Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

Skill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Basics of PLC

Addressing for Ladder Logic Programming

Bit-Byte Word Concepts

What is a bit?

Bit is short for 'binary digit.' It's a single digit in a binary number, and it can be either 1 or 0.

Inside a computer, you can think of a bit as being a mechanical switch, which can be either switched on or off (the earliest computers actually stored information in memory using mechanical switches, with electromagnets to turn each one on or off).

Now if you only have one of these switches, you can only store two different states, on or off. This is useful in itself, you can record that something is either true or false.

But if you have, say, eight of them, you can store 256 different combinations of on and off states between the eight switches.

What is a byte?

A byte is 8 bits. That's the definition. With 8 bits you can store any number between 0 and 255, since there are 256 different combinations of 1 and 0 to choose from.

Why eight bits? The original intention was that, when storing text, 8 bits would be enough to assign a unique number every possible language character you might want to use in your document. The idea was that each character in a file would take up one byte of memory (in most cases, this is still true).

Let's see: there are 26 uppercase letters (A-Z), 26 lowercase (a-z), 10 numerical digits (0-9), 32 punctuation characters and other symbols on a US keyboard, the space character that’s already 94 different characters. Then there's a few characters for creating newlines, a tab character for indentations, there's even a 'bell' character which programs would output in order to make the user's terminal beep. You can see how it all adds up

In practice, only characters up to 127 were ever standardized (the standard is called ASCII, which stands for American Standard Code for Information Interchange, because in the early days, one of the eight bits was set aside for error testing purposes (back when computers were far less reliable), and 7 bits only gives you 128 different combinations.

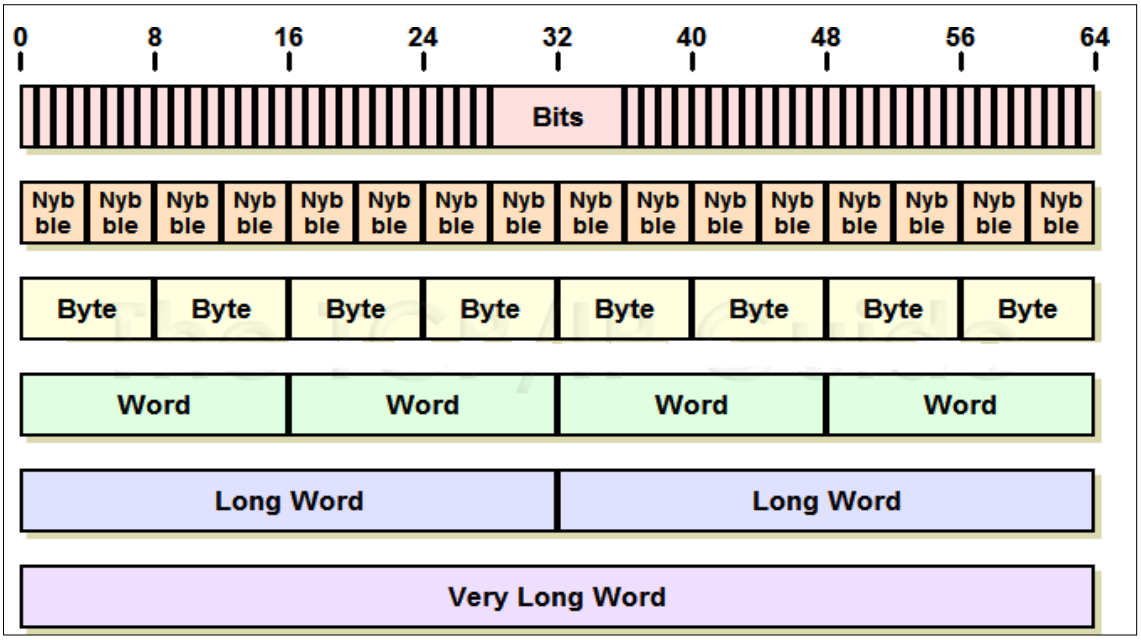
What is a word?

