

CHAPTER

10

Logic Gates

To understand the point of this laboratory exercise, you will need to know a few things about the robot. The robot you are building has a memory chip that stores all of the commands that you give it as instructions. Each instruction contains four binary entries that correspond to a beep signal, an LED flash signal, a right wheel drive signal, and a left wheel drive signal. The value of each bit in a particular instruction tells the robot whether or not to activate the corresponding functions - 0 = no, 1 = yes. When you send the robot on its way, it does all of the operations associated with each instruction simultaneously. The robot steps through its list of stored instructions one instruction unit at a time. It carries out the current instruction for one full period of the clock.

Every instruction is assigned a number (called its address) that is an integer from 0 to 255. The robot starts executing the instruction stored at address 0 in its memory of instructions. The next clock cycle it executes the instruction stored at address 1 in its memory, and so on. The memory IC chip does not automatically send out instructions in order. Instead, another chip, a counter, tells the memory chip which instruction to send out. The counter chip starts at 0, and increments the address of the instruction by one every clock cycle. The memory chip looks up the instruction stored at the address given by the counter and sends out the corresponding instruction to be executed. The memory chip can store up to 256 instructions. When the counter reaches a count of 255, instead of increasing to 256, it jumps back down to a count of 0.

Remember that the clock generates pulses about one to four times per second. With 256 stored instructions, the robot will take one to four minutes to completely go through all of the instructions stored

Two basic kinds of gates are the AND gate and the OR gate. A simple AND gate has two inputs, A and B, and one output, Z. All inputs to an AND gate must be 1 for the output to be 1. If both A and B are 1 then the output is 1. Otherwise, the output is 0. Fig. 10.1 shows the symbol for an AND gate and its truth table is given in Table 10.1 .

Input: A	Input: B	Output: AB
0	0	0
0	1	0
1	0	0
1	1	1

Table 10.1 Truth Table for an AND gate.

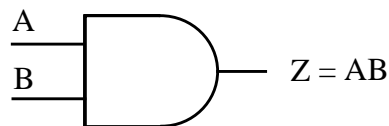


Figure 10.1 Circuit Schematic Symbol for an AND gate.

In equation form, two variables beside each other mean AND. “A and B” is written as AB. You can think of AND as the binary version of multiplication.

A basic OR gate has two inputs, A and B, and an output. The output is 1 if either A or B, or both, are 1, and 0 only when both A and B are 0. The symbol for a two-input OR gate is shown in Fig. 10.2 and its truth table is shown in Table 10.2 .

Input: A	Input: B	Output: A+B
0	0	0
0	1	1
1	0	1
1	1	1

Table 10.2 Truth Table for an OR gate.

A+B is the algebraic representation of “A or B.” You can think of OR as the binary version of addition except that $1+1=1$ instead of 2 since 2 is not a binary number.

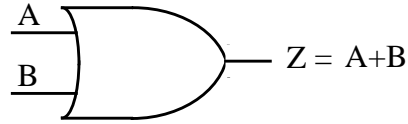


Figure 10.2 Circuit Schematic Symbol for an OR gate.

NAND and NOR gates are constructed by adding an inverter after AND and OR operators respectively as shown in Fig. 10.3. An interesting property of NAND and NOR gates is that each can be connected to simulate inverters, AND gates, and OR gates. For this reason, the robot circuit uses only NAND gates.

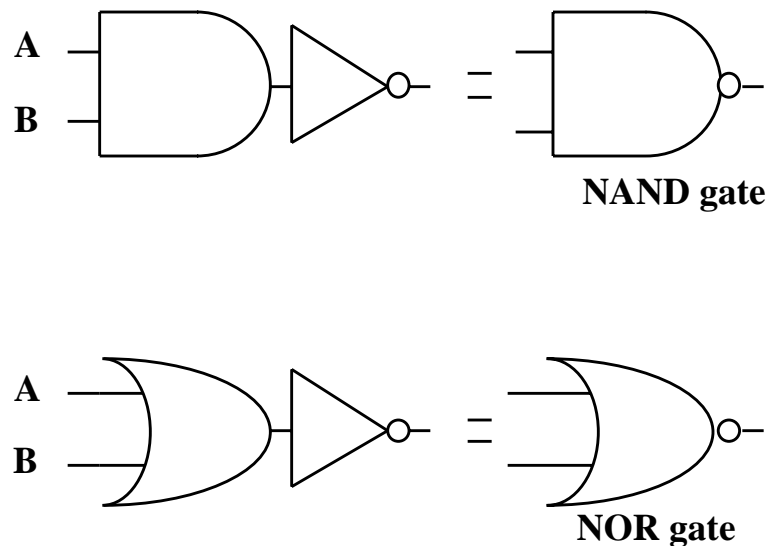


Figure 10.3 Converting AND into NAND and OR into NOR.

