WAM
Chapter 3

Before we start Chapter 3 here is something you should know.

When choosing components you need to consider

- **voltage ratings**
- **current and/or power ratings**
- **whether or not time constraints apply**

Example:

We connected LED's to the Basic Stamp. It worked because the stamp supplied enough voltage and current.

How many LED's can we safely connect to a single output line?

How many LED's can the entire stamp supply without extra circuitry?

We used a 470 Ω resistor.

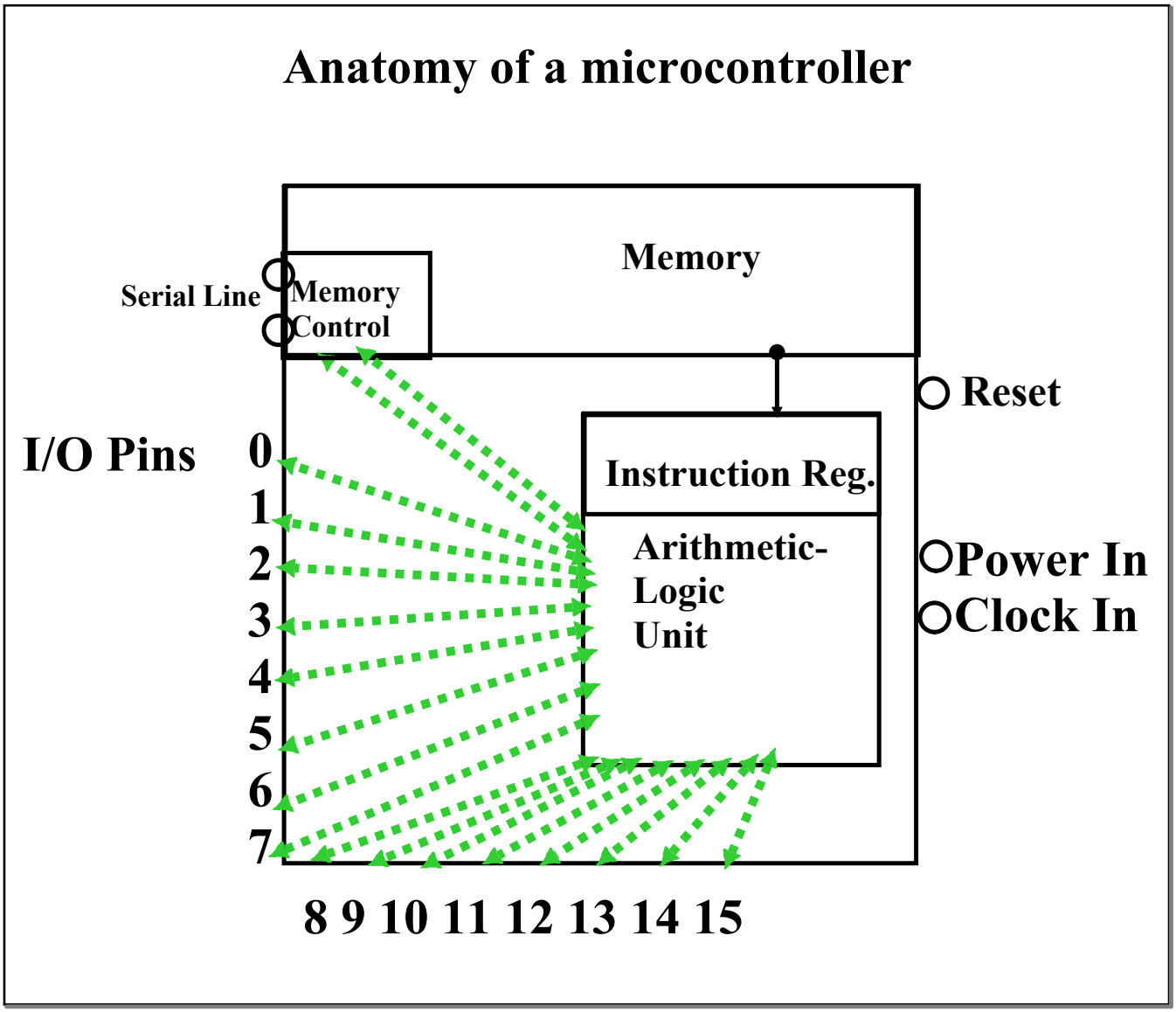
Current = Voltage/Resistance

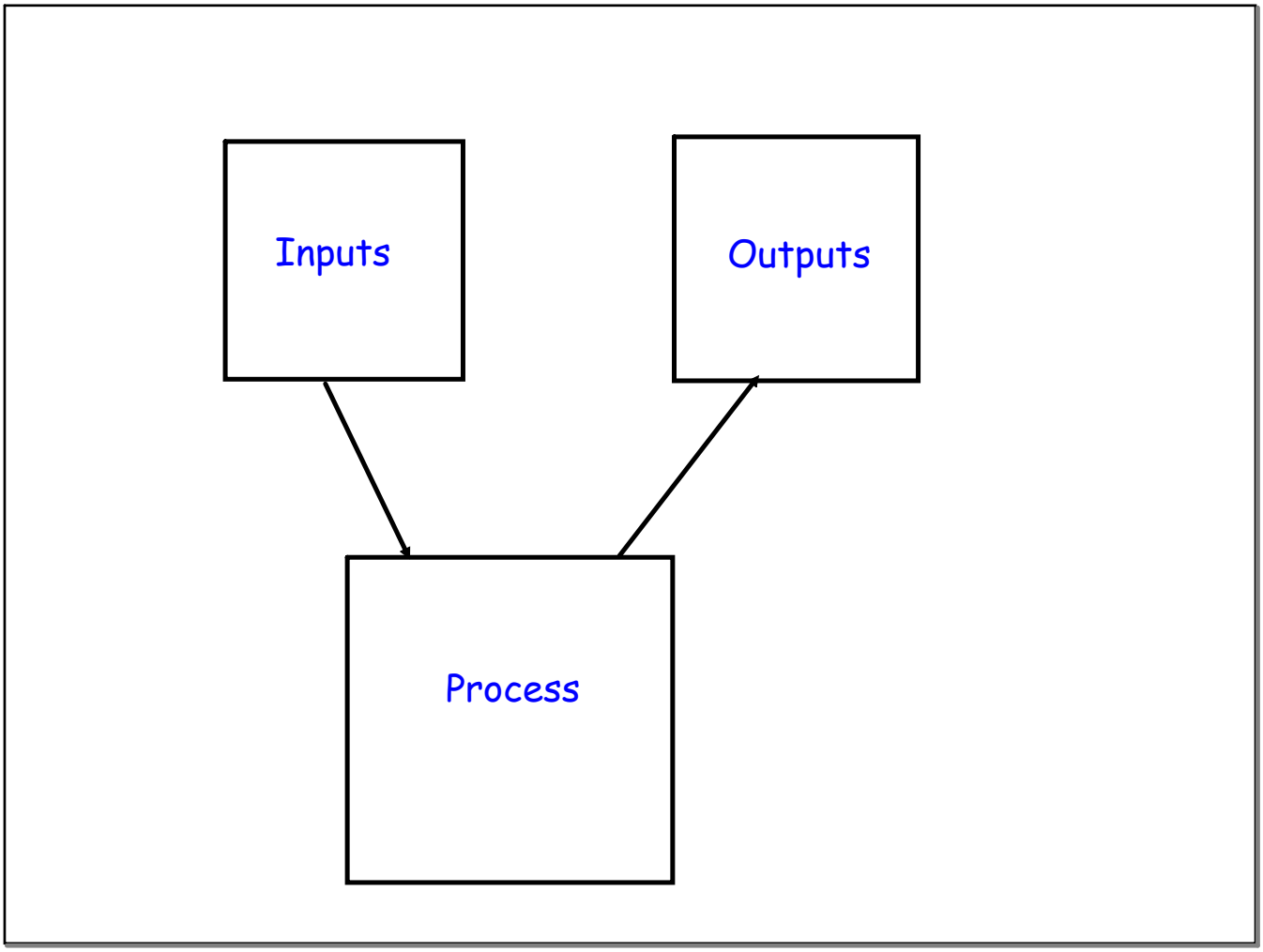
Ignoring the resistance of the LED, then:

Current = 5V/470Ω = 0.0106 Amp = 10.6 mA

Therefore, we were using about 11 mA for one LED. What would happen if we had several LED's on at the same time?

