Programmable Logic Controllers Fundamentals

Course No: E01-008
Credit: 1 PDH

Tracy Adams, P.E.

Continuing Education and Development, Inc.

P: (877) 322-5800
info@cedengineering.ca

Programmable Logical Controllers Fundamentals

Programmable Logic Controllers (PLC) impacted automation the way the Internal Combustion Engine impacted horse drawn wagons. A programmable logic controller is an electronic system that receives input signals, analyzes them according to a user-generated program commonly known as a LADDER diagram. and produces output signals for process control.

Prior to PLC's controls were done via hardwired logic. This consisted of electrical or pneumatic relays, drum sequencers and cam timers. If a manufacturer needed to change a production line, it meant taking the line down to change or relocate controllers, rewire all the control signals, and add new instruments or relays if new functions were required. This was a very labor intensive process.

When digital computers were developed, they were added to the manufacturing process but they required special environments and very costly programmers. In 1968 GM put out a request for proposal for the electronic replacement of hardwired relay control systems. The winner was Bedford and Associates and the first PLC was the result. Bedford and Associates started a new company, MODICON, to manufacture PLCs. MODICON stands for MOdular DIgital CONtrollers. MODICON is still manufacturing PLC's today.

A PLC is a programmable digital system such as a personal computer or PC. Of course, there are differences but there are many similarities.

- Each PLC contains a central processing unit or CPU, which is a single chip such as a Pentium or an 80486 etc.
- Each PLC contains a pre-written program for determining the systems behavior known as a BIOS in the PC and as FIRMWARE in the PLC.
- Each PLC contains a certain amount of RAM or random access memory for the storage of data and program instructions.
- Each PLC contains some sort of intelligent I/O hardware, which supports the reading and writing of data.
- Many PLC's contain a Boolean co-processor to perform logic functions whereas the PC may have a math co-processor to perform mathematical calculations.
- Each PLC contains methods of communicating with the outside world such as specialized modules in keyboards, monitors, serial ports and mice in the PC.
- Each has a bus that allows the addition of devices with special behavior. These devices

must adhere to some pre-determined scheme in order to be known to the system. An example in the PLC would be the addition of a digital input module in the back plane or rack. In the PC perhaps would be the addition of a modem in an ISA slot for connection to the Internet etc. Some PLC's even support the Plug and Play concept for many of their I/O modules.

- Each PLC is capable of executing or running user programs or algorithms that manipulate the system hardware and or behavior in some manner.

A CPU can be thought of as a software machine. It has no moving parts; however, it can perform the tasks it was designed for. A language was created along with the CPU that allows a user to manipulate it. This is the lowest level of programming and is known as MACHINE CODE programming or ASSEMBLY LANGUAGE.

In order for a CPU to function, it needs other specialized chips around it to do certain tasks. These chips all make up a system that requires a set of procedures to function. A BIOS or Basic Input Output System is such a set. A user needs an easy way to manipulate this I/O system. DOS and Windows are examples of an OPERATING SYSTEM with a user interface. DOS uses the command line (C:\) with built-in functions such as "DIR" and Windows uses the GUI or Graphical User Interface.

A PLC already knows to update I/O values, communicate with others of its kind, and execute the user's ladder diagram. Its manufacturer created an I/O operating system or FIRMWARE to make it do so.

The user creates a ladder that directs the PLC on what to do with the data and I/O values and when to do it. The manufacturer of the PLC has created several tools to do this such as the programming software that runs on a computer and possibly a hand held programmer that plugs directly into the PLC.

The ladder appears as an electrical control schematic with typical parallel power rails, one hot and one return with interconnecting links called RUNGS. These rungs contain items such as COILS, CONTACTS, TIMERS and COUNTERS and use a concept called POWER FLOW.

This simplified view may cause a false impression of what is actually going on inside the PLC and what the ladder really is.

Let's look into the PLC's CPU or, more to the point, any DIGITAL or BINARY device. Only two logical states or BIT VALUES exist. These are ((ON) (1) (TRUE) (HIGH) (+XVDC)) and ((OFF) (0) (FALSE) (LOW) (0VDC)). Everything is represented using 1's & 0's because that's all the CPU can use.

In programming binary systems, the idea of Boolean logic is applied. Two bits are compared with a resulting single bit result.

The Boolean functions:

- AND — The resulting bit is a 1 only if BOTH compared bits are 1.

  1 AND 1 = 1
  1 AND 0 = 0
  0 AND 0 = 0

- OR — The resulting bit is a 1 if ANY compared bit is 1
  1 OR 1 = 1
  1 OR 0 = 1
  0 OR 0 = 0

- EXCLUSIVE OR — The resulting bit is a 1 if ONE OR the other compared bit is 1, BUT NOT BOTH
  1 XOR 1 = 0
  1 XOR 0 = 1
  0 XOR 0 = 0

