EEET2481 – Assessment 2 (20%)

Timers & Interrupts for NUC140 MCU

# Objective

For this assessment, students will work in pair to carry out the following exercises. By the end of the assessment, students should get familiar with using Timers and Interrupts functions of NUC140 MCU.

# Exercise 1 – Timer (3 marks)

Write an embedded software program to generate a 2.4 kHz clock signal at the GPIOB.11 pin of the NUC140 MCU using Timer Controller.

Chart

Description automatically generated with low confidence

Your program needs to also satisfy the following specifications:

|  |  |
| --- | --- |
| CPU clock frequency | 22.1184 MHz |
| Timer Controller | Timer 3 |
| Timer Controller clock frequency | 50 kHz |

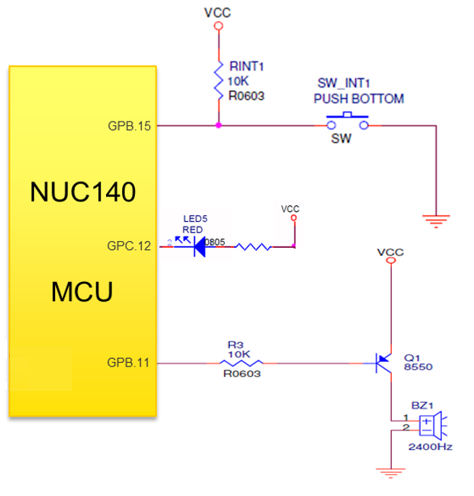
*Note – you need to decide what clock source (which is not necessarily to be CPU clock frequency) and prescaling required to have the Timer 3 clock at 50 kHz.*

Submission for this Exercise includes:

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. A screenshot of your waveform showing the correct output clock frequency. You can use either OSC or Virtual Bench to measure the timing.
3. A short clip (not longer than 2 minutes) in which one of you will present the system and explain the key configurations that must be done for CPU, GPIOs, CLOCK, TIMERs (and other relevant system and peripheral devices in the NUC140 MCU if any) and how your program works in general.

# Exercise 2 – External Interrupt (5 marks)

Write an embedded software program developed for hardware in the following figure:



Your program needs to also satisfy the following specifications:

* After system reset, both LED5 and the buzzer BZ1 is OFF.
* When a user presses then releases the push-button SW\_INT1 **three** times, the LED5 will be constantly ON/OFF after every 0.5 second.
* When a user presses then releases the push-button SW\_INT1 **five** times, the Buzzer BZ1 will constantly beep at 2.5 kHz frequency.
* When a user presses then releases the push-button SW\_INT1 **eight** times, both LED5 and the buzzer BZ1 is OFF.

The embedded software program must use GPIO Interrupt to detect the press-then-release events at the push-button **SW\_INT1** and proceed accordingly.

Submission for this Exercise includes:

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. A screenshot of your waveform showing the correct output clock frequency.
3. A short clip (not longer than 3 minutes) in which one of you will present the system and explain the key configurations that must be done for CPU, GPIOs, CLOCK, TIMERs, INTERRUPT (and other relevant system and peripheral devices in the NUC140 MCU if any) and how your program works in general.

# Exercise 3 – SysTick Interrupt (2 marks)

For this exercise, you need to write a program in which we will use SysTick Timer Interrupt to generate a delay of 1 second. To verify, we will use the Interrupt to toggle an LED with the frequency of 0.5Hz (1 second ON, 1 second OFF).

You need to research on what setup is required for enabling SysTick interrupt and what ISR required to process SysTick request.

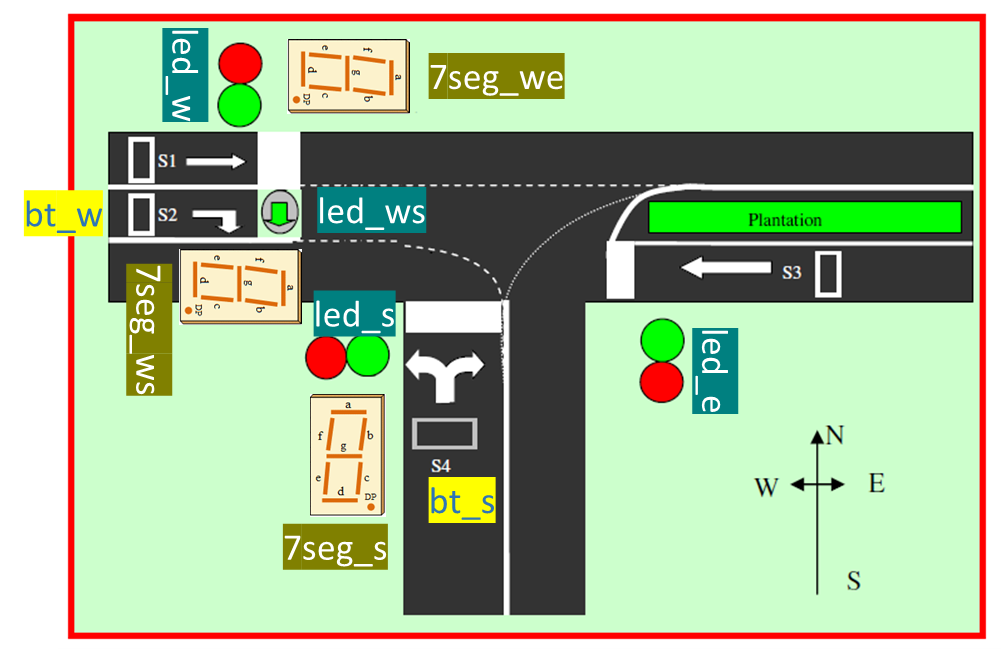
Submission for this Exercise includes:

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. A screenshot of your waveform showing the correct output clock frequency.

