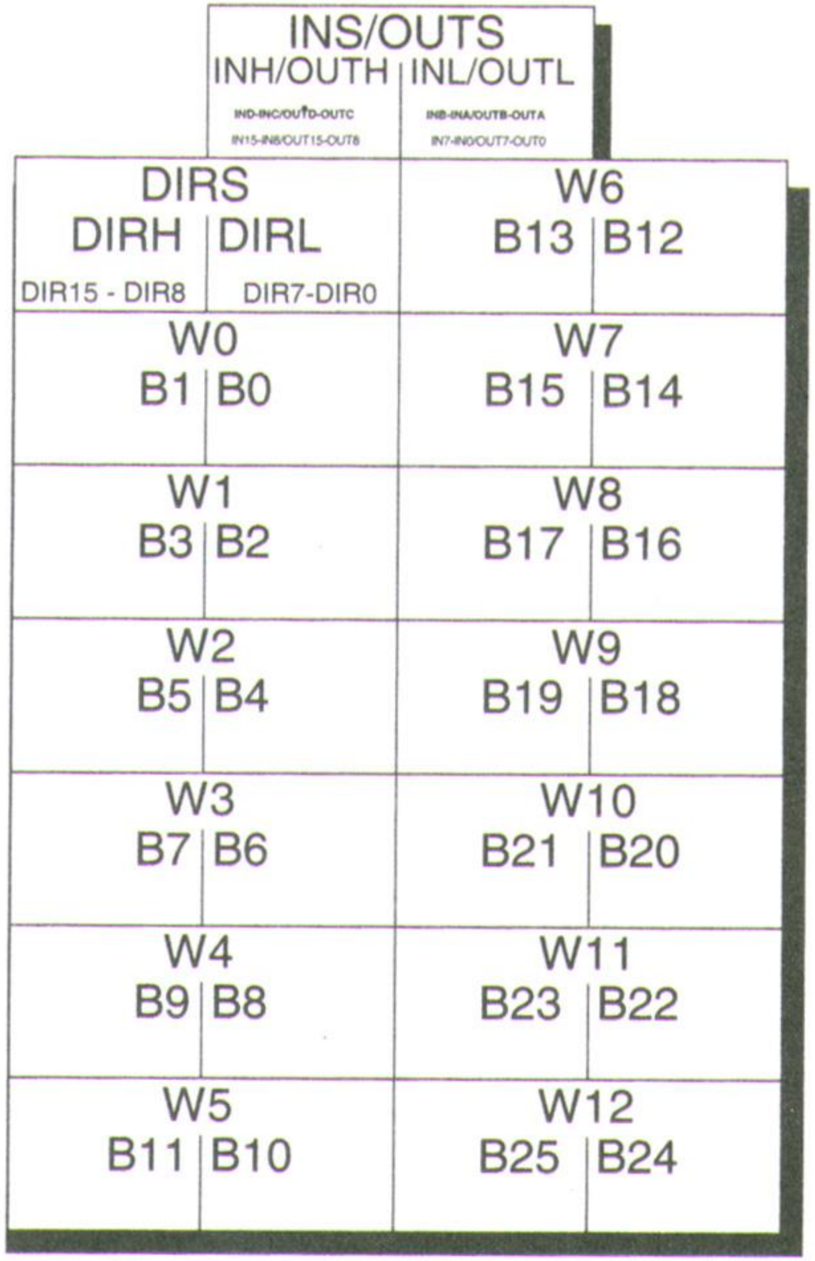
- When the switch is released, V_{ss} (0V) is sensed at the input of P3



- **The 10KΩ resistor prevents a short circuit from V_{dd} to V_{ss}**

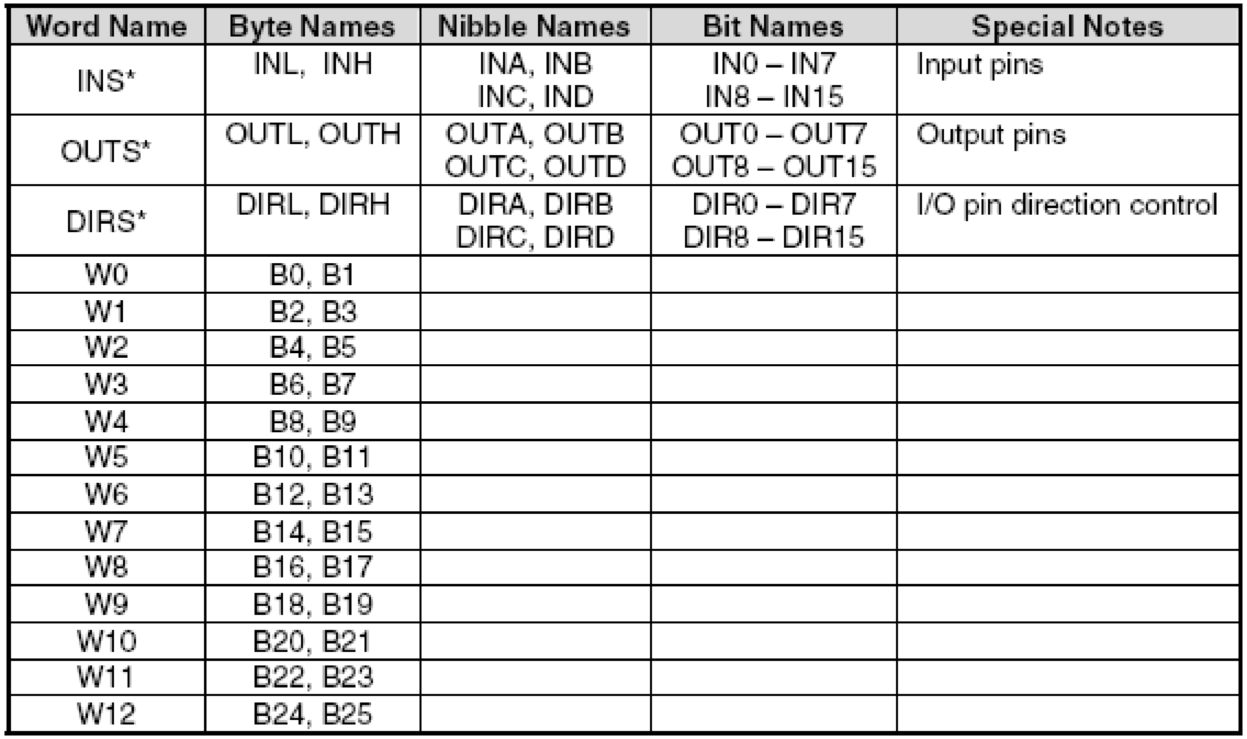
BS2 Programming Tutorial: BS2 Memory Map —I

