The flashing “Merry Christmas” sign[1]

© Steve Lee, Christmas 2016
EpiphanyBySteveLee.com

Introduction:

This project offers a very simple 556-based circuit used to drive a flashing string of colored LEDs arranged to spell “Merry Christmas” (plus a few LEDs arranged in a Christmas tree formation). The circuit requires one LM7808 or 7809 voltage regulator, roughly 120 colored LEDs, one 556 dual timer IC, one CMOS 4001 quad NOR gate IC, and a handful of discrete components (resistors and caps, as shown in the schematic). Total cost is around $10 if the parts are bought in bulk (e.g. one Jameco.com LED grab bag + chips, etc.).

How it works:

The heart of this circuit is a single 556 Dual Timer IC (i.e. one chip that contains a pair of 555 timers). Both timers (U1a and U1B) are configured as free-running oscillators, with the first (U1a) providing a rapid “twinkling”/”shimmering” effect, and the second (U1b) providing the slower toggle rate of roughly one second “high” and one second “low”, to provide the toggling select between which half of the sign is lite (“Merry” or “Christmas”).

To create the “twinkling”/”shimmering” effect, U1a generates an asymmetric square wave, with the “high” time equal to ~200 mSec and a “low” time equal to ~50 mSec. The “high” and “low” state time for each oscillator is calculated according to the following formulas:

$$T_H = 0.7 * (R_A + R_B) C$$

$$T_L = 0.7 * (R_B) C$$

The outputs from both U1a and U1b are combined together in the NOR logic gates to create both “A●B!” and “A●B” select lines (where “●” indicates logical AND, and “B!” = the inverse of B). The output of the logic section is then used to select and “shimmer” the bank of LEDs that form the two halves of the sign, alternately illuminating first the LEDs for the “Merry” half of the sign, and then the LEDs for the “Christmas” half.

For added effect, one red LED in the green Christmas tree LED group is connected to the “Merry” LED bank, and another red LED in the tree to the “Christmas” LED bank, giving the tree its own pair of alternating tiny flashing Christmas tree lights (see photos).

Building and troubleshooting:

A thin 5” x 9” plastic sheet cut from an old shower liner left over from a bathroom remodeling project was used as the mounting base for the sign's LEDs. Small mounting holes