You often hear about 32-bit or 64-bit computer architectures. A word is basically the

number of bits a particular computer's CPU can deal with in one go. It varies depending on the computer architecture you're using.

Imagine looking at an imaginary computer's circuitry very closely. On a 32-bit machine, you would see 32 wires running parallel to each other between the computer's memory controller and the CPU, for the purpose of giving the CPU access to one particular word of memory.

Actually, there would be an additional 32 wires (perhaps less) for the CPU to select a particular memory address to access. If a CPU can access 32 bits of memory in one go, then it turns out that it makes a lot of sense to address the computer's memory using ≤32 bits. (This happens to be why the 32-bit version of Windows can't deal with more than 2GB of RAM, but the 64-bit version can.)



Different Logic Gates Circuit Diagrams

In electronics, a **logic gate** is an idealized or physical device implementing a Boolean function; that is, it performs a logical operation on one or more logical inputs, and produces a single logical output. Depending on the context, the term may refer to an **ideal logic gate**, one that has for instance zero rise time and unlimited fan-out, or it may refer to a non-ideal physical device[1] (see Ideal and real op-amps for comparison).

Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using vacuum tubes, electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements. With amplification, logic gates can be cascaded in the same way that Boolean functions

can be composed, allowing the construction of a physical model of all of Boolean logic, and therefore, all of the algorithms and mathematics that can be described with Boolean logic.

Logic circuits include such devices as multiplexers, registers, arithmetic logic units (ALUs), and computer memory, all the way up through complete microprocessors, which may contain more than 100 million gates. In modern practice, most gates are made from field-effect transistors (FETs), particularly MOSFETs (metal–oxide–semiconductor field-effect transistors).

Compound logic gates AND-OR-Invert (AOI) and OR-AND-Invert (OAI) are often employed in circuit design because their construction using MOSFETs is simpler and more efficient than the sum of the individual gates.

In reversible logic, Toffoli gates are used.

A truth table shows how a logic circuit's output responds to various combinations of the inputs, using logic 1 for true and logic 0 for false. All permutations of the inputs are listed on the left, and the output of the circuit is listed on the right. The desired output can be achieved by a combination of logic gates. A truth table for two inputs is shown, but it can be extended to any number of inputs. The input columns are usually constructed in the order of binary counting with a number of bits equal to the number of inputs.

Truth tables are an important tool for evaluating statements and arguments. We can create our own truth tables using following steps:

1. Translate statements of ordinary language.
2. Break all complex statements into smaller parts.
3. Determine how many columns are required.
4. Determine how many rows are required.
5. Determine the truth values of statement letters.
6. Determine the truth values of complex statements.

A law of Boolean algebra is an identity such as $x \vee (y \vee z) = (x \vee y) \vee z$ between two Boolean terms, where a Boolean term is defined as an expression built up from variables and the constants 0 and 1 using the operations $\wedge$, $\vee$, and $\neg$. The concept can be extended to terms involving other Boolean operations such as $\oplus$, $\rightarrow$, and $\equiv$, but such extensions are unnecessary for the purposes
to which the laws are put. Such purposes include the definition of a Boolean algebra as any model of the Boolean laws, and as a means for deriving new laws from old as in the derivation of $x \vee (y \wedge z) = x \vee (z \wedge y)$ from $y \wedge z = z \wedge y$ as treated in the section on axiomatization.

Monotone laws

Boolean algebra satisfies many of the same laws as ordinary algebra when one matches up ∨ with addition and ∧ with multiplication. In particular the following laws are common to both kinds of algebra.