- BS2 has 32 bytes of RAM
- 26 bytes of RAM can be used for data/variables.
- 6 bytes are reserved for I/O:
    - 2 bytes for INS register
    - 2 bytes for OUTS register
    - 2 bytes for DIRS register



BS2 Programming Tutorial: BS2 Memory Map —II

- 16 words, of two bytes each for a total of 32 bytes.
- All bits are individually addressable through variable modifiers.
- The bits within the upper three words are also individually addressable though the pre-defined names shown.
- All registers are word, byte, nibble, and bit addressable.



Word Name	Byte Names	Nibble Names	Bit Names	Special Notes
INS*	INL, INH	INA, INB INC, IND	IN0 – IN7 IN8 – IN15	Input pins
OUTS*	OUTL, OUTH	OUTA, OUTB OUTC, OUTD	OUT0 – OUT7 OUT8 – OUT15	Output pins
DIRS*	DIRL, DIRH	DIRA, DIRB DIRC, DIRD	DIR0 – DIR7 DIR8 – DIR15	I/O pin direction control
W0	B0, B1			
W1	B2, B3			
W2	B4, B5			
W3	B6, B7			
W4	B8, B9			
W5	B10, B11			
W6	B12, B13			
W7	B14, B15			
W8	B16, B17			
W9	B18, B19			
W10	B20, B21			
W11	B22, B23			
W12	B24, B25			

BS2 Programming Tutorial: BS2 Memory Map —III

- Bit: 1 digit digital number (0 or 1)
- Nibble: A digital number containing 4 bits
- Byte: A digital number containing 8 bits (or 2 nibs)
- Word: A digital number containing 16 bits (or 4 nibs, or 2 bytes)

Word	**Byte**
W0	B0, B1
W1	B2, B3
⋮	⋮
W12	B24, B25

13 digital words 26 bytes

BS2 Programming Tutorial —Variable Types

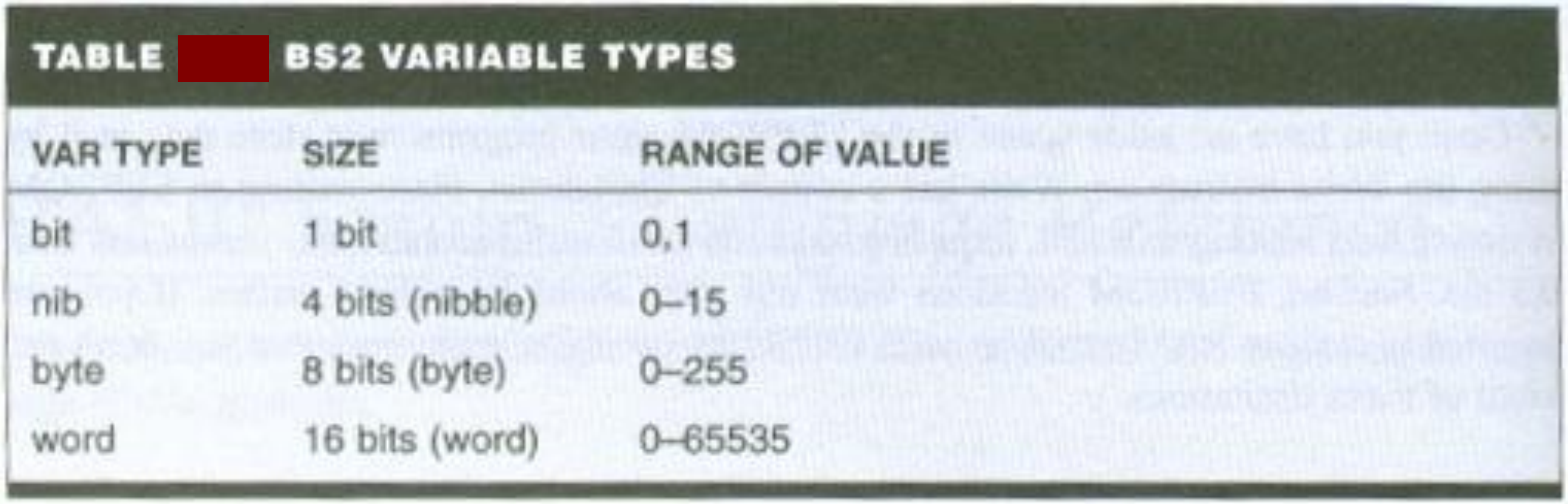


TABLE BS2 VARIABLE TYPES

VAR TYPE	SIZE	RANGE OF VALUE
bit	1 bit	0,1
nib	4 bits (nibble)	0–15
byte	8 bits (byte)	0–255
word	16 bits (word)	0–65535

```
OnOff var bit
InOutPins var nib
ADCin var byte
Count var word
```

Aside: Binary, Decimal, and Hexadecimal Numbers

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Binary	Decimal	Hexadecimal
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

BS2 Programming Tutorial: Assigning Data to BS2 Memory Locations

- b0=10 → byte size memory location #0 will store decimal number 10
- b0=%00001010 → byte size memory location #0 will store binary number 00001010 (equivalent to decimal number 10)
- b0=$0A → byte size memory location #0 will store hexadecimal number 0A (equivalent to decimal number 10)

BS2 Programming Tutorial: Working with Memory

- b1=20 → byte size memory location #1 will store decimal number 20
- With b0=10 and b1=20, use of b2=b0+b1 → byte size memory location #2 will store sum of b0 and b1
- Let b0=200 and b1=175. Then what is the result of b2=b0+b1?

Aside: Fixed-Size Integers

- When a car passes 99,999 miles its 5-digit odometer reads 00000.
- Consider the following PBasic code snippet:

```
FixSizVar Var Byte
FixSizVar=255 ‘i.e., FixSizVar=%11111111
FixSizVar=FixSizVar+1
Debug Bin FixSizVar
```

- Above code is performing the following operation

```
%11111111
+      %1
```

- The process of adding 1 to 11111111 results in a series of carry-the-one operations moving from right to left.
- The one carried out of the eighth bit (MSB) has no place to go (a byte contains only eight bits).
- → Returned value of FixSizVar is %00000000.
- By using FixSizVar Var Word we can overcome the above problem. Now FixSizVar can store numbers up to 65535.

