LPC4088 LED Device Driver

CM0506 Small Embedded Systems

Dr Alun Moon

Seminar 5a

LED Driver

Without an operating system, there is not the need for the complexity of a Hardware Abstraction Layer. It can be argued that since the the hardware of the embedded system is fixed at design time, the flexibility of the HAL is not needed.

We still need an API abstracting the LEDs behind a device driver. That way we can keep all the hardware dependencies behind the API. A clean interface, one not exposing any knowledge of the underlying hardware still needs to be developed.

Further more the device driver needs to be efficient at run time as it may be called as part of interrupt handlers.

Driver API

A driver API needs the following main features.

- A means of initialising the devices
- A means of controlling the devices
- A means of accessing their state.

The style used has an enumerated data-type, which gives logical names by which the LEDs can be referred to in code.

Question:
Why are there both `left_green` and `LED1` names defined?

```
enum LED {
 LED1, LED2, LED3, LED4,
 left_green=LED1, right_green,
 left_blue, right_blue
};
```

The functions to control the LEDs take this as a parameter,

```
led_on(enum LED name);
led_off(enum LED name);
led_toggle(enum LED name);
int led_state(enum LED name);
```

These, along with the prototype for `led_init()` can be put in the header `led.h`.

API and Driver Library

The two files `led.c` and `led.h` make up the device-driver library. The code for the library (`led.c`) can be distributed as source-code, or a compiled object file. The header file `led.h` defines the publicly visible part of the API and defines the functions and any useful constants needed by a program to use the functions provided by the driver.

Exercise 1: Git download of initial code

Retrieve the project from the GIT repository.

```
$ git clone https://github.com/dr-alun-moon/led-driver
```

Examine the files for the LED driver `led.h` and `led.c`.

1. Can you follow the way the code is structured?
2. Why are the SET and CLR registers used to turn different LEDs on?

LED circuits and effects on code. The green (LED1 and LED2) and blue (LED3 and LED4) LEDs are connected to the processor using different circuits (figures 1 and 2 [Emb, 2014b]).

LED 1 & 2 ARE CONNECTED AS PART OF THE USB SYSTEM. The circuit in figure 1 is controlled by the PNP transistor. This transistor is "on" when the voltage to the base is low. A logic 0 (LOW) switches on the device. This is why in the code, writing to the CLR register (clearing the bit) the LED turns on.

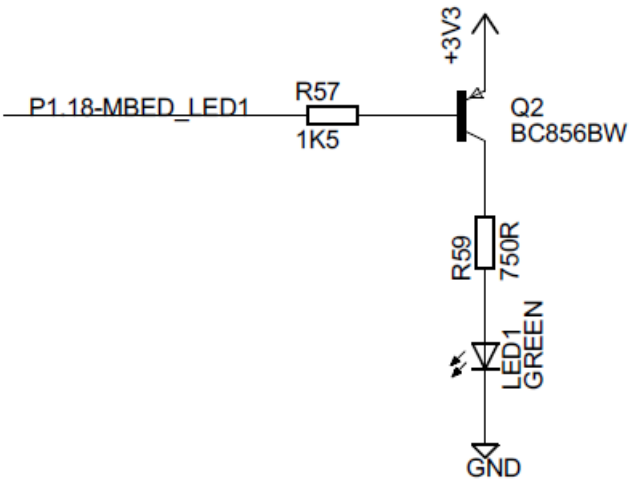

Figure 1: Connection for LED 1 & 2

LED 3 & 3 ARE CONNECTED TO OTHER GPIO PINS. The circuit is in figure 2. The LED has current flowing through it when the

connection is high. A logic 1 (HIGH) switches on the device. This is why in the code, writing to the SET register (setting the bit) the LED turns on.

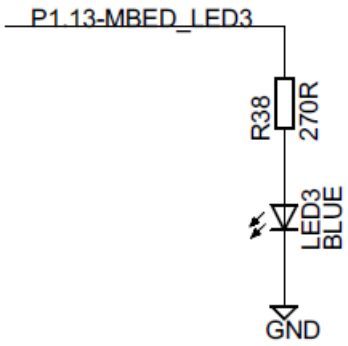

Figure 2: Connection for LED 3 & 4

Exercise 2: Extend the driver to Base-board LEDs

From the information in the Base Board schematic [Emb, 2014a], use the information to extend the driver to the Red, Green, and Blue LEDs. Things to note include:

- What Port is the LED connected to?
- Which Pin is the LED connected to?
- Is the circuit active-high or active-low?

Solution

A solution is in Git

```
$ git checkout solution
```

Question 1: enum and switch

In the code I've used a combination of `enum` for LED names, and then a `switch` to select which code to use. Why?

- What does the compiler say if the `switch` does not check all of the `enum` values?
- What does the compiler warn about in the interrupt handler `ledone`?
- Are these useful features?
- How efficient is the compiled code for a switch?

If you want to have a go at optimising the code, try it out. You will find that there is a tension between compiled code side and ease of writing the C code. In reality hand optimising is a tricky buiseness, and rarely done well.

References

LPC4088 Experiment Base Board rev A. Embedded Artists, September 2014a.

LPC4088 Quickstart Board rev B. Embedded Artists, August 2014b.