- NOT — Acts on a single bit as follows
  NOT 1 = 0
  NOT 0 = 1

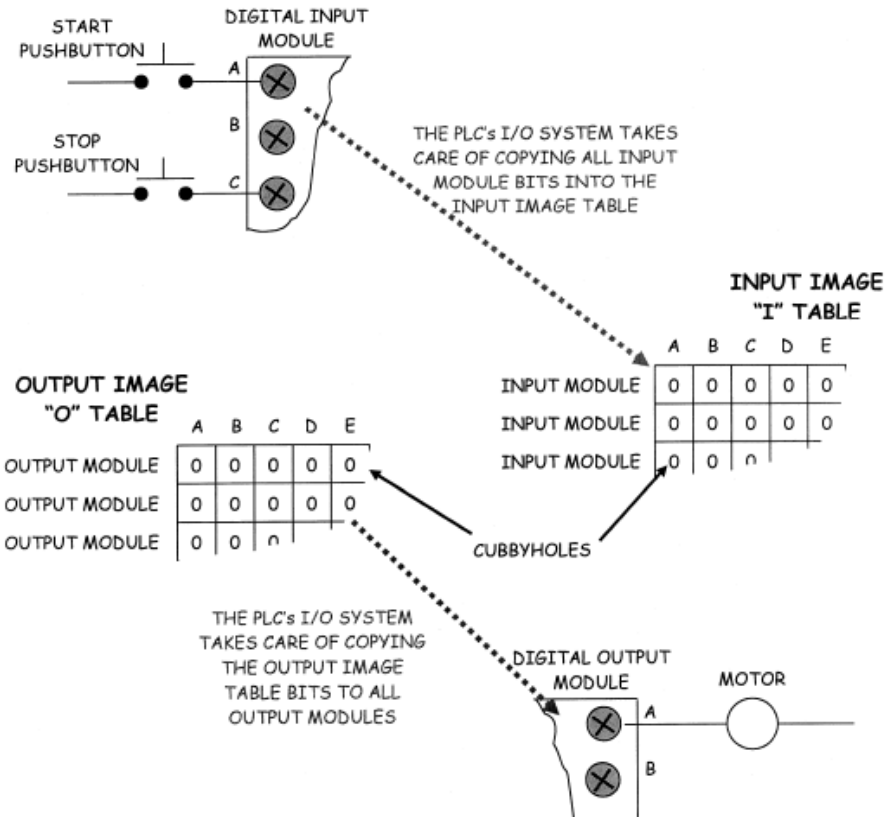
Think about a control systems operating ALGORITHM:

> *IF the motor start pushbutton is pressed OR the motor is running AND the motor stop pushbutton is NOT pressed THEN start the motor ELSE stop the motor. (We'll analyze this later).*

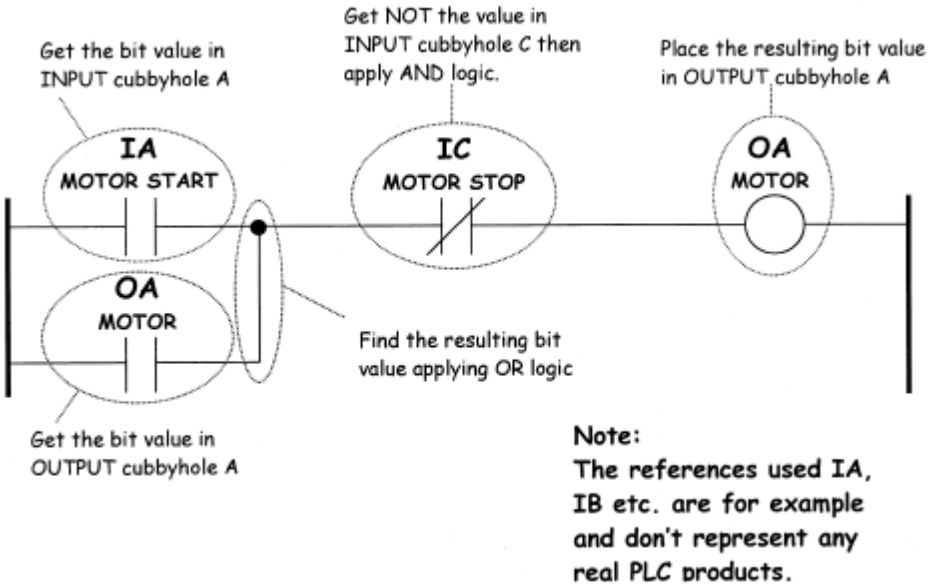
These Boolean functions are so basic and important that individual chips can be obtained that contain nothing but AND, OR, XOR and NOT gates. These were some of the first IC's available. CPU's and other complex logic circuits are made up of these basic tools.

A PLC deals with a switch wired to its input card or a relay wired to an output card as a bit.

These bit values are stored in an IMAGE TABLE. There is an INPUT and an OUTPUT image table as well as others. Think of each location in these tables as a CUBBYHOLE. If the input switch is closed, the input cubbyhole contains a 1 if open a 0. If the output cubbyhole contains a 1 then the output relay is on; if 0 it's off.



Here's an example ALGORITHM. Referencing the previous diagram, IF the motor start pushbutton is pressed OR the motor is running AND the motor stop pushbutton is NOT pressed THEN start the motor ELSE stop the motor.



