# 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** | 09/10/2015 |

## Questions from The Lab

|  |  |  |
| --- | --- | --- |
| **Question** | | **Answer** |
| 1 | Explain how the value written to the GPIO bit used to turn on the external LED differs from that used to turn on one of the on-board LEDs (as in lab 1) and why this difference occurs. | We use SET and CLEAR registers instead of writing directly what we want as output |
| 2 | Describe the behaviour of the button from the user's point of view, in the given code. Does the system always respond to a button press? Does it respond immediately? | When the user presses the button, the idea is that it stop the LED ON by 500ms, but using the original code it doesn’t answer immediately. Sometimes you have to hold a little to make it work properly. |
| 3 | Explain briefly how your design (i) achieves an ‘immediate’ response to the button (*what length of time is ‘immediate’*); and (ii) avoids a single physical button press being counts more than once. | i) Before the new code, the uC locks itself in one loop to let the light ON in certain amount of time, due to the function Delay(x).  To solve that, I created a task which increments a variable in each loop, without any Delay() call. This will left the uC free for other tasks.  Each time the loop passes by this var, it checks if it is equal to the aimed threshold (blink\_time) and, if it is, invert the the actual LED state (on/off). For the user it will appears to be a delayed LED for X amount of time because the blink\_time is defined as:  Which  ii) Create a buttonPressed variable which is set when the button is pressed and un-set when the button is left. This ensures that, if the button is holded by the user, the program will not be hold or reseted multiple times because of it. |
| 4 | The diagram in section 2.4 shows that each button press may lead to multiple levels changes (button bounce). Explain how this could prevent the program working correctly. | This could register various button presses, due to multiple high and low levels in a very small amount of time. |

## Viva Record

1. **Task 1**: Demonstrate external LED flashing
2. **Task 2**: Demonstrate use of the button to vary the rate of flashing
3. **Task 3**: Demonstrate the use of the oscilloscope

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

## Question about concepts

Answer the following questions concisely:

1. Answer the following questions about polling an input. [***5 to 20 words each***]
   1. What is meant by polling?

Is a method which the processor keep checking the bits/register/variables/devices constantly, using normally a infinite loop for it (example: while(1))

* 1. What determines the polling rate (or frequency)?

In this case, the polling rate is ruled by the Delay function. But, at low-level, it depends on the processor clock + how much ticks you want to wait before the next cycle.

1. Answer the following questions about oscilloscopes. [***5 to 20 words each***]
   1. What does an oscilloscope display?

The wave form in the selected channel

* 1. What is meant by a ‘trigger’ and why is it needed?

The trigger is a way to stabilize the output, making the image steady and useful for further analysis.

## Feedback

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