Electrical and Computer Engineering
Aristotle University Of Thessaloniki

Microprocessors and Peripherals
Report 2

Meliopoulos Marios #9680
Papadakis Konstantinos Fotios #10371
16/5/24

1.1 Assignment info

This assignment is implemented in a main.c file where all the necessary methods and processes take place, along with the complementary drivers(.h and .c files) provided by the instructors.

1.2 Problems

Some of the problems during this assignment included difficulty getting started since the coding relied heavily on functions provided in external libraries which we missed in the beginning. Searching on Google or LLMs to find possible solutions for problems was also quite ineffective. This led to a medium floor level of difficulty. Once we were able to understand the different functions provided in the libraries the assignment difficulty dropped significantly.

Another small problem was that the function *gpio_toggle* was not working with no clear indication as to why. We were able to find a workaround by changing the function a bit, making it work in two steps instead of one, but this most likely was a problem coming from the use of debug mode. After a while, we reverted to the original function and it was working without a hiccup.

Another issue was that sometimes we observed irregular interactions between our computer and our microprocessor. Most of the time unplugging it and then plugging it back in was the solution

1.3 Testing

To test the behavior of our code we tried to give different inputs in every possible combination to make sure that the results were correct and consistent. These combinations included

Providing an odd number as input, providing an even number as input, pressing the button with no input, pressing the button with even numbers as input(while the LED was off and while the LED was on), pressing the button with odd input while the led was flashing on and while it was

off, giving odd/even input then pressing the button then giving the opposite input and pressing the button again.

In all of these test cases, the program was running correctly.

2.1 UART input

We initially modified the range of input types of characters that can be given as input to include only numbers, backspace, and enter. On the bottom of the given UART code, we also added a check for when no number was given. If the input is valid we extract from *buff* the last digit of the input to use it in later steps.

2.2 Odd input

The code finds whether the last digit of the given input is odd or even. If it is odd a timer is enabled which we had initialized at 500000us. Enabling the timer calls the function *timer callback* every 0,5 seconds, as seen at the initialization of the timer, which toggles the board's LED at that period.

2.3 Even input

If the last digit of the given input is even, the *timer* is deactivated and the LED light remains in its current state.

2.4 Button setup and ISR

To set up the button we set the pin associated with the button as an input and in pullup mode. When that pin signal is rising an ISR is called which updates a counter that counts the number of times that the button has been pressed, then calls a function that toggles the LED, and finally calls a function that prints the number of times that the button has been pressed.

2.5 Toggle led

The function that toggles the LED simply toggles the value of the pin associated with the LED light and then calls a function that prints the current state of the LED.

2.6 Print button

The method that prints the number of times that the button is called turns the value of the corresponding counter into a string and then prints it via UART.