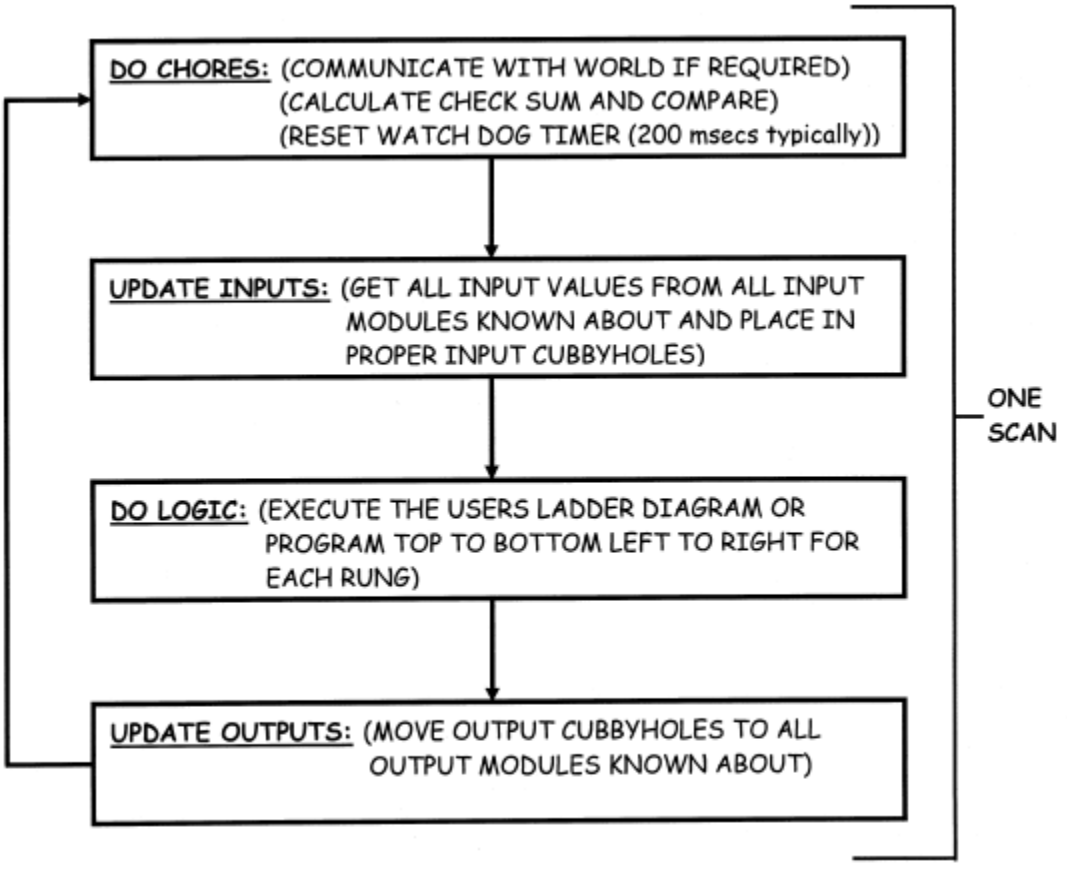
Above is a display you might see using programming software for a PLC. If you were to view the program using a hand held device you might see the following:

STR IA	store input image table location A
OR OA	OR with output image table location A
AND NOT IC	AND with NOT input image table location C
OUTOA	place result in output image table location A
END	end of listing

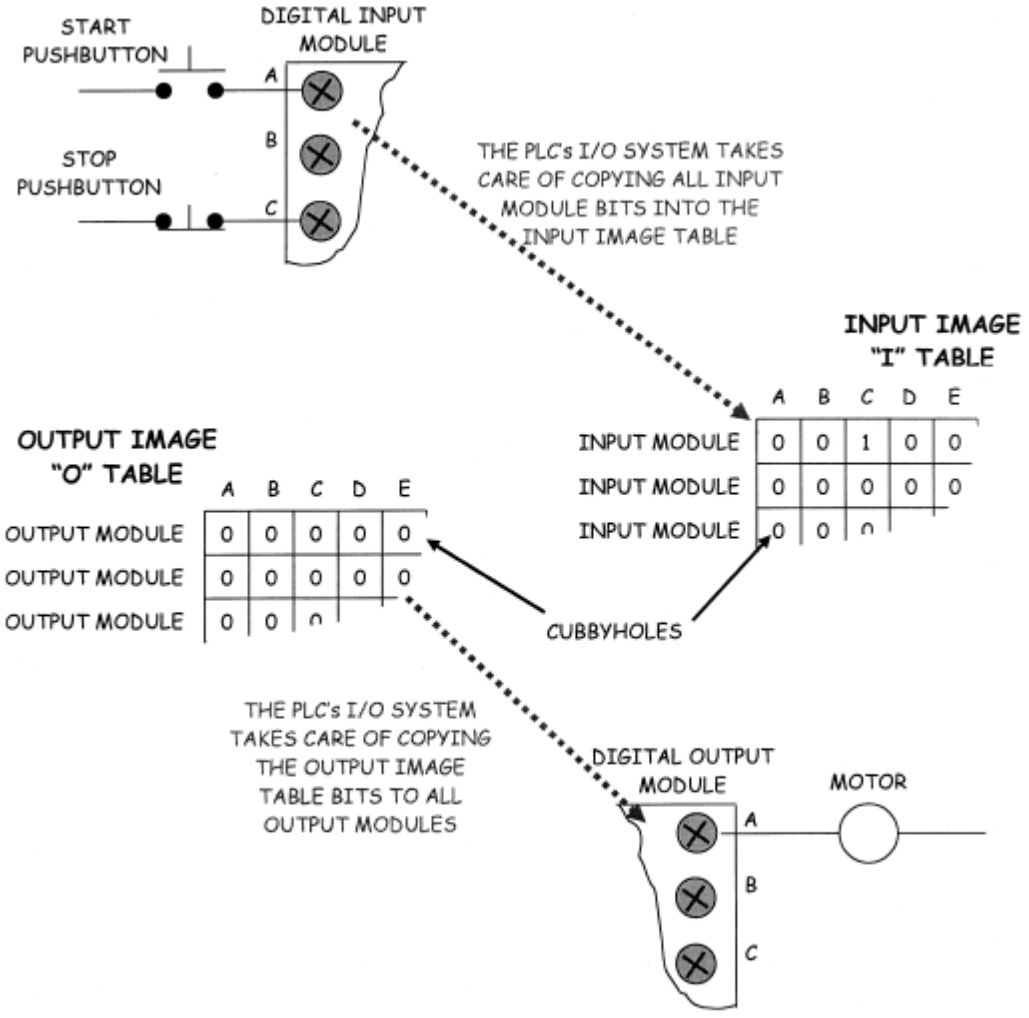
A program will probably contain more than one rung. The CPU SCANS EACH RUNG SEPARATELY TOP TO BOTTOM, in other words it does any OR logic. Then LEFT to RIGHT doing any AND logic.