BS2 Programming Tutorial: Working with Memory

- Recall byte size memory location can store any number 0 to 255.
- 200+175 is 375, larger than 255!
- $(375)_2 = \underbrace{00000001}_{\text{upper byte}}\,\underbrace{01110111}_{\text{lower byte}} \longrightarrow$

$$\boxed{\begin{aligned} \mathbf{01110111} &= \mathbf{2^7(0)+2^6(1)+2^5(1)+2^4(1)} \\ &\quad \mathbf{+2^3(0)+2^2(1)+2^1(1)+2^0(1)} \\ &= \mathbf{119} \end{aligned}}$$

- Since only one byte memory location is available to store 375, the lower byte is saved in the memory and the upper byte is lost.
- The lower byte is equivalent to 119 in decimal number system.
- Thus, use of

```
b3=375
debug DEC b3
```

will yield 119! Thus, need to use at least a word size variable to store numbers larger than 255.

BS2 Programming Tutorial: Variable Declarations —I

- When one uses commands such as

  ```
  b0=10
  b1=20
  ```

  BS2's specific byte size memory locations are used to store 10 and 20.
- A program is more readable when meaningful variables are used to store data instead of just the memory registers.
- For example, when one uses

  ```
  length VAR byte
  width VAR byte
  area VAR word
  ```

  BS2 is asked to set aside two byte size memory locations for length and width variables and a word size memory location for the area variable.
- Declaration of variables requires the size of memory needed to store the variable.

BS2 Programming Tutorial: Variable Declarations —II

- Example:

```
length VAR byte
width VAR byte
area VAR word
length=15
width=25
area=length*width
debug “Area in DEC=” DEC area
```

- Debug window will correctly return 375 as the value of area since a digital word stores the area.

Debug Declaration and Constants

- When using the debug command to display numerical value stored in a variable, appropriate declaration must be used to tell BS2 the display format (decimal, binary, Hex, etc).
  - DEC: is used to display the result in decimal number systems.
  - BIN: to display the result in binary number system.
  - HEX: to display the result in hexadecimal number system.
- When a constant is to be stored in a user specified variable name, use the following syntax "var_name CON value":

  gain CON 6

  which stores numerical value 6 in a variable name gain and tells BS2 to treat gain as a constant.

