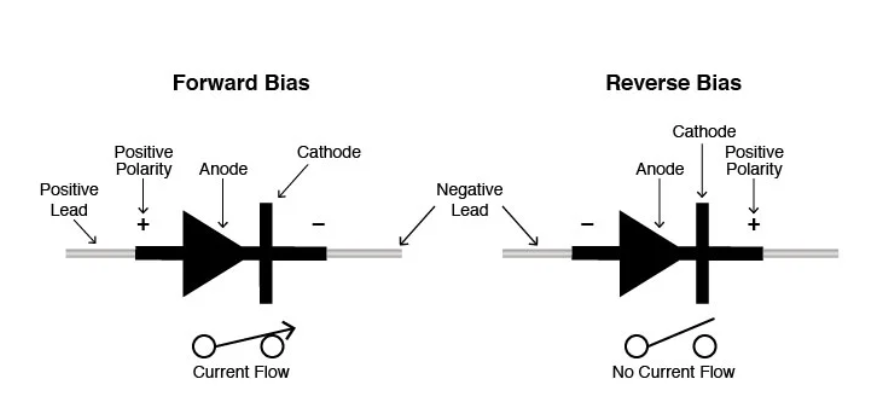
Manual on understanding basic concepts of electronics and bit manipulations on a register

Diode:

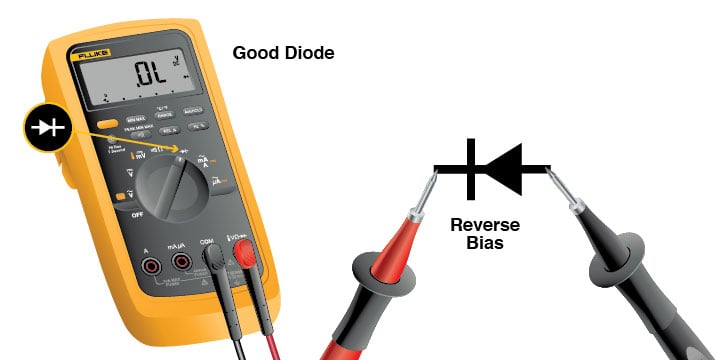
A **diode** is a semiconductor device that essentially acts as a one-way switch for current. It allows current to flow easily in one direction, but severely restricts current from flowing in the opposite direction.

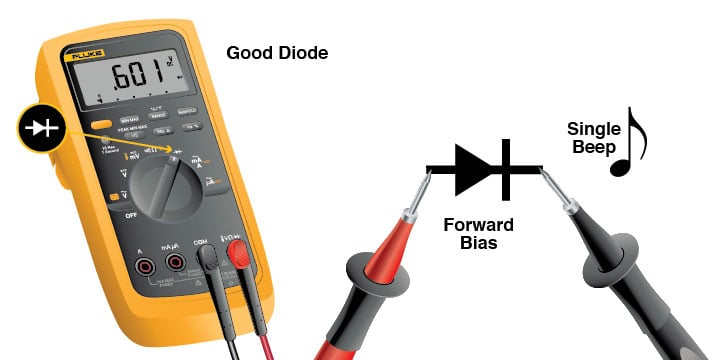
Diodes have polarity, determined by an **anode** (positive lead) and **cathode** (negative lead). Most diodes allow current to flow only when positive voltage is applied to the anode

When a diode allows current flow, it is **forward-biased**. When a diode is **reverse-biased**, it acts as an insulator and does not permit current to flow.

