Microcontrollers

Press Play: Interactive Device Design | July 02, 2012

Homework Debrief

Examples | Critique (I Like, I Wish)

Finding the Answer vs. Being Told

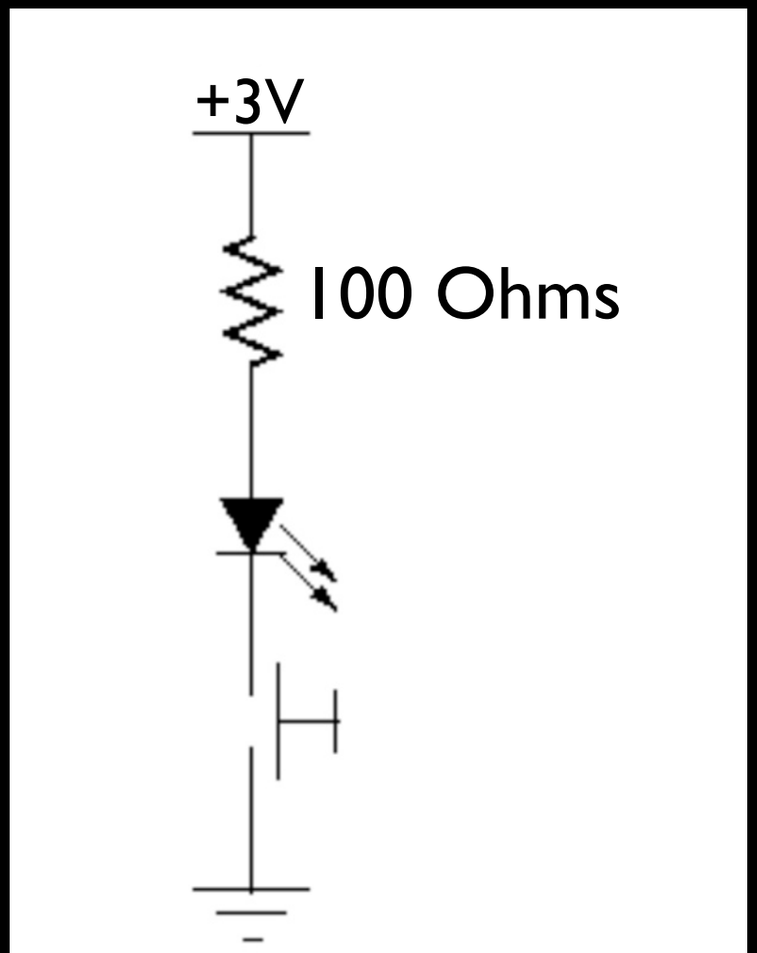
Lab Prep!