BS2 Programming Tutorial —Variable Modifiers

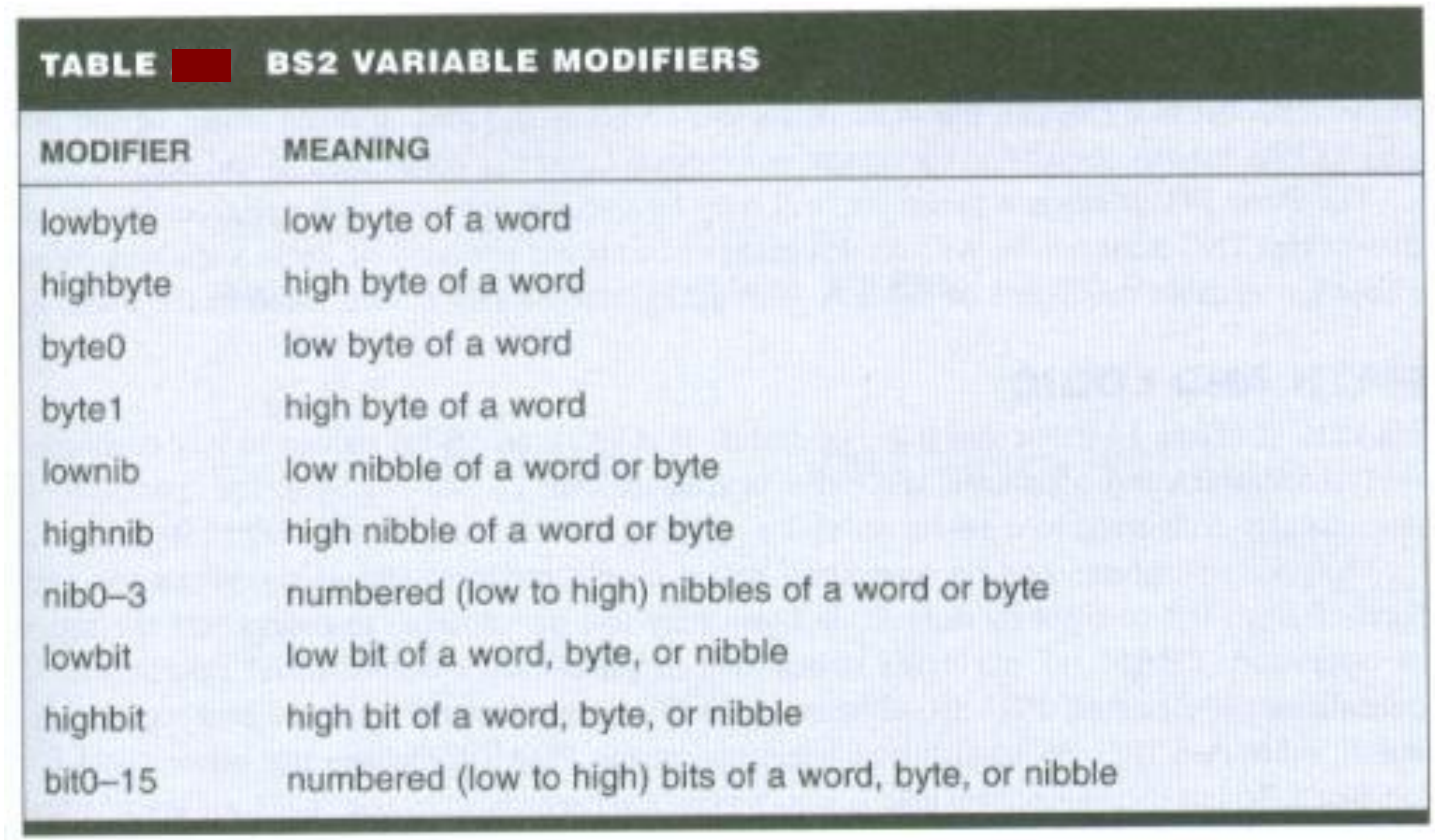
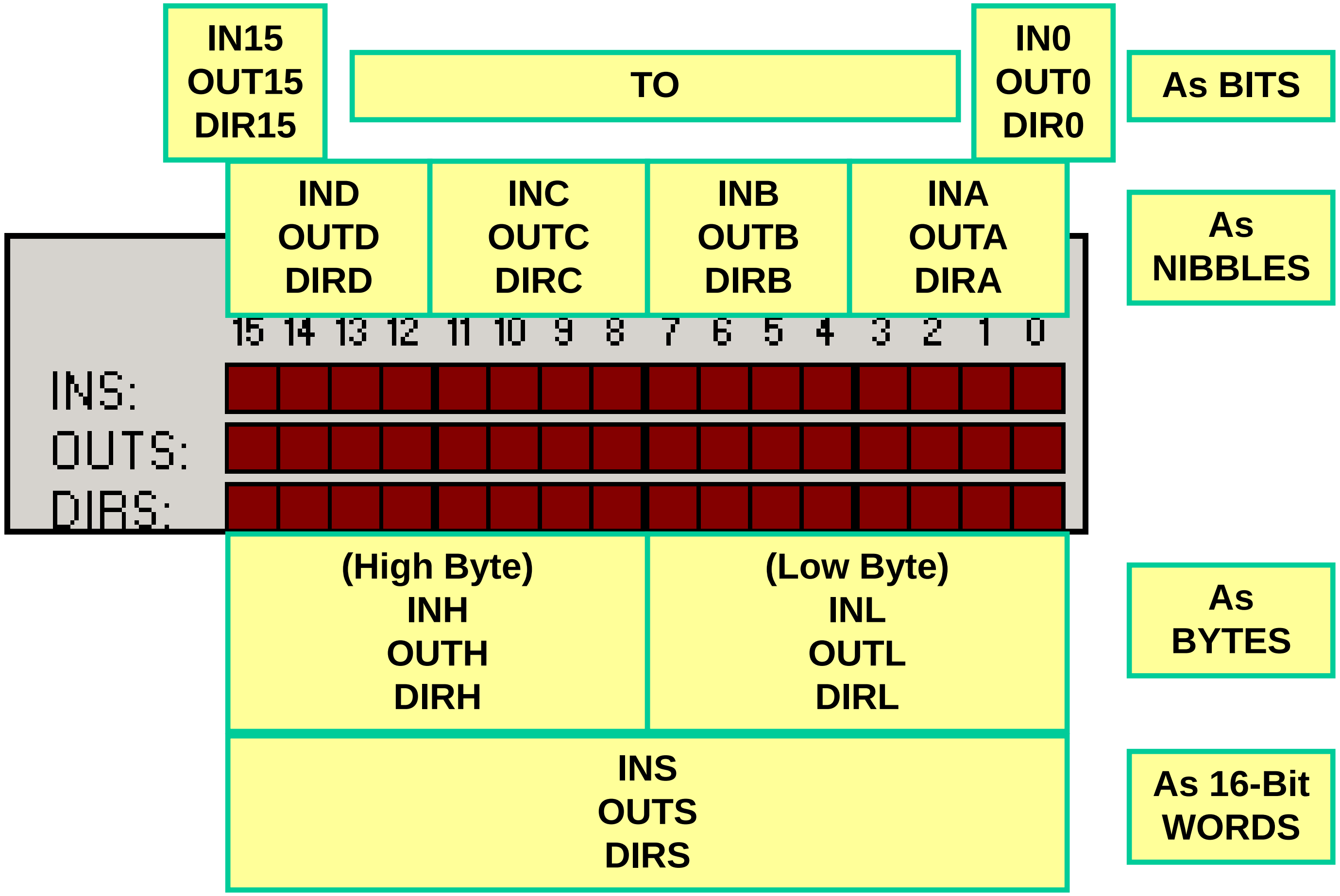


TABLE [redacted] BS2 VARIABLE MODIFIERS

MODIFIER	MEANING
lowbyte	low byte of a word
highbyte	high byte of a word
byte0	low byte of a word
byte1	high byte of a word
lownib	low nibble of a word or byte
highnib	high nibble of a word or byte
nib0–3	numbered (low to high) nibbles of a word or byte
lowbit	low bit of a word, byte, or nibble
highbit	high bit of a word, byte, or nibble
bit0–15	numbered (low to high) bits of a word, byte, or nibble

I/O Register Modifiers

- The I/O can also be addressed as nibbles, bytes or the entire word.



I/O Registers —DIRS

DIRS: 1 for output, 0 for input

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIRD				DIRC				DIRB				DIRA			
DIRH								DIRL							

- DIR0=0 and DIR1=1 → P0 is input pin and P1 is output pin, respectively.
- DIRA=%0110 → P0: input, P1: output, P2: output, P3: input
  - This can also be written as DIRA=6; however clearly binary form is more readable.

I/O Registers —OUTS

OUTS: 1 high, 0 for low

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTD				OUTC				OUTB				OUTA			
OUTH								OUTL							

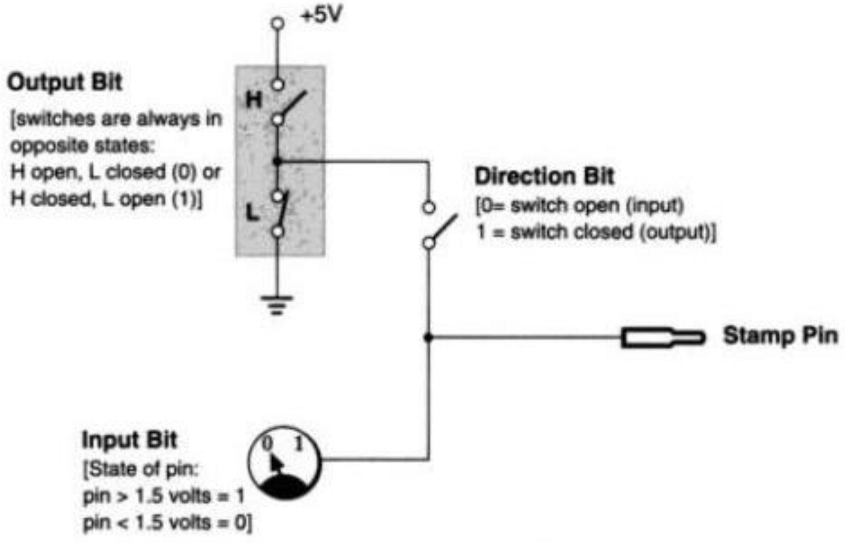
- OUT0=0 and OUT1=1 → P0 is driven low and P1 is driven high, respectively.
- OUTA=%0110 → P0: low, P1: high, P2: high, P3: low
- Aside: INS works similar to OUTS. INS is used to access state of various input pins.

