SLP-assignment #3: led

(14 points, in groups of two)

Reimplement the LED module of the `libspicboard` in the file `led.c`. The description of the interface can be found in the API documentation on the website:

`https://sys.cs.fau.de/lehre/ss24/spic/uebung/spicboard/libapi/extern/group__LED.html`

Additionally, you have to write a test program `test-led.c` that tests all public functions of the module.

LED-Module

- Make sure to stick exactly to the given interface. Therefore, you have to include `led.h` from the `libspicboard` in your file `led.c`.
- Further functions as well as global variables that are not defined inside this header have to be declared such way that their visibility is restricted to the module itself.
- Do not use `if`-cascades and `switch`-statements (or similar C constructs) for addressing the LEDs via the pins. Instead, you should use arrays (and possibly pointers) to determine the correct pin and port numbers for each LED. To do this, you can assume that the `LED`-enum type is enumerated from 0 to 7 (see `led.h`) and use it as an array index.
- Pay attention to the initialization of the I/O ports that has to happen in advance to their first use. This initialization, however, should be hidden in the module (like for the original).

Test Program

- Your test program should check all functions of the LED module. Do not forget to also check the behaviour for wrong inputs. All returned values have to be checked by machine.
- The test program should test a representative choice of parameters for each function. Remember to include edge cases (e.g. no LEDs, all LEDs, etc.).
- Also check how your test program behaves when running it with the reference implementation from the `libspicboard` and compare the results.
- For the test program you are free to use other modules from the `libspicboard`. E.g., you could use the ADC module to read the photosensor and represent the brightness with the LEDs as a level indicator.

Hints:

- As for this task, multiple source files have to be combined to a single executable file, you have to pay close attention to the correct naming of the files (`led.c` and `test-led.c`). Otherwise, the underlying build system will not work as intended.
- The linker will always use the "local" functions that are defined in the local source file. Only if a function is not defined there, it searches the library for the function. To test the functions from the `libspicboard` instead of the function from your `led.c`, you can comment out you own implementation e.g. by

  `#if 0`

  at the beginning of the file or before the function and

  `#endif`

  at the end of the file or after the function. You can reactivate your implementation by setting the `0` to a `1`.
- Always give a reason why you use the `volatile` keyword. If the same reasoning holds for multiple variables, you can justify them together.

Deadline

T01	19.05.2024	18:00:00
T02	19.05.2024	18:00:00
T03	20.05.2024	18:00:00
T04	21.05.2024	18:00:00
T05	21.05.2024	18:00:00
T06	22.05.2024	18:00:00
T07	22.05.2024	18:00:00
T08	23.05.2024	18:00:00
T09	20.05.2024	18:00:00