Remember that the CPU has firmware that controls it. It performs a preset sequence of events over and over. You can depend on this sequence and become familiar with it.

Here is a typical SCAN sequence:

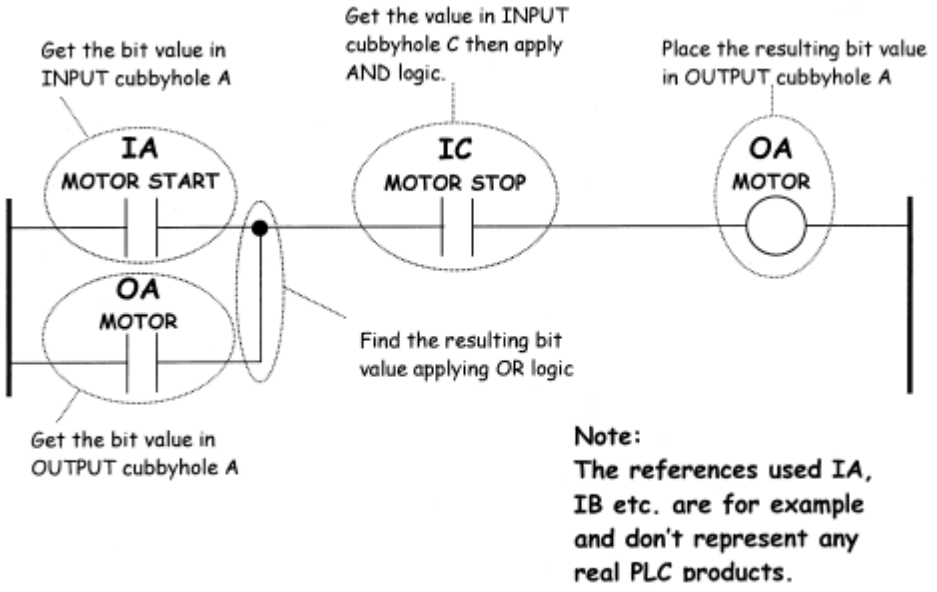


Now let's change the motor stop pushbutton from normally open "NO" to normally closed "NC" (which by the way is the proper application for safety) and see how the logic changes:



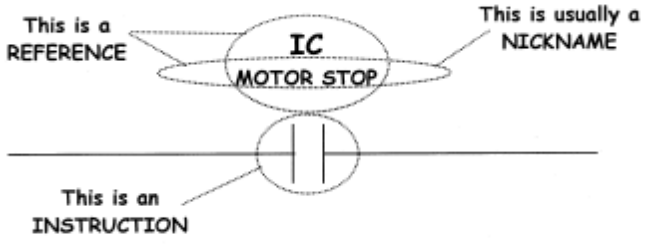
Now the algorithm changes to:

IF the motor start pushbutton is pressed OR the motor is running AND the motor stop pushbutton is NOT pressed THEN start the motor ELSE stop the motor.



Notice that the type of instruction used in reference to MOTOR STOP "IC" depends on the way you want the logic interpreted and the type of switch (NO, NC) actually used.

What we have been using within these logic (LADDER) diagrams is known as INSTRUCTIONS and REFERENCES:



NICKNAMES which are supported in most but not all brands of PLC's are user created ID's for a cubbyhole. When the program is being created and a reference to IC is needed you could instead type MOTOR STOP.

The INSTRUCTION is what action the PLC is to perform. Don't forget that "LINES" known usually as SHUNTS (VERTICAL and HORIZONTAL) are instructions also:

Bits are all that a CPU can manipulate. But there must be some way of representing things other than discrete items. What if the PLC must know how many of something to make? Bits are grouped into WORDS to represent VALUES. These words are usually 16 bits in length.