I/O Registers —BS2 Power Up/Reset

- When the BASIC Stamp is powered up, or reset, all memory locations are cleared to 0
  - All pins are inputs (DIRS = %0000000000000000).
- If the PBASIC program sets all the I/O pins to outputs (DIRS = %1111111111111111)
  - All pins will initially output low, since the output latch (OUTS) is cleared to all zeros upon power-up or reset, as well.

I/O—Behind the Scenes

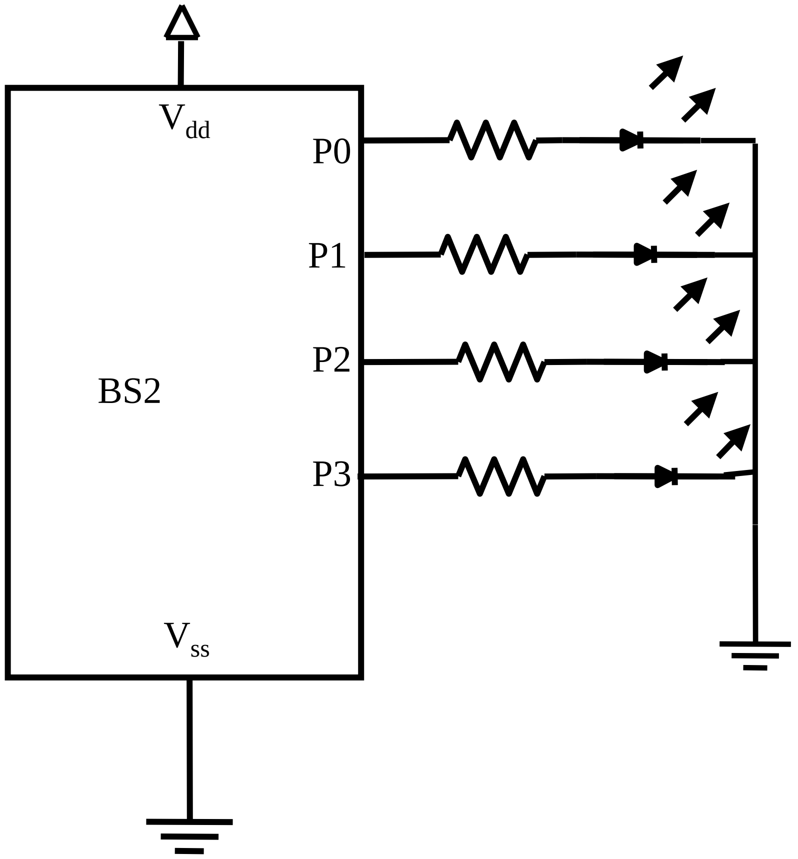
- An I/O pin's behavior is determined by the direction bit, output bit, and input bit
- Output bit, e.g., OUT0
  - a pair of MOSFET switches controlled by BS2
  - always in opposite states (one H, one L)
  - OUT0=1 puts a 1 in the output bit of P0, regardless of the direction bit of P0
- Direction bit, e.g., DIR0
  - a MOSFET switch that is open when DIR0=0 and closed when DIR1=1
  - closing the direction bit switch puts the output bit OUT0 on P0
- Input bit, e.g., IN0
  - always connected to P0 and its state always matches the state of P0
  - when DIR0=1, IN0 will be equal to OUT0
  - when DIR0=0, IN0 will match the state of P0



Input/output circuitry of a Stamp pin.

Dealing with Outputs —I

- Consider the following diagram with 4 LEDs connected to P0-P3 where BS2 sources current.



Dealing with Outputs (P0) —Code Examples

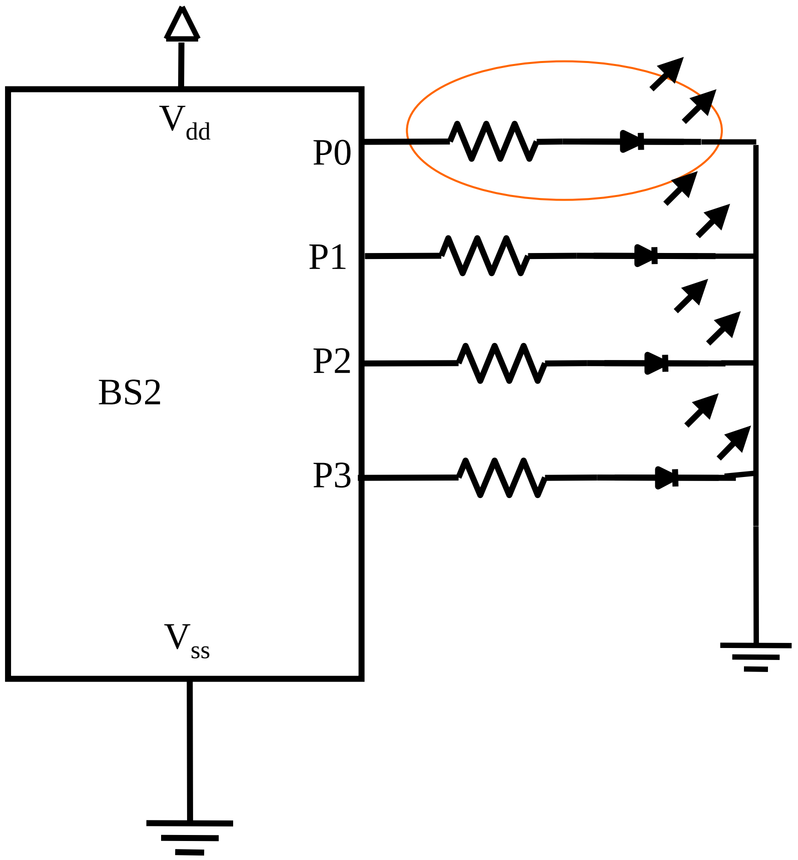
```
high 0
```

```
high 0
pause 100
low 0
```

```
output 0
high 0
```

```
dir0=1
out0=1
```

