Chapter 11

Boolean Algebra

Boolean Algebra

When we hear the word “Algebra", we usually think of high school algebra with its positive and negative numbers, simultaneous equations, quadratic equations, etc. However, this is not the only kind of algebra.

Relax ! Boolean algebra is considerably easier than high school algebra.

George Boole (1815-1864) invented a new algebra to describe logic and thought. Since the time of the early Greek philosophers, much logic and reasoning has been done using *true* and *false* statements. For many years mathematicians tried to include the laws of true-false logic in the realm of algebra but failed. Boole succeeded with the publication of "An Investigation of the Laws of Thought,..." in 1854. What Boole did was to symbolize logic by a new kind of algebra. In other words, he showed that some types of thinking and reasoning could be done by manipulating symbols.

Boole's algebra stayed in the domain of pure mathematics until almost a century later. In 1938, Claude Shannon wrote a paper titled "A Symbolic Analysis of Relay and Switching Circuits." This paper applied the new algebra to switching circuits and since then Boolean algebra has been widely used in telephone and digital systems. Most computers use this algebra as the basis for all of their operations.

In this chapter, you will become familiar with the basics of Boolean algebra so you can program the PMC. There are only three functions to learn, OR, AND and NOT.

Boolean algebra differs from ordinary algebra in some ways. In ordinary algebra when we solve an equation for its roots, we can get any real number: positive, negative, fractional and so forth. In other words, the set of numbers in ordinary algebra is infinite.

In Boolean algebra when we solve an equation, we get either a 1 or a 0. No other answers are possible because the set of numbers includes only the binary digits 1 and 0. These numbers are also referred to as true and false, and high and low. In the PMC system, a 1 would represent a closed switch input, or an output that is ON. An 0 represents an open switch, or an output that is OFF. This lack of other numbers gives rise to a number of new thoughts, such as the meaning of the plus sign.

In Boolean algebra there are three basic functions, OR, AND and NOT.

THE “OR” FUNCTION

The OR function means that the output is true if *any* input is true. In other words, the OR function is an *any or all* function.

If Q is the output and A and B are inputs, **the simple OR equation is written:**

Q is the output on (relay)
A and B are inputs (switches)
Q = A + B

This means that A and B are OR'ed together. The OR function has a 1 output when either *A, or B,* or both are 1. In PMC the load is ON if either input switch A, or B are ON.

The following table lists the input-output conditions of the OR function. This table, called a "truth table", shows all the input-output possibilities for a logic circuit. 1= ON 0 = OFF.

Inputs		Output
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

The point to remember about the OR function is that it has a 1 output when either *A, or B,* or both are 1.

THE "AND" FUNCTION

The AND function means that the output is <u>true</u> only if *all* the inputs are <u>true</u>. In other words, the AND function is an <u>*all or nothing*</u> function.

The simple AND equation is written:

Q = A * B

This means that A and B are AND'ed together. **The truth table for this function is as shown below:**

Inputs		Output
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

In PMC, the load Q will be ON only if both switches A and B are closed.

THE "NOT" FUNCTION

The NOT function is a simple function that inverts the input. The output is *not* the same as the input. This is normally written as a prime, or an over-score.

The simple NOT function is written:

$$Q = A' \text{ or } \overline{A}$$

For purposes of our typing, we will use an exclamation point in front of a term to mean NOT. **So the function will be written as:**

Q = !A

The truth table for the NOT function is:

Input	Output
A	Q
0	1
1	0

Through the use of just these three functions, all logic can be expressed. They can be combined into various equations to create more complex functions that are useful in describing logic for control systems.

With these few functions, you will be able to describe the electrical functions of a vehicle.

SOME EXAMPLES

Let's look at a couple of examples. The first example is the function of a simple light switch, that we will call input A. A can either be 0 for OFF, or 1 for ON. The output to the light is Q.

The switch function can be described as:

Q = A

Simple isn't it. The output to the light is the same as the output of the switch. When the switch is ON, the light is ON. When the switch is OFF, the light is OFF.

Now let's say that we want the switch to be turned ON from two different switches, A, or B. This might be useful to operate interior lights from door switches. The switch on the first door will be A, the switch on the second door will be B, and the output to the light Q.

That equation would be written:

Q = A + B

This means if either switch A, *or* B are ON, the light will be ON. Or in other words, in our example, if either door or both is open, the interior light will be ON.

Let's take another example of light that wants to be ON only if two switches are ON. We will turn ON the fog lights only if the headlights are ON. Let's call the fog light switch A, the headlight switch B, and the fog light output Q.

The fog light equation would be written as follows:

Q = A * B

This means that both the headlight switch *and* the fog light switch have to be ON to turn the fog lights ON.