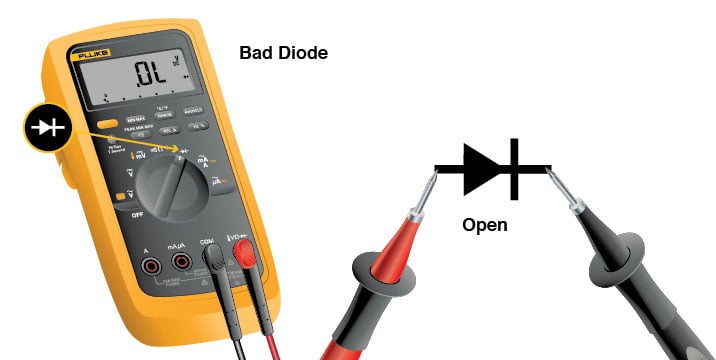


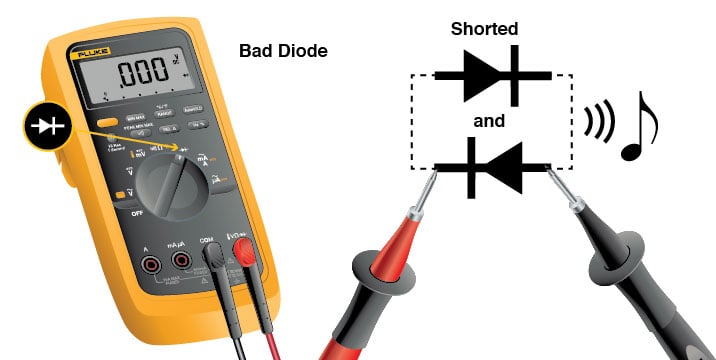
A digital multimeter's diode test diode produces a small voltage between the test leads enough to forward-bias a diode junction. Normal voltage drop is 0.5 V to 0.8 V. The forward-biased resistance of a good diode should range from 1000 ohms to 10 ohms. When reverse-biased, a digital multimeter's display will read OL (which indicates very high resistance).





Diodes are assigned current ratings. If the rating is exceeded and the diode fails, it may short and either a) allow current to flow in both directions or b) halt current from flowing in either direction.





Note: LEDs work jut like the diodes for testing follow the same procedure above given.

Biasing:

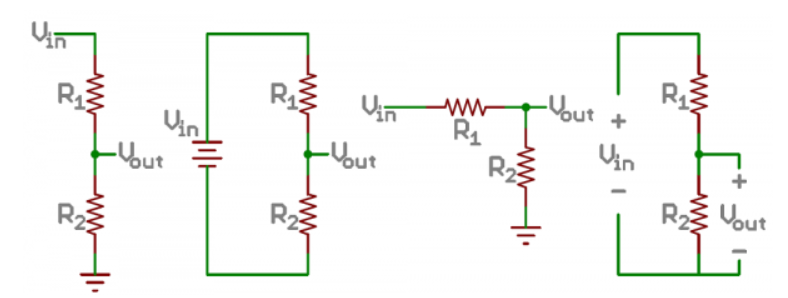
Biasing is the process of intentionally applying direct current (DC) voltage or current to a device to control a circuit. It can be used in diodes or electrical devices to allow a larger flow of current in a specific direction.

Voltage divider:

A **voltage divider** is a simple circuit which turns a large voltage into a smaller one. Using just two series resistors and an input voltage, we can create an output voltage that is a fraction of the input. Voltage dividers are one of the most fundamental circuits in electronics. If learning Ohm's law was like being introduced to the ABC's, learning about voltage dividers would be like learning how to spell *cat*.

These are examples of potentiometers - variable resistors which can be used to create an adjustable voltage divider. We'll learn more about these soon.

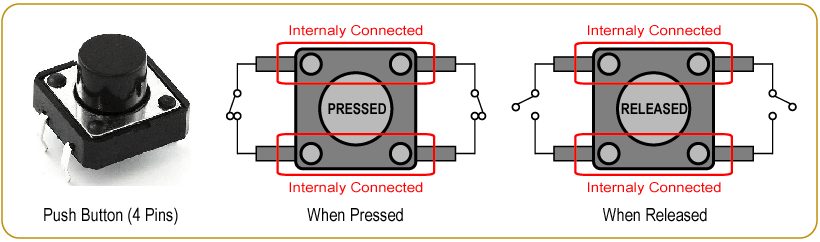
Ideal voltage divider: There are two important parts to the voltage divider: the circuit and the equation.



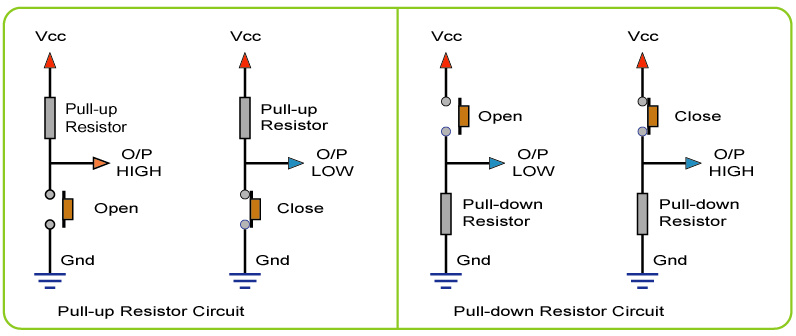
Push button:

A push button switch is a mechanical device used to control an electrical circuit in which the operator manually presses a button to actuate an internal switching mechanism. They come in a variety of shapes, sizes, and configurations, depending on the design requirements.

A typical push button switch has two active terminals that may be normally open or closed and changes its state when pressed or depressed. Sometimes we get 4 pins with push buttons having a paired (internally connected) terminals of each pin which is illustrated in the following figure. There are also some push buttons are normally closed but open when is pressed. Those are used for some special purposes.