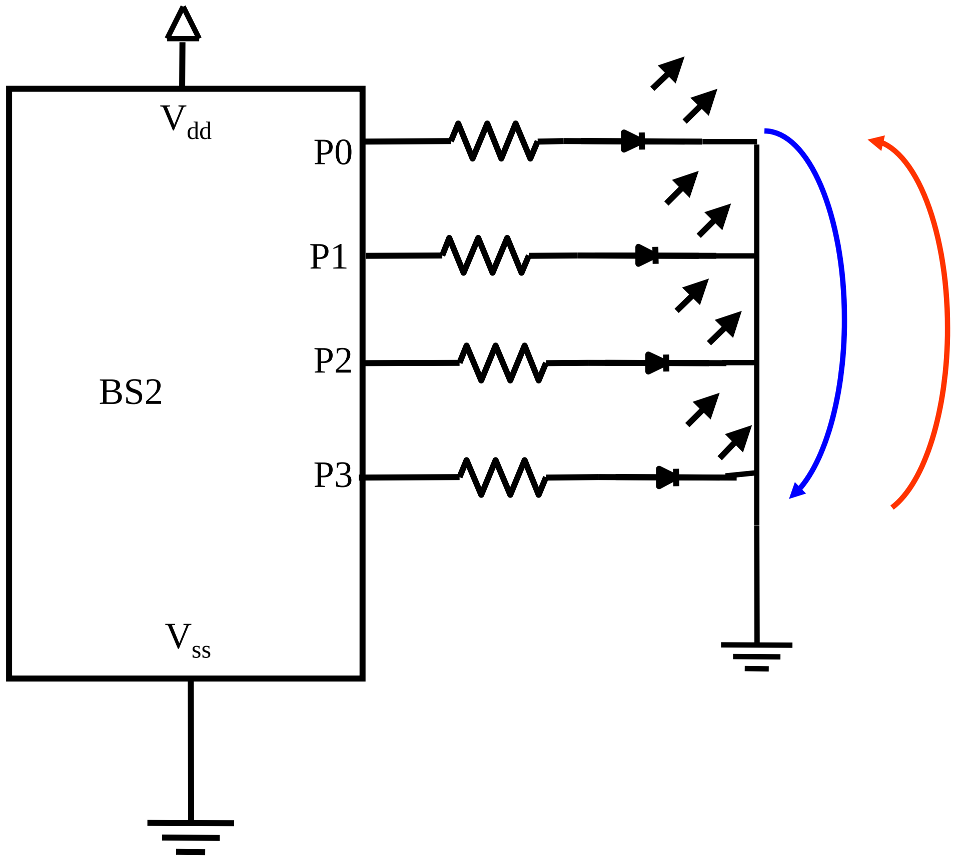
```
Ledpin CON 0
output Ledpin
high Ledpin
pause 100
low Ledpin
```



Dealing with Outputs (P0 to P4) : For…Next

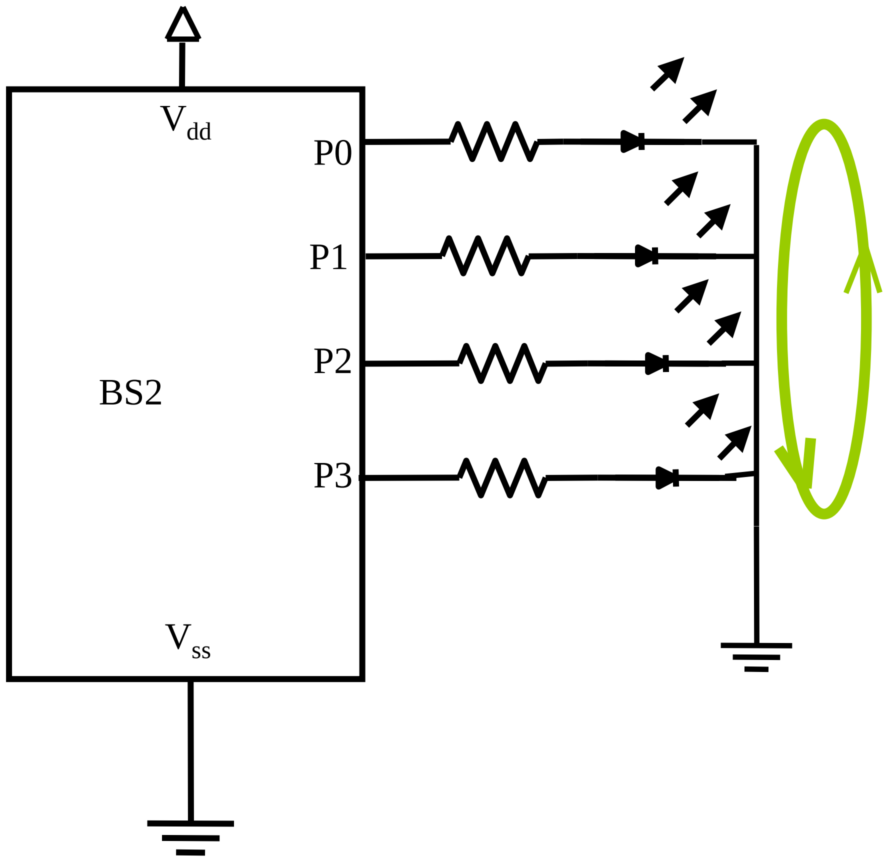
```
Ledpins Var nib
    for Ledpins = 0 to 3
        output Ledpins
        high Ledpins
        pause 1000
        low Ledpins
    next
```

```
Ledpins Var Nib
    For Ledpins = 3 to 0
        output Ledpins
        high Ledpins
        pause 1000
        low Ledpins
    next
```



Forward/Backward LED Display: Labels, Goto, If…Then

```
LEDs VAR OutL
DlyTime CON 1000
DirL = %00001111
LEDs = %00000001
FwdDisp:
pause DlyTime
LEDs = LEDs<<1
If LEDS = 00001000 then RevDisp
Goto FwdDisp
RevDisp:
Pause DlyTime
LEDs = LEDs>> 1
If LEDs = 00000001 then FwdDisp
Goto RevDisp
End
```



