Lab 1: Toggling LED using GPIO with/without low power features

組別: 鎂鋅藥酒

All of the following experiments are using the board Nucleo-F401RE.

Part 0: Computing square root using assembly only

Using Keil μVision, we learned how function calls and context switches are achieved in the assembly level, and that how to achieve higher order abstraction using these as build blocks.

Part 1: Using GPIO to toggle the LED on the development board

Through practicing with GPIO, we learned how an embedded system could interact with its peripherals: First, because that current embedded systems by default have many power-saving features enabled, that is, the pins are “turned off” and we are ignorant of them, we have to toggle its corresponding clock on, s.t. it’s possible for us to receive or send signals through that pin. Secondly, we have to set proper modes for the register, e.g. the register controlling the LED in this lab, s.t. the recognition/reproduce of the signal would be more stable.

Part 2: Using interrupt to toggle LED in low power mode  
 The previous example would do the job, but by doing so we consume too much energy: we simply put it in an infinite loop, checking the register again and again without halt. In this lab, we learned a way to address this issue: by utilizing the WFI (wait for interrupt) feature on the MCU, we could put the MCU in sleep mode, waking it up only when some interrupt occurs. In fact, this kind of technique could be categorized into what’s so called “event-driven programming”, that is, the current behaviors would mostly depend on outside events. In the embedded system case, we could put the system into power-saving mode, only registering what we shall do later and handle tasks that’s of higher priority, leaving most of the computational job for some time when there’s enough energy supply.

Optional: Using mbed API to implement the previous functions

Mbed library may seem very troublesome at first when you don’t have any document at hand, but since they provide nice abstraction over assembly language and miscellaneous machine-specific registers, it turns out to be a very powerful tool and could greatly accelerate the developing schedule. Most importantly, it makes the code much comprehensive, which we much appreciated while reconstructing the previous problems with it.