If we connect the push button switch directly to a Microcontroller to get digital input, It means switch one pin is connected to Ground or 5v Vcc and another pin connected to 2 Microcontroller digital pin. In this case, the Microcontroller is read unstable input from the push button.  
  
So, we need to connect a “pull-up” or “pull-down” resistors circuit to stabilizes the input, when using the switch.



**Pull-up Resistors :** If the push button one pin is connected to the Vcc through a resistor and the other pin is connected to the ground, this circuit known as the pull-up resistor circuit. In this case, the push button output is High(1) when the button is open, and the output of the push button is Low(0) when the button is pressed.

**Pull-down Resistors :** If the push button one pin is connected to the ground through a resistor and the other pin is connected to the Vcc, this circuit known as the pull-down resistor circuit. In this case, the push button output is Low(0) when the button is open, and the output of the push button is High(1) when the button is pressed.

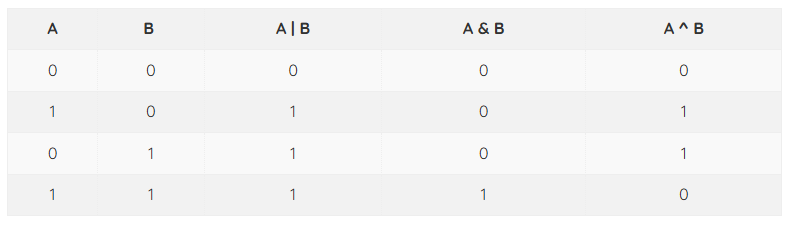
Bit-manipulation:

Manipulating individual bits is one of the most important and fundamental concepts to understand when programming microcontrollers. Bit manipulation is necessary to read the status of components, set parameters, and change the state of output pins. It is important to know how to individually change the states of if certain bits while leaving others unchanged. Here we will take a look at some of the bitwise operators that are available to us in C and show how they are used to control an LEDs so on .

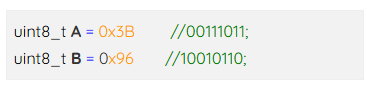
The bitwise or bit-level operator lay’s foundation for bitwise operations in embedded programming. We hope you already have sound knowledge of binary number systems in order to follow this tutorial. Many newbies found themselves confuse with **Bitwise Vs. Boolean** operations. A **bitwise expression** is used when we want to modify a variable by changing some or any of the individual bits of the variable.**&** and **|** are bitwise operators for **AND** & **OR**respectively. We’ll see in later part of this post, how it will be used to set, clear and extract a bit using bitwise operations. A **Boolean expression** is used when we want to know something about a variable is it equal to 12, is it greater than 12 or is it less than 5 etc. Boolean operators are: **&&** (logical and), **||** (logical or), **!** (logical not).  **X** && **Y** returns TRUE only if both **X** and **Y** are TRUE. **X** || **Y** returns TRUE if either **X** or **Y** are TRUE.

Basic bitwise operations:

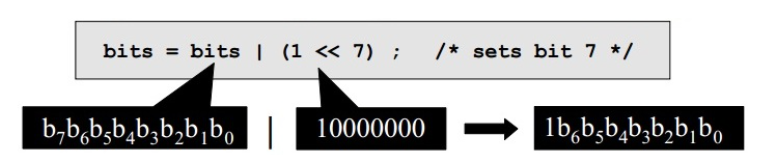
Now let’s concentrate only on bitwise operations. We’ll learn how these bitwise operations allows us for Setting, Inverting, Toggling, Clearing, Extracting and Inserting bits in embedded programming. Here is a table which summarizes operations with 2-operands.



Let’s take variable **A** and **B**. We’ll perform bitwise operation on these two variables. This example will help you understand their operation. Let’s first declare these two variables:



Let’s say we have variable called **bits** and we have asked to set the bit-7. This can be achieved by writing this single line of code.



By assuming the 32-bit register maximum value it can have 0xFFFF FFFF the equivalent binary value is all 1s in the given register.

Setting bit using bitwise operator on a register:

\*REG = \*REG | (1<<BIT\_POSITION);

Or

\*REG |= (1 << BIT\_POSITION);

Clearing bit using bitwise operator on a register:

\*REG = \*REG & ~(1<<BIT\_POSITION);

Or

\*REG &= (1<<BIT\_POSITION);

Toggling bit using bitwise operator on a register:

\*REG = \*REG ^ (1<<BIT\_POSITION);

Or

\*REG ^= (1<<BIT\_POSITION);