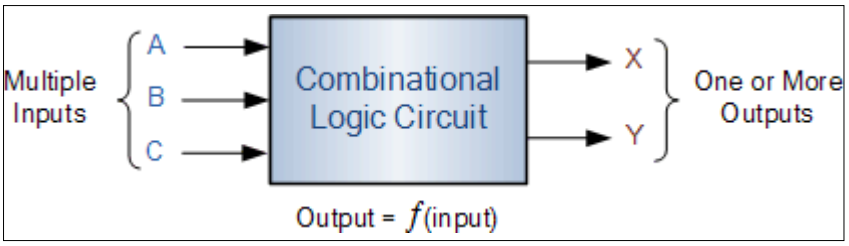
Combination Logic Circuits

Unlike Sequential Logic Circuits whose outputs are dependent on both their present inputs and their previous output state giving them some form of Memory, the outputs of **Combinational Logic Circuits** are only determined by the logical function of their current input state, logic "0" or logic "1", at any given instant in time.

The result is that combinational logic circuits have no feedback, and any changes to the signals being applied to their inputs will immediately have an effect at the output. In other words, in a **Combinational Logic Circuit**, the output is dependent at all times on the combination of its inputs. So if one of its inputs condition changes state, from 0-1 or 1-0, so too will the resulting output as by default combinational logic circuits have "no memory", "timing" or "feedback loops" within their design.

Combinational Logic

Combinational Logic Circuits are made up from basic logic **NAND**, **NOR** or **NOT** gates that are "combined" or connected together to produce more complicated switching circuits. These logic gates are the building blocks of Combinational Logic Circuits. An example of a combinational circuit is a decoder, which converts the binary code data present at its input into a number of different output lines, one at a time producing an equivalent decimal code at its output.



Combinational logic circuits can be very simple or very complicated and any combinational circuit can be implemented with only NAND and NOR gates as these are classed as "universal" gates.

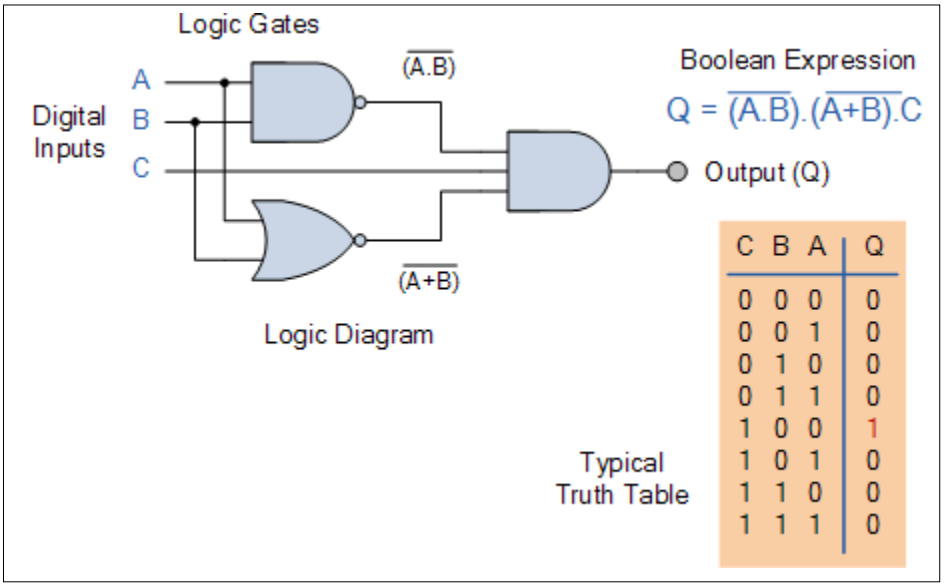
The three main ways of specifying the function of a combinational logic circuit are:

1. **Boolean algebra** – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False

that results in a logic "1" output.

2. **Truth Table** – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.
3. **Logic Diagram** – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol that implements the logic circuit.

And all three of these logic circuit representations are shown below.



As combinational logic circuits are made up from individual logic gates only, they can also be considered as "decision making circuits" and combinational logic is about combining logic gates together to process two or more signals in order to produce at least one output signal according to the logical function of each logic gate. Common combinational circuits made up from individual logic gates that carry out a desired application include **Multiplexers**, **De- multiplexers**, **Encoders**,**Decoders**, **Full** and **Half Adders** etc.