

Re: PMOS in parallel with NMOS

Source: <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2008-05/msg00415.html>

- *From:* Tomás Ó hÉilidhe <toe@xxxxxxxxxxxx>
 - *Date:* Thu, 8 May 2008 09:58:49 -0700 (PDT)
-

On May 8, 4:20 am, rickman <gnu...@xxxxxxxxxx> wrote:

Before I bother to address the design issues you present, I want you to go back and reread the last couple of messages you posted and tell me exactly what circuit you have described. I only see described a pair of transistors with their B-E junctions connected directly to the power supply. I have no idea if you are talking about an emitter follower configuration or a common emitter configuration.

Take a PNP bi-polar transistor.
Connect 0 volts directly to the base.
Connect 5 volts directly to the emitter.
Connect the anode of the LED to the collector, and the cathode of the LED to ground.

Now obviously, in everyday life, you put a resistor going into the base, and also a resistor in series with the LED. Why? To stop:

- 1) Damage to the transistor from having too much base current
- 2) Damage to the LED from having too much current thru it

Regarding the LED, well a friend of mine has told me of experiments where people flashed a normal LED with as much as an entire ampere, and it worked fine because the duty cycle and the pulse width were sufficiently low that the LED didn't get damaged. This is quite easy to see if you take a green diode; if there's too much current, it will glow yellow. On my own board, my green LED's stay green.

It seems quite conceivable to me that I'm experiencing the same thing with the transistor, i.e. I'm putting a massive current thru it but it's OK because the pulse width and duty cycle are low enough that it won't get damaged.

Another thing I'll look into is the current limit on the microcontroller pins -- specifically, what happens if you try to draw too much current. Maybe the microcontroller will die, or maybe I'll

Re: PMOS in parallel with NMOS

just get an output voltage less than 5 volts. Who knows? I'll look into it.

However, if you think the output capacitance of the MCU pin is limited the current to the transistor, whew! you have a lot to learn.

I'm sure I could find a sufficiently low pulse width and duty cycle that would make this happen. Whether this pulse width and duty cycle is within the MCU's capability, I don't know.

BTW, what is the speed of your multiplexing? Is it above 100 MHz?

There's 16 columns. Each column stays lit for 220 microseconds. The shifting time is negligible compared to the 220 microseconds, so it takes about 3.5 milliseconds to perform one full flash. That gives a frequency of about 284 Hz.

The rise time of a typical I/O pin on an MCU is on the order of a few ns. So the multiplexer would have to be running seriously fast for the output capacitance to have any effect on it.

Noted, that answers my question above ^

Have you tried reading some books on embedded system design? Or are you planning to learn it all on your own from first principles?

None specifically on embedded systems, just more generic electronics.

If you go back to the MOSFET parts, this circuit will work ok. The gate does not need current drive. When the MCU is not driving the

Re: PMOS in parallel with NMOS

gates can be pulled to the mid voltage point with a pair of resistors and you can pick MOSFETs with high enough threshold voltage that there will be no chance of either LED being on, as long as you use an emitter follower circuit. You do still need a current limiting resistor in the drain leg.

I'll look into it, thanks.

If you think all of this is unneeded, then please explain why the MCU I/O pin burned out when the LED didn't have a current limiting resistor? And no BS. Either figure it out so that you understand it and can explain it, or don't bother replying. The MCU is not burning out why you think it is

Each LED gets flashed as follows:

Pulse width = 200 microseconds

Duty Cycle = 1/16

The board works perfectly when set like this.

However, if I increase the pulse width to 300 microseconds, the board dies. Perhaps it's the microcontroller that's being killed, perhaps it's the shift register that's being killed. Maybe even the transistors, I don't know.

Thinking about it logically, what exactly changes when I increase the pulse width? Well, current flows for a longer amount of time. The significance of this? Well, current flow produces heat, so maybe too much heat is building up for the duty cycle to compensate for. I don't know, I'll look into it.

.