Frankenlight | Arduino Programming

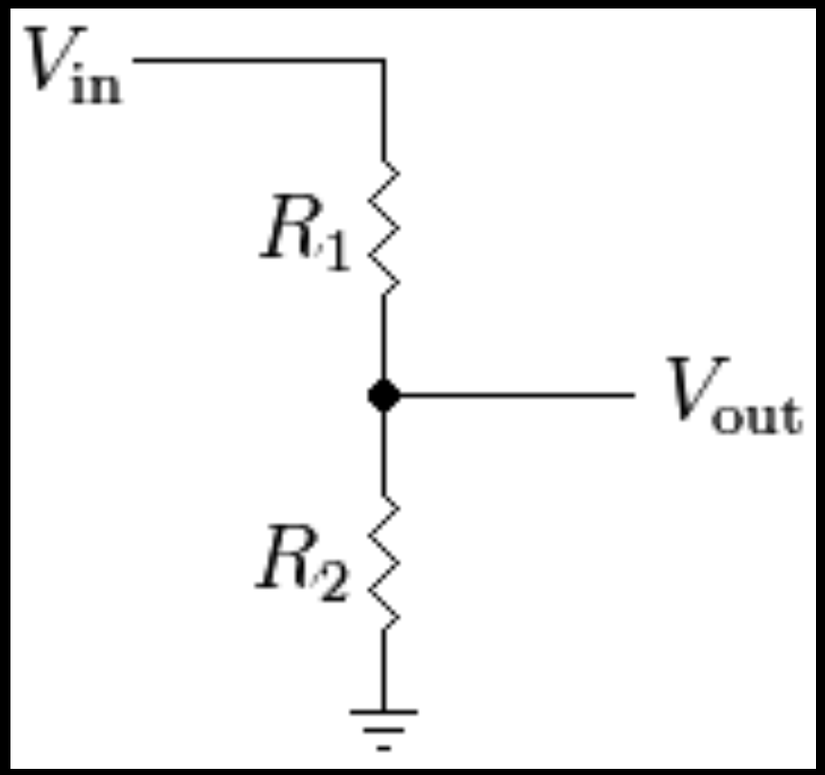
Basic Sensor Circuit

Button Circuit | Voltage Divider Circuit

A Button Circuit

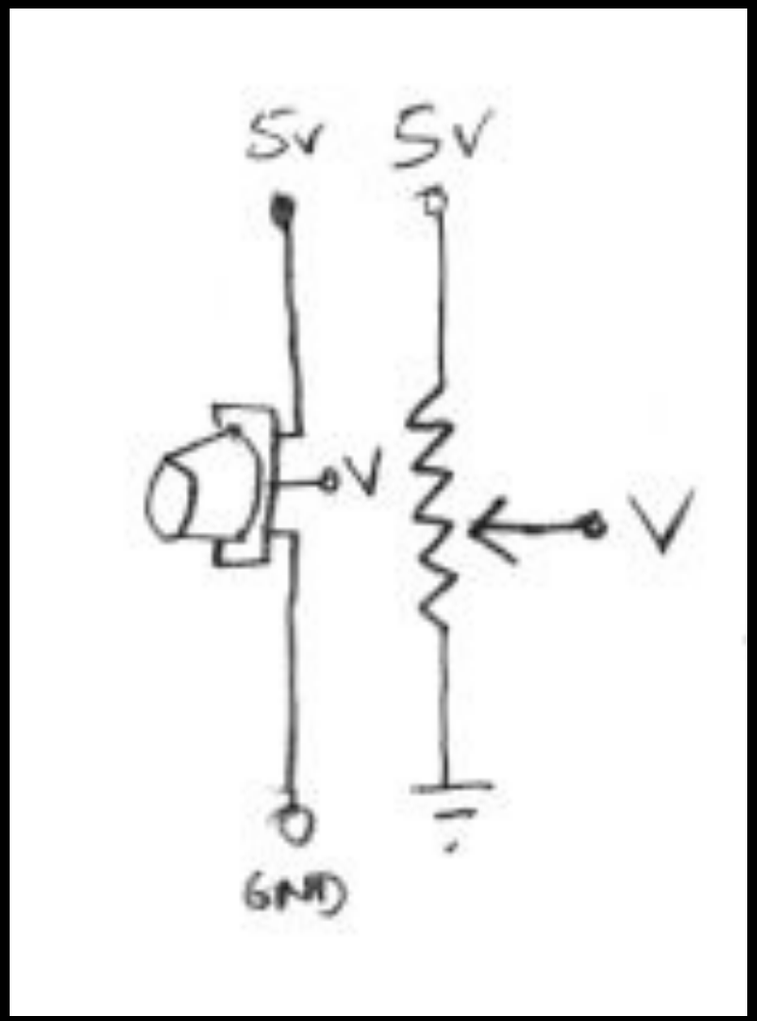