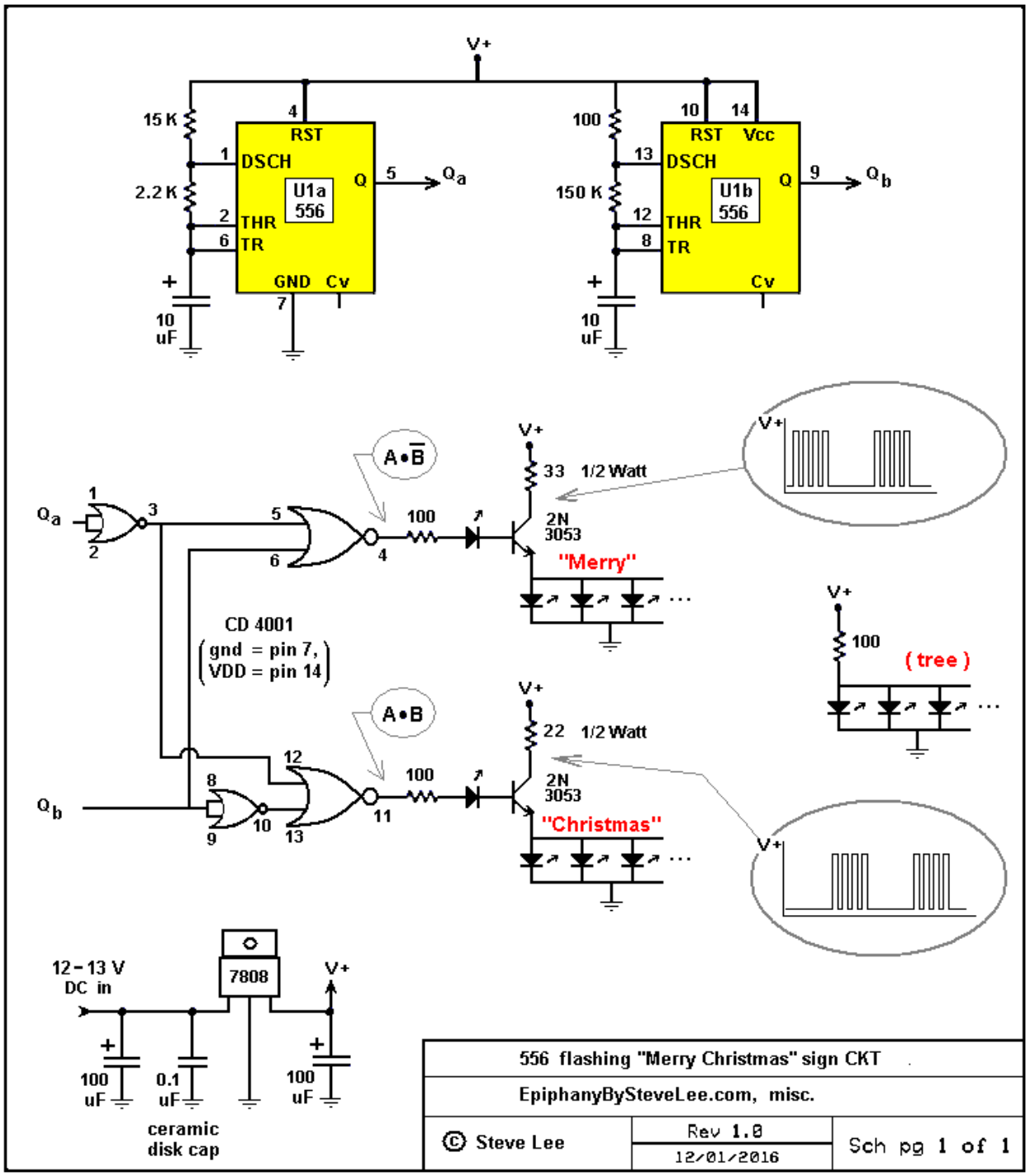
1 See “EpiphanyBySteveLee.com” for additional projects and tutorials.