The circuit you will be building in this lab is shown in Fig. 10.3. Remember, the purpose of the circuit is to provide a logic 1 whenever we want to reset the counter. The output of the circuit should stay at 0 unless the reset switch is pushed, or a reset instruction is encountered. A NAND gate output is 1 if either of its inputs are 0. Consequently, the inputs to NAND gate 2, pins 9 and 8, are both 1 unless a reset signal is sent. The input to the NAND gate that is connected to the push-button reset switch is normally held high by

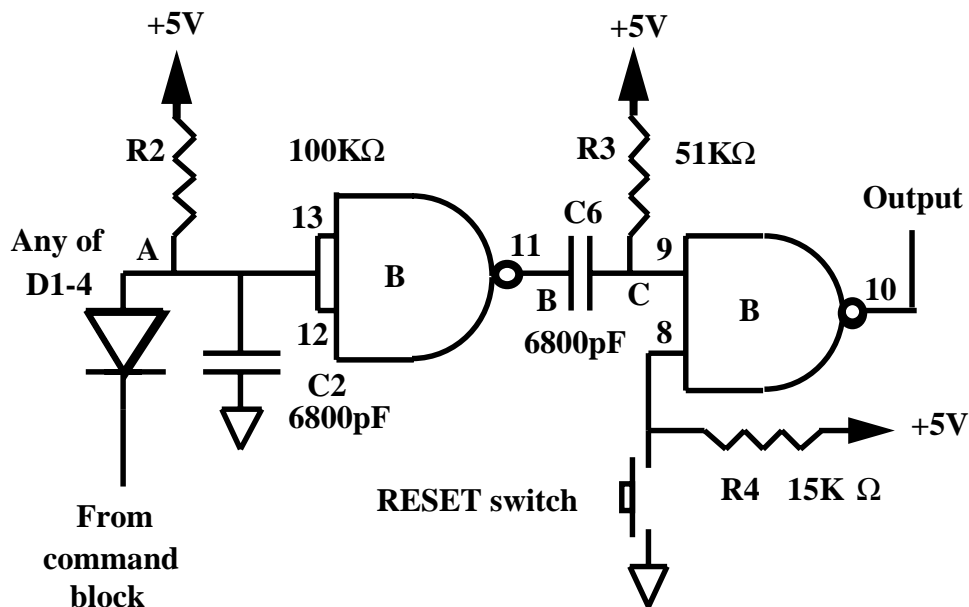


Figure 10.4 The Robot Restart Circuit.

the 5V connection through the 15KΩ resistor, but when the switch is closed, the NAND gate input becomes 0V, or logic 0. The output of the circuit then becomes a 1, and stays there as long as you are pressing the push-button switch, regardless of what the other input to the NAND gate is doing.

A programmed reset changes the left NAND input to a 0, and it works as follows: There are 4 diodes, one connected to each command type. If the current command is to have right and left motors on, then the signals for those two commands would be high and the signals for the beeper and light would be low. If any one of the command signals is low, then point A will sit at about 0.7V and that diode will conduct the current flowing through R2. If all 4 command inputs happen to be high, then none of the diodes can turn on. The current flowing through R2 will charge up C2 until it reaches the threshold of the NAND gate. In essence, the 4 diodes and the resistor are acting like a 4 input AND gate: the output only goes high if all 4 inputs are all high. In the early days of computers, logic devices were actually built using diodes in just this manner.