The BINARY (TWO SYMBOLS) numbering system is used to put values into bit patterns for words. Remember our only symbols are 0 & 1. Another way of referring to this is BASE2.

We use a system known as DECIMAL (ten symbols) every day. Our symbols are 0123456789. This is a BASE10 system. If you are counting and get to a value of 9 you are out of symbols and start over with 10. We call this TEN but actually it's ONE ZERO base10. We would say TWENTY-FIVE for 25 but again this is really TWO FIVE base10. This probably seems a little simplistic but it's the basis for what follows.

If you were counting items, before any count you have [0 base10] or [0 base2]. After the first count you have [1 base10] and [1 base2]. After the second count you have [2 base10] and [10 base2]. After ten counts you have [10 base10] and [1010 base2]. So knowing the base is important when a looking at a number.

What does [10 base10] mean? The 1 indicates that all of the symbols have been used once, and the 0 indicates that no additional symbols were used. And [99 base10] means that all of the symbols were used 9 times and 9 additional symbols have been used. If an additional count is added then [100 base10] is the new number. Again this means that all ten symbols have been used ten times or ONE TIMES ONE HUNDRED PLUS ZERO TIMES TEN PLUS ZERO TIMES ONE = ONE HUNDRED.

Applying this principle to base2 you will see that [10 base2] means that all of the symbols were used once and no additional symbols were used. So if you want to convert this to our base10 system you would say ONE TIMES TWO PLUS ZERO TIMES ONE = TWO or [2 base10]. Also [100 base2] means that all of the symbols were used twice with no additional symbols. So ONE TIMES FOUR PLUS ZERO TIMES TWO PLUS ZERO TIMES ONE = FOUR [4 base10].

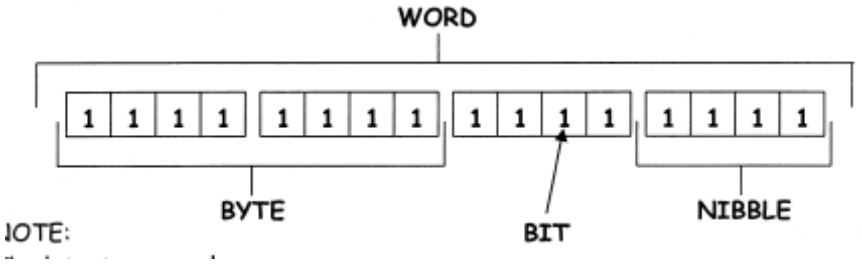
Each place to the left holds a value equal to the previous position times the base of the system:

Base10 (decimal):	10000	1000	100	10	1
base2 (binary):	16	8	4	2	1
base8 (octal):	4096	512	64	8	1

Notice that the value accumulates from right to left and not from left to right. We say that the least significant symbol or bit (LSB) is on the right and the most significant (MSB) is on the left. Notice also that the number of places for symbols determines the maximum count that can be manipulated and or displayed.

There is another form of bit pattern that is used for value input and display that needs to be understood. This system is known as BINARY CODED DECIMAL or BCD. Each a word

contains 16 bits. A BYTE contains 8 bits and is half a word, and a NIBBLE contains 4 bits and is one quarter of a word.



A word is made up of 4 nibbles. If you analyze any nibble (4 bits) you will see that the maximum value that it can represent is 15 base10. That is:

(1 X 8) + (1 X 4) + (1 X 2) + 1 = 15. (1111 base2)

Before Human Machine Interfaces (HMI) were around, BCD thumbwheels and displays were commonly used for value input and display in control systems.

A single thumbwheel has one digit that can display 0 to 9 and controls 4 wires in a binary format. The wires will indicate from 0000 base2 to 1001 base2. Four thumbwheels side by side indicate from 0000 to 9999 controlling 16 wires as four separate groups. So 9876 displayed is sent as 1001 1000 1011 0110.

The thumbwheels were usually wired to a 16 point digital input card and the 16 input cubbyholes are read as a word. But note:

9999 BCD = 1001 1001 1001 1001 base2

And

9999 base10 = 0010 0111 0000 1111 base2

Math in a PLC is performed using the base10 values of words. Most PLC's include a function to convert BCD to base10 or vice versa. This shows the importance of keeping track of where data comes from and what type it is.

One final format extends the idea of BCD to include all possible 4-bit values 0 to 15. Words are not always used as numeric values in programming. Many times the bit pattern in a word is what is being used. It would be difficult to enter a 16 bit pattern if you always had to convert it into a base10 value. So the HEXADECIMAL numbering system is used. In this format 0 to 9 base10 [0000 to 1001 base2] are as in BCD; however, 10 to 15 [1010 to 1111 base2] are indicated using the letters A through E. In other words the bit pattern 1111 1100 1010 0011 base2 would be FCA3 HEX. This format only requires using these 16 symbols

and bit patterns. Thumbwheels and displays are available that use the HEX format.

Here is an example of using a bit pattern in programming. There are two BCD thumbwheels. The thumbwheel can send binary values from 00 to 99 using eight wires connected to the first eight points on a PLC's 16 point input module.