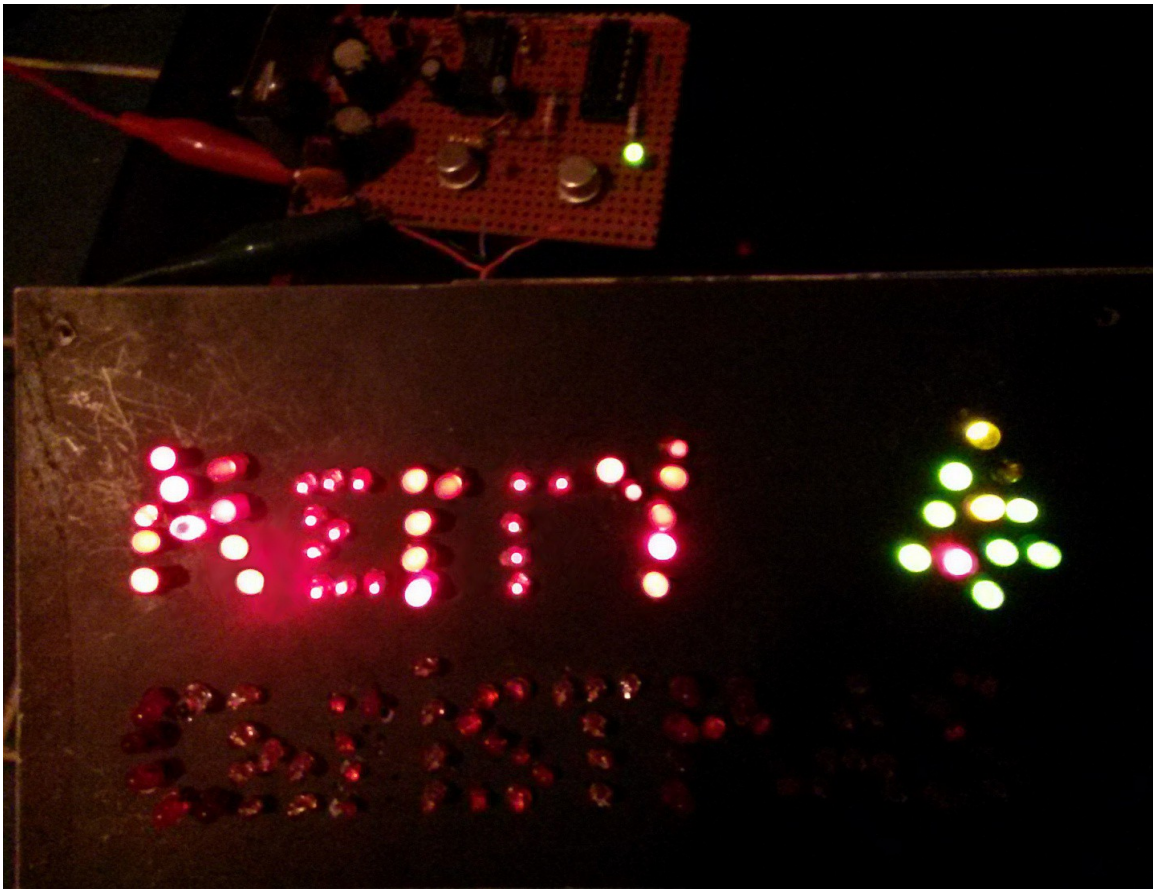
were pre-drilled into the plastic sheet, one pair of holes for each LED used to make up the letters in the sign. (Note that we highly recommend you first pencil in the placement of each LED on the sheet well prior to drilling any holes.) Once the final placement design was settled, and all holes then drilled, the plastic mounting base was spray painted black to increase the viewing contrast with the LEDs.

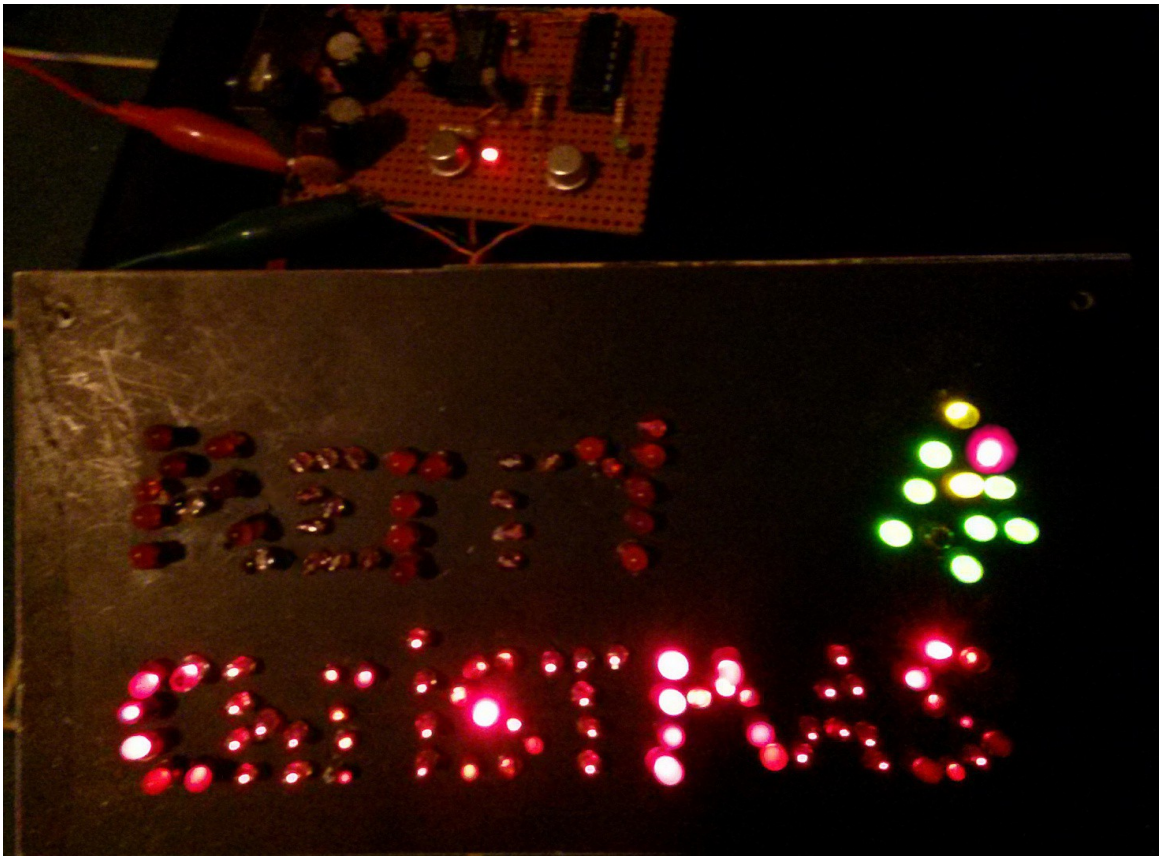
The circuit itself is very straight forward, making the hardest and most tedious part of the whole project finding enough similar LEDs, and then mounting/soldering all LEDs in place in the plastic mounting base. To reduce the post-construction troubleshooting pain, each LED was pre-tested to confirm each worked, and then marked for polarity with a permanent marker before mounting and soldering each LED in place. Each completed letter/section was then tested to confirm everything was working before moving on to the next. If either bank of LEDs fail to light, verify the 2N3053's are toggling with Qa and Qb. Next check the wires between the 2N3053's and LEDs (no broken connections or cold solder joints), as well as the ground.