A Voltage Divider Circuit

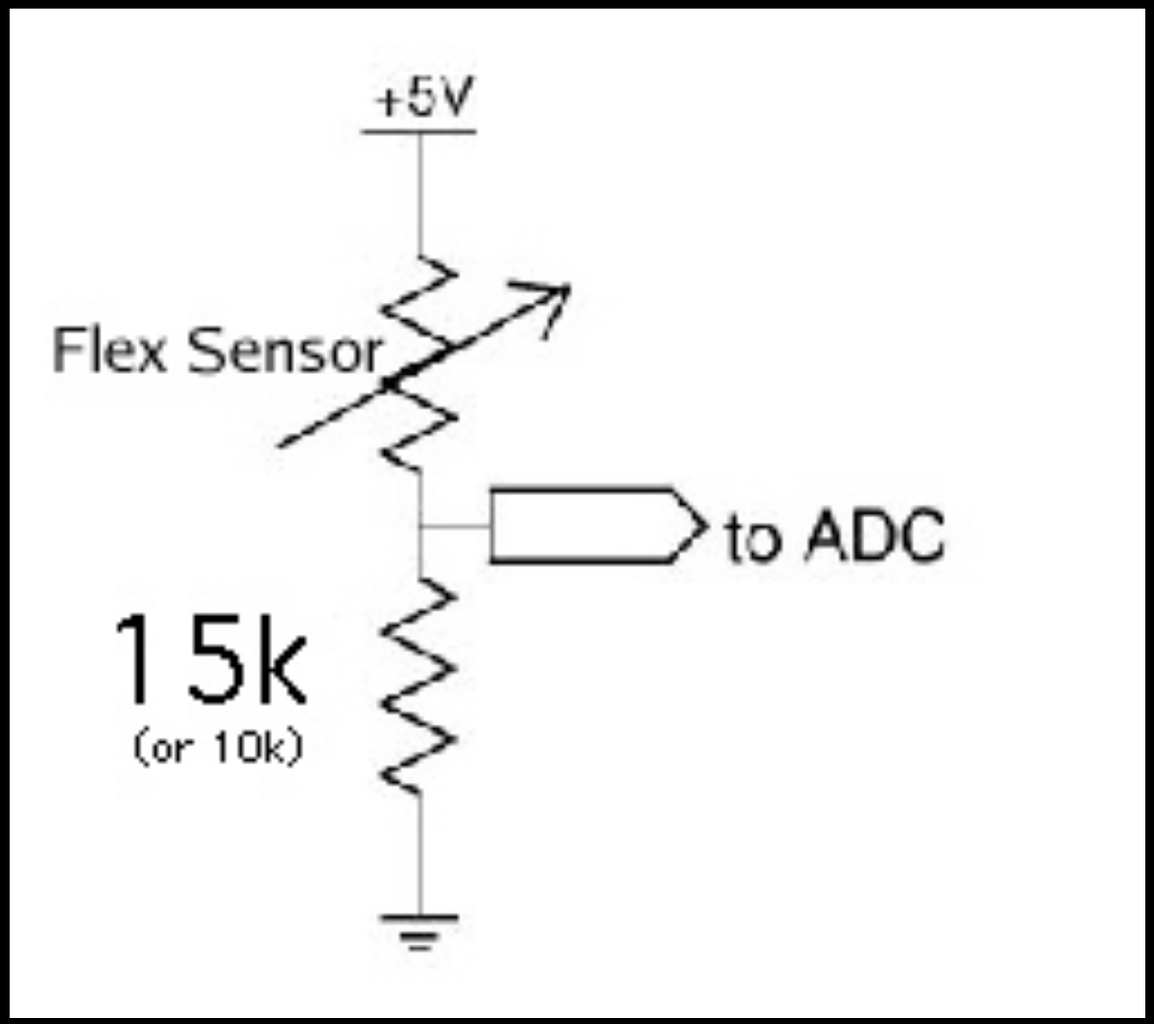


$$V_{out} = \frac{R2}{R1 + R2} \times V_{in}$$

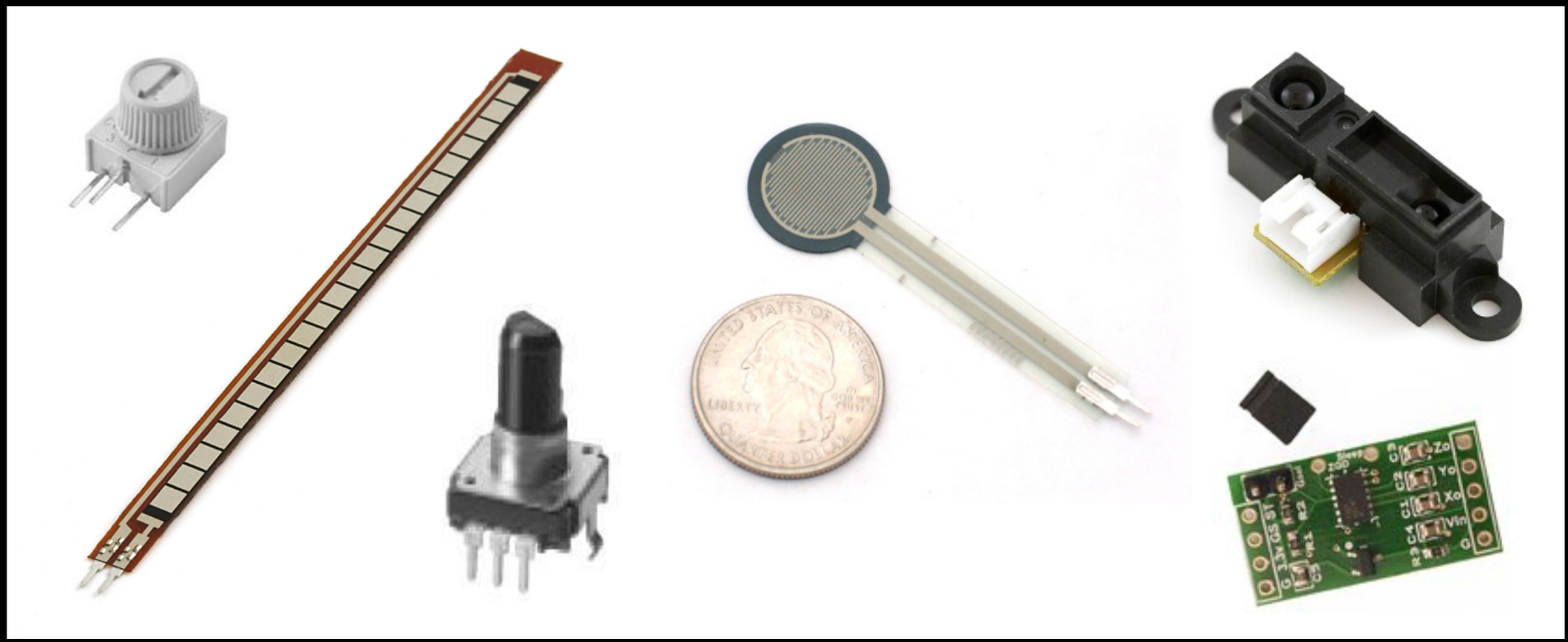
A Potentiometer Circuit



