# Answer Sheet

*This sheet should be printed out and handed in during the lab session. It can be completed either electronically or by hand.*

|  |  |
| --- | --- |
| **Name** | Bruno Luiz da Silva |
| **Student number** | cb150724708 |
| **Date submitted** | 30/10/2015 |

## Questions from The Lab

|  |  |  |
| --- | --- | --- |
| **Question** | | **Answer** |
| 1 | How many tasks are there in the provided program?  Do these tasks interact or run independently? If they interact, explain how. If they do not, explain why no interaction is needed. | Two tasks: task1FlashRed and task2FlashGreen  They run independently and they don’t need interaction as each one is self-contained: one is meant to blink the red LED and the other to blink the green LED. |
| 2 | In task 1, what proportion of the cycle is spent executing user code, rather than waiting? | Oscilloscope: 2.15 us was spend executing the user code  Checking the assembly code: 62 operations 🡺 62/(4800000) = 12.08 us |
| 3 | In task 2, what is the interval between the falling edge of the button press and the PIO going high?  Use this to estimate the time before the interrupt handler starts. Explain your estimate | Oscilloscope: 1.56us  The wave output is from the button press until the output goes high, which activated in the beginning of the interrupt handler. We can consider that the uC take half of this time (0.75us) to activate the interrupt. |

## Viva Record

Task 1: Demonstrate the use of the oscilloscope to time the cycle.

Task 2: Demonstrate the use of the oscilloscope to time the interrupt.

Task 3: Demonstrate the complete reaction timer.

|  |  |
| --- | --- |
| **Viva comment (completed by TA / lecturer)** |  |
| **Name:** |  |

## Question about concepts

Answer the following questions concisely (up to 30 words each):

1. What is an interrupt?   
   It is a signal sent to the uC that stops the current task to execute some critical task at a defined handler  
     
     
   Explain the advantages and disadvantages of using interrupts rather than polling to handle an inputs.

It enables the developer to execute critical tasks before the next iteration of the pooling cycle. The disadvantage is that you have a limited number of IRQ handlers.

1. A system has 2 buttons and 3 LEDs. The first button toggles between two flash rates – fast and slow; the second chooses between a 3 colour sequence (only 1 LED on at once) and 7 colour sequence (multiple LEDs may be on at once). It is proposed to implement this system using a cyclic design. Consider whether the code can be divided into multiple tasks. If you think it can, suggests the tasks, outlining i) what each task does ii) what variables it needs iii) how it communicates with other tasks. If you think de-composition is not possible, explain why.

There is a global struct called Settings to make the inter-task communication possible.

The two import proprieties of the struct are:

- speed: save the choosen speed (based on the user config with the button)

- colourMode: save the choosen colour mode (based on the user config with the button)

- speedToogle() and colourModeToogle(): just toogle the value of the above properties

Three tasks system:

- buttonColourModeTask: checks the if the button for colourMode was pressed

# Write: - settings.colourMode

- buttonSpeedTask: checks the if the button for speed was pressed

# Write: - settings.speed

- ledDemoTask: demonstrate the LED depending on the user config

# Read: - settings.colourMode to define which function to call

# ledPerLedDemo() and multipleLedsDemo()

# Read: - settings.speed to blink it faster or slower

I choose pooling instead of an interruption system because, for me, the button press is not so critical to use the available interrupt handlers. If you need to add more critical tests to the project, the handlers will be available for those.

## Feedback

|  |  |  |
| --- | --- | --- |
|  | | |
| **Marker** | **Date** | **Grade** |
|  |  |  |