Released Products	BS2-IC	BS2e-IC
Package	24-pin DIP	24-pin DIP
Package Size (L x W x H)	1.2"x0.6"x0.4"	1.2"x0.6"x0.4"
Environment	-40°C - +85°C (-40°F - +185°F) **	0° - 70°C* (32° - 158° F) **
Microcontroller	Microchip PIC16C57c	Ubicom SX28AC
Processor Speed	20 MHz	20 MHz
Program Execution Speed	~4,000 instructions/sec.	~4,000 instructions/sec.
RAM Size	32 Bytes (6 I/O, 26 Variable)	32 Bytes (6 I/O, 26 Variable)
Scratch Pad RAM	N/A	64 Bytes
EEPROM (Program) Size	2K Bytes, ~500 instructions	8x2K Bytes, ~4,000 instructi
Number of I/O pins	16 +2 Dedicated Serial	16 +2 Dedicated Serial
Voltage Requirements	5 - 15 vdc	5 - 12 vdc
Current Draw @ 5V	3 mA Run / 50 µA Sleep	25 mA Run / 200 µA Sleep
Source / Sink Current per I/O	20 mA / 25 mA	30 mA / 30 mA
Source / Sink Current per unit	40 mA / 50 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins
PBASIC Commands	42	45
PC Programming Interface	Serial Port (9600 baud)	Serial Port (9600 baud)
Windows Text Editor	N/A	N/A
	Stampw.exe (v1.04 and up)	Stampw.exe (v1.096 and up)


Anatomy of a microcontroller
Memory
Serial Line
Memory
Control
Reset
I/O Pins
0
1
2
3
4
5
6
7
Instruction Reg.
Arithmetic-
Logic
Unit
Power In
Clock In
8 9 10 11 12 13 14 15

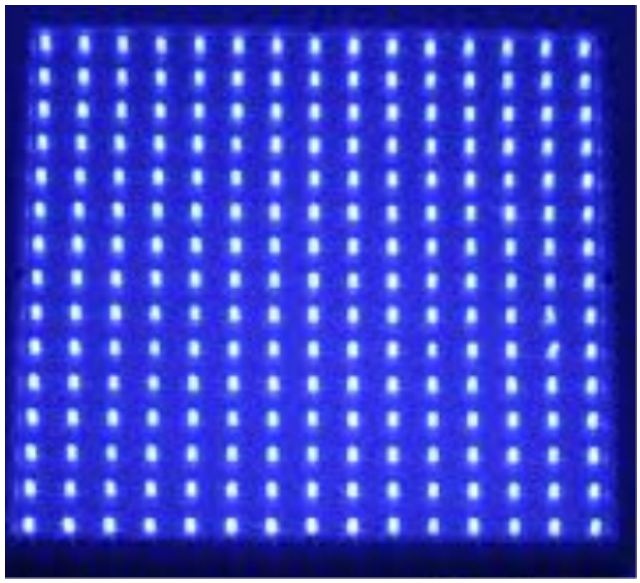

Inputs
Outputs
Process

Now from chapter 2 you should remember how to program outputs

High n

Low n

n = 0-15



Programming inputs looks very much the same

In n

n = 0 - 15

Making Decisions

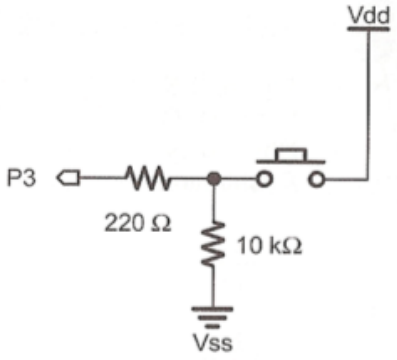
Is the button pressed?
- **Yes - Light the RED led**
- **No - Do nothing**

Is the button pressed?
- **Yes - Light the RED led**
- **No - Light the Green led**

But how do we connect the buttons?