To get the BCD value into a usable format, the following algorithm might be used:

Thumbwheel indicates 77 (0111 0111)

Move the 16 input bits into WORD1: 0000 0000 0111 0111
Convert WORD1 BCD to base2: 0000 0000 0100 1101
Use WORD1 as gallons to pump: 77 base10

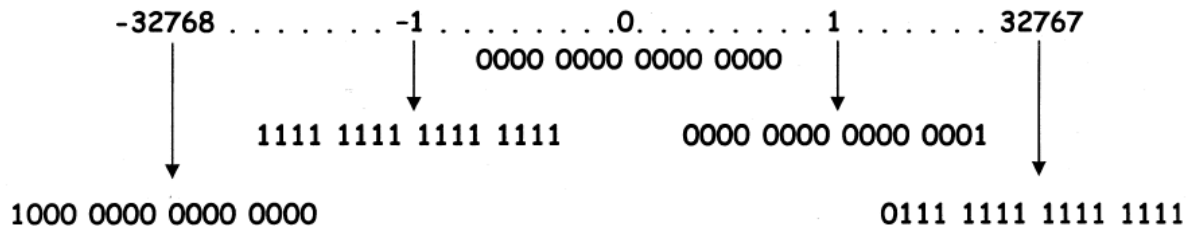
Up to now "word" has been used to represent a group of bits taken together. The proper term is REGISTER. As in a word a single register is assumed to be 16 bits wide. Registers are contained in a table just like bit data with the difference being that the each uses 16 cubbyholes.

REGISTER TABLE

REGISTER 1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
REGISTER 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
REGISTER 3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
REGISTER….	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Most PLC's perform math using INTEGERS or WHOLE numbers such as 12, -3, 2124 etc. There are no fractional or decimal parts and the numbers can be SIGNED negative or positive. We already know that a register contains sixteen bits and if all bits were 1 the base10 value would be 65535. The PLC uses the most significant bit (bit 16) to indicate the sign. If the bit value is 1 the number is considered to be negative. Because only fifteen bits are left to indicate value, the range of 16 bit integers is limited to +32767 to -32768. Usually two registers can be used together then the range is +2147483647 to -2147483648. This is called DOUBLE INTEGER.

Negative numbers are stored in TWOS COMPLEMENT NOTATION. To find the twos complement of a bit pattern, you first NOT all of the bits then add 1. The twos complement of 0000 0000 0000 0001 is 1111 1111 1111 1111. This means that 1000 0000 0000 0000 base2 is actually -32768 base10, and 1111 1111 1111 1111 base2 is actually -1 base10. See the number line below for an example.



Real or floating point numbers commonly known as FLOATS use a different encoding scheme altogether. A float is a number that contains a fractional part such as 2.789623. Looking at 32 bits that were encoded to indicate a single precision float, you would only see 1's and 0's.

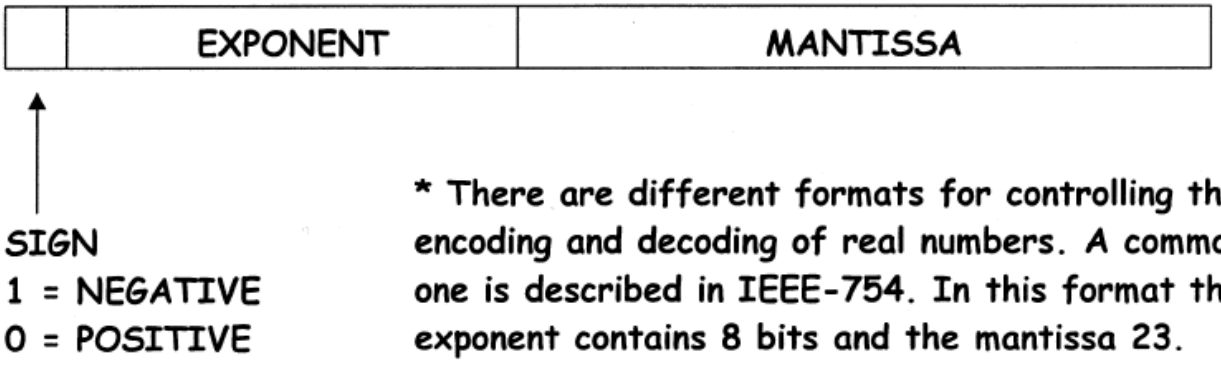
To show numbers in the base10 system we just extend our positional notation. 123.45 is the same thing as saying:

1 X 100 + 2 X 10 + 3 X 1 + 4 X 1/10 + 5 X 1/100

An example in the base2 system, 1010.10 would indicate:

1 X 8 + 0 X 4 + 1 X 2 + 0 X 1 + 1 X 1/2 + 0 X ¼

Real numbers are encoded in 32 bit SINGLE PRECISION as follows:



Because there are different formats, let's create an imaginary 9 bit system to see how one might work. Our system uses one sign bit, four exponent bits and 4 mantissa bits. The mantissa will always be normalized to contain one digit to the left of the decimal point. The exponent is 4 bit and the MSB indicates the sign.