If…Then

- If (condition 1) or (condition 2) then (do X)
- If (condition 1) and (condition 2) then (do X)

Condition 1		Condition 2	Result
False	OR	False	False
False	OR	True	True
True	OR	False	True
True	OR	True	True

Condition 1		Condition 2	Result
False	AND	False	False
False	AND	True	False
True	AND	False	False
True	AND	True	True

Creating Alias for a Variable

- Given a byte size variable BytVar, to create a variable to get lower nibble value of BytVar use

  LowNib_BytVar VAR BytVar.LOWNIB

- To determine if a number stored in a counter variable “Ctr” is odd/even, create an alias:

  $$\text{Lowbit\_Ctr VAR Ctr}\underbrace{\text{.LOWBIT}}_{\text{or Ctr.BIT0}}$$

  – Now if Lowbit_Ctr is 1, the number in Ctr is odd else it is even!

Negative Numbers with Fixed-Size Integers

- Suppose we roll the odometer of a car backwards.
- After rolling down to the odometer to 00000 if we kept going, the odometer will next read 99999 (not -1)!
- Consider the following PBasic code snippet:

```
FixSizVar Var Byte
FixSizVar=0-1
Debug Bin FixSizVar
```

- Similar to the odometer example, the above code will produce 255, not -1. Recall, binary representation of 255 is %11111111. Similarly,
  - FixSizVar=0-2 will produce 254 not -2. Recall, binary representation of 254 is %11111110.
  - FixSizVar=0-3 will produce 253 not -3. Recall, binary representation of 253 is %11111101.
- The negative numbers follow a pattern know as two's complement.
- Instead of subtracting a number from 0 to produce its negative, we can do the following.
  - Represent the given number in binary → Invert all bits → Add 1 to the result
- Binary representation of -1, -2, -3, etc:

$$1 \xrightarrow{\text{D2B}} 00000001 \xrightarrow{\text{InvB}} 11111110 \xrightarrow{+1} 11111111$$

$$2 \xrightarrow{\text{D2B}} 00000010 \xrightarrow{\text{InvB}} 11111101 \xrightarrow{+1} 11111110$$

$$3 \xrightarrow{\text{D2B}} 00000011 \xrightarrow{\text{InvB}} 11111100 \xrightarrow{+1} 11111101$$

- To represent positive and negative values using a byte sized variable, by using the MSB to encode sign of the number, we can represent values from -127 to +127.
- If MSB=0, then we have a positive number and no special treatment is needed.
- If MSB=1, then we have a negative number.

2’s Complement Framework for Negative Numbers —I

- Let us restrict ourselves to working with NIB for now.
- For positive numbers: one NIB can give decimal numbers 0 to 15
- In order to use both positive and negative numbers:
  - we use the 3 low bits to encode the number
  - we use the MSB to encode the sign of number, 0 for +ve and 1 for –ve
- Now we can store decimal numbers 0 to 7.
- For positive numbers: we have

  0000→0

  0001→1

  0010→2

  0011→3

  0100→4

  0101→5

  0110→6

  0111→7

2’s Complement Framework for Negative Numbers —II

- Now for negative numbers, we use the following methodology.
- Let us say we are given a number: 1010
- Since the MSB is 1, we know we are dealing with a –ve number.
- To get the decimal equivalent of 1010, we first subtract 1 from 1010. For this, we get the decimal value of 1010 considering a NIB is being used to store 0 to 15:
  – 1010→10 so 10-1=9→1001
- Now we take the binary representation of 9 and invert each one of its bits, which yields:
  – 0110, whose decimal equivalent is 6.
- → The negative number for 1010 is –6!

2’s Complement Framework for Negative Numbers —III

- Decimal representation of 4-bit binary number where MSB is used to encode sign of number

$$1001 \xrightarrow{B2D} 9 \xrightarrow{-1} 8 \xrightarrow{D2B} 1000 \xrightarrow{InvB} 0111 \xrightarrow{B2D} -7$$

$$1010 \xrightarrow{B2D} 10 \xrightarrow{-1} 9 \xrightarrow{D2B} 1001 \xrightarrow{InvB} 0110 \xrightarrow{B2D} -6$$

$$1011 \xrightarrow{B2D} 11 \xrightarrow{-1} 10 \xrightarrow{D2B} 1010 \xrightarrow{InvB} 0101 \xrightarrow{B2D} -5$$

$$1100 \xrightarrow{B2D} 12 \xrightarrow{-1} 11 \xrightarrow{D2B} 1011 \xrightarrow{InvB} 0100 \xrightarrow{B2D} -4$$

$$1101 \xrightarrow{B2D} 13 \xrightarrow{-1} 12 \xrightarrow{D2B} 1100 \xrightarrow{InvB} 0011 \xrightarrow{B2D} -3$$

$$1110 \xrightarrow{B2D} 14 \xrightarrow{-1} 13 \xrightarrow{D2B} 1101 \xrightarrow{InvB} 0010 \xrightarrow{B2D} -2$$

$$1111 \xrightarrow{B2D} 15 \xrightarrow{-1} 14 \xrightarrow{D2B} 1110 \xrightarrow{InvB} 0001 \xrightarrow{B2D} -1$$

2's Complement Framework for Negative Numbers —IV

- Compute 7-5 = ?

$$a = 7 \Rightarrow 0111$$

$$b = 5 \Rightarrow 0101$$

–To get –5, use the following:

$$-5 \xrightarrow{D2B} 0101 \xrightarrow{InvB} 1010 \xrightarrow{B2D} 10 \xrightarrow{+1} 11 \xrightarrow{D2B} 1011$$

–So $-5 \Rightarrow 1011 \xrightarrow{B2D} 11$

–So if we form $7-5 = \underset{7}{0111} + \underset{11}{1011} = 18 \xrightarrow{D2B} 10010$

but nib can store only the lower 4 bits of above $\Rightarrow 0010$

$\Rightarrow 2$

In decimal system

–So we get the correct answer: <u>7-5=2</u> !!

2's Complement Framework for Negative Numbers —V

•To compute 5-7 = ?