A Flex Sensor Circuit

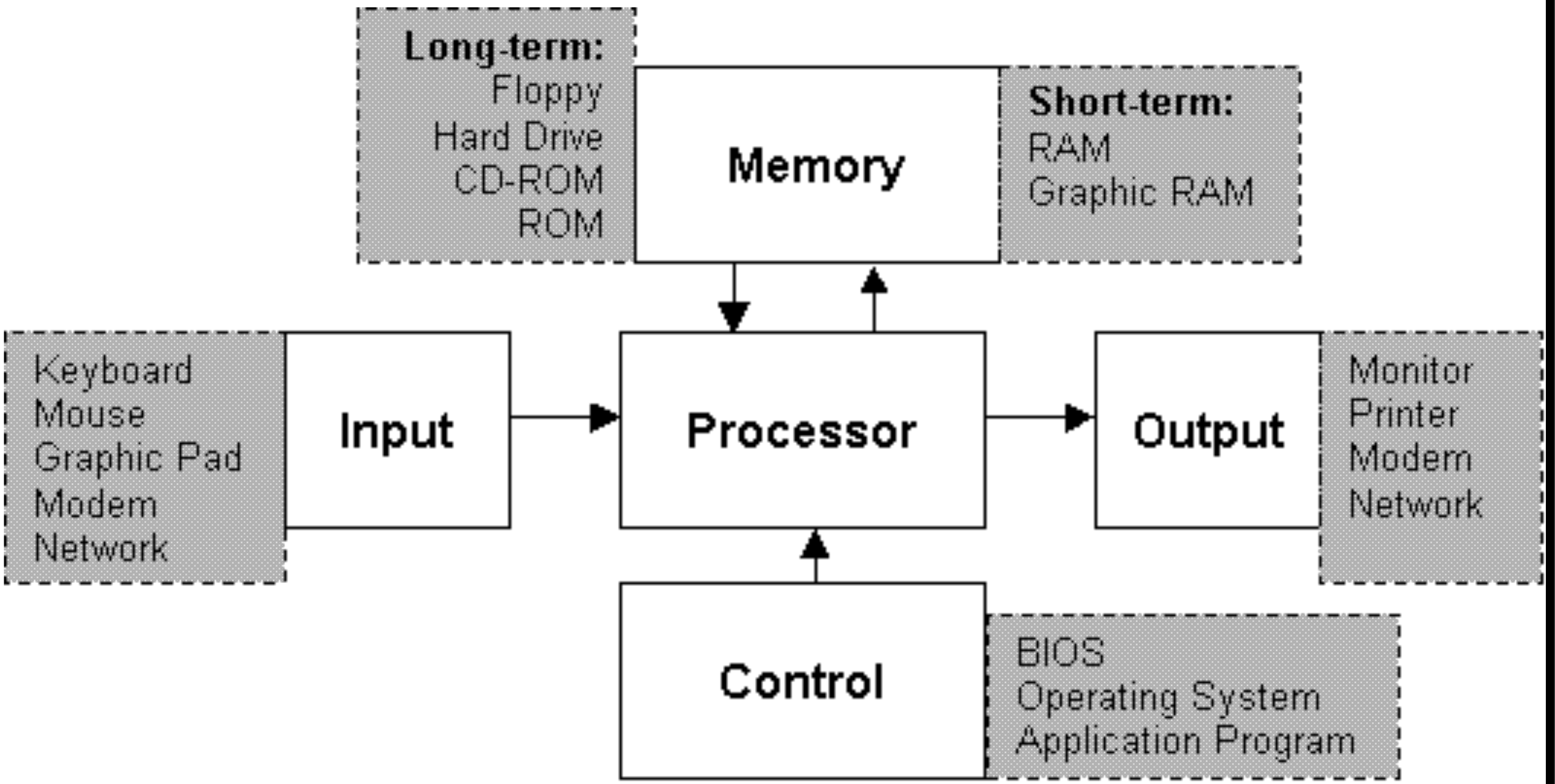


Your Lab Kit Sensors



Micro-Controllers Are

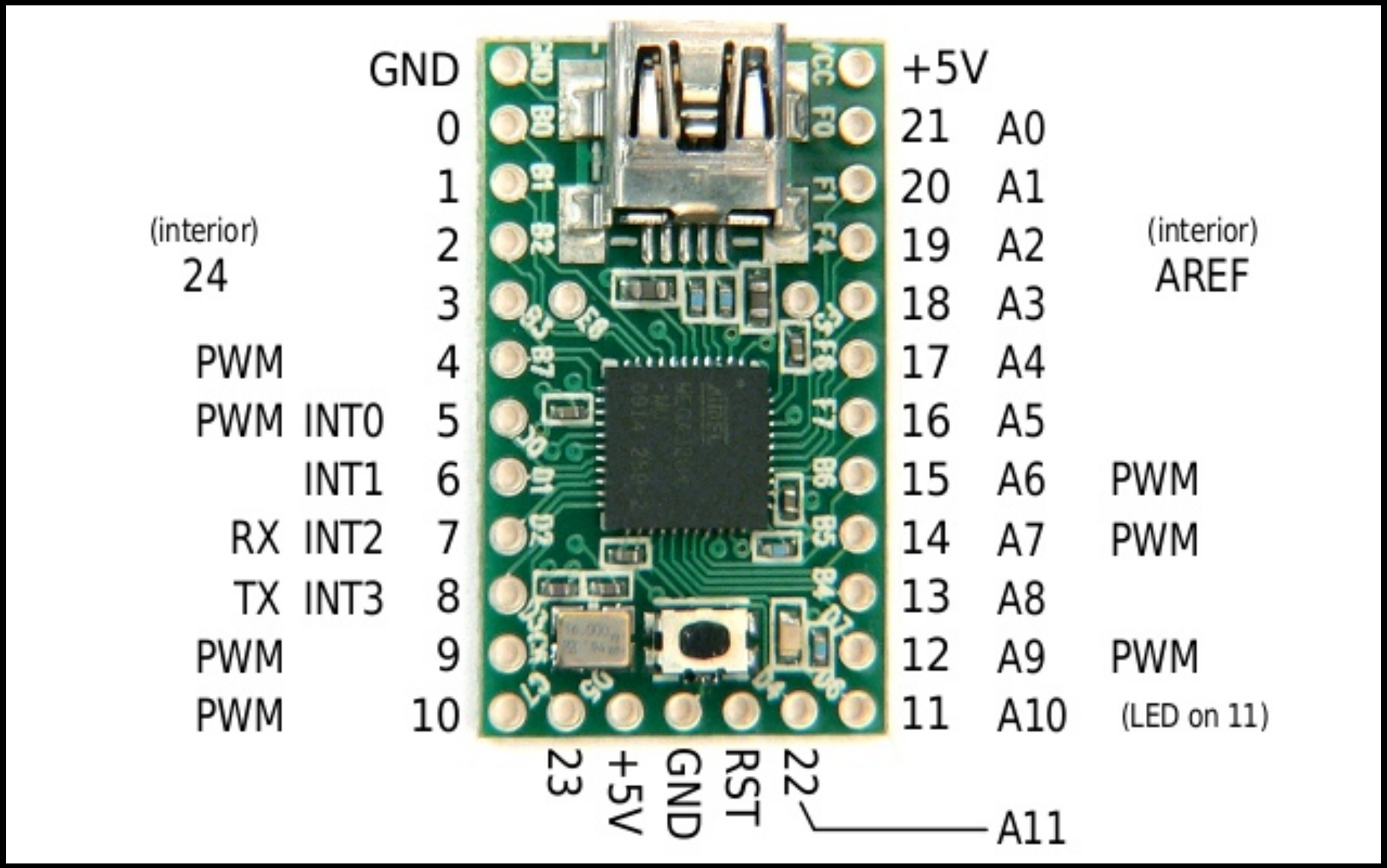
Very Small Computers


Long-term:
Floppy
Hard Drive
CD-ROM
ROM
Memory
Short-term:
RAM
Graphic RAM
Keyboard
Mouse
Graphic Pad
Modem
Network
Input
Processor
Output
Monitor
Printer
Modem
Network
Control
BIOS
Operating System
Application Program

