lab 2
Getting started with GPIOs

M. Briday

October 15, 2021

1 Objective

The objective of this first lab is start an application development process by trying to stay as close as possible to the hardware.

This lab is split in 2 parts:

- The first one focuses on getting started with a very simple application to use debugging tools (gdb integration).
- The second part focuses on the development of a driver to control a set of leds that are connected using the charlieplexing method.

2 Getting started

Some external files give information on:

- Setting the compiler:
  `https://gitlab.univ-nantes.fr/briday-m/itii-mac-tp-etu/-/blob/master/doc/compiler.md`
- Qt Creator (GUI) configuration:
  `https://gitlab.univ-nantes.fr/briday-m/itii-mac-tp-etu/blob/master/doc/qtCreator.md`
- The Coro lab board characteristics:
  `https://gitlab.univ-nantes.fr/briday-m/itii-mac-tp-etu/blob/master/doc/labBoard.md`

Note: On Macs in `D102`, the tools are installed (compiler, debugger, stlink, cmake, qtcreator ide), but the QtCreator configuration should be done!

All the specific files are in the `sys/` folder:

- `startup_ARMCM4.c` the startup file (define the interrupt vector and the reset Handler)
- `startup_clock.c` configure the clock tree at startup.
- `stm32f303K8.ld` is the link script, used by the linker to allocate the code/data in memory
- `CMSIS/Device/ST/STM32F3xx/Include/stm32f303x8.h` contains the symbol definitions provided by ST for this chip. This is a 12000 lines files… but the GUI can parse it and help you with code completion.

Question 1 *update the squeleton provided so that the led is light on only when the button (D6) is pushed. Your peripheral accesses should use symbol definitions.*

Question 2 *add a breakpoint in your application, just after the C source line that lights the led on. Try to use the integrated debugger to stop the application, update variables and GPIO state.*

Question 3 *update your application so that the led is toggled each time we push the button. You may use a non blocking finite state machine (with states for instance* ***RELEASED****,* ***PUSHING****,* ***PUSHED*** *and* ***RELESASING****), as chown at the end of the GPIO chapter.*

3 Charlieplexing

Charlieplexing[1] is a 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.

The lab board has 6 leds `L0` to `L5` connected to 3 I/Os, using the schematic in figure 1.

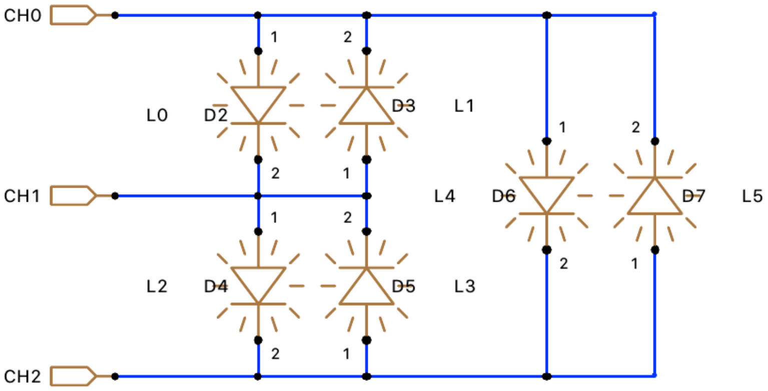

Figure 1: Schematic of the Charlieplexing leds of the lab board

[1] `https://en.wikipedia.org/wiki/Charlieplexing`

Only one led can light on at a time, and we have to use the 3-states outputs of the MCU pins (0, 1 or high impedance = Z)[2]

Question 4 *What is the truth table of the system, with the 3 inputs from the MCU, and the 6 output leds?*

3.1 Driving a single led

the following function that can drive a single led:

```
//Just light on a single led
void setLed(int ledId);
```

To test the function, we can write a simple application that lights on the next led each time the button is pushed.

Question 5 *implement and test the basic charlieplexing application. Take a particular attention to check the arguments at the start of the function (few values are OK for `ledId`).*

3.2 Driving the whole set of leds

We now want to make a fast scanning function that allows the display of several LEDs at the same time, thanks to the retinal persistence.

The function will be:

```
//drive many leds at a time.
void charlieplexing(uint8_t mask);
```

The first low 6 bits of `mask` are associated to the state of one led. This function should call the `setLed()` function many times, but should not access the hardware directly.

The `charlieplexing()` function should be called very often in the main loop of the application. It may be implemented in 2 ways:

- the function updates all the leds at each call;
- the function updates only one led at each call (it will result in a better display quality, as each led will light the same time if the loop duration is constant).

Question 6 *Make a simple application that lights all the leds except `L0` in the setup, and:*

- *shift the led off to the right when we push the button D6*
- *shift the led off the left when we push the button D5*

[2] To get the Z state, we can either configure the pin as an open drain output 1, or as an input.