Definition

Charlieplexing (also known as tristate multiplexing, reduced pin-count LED multiplexing, complementary LED drive and crossplexing) is a multiplexing technique for driving a multiplexed display in which relatively few I/O pins on a microcontroller are used e.g. to drive an array of LEDs. It was invented by Charlie Allen IN 1995. It does this by utilizing the tri-state logic capabilities of microcontrollers in order to gain efficiency over traditional multiplexing.

As with any multiplexing, there is a requirement to cycle through the in-use LEDs rapidly so that the persistence of the human eye perceives the display to be lit as a whole. Multiplexing can generally be seen by a strobing effect and skewing if the eye's focal point is moved past the display rapidly.

Use cases

Charlieplexing is used to multiplex many output signals to relatively few GPIO on a microcontroller. This can be used to power output devices like LED’s, piezo buzzers, etc. Note that when charlieplexing anything other than LED’s, you will need to connect a diode in series with the device (say a piezo buzzer) to control current flow.

Charlieplexing can also be used to multiplex digital input signals into a microcontroller. The same diode circuits are used, except a switch is placed in series with each diode. To read whether a switch is open or closed, the microcontroller configures one pin as an input with an internal pull-up resistor. The other pin is configured as an output and set to the low logic level. If the input pin reads low, then the switch is closed, and if the input pin reads high, then the switch is open. One potential application for this is to read a standard (4×3) 12-key numeric keypad using only 4 I/O lines.

Implementation

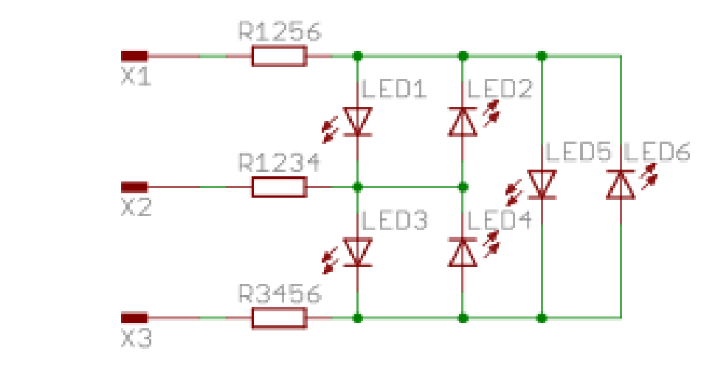
The charlieplexing technique is based on the concept of tri-state logic. Tri-state logic means that the pins of a microcontroller can be in three possible states. High, Low and Input. The Input state is known as the high impendence state. This pin state disconnects the pin from the circuit, even while other pins of the microcontroller are till operational.

The most common use case of charlieplexing is to drive many LED’s so we will be using this use case for our case study. The formula for charlieplexing multiple LED’s is

npins =n(n-1)LED’s

This formula give you the required pins for a giving number of LED’s

Charlieplexing is made possible because of the characteristic of diode to allow current to flow in only one direction. An LED also possesses this characteristic. This means that if these LED’s are arranged in the correct order. Each LED can be lit individually by just changing the current flow. Lets look at an example.

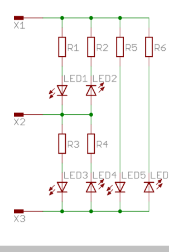
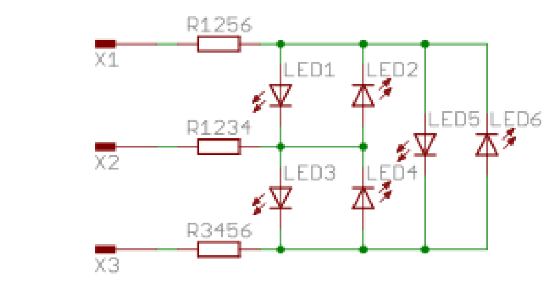


Let’s consider LED 1 in the circuit above. If we wish to turn it on, all we need to do is set pin X1 high,pin X2 low and pin X3 to a high impedance state of Input. This means that any led in the circuit (or any circuit) can be turned on be setting the correct pins “high” and “low” and setting the remaining as “Inputs”.

When designing your circuit, there are two possible resistor configurations. Placing you resistors in series with you GPIO. This is done if all your LED’s are the same. Meaning they have the same nominal current.

Placing the resistors in series with each LED if the nominal current is different for each LED.

Don’t forget that you have to calculate the resistance required for each use case. Example schematic below.



Limitations

Peak current:

Due to the decreased duty cycle, the current requirement of a Charlieplexed display increases much faster than it would with a traditionally multiplexed display. As the display gets larger, the average current flowing through the LED must be (roughly) constant in order for it to maintain constant brightness, thus requiring the peak current to increase proportionally. This causes a number of issues that limit the practical size of a Charlieplexed display.

Forward voltage:

When using LEDs with different forward voltages, such as when using different color LEDs, some LEDs can light when not desired. This issue can easily be avoided by comparing forward voltages of the LEDs used in the matrix and checking for compatibility issues. Or, more simply, using LEDs that all have the same forward voltage. This is also a problem where the LEDs are using individual resistors instead of shared resistors, if there is a path through two LEDs that has less LED drop than the supply voltage these LEDs may also illuminate at unintended times.