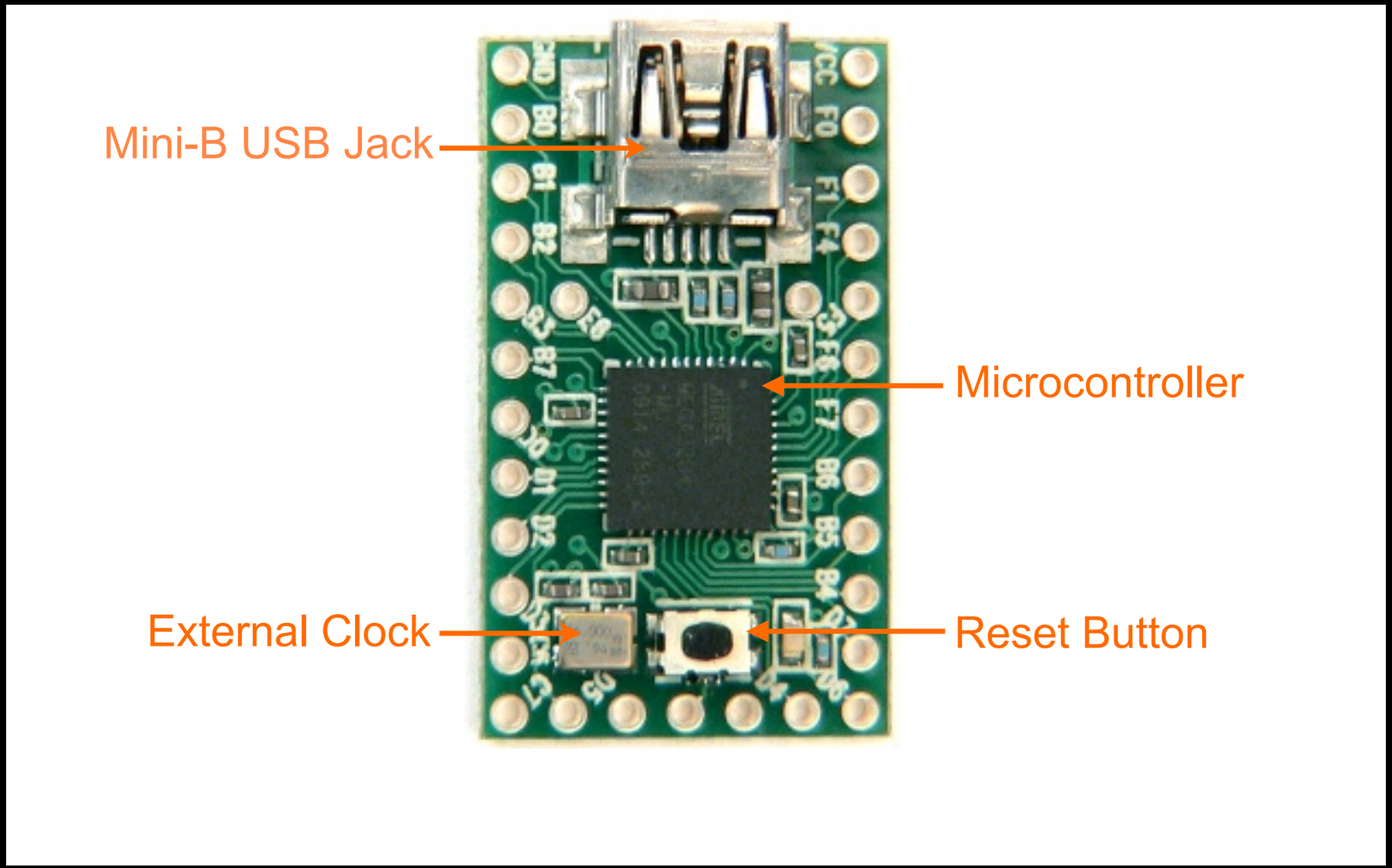
Microcontroller Architecture

Clock | Program Memory | Data Memory | Registers | Code

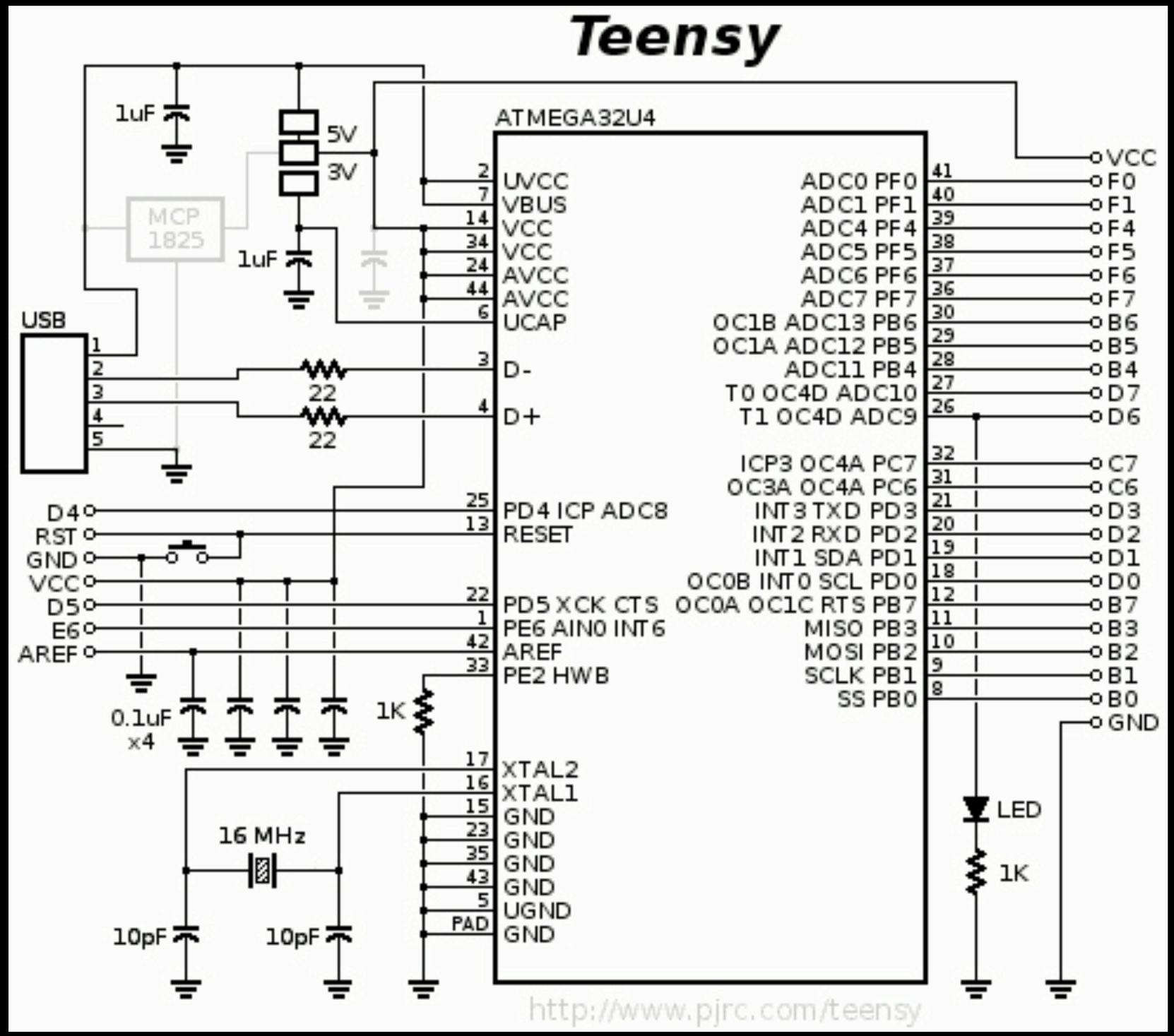
Physical Hardware:



Physical Hardware:



Physical Hardware:



Bits and Bytes:

1 Bit

0 = LO = 0 Volts

1 = HI = 5 Volts

2 Bits

00 = 0

01 = 1

10 = 2

11 = 3

3 Bits

000 = 0 ◀

001 = 1

010 = 2 ◀

011 = 3

100 = 4 ◀

101 = 5

110 = 6

111 = 7

4 Bits

1000 = 8 ◀

...