•Using the table:

$$5 = 0101 \Rightarrow 5$$

$$\underbrace{-7 = 1001} \Rightarrow 9$$

•Then,

$$5 + (-7) = 0101 + 1001 \Rightarrow 5 + 9 = 14$$

•Again, from table

$$14 \xrightarrow{D2B} 1110 \Rightarrow -2$$

–Since highest bit of nib is 1, we have a negative number.

Bit Inversion using Exclusive OR Operator

- The exclusive OR operator “^” works on a bit-wise basis.
- ^ compares corresponding bits in its two arguments:
  - set the result bit for current location to 1 if either of the two bits in argument are 1
  - set the result bit for current location to 0 if both bits in argument are 1
- Implementing bit inversion using ^:
  - given a = 1010, to invert all bits use: 1010^1111→0101

Arithmetic Operations via Bit Shifting

- Multiply a binary number by 2: e.g., choose a random binary number, say 00101101
- First, write 00101101 as a sum of powers of two

$$0010\ 1101 = (1 \times 2^5) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^0)$$

- Now, multiply 00101101 by 2:

$$0010\ 1101 \times 2 = 2(1 \times 2^5) + 2(1 \times 2^3) + 2(1 \times 2^2) + 2(1 \times 2^0)$$

$$= (1 \times 2^6) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^1)$$

- Note that multiplication of the original number by 2 simply causes the exponents on the 2s increase by one each. The result of the multiplication is still a sum of powers of two with corresponding binary number:

$$0010\ 1101 \times 2 = (1 * 2^6) + (1 * 2^4) + (1 * 2^3) + (1 * 2^1) = 0101\ 1010$$

- Note that the resulting binary number is the original number where each bit has been shifted left by one position. (A left shift increases the weight on each bit by one place!)
- To multiply a binary number by 2^n, simply shift the bits of the original binary number n places to left!
- Similarly, to divide a binary number by 2, simply shift the bits of the original binary number right by one position. (Integer division only, can't get fractional value out of this!)

Multiply High operator **

- The ** operator (Multiply High operator): multiplies two numbers and returns the high 16 bits of the result
- Application: Multiply a number by a fractional value less than one
  - Example: Multiply 10000 by 0.72562
- First, Express the fraction in units of 1/65536, i.e., multiply the fractional value by 65536
  - The fraction 0.72562 is represented by 0.72562 × 65536=47554
- Next, compute 10000**47554
- Binary representation of

** operator drops these lower 16 bits

$$10000 \times 47554 \rightarrow \underline{1110001011000}\boxed{0010101000100000}$$

Decimal value = 7256

To get 10000× 0.72562, divide the above by 65536

→ divide by 2 sixteen time

→shift bits to right 16 times

→ keep only the high 16 bits.

```
Frac CON 47554 ' = 0.72562 x 65536
value VAR Word
value = 10000
value = value ** Frac ' Multiply 10000 by 0.72562
DEBUG ? value ' Show result (7256)
```

Multiply High operator **: Example

- Example : Distance measurement using PING))) sensor:
  - –*D=0.5*×*C*×*T*, *D*: distance, *C*: speed of sound in air (344.8m/s), *T*: round-trip time (2μs units)

$$D_{\text{cm}} = \left(\frac{1}{2}\right) \times \overbrace{\left(100 \times C\right)}^{\text{Speed: cm/s}} \times \left(\overbrace{T \times 2 \times 10^{-6}}^{\text{Time: seconds}}\right) = \frac{C \times T}{10^4} = \frac{344.8}{10^4} \times T = T \times 0.03448$$

  - –0.03448 → INT(0.0348 × 65536) = 2260

```
MeasDist VAR Word
RtripTime VAR Word
ConConst Const 2260
MeasDist = RtripTime **ConConst
DEBUG DEC MeasDist, “ cm”
```

Multiply Middle operator */

- The */ operator (Multiply Middle operator): multiplies two numbers and returns the middle 16 bits of the result
- Application: Multiply a number by a whole number and a fraction
  - Example: Multiply 100 by 1.5
- The whole number is the upper byte of the multiplier (0 to 255 whole units) and the fraction is the lower byte of the multiplier (0 to 255 units of 1/256 each
- When computing 100*1.5, the whole number (upper byte of multiplier)=1, and the fractional part (lower byte of multiplier)=128, since 128/256 = 0.5.
- Easy to express the multiplier in hex—as $0180—since hex keeps the contents of the upper and lower bytes separate.
- To calculate the multiplier for use with the */ operator, multiply the original (mixed) value by 256, convert to an integer, and corresponding hex value.
  - Example: Pi (π=3.14159). The */ constant would be INT(3.14159 * 256) = 804 ($0324).

```
value1 VAR Word
value1 = 100
value1 = value1*/ $0180 ' Multiply by 1.5 [1 + (128/256)]
DEBUG ? value1 ' Show result (150)
```

Multiply Middle operator */: Examples

- Example 1: Pi (π=3.14159). The */ constant would be
  - INT(3.14159 * 256) = 804 ($0324).
  - HBytePi=3 ($03), LBytePi=INT(0.14159 * 256) = 36 ($24)→WordPi=$0324

- Example 2: Compute 100×1.5.
  - 1.5 → HByte1_5=1 ($01), LByte1_5=INT(0.5× 256) = 128 ($80)→WordPi=$0180

```
value1 VAR Word
value1 = 100
value1 = value1*/ $0180 ' Multiply by 1.5 [1 + (128/256)]
DEBUG ? value1 ' Show result (150)
```

Hands-on Exercises—I

LED: What's a Microcontroller?	Chapter 2
LED: StampWorks Manual (See Parallax CD)	p25 — p56
Button: What's a Microcontroller?	Chapter 3
Button: StampWorks Manual (See Parallax CD)	Exp. #15
If…then: BASIC Stamp Syntax and Reference Manual 2.2	p231— p242
For…next: BASIC Stamp Syntax and Reference Manual 2.2	p191— p197
Variables: BASIC Stamp Syntax and Reference Manual 2.2	p85— p86

Hands-on Exercises—II

**: BASIC Stamp Syntax and Reference Manual 2.2	p111— p112
*/: BASIC Stamp Syntax and Reference Manual 2.2	p112
//: BASIC Stamp Syntax and Reference Manual 2.2	p113— p114
Binary operators: BASIC Stamp Syntax and Reference Manual 2.2	p109— p122