Some examples follow:

0	0000	1100	= 1.100 base2 X 1 = 1 + 1/2 = 1.5 base10

0	0111	1101	= 1.101 base2 X 2^{7} = 11010000 base2 = 128 + 64 + 16 = 208 base10

0	1000	1010	= 1.01 base2 X 2^{-8} = .0000000101 base2 = 1/256 + 1/1024 = .0048828125 base10

Things to note are that the mantissa and exponent are contained in a fixed number of bits so the range of values as well as the precision is limited. Floating points come in 32 bit SINGLE PRECISION and 64 bit DOUBLE PRECISION. As with all other data types, the program creator is responsible for proper usage.

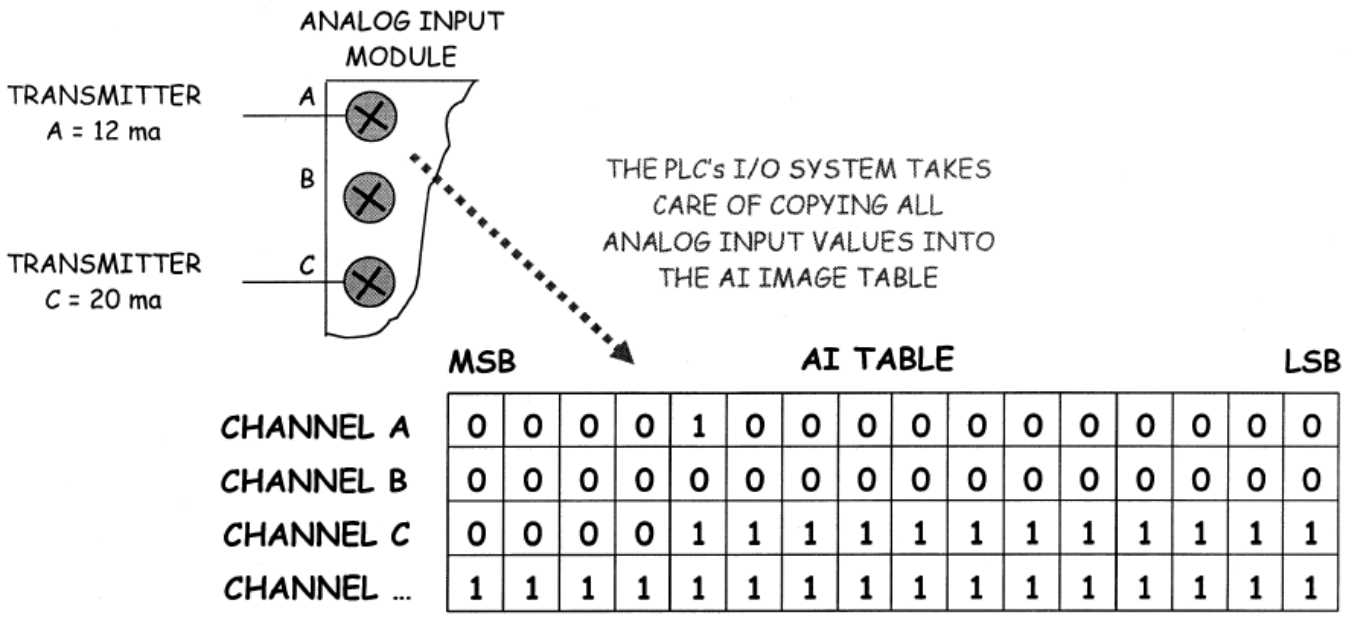
Data registers are used for analog values. A PLC can usually handle any of the standard industrial formats for instrumentation signals. These include 0 to 10 VDC, -10 to +10 VDC, 1 to 5 VDC, 4 to 20 ma and -20 to +20 ma. In order to handle these analog values specialized modules or cards are used.

An analog input module contains an ANALOG to DIGITAL CONVERTER(s) or ADC which converts lets says 4 to 20 ma into a bit pattern. An analog output card contains a DIGITAL to ANALOG CONVERTER(s) DAC which does exactly the opposite.

A standard data register is sixteen bits wide. Converter circuits are characterized by how many bits they can deal with. A typical converter would be 12 BIT. There are 14, 16 and others available but the following discussion is on 12 bit circuits. In base2 12 bits give a numeric range of 0 base 10 to 4095 base10. This is known as the RESOLUTION capability of the module.

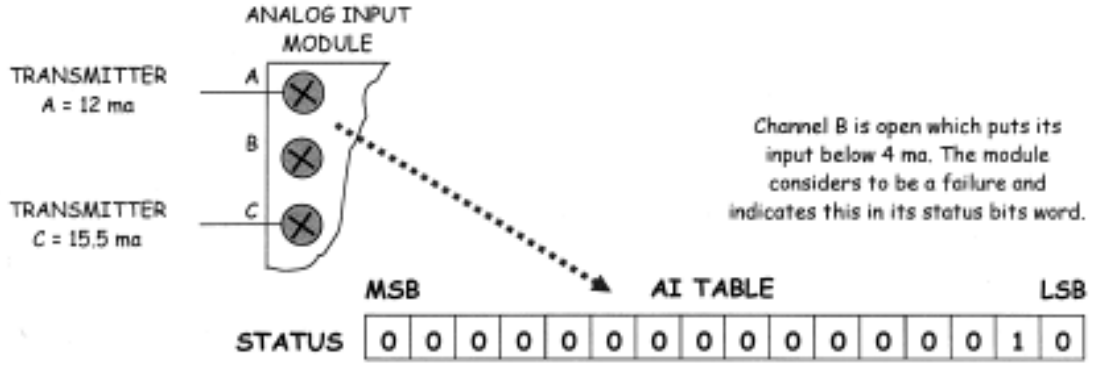
An analog input card is serviced by the CPU just like digital inputs in a manner that it has certain pre-configured cubbyholes where its values are copied. The difference being is that words are used instead of bits. There are two general formats that an analog card uses to provide these values to the CPU.

