Metropolia

Exercise - PWM and Interrupts

Instructor: Antti Piironen

Equipment

In this exercise you will need the following equipment

- CY8KIT-059 PSoC5 prototyping board
- Servo motor, preferably a low power (SG90 or similar)

Setup

Create a PSoC project for CY8KIT-059 kit by using the previous exercise as a template:

- **OPTIONAL:** Open the Exercise 1 project, and save it as a Template (Project->Copy to My Templates)
- Create a new project from the previous template:
  - New project directory
  - Rename of the project = “FirstnamePSoC5Servo”
- Clear main.c from any extra code, except for the loop-forever and calls to initialization routines.
- Configure the PWM Component:
  - PWM output should be connected to pin P2.0 to control a servo motor.
- Modify the Clock Component for PWM usage
- Don’t make any wired connections until you are ready to test your system.

Task

Your task is to design the system to control the movement of the servo based on a PWM Component:

- The system should change the position of the servo on each key press.
- Initially the system starts from a neutral position (1.5ms pulse width).
- Each key press causes the servo to change the position (min of 5 different positions).
- Key press must be detected using interrupts.
  - For help on PSoC5 Interrupts read AN54460
- **To get maximum points** from this exercise, the servo must change the position smoothly. This can be done by using interrupts:
  - Enable PWM interrupts
  - Write an interrupt service routine for PWM such that the button press changes only the target where the position should be set, but the PWM ISR will increment/decrement the PWM width gradually

Metropolia

    - Remember to clear PWM interrupt by reading the status register
- Hint: Do not try to do all functionalities at once. Work gradually and test each added feature before moving to the next feature. It is also a good idea to keep different versions in case you want to step back and start over.

Exercise report return

All program files (of your own… main.c etc.) must be documented appropriately. Minimum requirement: each file starts with a commented area with description, author, date, and course ID. The fundamental principle is that anyone with some programming background can understand how the code works.

Use descriptive names for all added modules and user defined signal pins.

Use the exercise return submission form (unless instructed otherwise) and provide:

- **Name**(s). If you work in a group, names of all members who contributed to the project.
- **Email** address to ensure that I can contact you in case the submission goes wrong.
- **Packed PSoC Creator project**:
    - ZIP: Open file browser, go to your PSoC working directory, select the whole project directory, and zip it.
        - You can return the zipped project
            - **either** as a link to a cloud service.
                - Upload your zip-file to your favourite cloud repository
                - Provide a link to your zip-file. .
            - **or** as a zipped file (and write “attachment” to the form)
    - Github: share the project with account “antti.piironen@metropolia.fi” in case the instructor is Antti Piironen.
        - In case your instructor is Kimmo Sauren then you should share with “kimmo.sauren@metropolia.fi”.
    - If I cannot open your project, I will send you a message, and you lose 1p.
- Link to your own code of the project:
    - Create a Google Docs (or some other shareable document file), and share it such that “anyone with the link can comment” or share it with “antti.piironen@metropolia.fi” (or “kimmo.sauren@metropolia.fi”).
    - Copy-paste all of your own code (**`main.c`** and other functions you wrote yourself) to the document file.
    - This way I can give you advice later on. If I cannot open or make comments to your code, you will automatically lose 1-2 points.

Metropolia

Use this form to return your exercise.