Introduction to Programmable Logic Controllers – Part I

Module 2: Number Systems and Logic Functions

In this module, the decimal number system and the binary number system are introduced. The conversion between a decimal number and a binary number is addressed. Basic binary number logic functions are also introduced.

The most commonly used number system is the decimal number system. This system uses ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. In a decimal number, the digits at different positions have different weights. For example, the decimal number

$2234 = 2 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 = 2 \times 1000 + 2 \times 100 + 3 \times 10 + 4 \times 1.$

The first "2" on the left side weighs ten times as much as the "2" next to it.

In the binary number system, there are only two digits: 0 and 1. A binary number consists of a string of 0s and 1s, each of which has a different weight. For example, the binary number

$$\begin{aligned} 101101 &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \end{aligned}$$

It is decimal number 45.

Following is a table for the equivalence of binary and decimal numbers from 0 to 15.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

To convert a decimal number to a binary number, the successive division method can be used. In this method, first divide the given decimal number by 2. The remainder becomes the most right bit of the binary number. Then, divide the quotient by 2. The remainder becomes the next bit on the left of the binary number. This division process is repeated until the quotient is 1, which becomes the most left bit of the binary number. As an example, convert the decimal number 115 to a binary number.

Divide 115 by 2. The quotient is 57 and the remainder is 1, which is the first bit on the right of the binary number.

Divide 57 by 2. The quotient is 28 and the remainder is 1, which is the next bit on the left side of the binary number.

Divide 28 by 2. The quotient is 14 and the remainder is 0, which is the next bit on the left side of the binary number.

Divide 14 by 2. The quotient is 7 and the remainder is 0, which is the next bit on the left side of the binary number.

Divide 7 by 2. The quotient is 3 and the remainder is 1, which is the next bit on the left side of the binary number.

Divide 3 by 2. The quotient is 1 and the remainder is 1, which is the next bit on the left of the binary number.

Now, the quotient is 1, which is the most left bit the binary number. The converted binary number is 1110011.

To convert a binary number to a decimal number, convert each bit of the binary number to a decimal number and add all of the bits together. For example, the binary number

$$\begin{aligned}110101 &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0\\ &= 32 + 16 + 0 + 4 + 0 + 1 = 53\end{aligned}$$

For practice, convert the decimal number 95 to a binary number. The answer is 1011111. Also convert the binary number 1000111 to a decimal number. The answer is 71.

To process numbers in electronic circuits, such as in a PLC or a computer, the numerical quantities must be represented by electrical signals. The binary number system is ideally suited for processing data electronically because only two voltage levels are required to represent all the different digits in the binary system. The voltages used to represent a binary 1 and a binary 0 are called *logic* levels. Ideally, one voltage level represents a HIGH (binary 1) and another voltage level represents a LOW (binary 0). In a practical digital circuit however, a HIGH can be any voltage between a specified minimum value and a specified maximum value. Also, a LOW can be any voltage between a specified minimum value and specified maximum value. But, there can't be any overlap between the accepted HIGH levels and accepted LOW levels.

Understanding basic logic operations of binary numbers is very helpful for learning PLC programming. The basic logic operations include AND, OR, and NOT. Figure 2.1 shows the symbols of these logic operations.

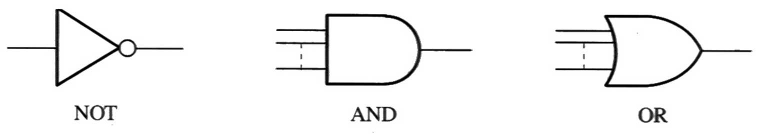

Figure 2.1 Symbols of Basic Logic Operations

An AND operation has two or more inputs and one output. Only when all the inputs are HIGH, will the output be HIGH. If one or more of the inputs is LOW, the output is LOW. The relation between the input(s) and the output of a logic function can be presented by a *truth table*, which is a table listing all the possible input combinations and corresponding output status. Following is a truth table of a three-input AND operation. The logic equation of this operation is X = A*B*C or X = ABC. The symbol " * " is for AND operation.

Input			Output
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

An OR operation also has two or more inputs and one output. If one or more of the inputs is HIGH, the output is HIGH. Only when all the inputs are LOW, will the output be LOW. Following is a truth table of a three-input OR operation. The logic equation of this operation is X = A+B+C. The symbol “ + ” is for OR operation.

Input			Output
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

A NOT operation has one input and one output. The output is always opposite to the input. The following truth table shows the input and output relation of a NOT operation. The logic equation of this operation is $X = \overline{A}$.

Input	Output
A	X
0	1
1	0

Related web sites:

http://www.geocities.com/regia_me/
http://www.danbbs.dk/~erikoest/binary.htm
http://www.play-hookey.com/digital/basic_gates.html