8 Bits = 1 Byte

1 1 1 1 0 0 1 1 = 2 + 1 + 128 + 64 + 32 + 16 = 243

Bits and Bytes:

- Binary has 2 possible digits: 0 and 1
- Decimal has 10 possible digits: 0 thru 9
- Hexadecimal has 16 possible "digits": 0 thru 9, A thru F

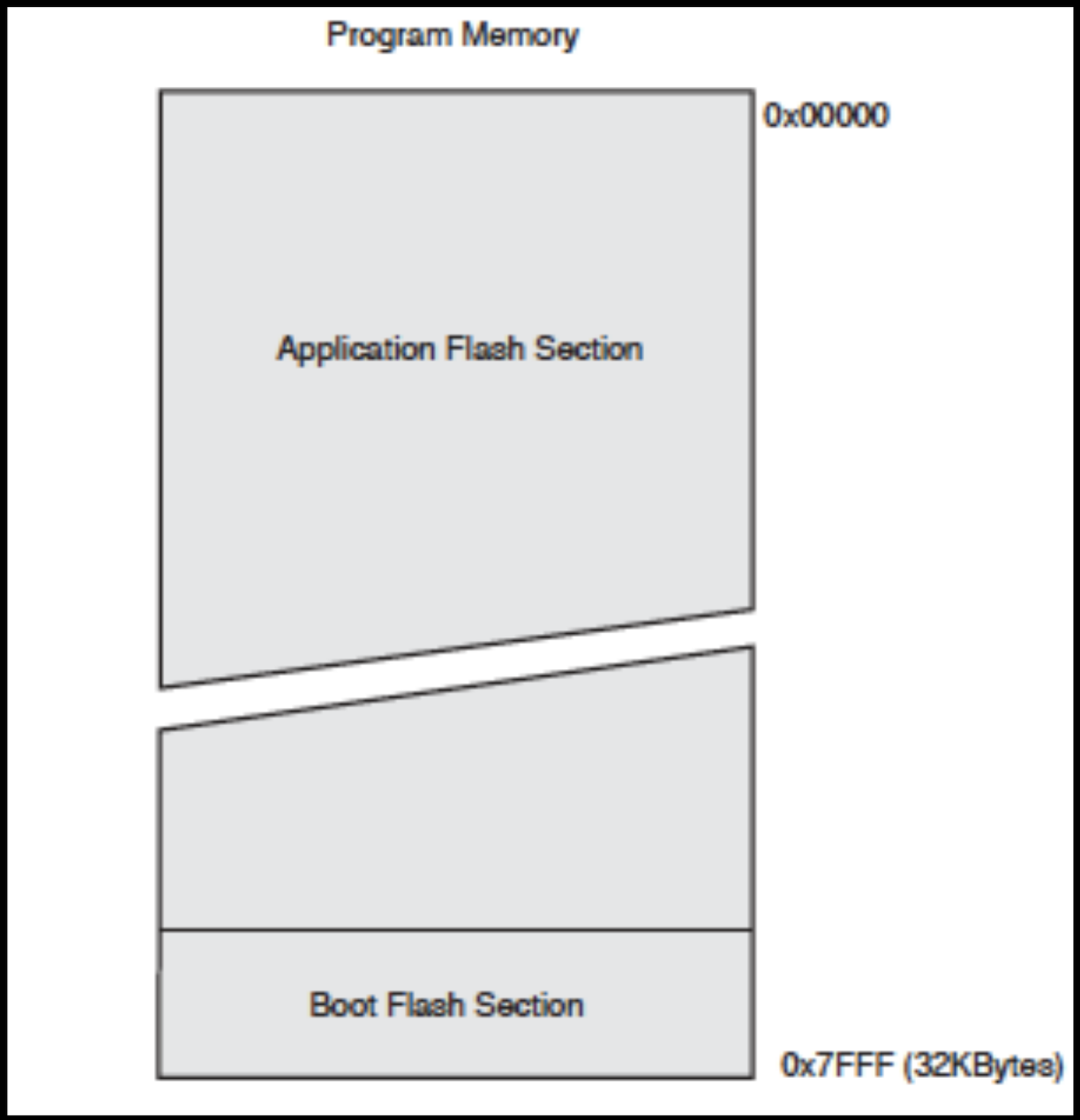
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Binary	Decimal	Hexadecimal
1111 0011	= 243	= F3

Bits and Bytes:

- 1 byte = 8 bits, yielding 256 unique values for each byte.
- All the information in the microcontroller is stored in byte-size chunks; we represent each byte of information as a two-digit hexadecimal number.
- 11110011 in binary = 243 in decimal = F3 in hexadecimal
- We typically write this as: b11110011 = 0xF3
- Memory addresses are 2 bytes long and are hex, as well, but preceded with $, e.g. $03DF.

