

# Deliverable 1: Morse Time

## 1 Introduction

As an additional opportunity to consolidate your skills, we offer four programming exercises. They make up 20% of your overall grade as outlined in the first exercise session. You are encouraged to discuss approaches and share experience with other students. But as it affects your grade, your submission must of course be your own original work.

The submissions are graded fully automatic once per hour after the grading machine is set up (usually a day or two after the release date above). Together with your achieved points you will be provided several logfiles which explain why your solution got this amount of points. You can upload a new solution as often as you like during the submission period. Should you find evidence that the grading does not follow the problem statement set out below, please contact us as soon as possible, so we can look into the problem and fix it before too many other students are affected.

## 2 Problem Description

### 2.1 Part A

In this part you have to write code that is able to send a given string in Morse code via LED1. You probably want to use one of the example projects as a starting point, rename it to **PartA**, and replace the example code with your solution.

Use the current ITU standard M.1677-1 for details on the Morse Code. You do not have to implement punctuation characters. Letters and spaces according to Sections 1.1.1 and 1.1.2 are sufficient. The length of a dot has to be  $100\text{ ms} \pm 10\text{ ms}$ , confer Section 2 of the standard for the length of the other signals. You can verify the length e.g. by checking the time it takes to send a certain message divided by the length of the message in number of dot equivalents. Send the string in the most simple form, i.e. not transmitting a wrong character, then sending the correction signal and then the correct character. Instead of the general transmission rules in Part II of the standard, send out the string repeatedly while the device is on, without any calling or invitation signals but a pause of 5 s after each transmission.

The string to send out is: **I CAN MORSE**

### 2.2 Part B

Copy the project from Part A, call the copy **PartB** and extend it as follows:

- The string is sent once after each press of Button1, not continuously.
- If Button2 is pressed, the time between the last and second last press of Button1 in milliseconds is sent once in Morse code via LED1.

- If Button1 has not been pressed yet, send a single 0 digit.
- If Button1 has only been pressed once, send the time between boot and the first press of Button1.
- The time has to be sent in decimal notation without leading zeroes or decimal point. Round off to full milliseconds.
- While a transmission of Morse code is ongoing, any button press may be ignored.
- If both buttons are pressed simultaneously, the one pressed down first wins.

Make sure measuring the time between button presses does not interfere with your transmission in Morse code. There are several solutions to this, e.g. using the SysTick or the CCU4.

### 3 Hints

- Check the Board User's Manual of the XMC Relax Kit (not the reference manual of the XMC4500) for the GPIOs connected to the buttons
  - You are supposed to find the correct Board User's Manual on your own, try searching the internet.
- The buttons on the XMC Relax Lite Kit are equipped with RC filters for debouncing

In case you opt for the CCU4 in Part B, here are a few hints:

- Do not be overwhelmed by the complexity of CCU4, you only need few of its features
- Remember how we used the CCU4 for PWM dimming in problem 2.4
- Use the search function of your PDF viewer to look up register names
- Cf. Table 25-12 to see which slice of which CCU4 instance can be connected to LED1
- Remarks on section 22.6.1 of the manual:
  - The XMC4500 boots with reset already applied to all peripherals
  - GCTRL is fine with its initial 0x00
  - Configuring the timer slice confines to setting up prescaler, period and compare value then requesting a shadow transfer
  - Step 7 is not needed
  - As we do not need to start timers synchronously, Step 9 reduces to setting the run bit of the respective slice
- Do not forget to connect the GPIO to the output of the CCU4, cf. Table 25-5
- The drivers in `xmclib` may ease one task or the other
- Structure your program wisely

## 4 Submission

### 4.1 What to Submit

- A ZIP archive containing both projects including all subfolders.
  - I.e. your ZIP has to contain two project folders in its root, named `PartA` and `PartB`.
- Make sure the `*.elf` files in your project are up to date, since they are used without modification.
- If your submission does not contain `*.elf` files, your project will be rebuilt, which adds a point of failure.

### 4.2 How to Submit

1. From any of your project directories (e.g. directory `PartA`) run `make deliverable`.
2. Make will create a `.zip`-file in `../`.
3. Upload this archive via Moodle.