Boolean Algebra

These first examples have been simple, using only two terms to describe an output. Although you will find that most of your system can be simply defined, output functions are not limited to just two input terms. As a more elaborate function, let's consider the control of an air conditioner evaporator fan on a bus. Obviously, the fan should run when the temperature is too high, so the thermostat should be one of its inputs. Let's say that we *also* want it to run if the defroster fans are ON, but *don't* want it to run if the alternator light is ON. What would the equation for this function be ?

Let's define our terms:

A1 = Evaporator Fan
B1 = Thermostat switch
B2 = Defroster Switch
C1 = Alternator Light

In Boolean terms, we want the evaporator fan to run if the thermostat switch is ON, OR if the Defroster Switch is ON, AND NOT if the Alternator light is ON.

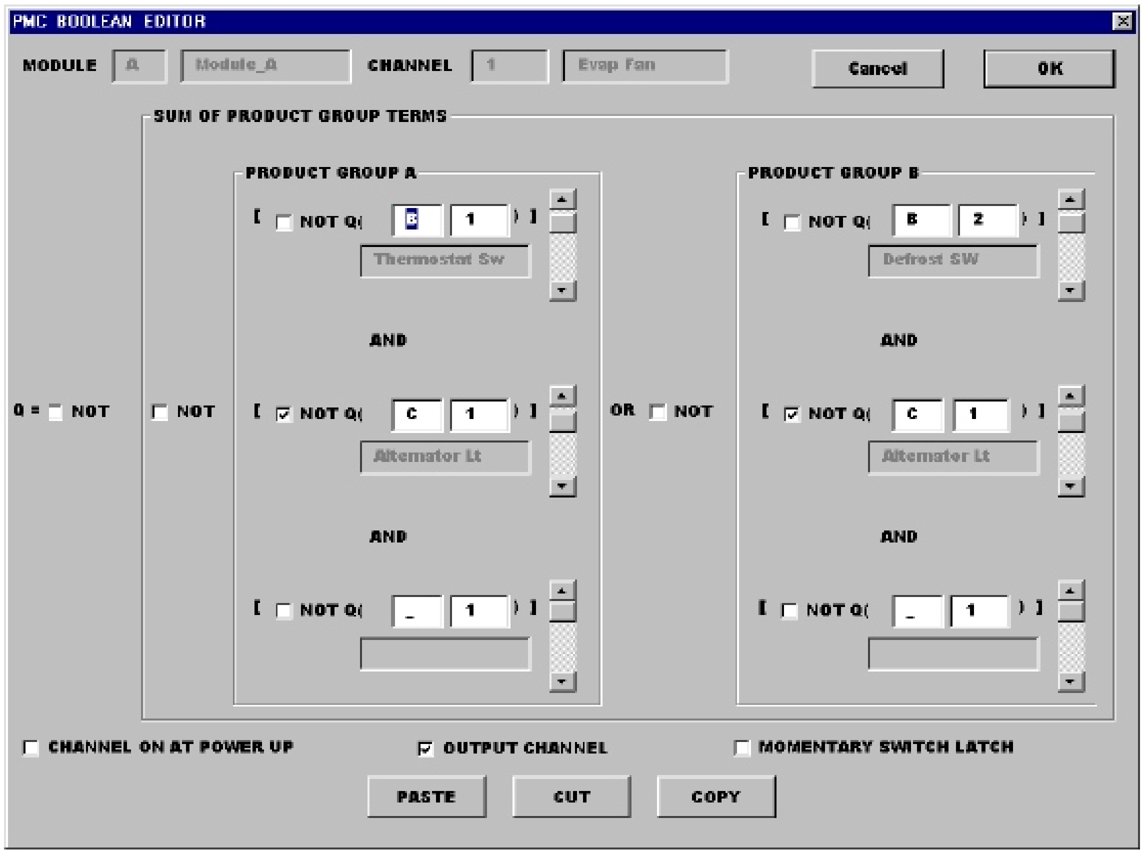
The equation would be written:

A1= (B1+ B2) * !C1

Stated another way:

A1= (B1*!C1)+(B2*!C1)

The PMC Screen will look like this:



PMC BOOLEAN EDITOR

MODULE A Module_A CHANNEL 1 Evap Fan

SUM OF PRODUCT GROUP TERMS

Q = ☐ NOT ☐ NOT

PRODUCT GROUP A

[☐ NOT Q(B 1)] Thermostat Sw

AND

[☑ NOT Q(C 1)] Alternator Lt

AND

[☐ NOT Q(_ 1)]

OR ☐ NOT

PRODUCT GROUP B

[☐ NOT Q(B 2)] Defrost SW

AND

[☑ NOT Q(C 1)] Alternator Lt

AND

[☐ NOT Q(_ 1)]

☐ CHANNEL ON AT POWER UP ☑ OUTPUT CHANNEL ☐ MOMENTARY SWITCH LATCH