

## LECTURE-1

This is a series of lectures written for those with mild electronics background to learn about the wild world of Robotics. I assume only that you know what electricity is and that you've touched an electrical component. Everything else is spelled out as much as possible. There is quite a lot here so take your time! It is also my intention to get book-hardened Engg. students to put down the calculator and to plug in an LED. Remember, if it smokes, at least you learned what not to do next time!

### What's a Microcontroller?

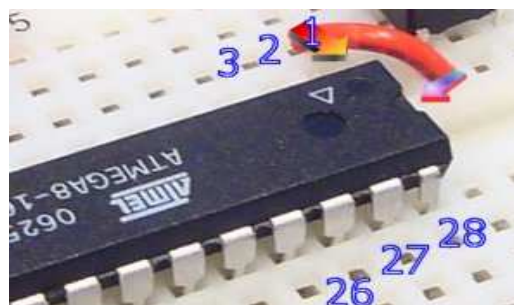
We all know what an OR gate is. An OR gate is a logic gate that takes two inputs and controls an output. We all have played with these types of gates, even possibly a DIP packaged OR gate with 4 OR gates built into it. This DIP package require a power pin and a ground pin. Electricity flowed through the IC and allowed it to operate. You may not be sure how the IC was built, but you understand that if you change the inputs, the output changes. One can do this by tying the inputs to either power (also known as VCC) or ground (GND).

A microcontroller is the same as an OR gate. We give some inputs, and have outputs. The crazy thing is that a micro runs code. Machine code to be specific. For instance, with a little bit of work, anyone can monitor the input of two pins A and B. And based on those inputs, can control an output pin C. So to replicate an OR gate:

```

if (A == 1 || B == 1)
{
  C = 1;
}
else
{
  C = 0;
}

```



It's C code! we can code up all sorts of different applications, compile code, load it onto a micro, power the micro, and the code runs. Very simple! Microcontrollers are used in all the electronics we take for granted such as your microwave, TV remote,

cell phone, mouse, printer, there's over 150 microcontrollers embedded into new cars! There's one waiting for you to depress the brakes (BRAKES == 1) and for the tires to lock up (LOCK\_UP == 1). When this happens, the micro releases the brakes, and you have ABS (anti-lock brake system).

In the old days, microcontrollers were OTP or one-time-programmable meaning we can only program the micro once, test the code, and if the code didn't work, throw it out and try again. Now micros are 'flash' based meaning they have flash memory built inside that allows their code to be written and rewritten thousands of times.

Flash micros are different than computers and RAM. Computers require tons of power and components to get up and running. Computers run HOT. Computers take forever and a day to boot. Micros are on and running code within milliseconds and if they're warm enough you can feel heat coming off of them, something is very wrong and you've probably blown the micro.

Now back to that OR gate IC. It had a bunch of pins, all dedicated to being either inputs or outputs of the various built-in OR gates (4 gates in one package = 8 inputs, 4 outputs, 2 power/gnd pins). 14 pins of fun. Now with a micro, the most basic pin function is GPIO - general purpose input/output. These GPIO pins can be configured as an input or an output. Each input pin can be monitored and acted upon. Example:

```
if (PORTC.2 == 1)  
  
then do something...
```

Each output pin can be pushed high or low. Example:

```
while(1)  
{  
RB3 = 1;  
delay_ms(1000);  
RB3 = 0;  
delay_ms(1000);  
}
```

Guess what that code does? It toggles a pin high/low every 2 seconds. Fancy right? This is the 'Hello World' of the microcontroller world and we see that LED blinking, it's just glorious!

## **Types of microcontrollers :-**

**PIC** - This is the classic micro from Microchip. Very simple, very proven, but it lacks many of the features that other mfg's are building into their chips.

**AVR** - This is basically a direct competitor of PICs. They do everything a PIC does, but in are better, faster, cheaper, and simpler.

**MSP** - These are very good micros by Texas Instruments (TI), not as beefy as AVR or PICs. However they truly excel at low-power applications. More on this later, but imagine running a complete system on one AA battery for 5 years. This is in the realm of nano-amp current consumption. Crazy!

**ARM** - ARMs are the new kids on the block and they are huge. Very powerful, very low-cost, they are taking over the world but can be really intimidating if never played with a micro before.

**8051** - The '8051 core' was the de facto standard in 8-bit (and 4-bit!) microcontrollers. Developed by Intel in the 1980s, it still seems to be the instruction set they love to teach in college. They are based on archaic, but field proven instruction sets. Very old tech, but these ICs have been significantly improved over the years (now Flash based, ADC, SPI, etc.).

**68HC08/11** - Another very common instruction set developed by Motorola. Extremely popular, and a micro commonly taught at university. These original micros often lack on-board RAM and flash based memory.

***We generally use the ATmega16 as the learning IC of choice. Why?***

- 20 MIPS (million instructions per second!) is powerful enough to do some really cool projects
- It's cheap!
- It's got all the goodies under the hood (UART, SPI, I2C, ADC, internal osc, PWM, kitchen sink, etc)
- 16K of program memory is enough for almost any beginner project
- The tools are free! (C compilers for many of the other micros cost a lot of money)
- The programming and debugging tools are low cost