The whole project took about a day to complete (not counting “post construction” “tweaking” time).

The “Merry Christmas” sign stencil, first draft.


V+
15 K
2.2 K
10 uF
4
RST
1
DSCH
2
THR
6
TR
GND
7
Cv
U1a
556
Q
5
Qa
100
150 K
10 uF
10
14
Vcc
13
12
8
9
Qb
U1b
556
A•B
33 1/2 Watt
2N 3053
"Merry"
100
(tree)
CD 4001
(gnd = pin 7,
VDD = pin 14)
22 1/2 Watt
"Christmas"
12 - 13 V
DC in
7808
100 uF
0.1 uF
ceramic disk cap
556 flashing "Merry Christmas" sign CKT
EpiphanyBySteveLee.com, misc.
© Steve Lee
Rev 1.0
12/01/2016
Sch pg 1 of 1

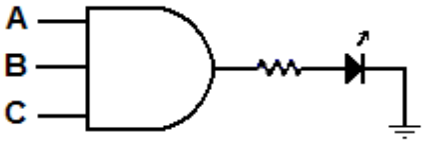

MERRY


CHRISTMAS

Appendix A: Notes on digital logic gates

In this circuit we used four logic NOR gates to combine the outputs from the two 556 square wave oscillators ("A" and "B") in such a way as to give us the "A●B!" and "A●B" select lines (read as: "A and B not" and "A and B"). Those not familiar with logic circuits may wonder how we developed a logical "AND" using logical "NOR" gates[2]. Well, it all has to do with that serviceable villain, Colonel Mustard.

If you've ever played the game "Clue", you know the object of the game is to correctly guess three things: A) the identity of the murderer, AND B) the weapon used, AND C) the room where the dastardly deed took place. If we were to design a simple logic circuit to light an LED when you get it right (i.e. output a logical HIGH), it would look something like this:



A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
1	1	1	1

$= A \bullet B \bullet C$

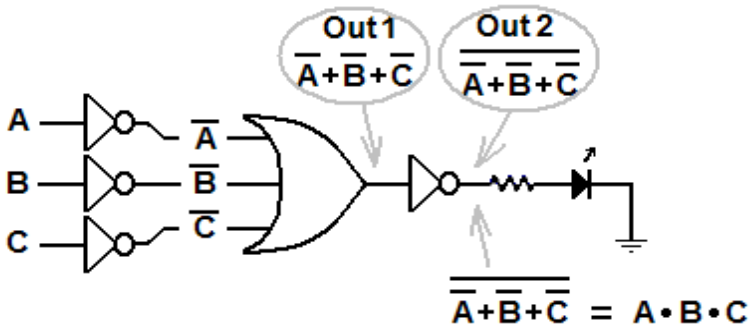
As the logic table to the right of the gate indicates, the only way to get a "High" out of the gate (i.e. indicating a correct guess in our "Clue" game), is to get ALL three inputs correct (i.e. a High on "A" AND "B" AND "C").