The inputs to the left NAND gate are connected together, so they are both either a 1 or a 0. If they are a 1, the output is a 0, and vice versa. The NAND gate is wired to act like an inverter. The input to the NAND inverter is connected by four diodes to each block of the

instruction unit output. Usually, one of the robot functions is not turned on, so one of the diodes is connected to a logic 0, or low voltage. This puts the input to the NAND inverter at about 0.7V, or Logic 0. The output, or point B, becomes a 1. Also, point C is held at 1 by the 5V source. Therefore, there is no voltage across C6, and no charge stored in it. When an instruction unit with all four robot functions comes, all of the diodes are turned off. Consequently, point A is pulled up to logic 1 by R2, and point B drops to logic 0. Since C6 had no charge stored, and since it cannot charge up instantaneously, point C drops to a low voltage with point B. A current then flows through R3 to charge up C6, and point C returns to logic 1 after a short time. However, for that short time when point C is a logic 0, the output of the right NAND gate becomes a 1, and that is sufficient to reset the counter. This is just like the pulse generator circuit you saw in a previous laboratory exercise. Fig. 10.5 shows the timing diagram for this circuit. The voltage level at each significant point is plotted on the same time axis so you can see how the program reset works when the RESET push-button switch is not being used.

One last aspect of this circuit: when you plug in the teach pendant, you do not want the counter reset at all. Because of this, attaching the teach pendant automatically connects point A to ground, holding points B and C high.



Laboratory Exercises

1. Before you test the circuit shown in Fig. 10.4 you will need to check out the operation of the NAND gates. There are two NAND gate chips used in your robot. You should use one of them to wire up the circuit of Fig. 10.4 (see step 4) before coming to lab, and you will use the other one to characterize the behavior of NAND gates (see steps 2 and 3). The Quad NAND gate has 4011 printed on the top. The internal gates are shown in Fig. 10.6.
2. First, power up the NAND gate IC by applying ground and +5 volts at pins 7 and 14 respectively. Use the

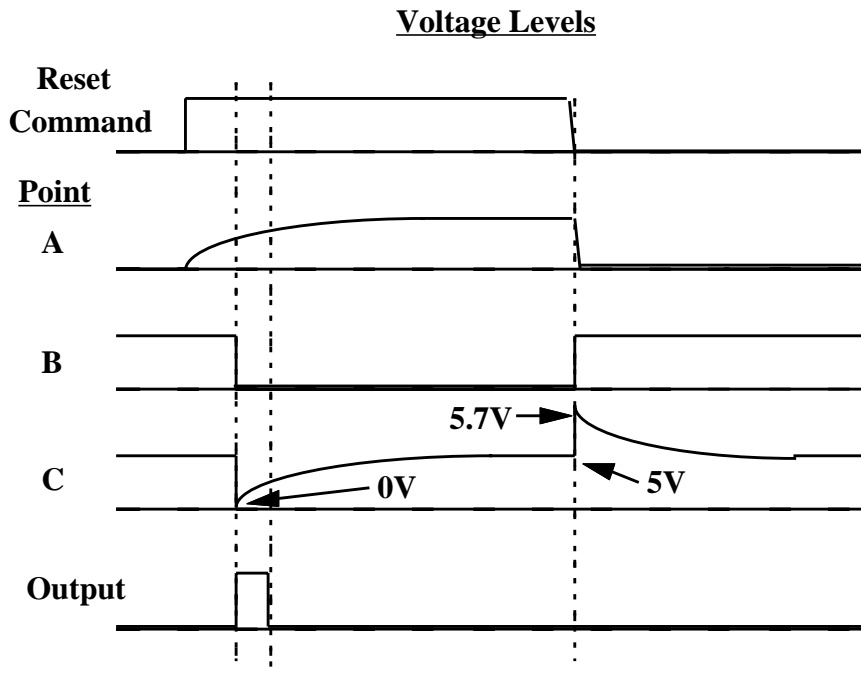


Figure 10.5 Timing Diagram for Robot Reset circuit.

NAND gate with input pins 8 and 9 and output pin 10 to test and measure the output voltage for all four logic input combinations: 0-0,0-1,1-0, and 1-1. To make a logic 0, connect the pin to ground. To make a logic 1, connect the pin to 5V. To observe the value of the output, connect an LED and resistor to pin 10 of the NAND gate as in Fig. 10.7.

3. Construct the circuit of Fig. 10.4 on the protoboard. Do this before you come to Lab. You should install 4 diodes, one for each command type. Simulate the command codes by wiring the bar end of the diode to either ground or +5V. That is, to make a logic 0, connect the pin to ground. To make a logic 1, connect the pin to 5V. Initially connect all 4 diodes to ground. To observe the value of the output, connect an LED and resistor to pin 11 of the NAND gate as in Fig. 10.7.
4. Connect two bare wires, one to pin 8 of the NAND gate, and the other wire to ground. By touching these two wires together with your fingertips, you can simulate that action of the push-button switch. Touching the wires together should cause the output to change to a

logic 1. Consequently, the LED on pin 10 should light. When you remove your fingers the output should change back to a logic 0. Consequently, the LED should go dark.

