EEE3096S - Practical 3
2023

BCOS 2 - ADCs, PWM and LCD

1 Overview

1.1 Design overview

Many embedded systems rely on the use of analog devices for various reasons. Certain sensors, for example, output an analog voltage or current which the digital brain of the system needs to be able to read. Similarly, embedded devices often allow users to adjust parameters by turning the knob of a rotary potentiometer. In order to be able to process these analog signals we need to make use of an Analog to Digital Converter, or ADC for short.

In this prac, you will be using C to read the voltage on a potentiometer's wiper using the onboard ADC on the STM32. You will then use that to calculate the duty cycle of a PWM signal which will be used to control the brightness of a built-in LED (pin PB7). You will also use a pushbutton interrupt to control when certain events occur and make use of the built-in LCD screen for outputting text and values.

All of the code needed to initialize the peripherals used in this practical is automatically generated by STMCubeIDE using the .ioc file; as such, you only need to edit the main.c file in the Core/src folder. If you want to see how the peripherals were configured, you can open up the STM configuration GUI by double-clicking on the .ioc file included in the project folder, but do NOT make any changes (or re-generate the code using the .ioc file) as this will cause problems with your main.c file.

2 Outcomes and Knowledge Areas

You will learn about the following aspects:

- ADCs
- PWM
- LCD
- You should acquaint yourself with the STM32 HAL/LL documentation, available here.

3 Deliverables

For this practical, you must:

- Submit a **PDF** of your main.c code on Amathuba/Gradescope. Do this by copy-pasting your code into a document and saving it as a PDF; do NOT submit screenshots of your code as the text needs to be searchable.
- After you have submitted your PDF, push your code to your shared repository on **GitHub** and then **demonstrate** your working implementation to a tutor in the lab; they will open your submission on Amathuba and mark your demo in real time.
- You will have **two weeks** to complete this practical, and as such, you will be allowed to conduct your demo during any lab session within these two weeks. By the deadline, you should have submitted your PDF, pushed your code to GitHub, and had your demo marked.

4 Hardware Required

- UCT STM Board

5 Walkthrough

1. Clone or download the git repository.
   $ git clone https://github.com/UCT-EE-OCW/EEE3096S-2023
2. /EEE3096S-2023/WorkPackage3/Prac3_student is the project folder that you will need; open up STMCubeIDE and then go to File --> Import --> Existing Code as Makefile Project --> Next --> Browse to the project folder and then select "MCU ARM GCC" as the Toolchain --> Finish.
3. Open up the main.c file under the Core/src folder and complete the Tasks below. Note: All code that you need to complete is marked with a "TODO" comment; do not edit any of the other provided code.
4. T**ASK 1**: For the first task you need to switch the frequency of the flashing LED (the leftmost LED, pin PB7) between 1 Hz and 2 Hz when Push Button 0 (pin PA0) is pressed. The function EXTI0_1_IRQHandler is called when an interrupt is generated on EXTI0 (i.e. PA0); complete this function's body to implement the switching of delay frequencies. Note that the function has already been declared and an empty function body has been provided.
   You may notice that occasionally the frequency does **not** seem to change when the button is pressed; explain why this happens and think of a way to fix the issue. HINT: The HAL GetTick() function may be useful here.

5. **TASK 2**: With our potentiometer (PO1, pin PA6) and the ADC already configured, your next task is to complete the functions pollADC and writeLCD, and then add code into the main while loop to read the ADC value and print it to the LCD. The function to start, poll and stop the ADC (and return the most recent value) has already been provided. HINT: Read the documentation of the HAL libraries, as well as the LCD library's C file (under folder Core/Inc), to get an idea of how to do this.
6. **TASK 3:** The function ADCtoCCR converts an ADC value to a PWM duty cycle value (or rather, the CCR (Capture/Compare Register) value that needs to be set for a specific duty cycle). Since the ADC is configured in 12-bit mode, the function's input would be integer value between 0 and 4095. TIM3 is configured to use channel 3 (pin PB0, our rightmost LED) with the prescaler set to 0 and the ARR (Auto Reload Register) set to 47999; this corresponds to a 1 kHz frequency for the PWM signal, with Duty Cycle = CCR /ARR.
Using this information, change the zero return value of the ADCtoCCR function to an appropriate equation and then use this in your main while loop to update the CCR based on the ADC value from POT1. Turning the POT should thus adjust the value on the LCD *and* the brightness of LED0.
7. **TASK 4**: Demonstrate all your working peripherals/code to a tutor. Your completed program should:
   - Toggle the delay frequency of LED7.
   - Poll the ADC (POT1).
   - Output the ADC value to the LCD screen.
   - Calculate the CCR based on the ADC value.
   - Set the PWM duty cycle (i.e., the brightness) of LED0 using the CCR.
8. Upload your main.c file to your shared repository. Your file structure should resemble this: STDNUM001_STDNUM002_EEE3096S/Prac3/main.c (assuming your shared repository is called STDNUM001_STDNUM002_EEE3096S).

6 Submission

A PDF submission is required for this practical:

1. The PDF document must comprise your main.c code in **text form**, not screenshots.
2. After submitting the PDF, demo the aforementioned functionalities to a tutor in the lab. Your PDF **must already be submitted** by the time you call a tutor to your bench for the demo, as they will then be able to open your submission on Amathuba/Gradescope and assign marks accordingly.
3. All marks for this practical will come from your demo and your submitted code.