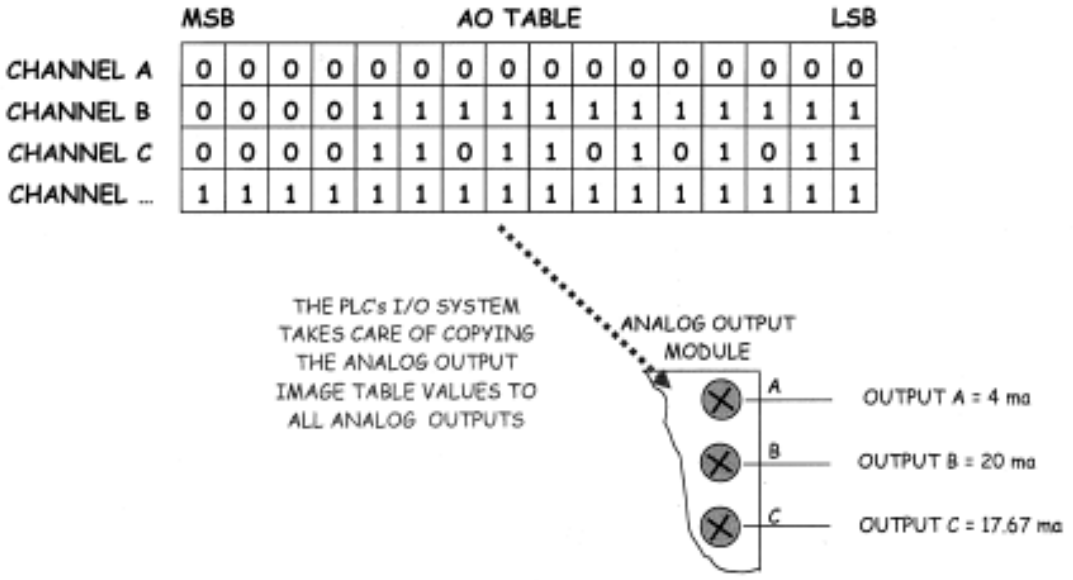
In the first mode, data is provided channel by channel and is copied cubbyhole word by cubbyhole word. The CPU stops by the analog card and gets all 8 words before moving on:



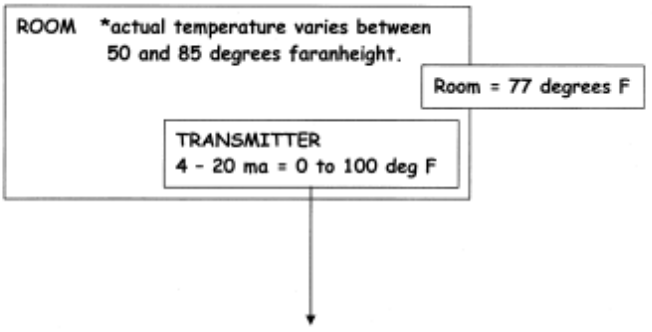
In the second mode, the CPU gets only two words on each scan. One word contains the converted value and the other contains the ID of the channel that was converted. This is known as MULTIPLEXING and means multiple things passed through a common point. This mode is used most commonly on cards that can handle different input signal types and different resolutions at the same time.

Most analog input cards provide at least one additional word that uses individual bits to indicate whether a particular channel should be considered good or VALID. This word is also copied into an image table by the CPU. In most cases a 1 bit indicates a bad channel.





For example, a temperature transmitter measures the temperature of a room and converts it into a 4 to 20 mA signal. What is the transmitter signal current level to the PLC?



Signal span = 20 - 4 mA = 16 mA
Temperature Span = 100 – 0 deg = 100 deg
Slope = 16 mA/100 deg = 0.16 mA per degree

77 deg x 0.16mA per deg = 12.32 mA

12.32mA + 4 mA = 16.32 mA

For an input card using a 12 bit resolution, what would the COUNT or value be in this input image table location that the PLC program will be using?

Count span = 4095 - 0 = 4095 (12 bit)
Signal span = 20 - 4 ma = 16 mA
Slope = 4095/16 ma = 255.9375 counts per mA

Count = 12.32 ma X 255.9375 counts per mA = 3153

PLC's give the means to use existing instruments and controls on different manufacturing and industrial processes by changing the program instead of tearing down and rebuilding conveyors, control panels, etc. They are made for use in industrial environments with high temperatures that a standard computer would not last in very long. They are simple to modify for adding new signals or different types of signals from initial construction. They are built around the concept of using very simple programming languages and mathematical functions that are easy to learn.

PLC's changed the world of automation, and they continue to evolve improving processor speed, adding new functionality, and providing the ability to use the latest methods of communication.