5. Using your function generator, create a 200 Hz square wave signal which goes between 0 and 5V. Connect one diode to the function generator. Connect all of the other diodes to +5V. Using the oscilloscope, look at this input simultaneously with the circuit's output, pin 10.
6. Finally, install the restart circuit onto the robot PC board. Install only one diode and leave the black banded side unconnected for now. Set the +20V part of your power supply to +9V and attach it to the +9V pin of the robot PC board. Attach the power supply common to one of the GND pins of the robot PC board. With SW1 on, check that the output of your voltage regulator is still +5V. Connect the function generator to one of the diode inputs, and use the oscilloscope to verify the functioning of the restart circuit. You can connect to the diode end by inserting any resistor into Pin 16 of the 22 pin socket and then connecting your function generator's signal to the socket-side lead of the resistor. Don't forget to connect the common side of the function generator and the oscilloscope to the common of the power supply. You can observe the output by sticking a resistor into pin 11 of the 16 pin socket and grabbing that resistor with the scope probe. Pin 11 is the reset input of the counter chip. Remember to try the RESET push-button switch as well. If there is a problem, identify and fix it, and then explain what went wrong. If you cannot solve the problem ask your TA for help.

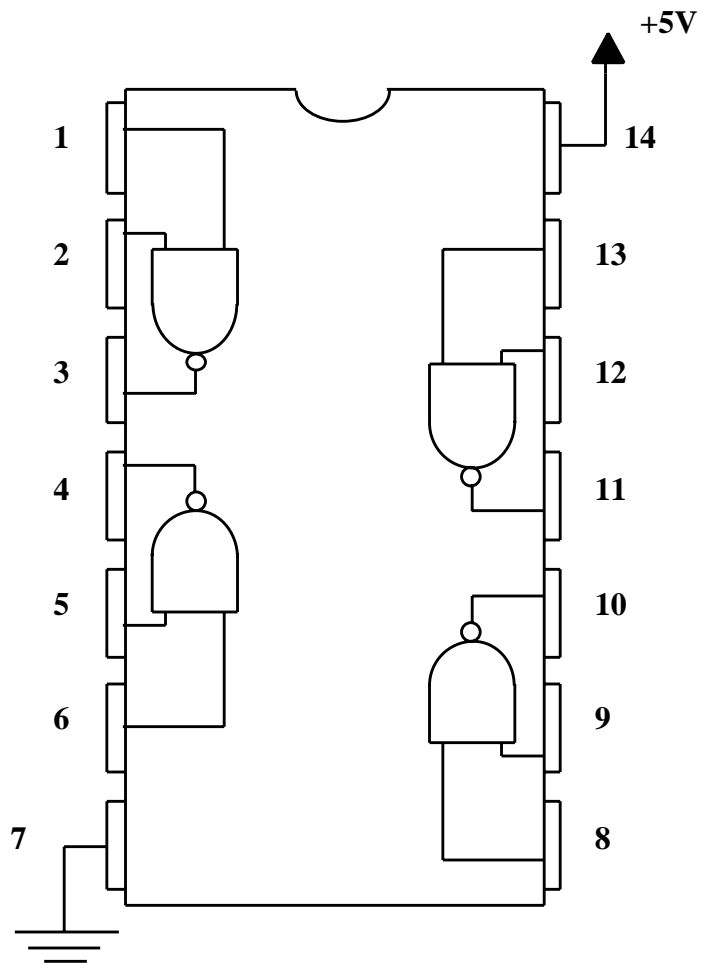


Figure 10.6 Pinout of CMOS 4011 NAND gate.

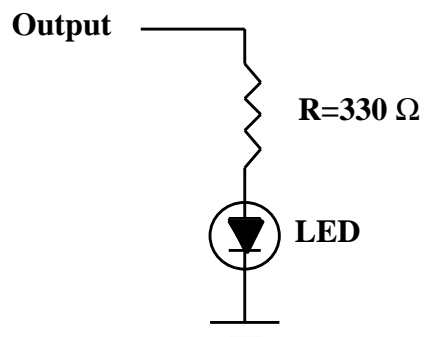


Figure 10.7 Using an LED to construct a digital signal indicator.