Program Memory:



Data Memory:

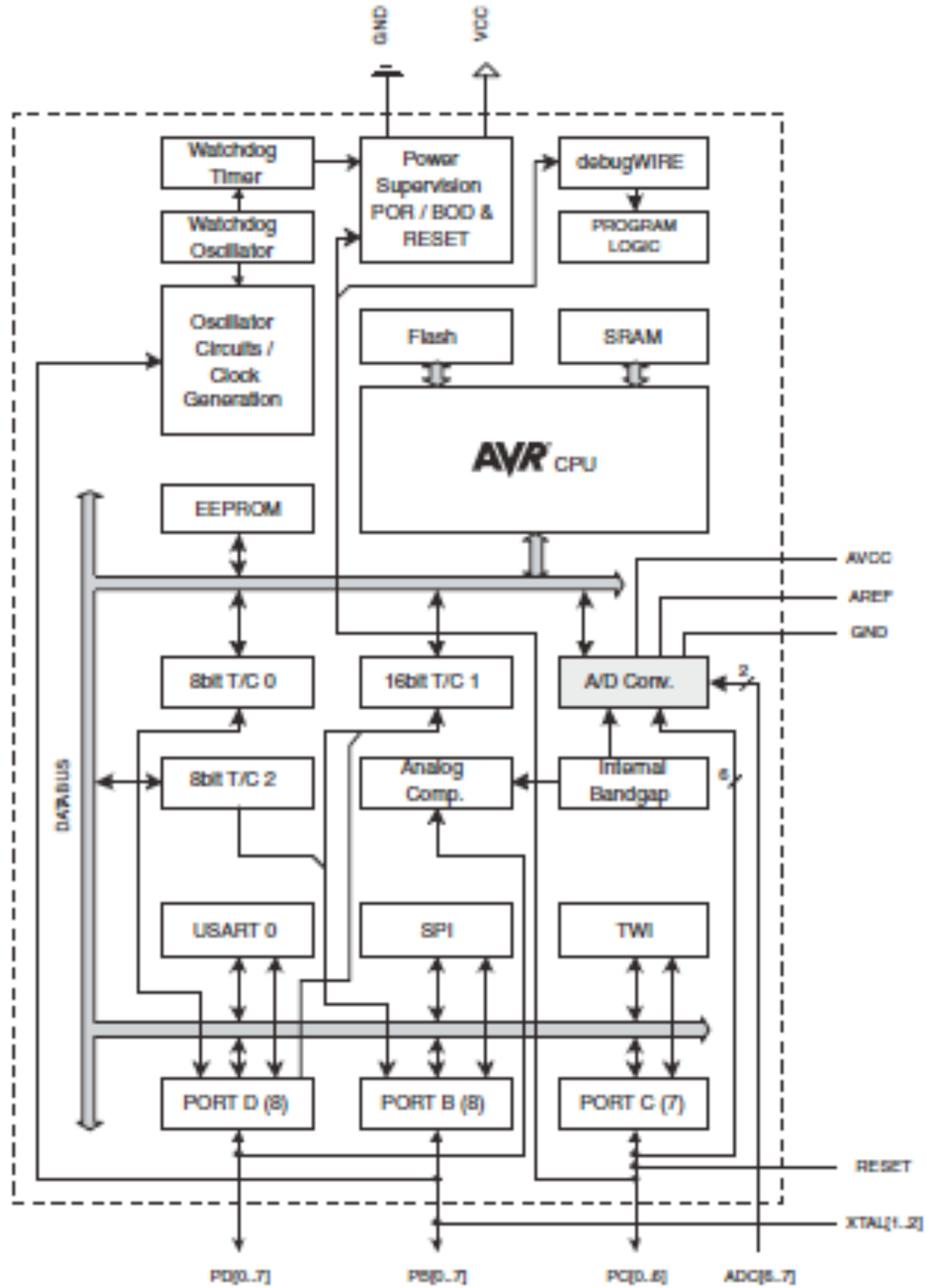
Address	Bit 7 (MSB)	6	5	4	3	2	1	0 (LSB)	Value in Memory (hex)	Value in Memory (decimal)
$0000	1	0	0	1	1	0	1	0	0x9A	154
$0001	0	1	1	0	1	1	1	1	0x6F	111
$0002	0	0	1	1	1	1	1	0	0x3E	62
...										
$07FD										
$07FE										
$07FF										

Address

IO Registers:

- PORT B: (PB7-PB0) 8-bit bi-directional IO
- PORT C: (PC 7, 6) 8-bit bi-directional IO
- PORT D: (PD7-0) 8-bit bi-directional IO
- PORT F: (PF7-4, PF1, PF0): analog inputs to A/D converter (can be used at 8-bit bi-directional IO)

Figure 2-1. Block Diagram



Data Direction Registers (DDR):

- Since the IO pins are configurable to be either input or output, the controller needs some place to store the directionality of each bit.

- These are stored in the Data Direction Registers. Like all the other registers, the DDRs have 1's and 0's, but its 1's and 0's indicate whether the corresponding port pin is an input (0) or output (1).

Port Features:

- Analog to Digital Conversion
- Pulse Width Modulation
- Timers & Counters
- External Interrupts
- Serial Peripheral Interface
- RX/TX

Arduino Software Environment

IDE | Structure of Arduino Programs | Flashing Programs

Sketch:

Blink | Arduino 1.0.1

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 hes the LED on Teensy 2.0
// Pin 6 hes the LED on Teensy++ 2.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

1 Teensy 2.0 on /dev/tty.usbmodem12341

Sketch:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6 has the LED on Teensy++ 2.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off my making the voltage LOW
  delay(1000);               // wait for a second
}
```

What happens when we flash code?

1. Code from libraries (if any) are included (linked).
2. Code is checked for errors (verified).
3. Code is “cross-compiled” into machine code (a.k.a. machine code or hex code) using avr-gcc.
4. Code is written to the program memory of the AVR over USB using the Teensy bootloader.

Flash Demonstration