As your opponent, I am naturally hoping that you will not get it correct and thus lose. So we ask what is required for you to lose (i.e. get a logic Low out of our gate)? To lose you only have to guess "A" wrong, OR "B" wrong, OR "C" wrong.

From this we see that a logical "OR" is effectively the inverse of a logical "AND" (and vice-verse). To demonstrate that, let's put each input through an inverter, and then into an "OR" gate. That gives us the exact inverse of our "AND" gate ("Out1" in the figure below). If we then put that output through another inverter, we then get the exact same output as our "AND" gate above ("Out2" in the figure below).

If we represent a logical "AND" with a dot operator ("●"), a logical "OR" with a plus operator ("+"), and a logical "INVERTER" with an over-score ("$\overline{\ }$"), we can write an equation to describe these logical operations (as shown in the figure below).

Note that one inverse over an "OR" operator (single line) converts that into an "AND" operator (and vice-verse), while a double inverse cancels itself out (since: "-1 x -1 = +1):



$\overline{A}$	$\overline{B}$	$\overline{C}$	Out 1	Out 2
1	1	1	1	0
1	1	0	1	0
1	0	1	1	0
0	1	1	1	0
0	0	0	0	1

2 Note that a "NOR" gate is just an "OR" gate with a "NOT gate" (an inverter) attached to its output.

From this project's design, it is clear that we can not only use a NOR gate to serve as a logical AND gate, but by tying the inputs of a NOR gate together, we can also use a spare NOR gate to serve as an INVERTER and thus “flip” signals as needed.

From all of this, one might be tempted to ask “if a single type of gate can be used to get both logical operations, why bother to design more than one type of gate?”

In a slow speed circuit such as in our “Merry Christmas” sign, one type of gate works fine. However, in any circuit designed to process digital signals at *high rates*, such as in your laptop computer running at 3 GHz (i.e. 3 billion clock cycles per second, with the time for one clock cycle = 1/F = 0.33 nSec), or in a processing amplifier used in a fiber optic data link running at 10 Gbps, the fewer gates you have strung together, the better., since each gate adds some amount of extra delay.

Though we represent each of these logical gates with a single symbol, in reality each logic gate is typically composed of many individual transistors. When a transistor is ON, a great many electrons are flowing through it (e.g. 1 mAmp = ~ 10^{16} electrons/second). The moment the transistor switches OFF (e.g. at the trailing edge of a square wave), all those electrons don't just suddenly disappear. In reality, it may take 10 – 15 nSec (nSec = 10^{-9} seconds) for all of those electrons in all the gate's transistors to drain out and thus allow the output transistor to switch OFF. Adding say ten extra gates in series to make several NOR gates represent several AND gates, can add ~100-150 extra nSec to the signal delay.

You may be tempted to think “okay, so it takes an extra 150 nSec to get my data; big deal”. If every signal in the system experienced the exact same delay, it might not be much of a problem in many applications. However when one signal presented to a gate/device down the line is delayed more than any of the other signals presented to that same device (including the clock signal), it can be a *very* big problem.

Consider the example shown in the figure below, where a delay in the “Chip Select !” (read as: “chip select not”, or inverted CS) signal by only a single clock cycle (e.g. 0.33 nSec) results in the device processing data while the data bus is in transition. That can cause the device to read and thus store/transmit the wrong data bit values, introducing bit errors into the data stream. As a result, in high speed applications even a few tens of nSec of delay can make a very big difference as to whether the circuit even works correctly or not.