There are two ways you can wire a button and they produce opposite results

One Schematic Diagram for a button



The 220 ohm resistor limits current (protecting the input)

The 10 k ohm "pull down" resistor holds the pin at ground unless the button is pushed.

When the button is NOT pushed (open) there is a low voltage at pin 3. [logic = 0]

When the button IS pushed (closed) there is a high voltage at pin 3. [logic = 1]

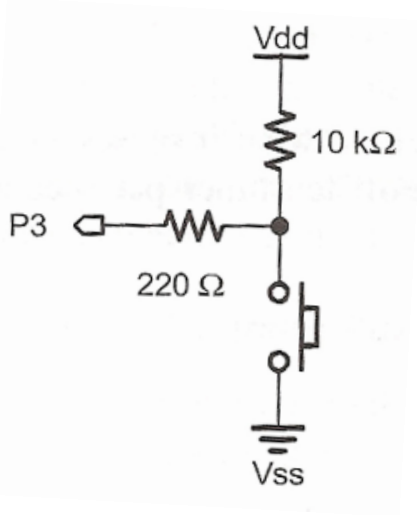
Alternate Schematic Diagram for a button

The 220 ohm resistor limits current (protecting the input)

The 10 k ohm "pull up" resistor holds the pin at Vdd unless the button is pushed.

When the button is NOT pushed (open) there is a high voltage at pin 3. [logic = 1]

When the button IS pushed (closed) there is a low voltage at pin 3. [logic = 0]



```
If (condition) then
        command
        command
        command
ENDIF

----------------------------------------
If button on pin 5 is pushed, then
light LED on pin 15
----------------------------------------

IF (In5 = 1) THEN
        High 15
ENDIF
```

If button on pin 3 is pushed then light LED on pin 11 should be on. If the button is not pushed then the LED on pin 12 should be on.

```
If (IN3 = 1) THEN
        HIGH 11
        LOW 12
ELSE
        LOW 11
        HIGH 12
ENDIF
```

```
If (condition) then
        command
        .
        .
        command
ELSEIF (condition)
        command
        command
ELSE
        command
ENDIF
```

Here is an exercise:

You have a Red LED and a Green LED.
You have a button

You want the Green light to be on
UNLESS the button is pushed.

If the button is pushed,
the green should go off
and the red should go on.

Step 1.
Decide on the logic you want to use.

Input:
Let's say that the button will be active LOW on pin 6

Output:
A HIGH on pin 15 will turn on the green LED
A HIGH on pin 14 will turn on the red LED

Step 2:
DrawSchematic Diagrams:

Button

GREEN LED

RED LED

STEP 3 Write the program

```
' {$STAMP BS2}
' {$PBASIC 2.5}
' Button on p6 is high until pushed (active low)
'If button not pushed (high) only green light is on
'If button pushed, only RED light is on
'Red light pin 15
'Green light  pin 14

DO
      IF (IN6 = 0) THEN  'button is pushed
            HIGH 15
            LOW 14
      ELSE                    'button not pushed
            LOW 15
            HIGH 14
      ENDIF
LOOP
```

Test:

When button NOT pushed, green light is on

When button is pushed then
Green light goes out
Pause for 1 second
Red light goes on and stays on
Pause for 10 seconds

Go back to monitoring button

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DO
        IF IN7=0 THEN  'button not pushed
                LOW 5
                HIGH 6
        ELSE  'button is pushed
                LOW 6
                PAUSE 1000
                HIGH 5
                PAUSE 10000
        ENDIF
LOOP
```

Chapter 3 WAM book

Important hardware concepts

- The button pin-outs - page 72
- Active HIGH button wiring - page 76
- Active LOW button wiring - page 79

Important software concepts

- High n
- Low n
- Pause n
- IN n
- IF (condition) THEN

...
{ELSE IF (condition) THEN
...}
ENDIF