# Exercise 4 – Traffic Light Controller using Finite State Machine Approach (10 marks)

*Note – this exercise was part of Lab 2 in EEET2505 from 2021C semester. For students who did that lab, you might find some similarity here. For those who have not done that lab, you can treat this as a brand-new exercise.*

For this exercise, you will be working with a traffic light controller, and then implement it on **NUC140 Board,** for a side road entering a busy road running east west. The diagram of the intersection can be seen as follows



As there will be less traffic entering or leaving the side road hence it is given with a lower priority. Therefore, your controller needs to control this scenario (according to the priority level)

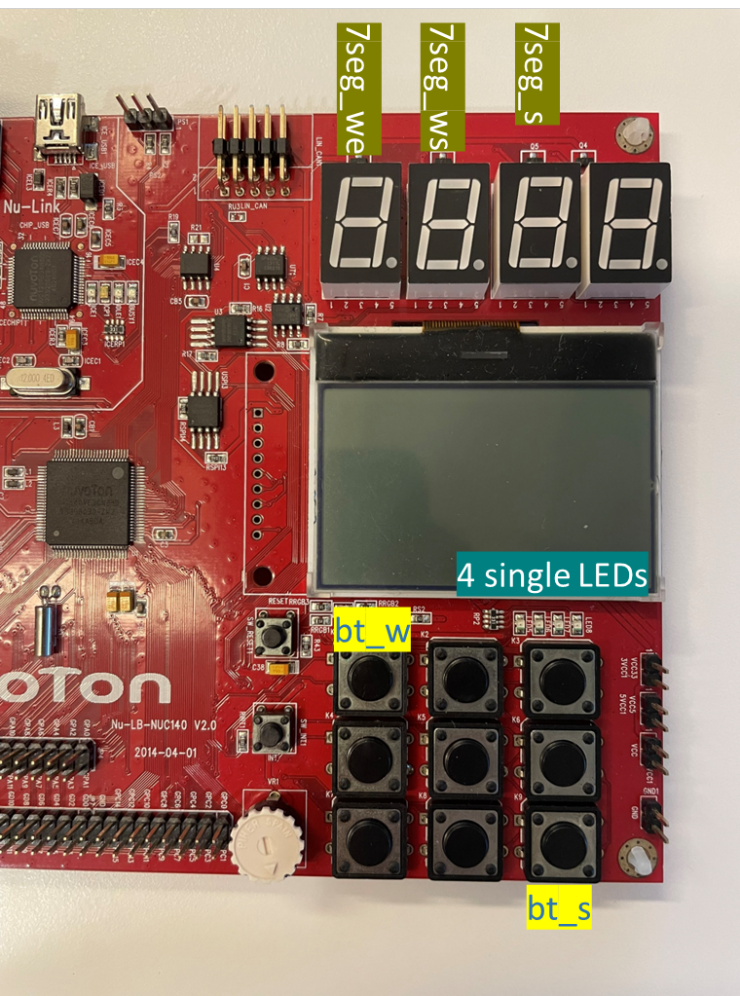
1. Normally East-West (EW) traffic on S1 and S3 lanes will go as usual back and forth. **Time for Red/Green is 3 seconds**
2. If there are vehicles travelling from West to South (WS), they approach S2 lane, wait at the cross line, a sensor picks this presence. After the traffic from EW is clear, the turning right light will be ON. **Time for Turning right here is 2 seconds.** After this, the traffic backs to EW going straight.
   1. During the turning, traffic on both S1 and S3 lanes are stopped.
3. If there is a car from the south approaching the intersection, wait at the cross line and a sensor there captures this and raise the request to the controller. After the traffic from EW and WS if any, the controller will then allow the traffic from the south to go. **Time for vehicles travelling this direction is 2 seconds.** After this, the traffic backs to EW going straight.

For this exercise, you must use the Finite State Machine (FSM) approach in designing your controller. You must produce a state diagram and submit along with your work.

In term of pin assignment, the following signals and their respective locations on NUC140 Board can be found:

|  |  |  |
| --- | --- | --- |
| Signals | Description | Location on NUC140 Board |
| **led\_w** | Single LEDs to indicate the traffic light from the respective direction  LED ON -> Green Light  LED OFF -> Red Light | **GPC12 (LED5)** |
| **led\_e** | **GPC13 (LED6)** |
| **led\_ws** | **GPC14 (LED7)** |
| **led\_s** | **CPC15 (LED8)** |
| **7seg\_we** | 7semeng LEDs to indicate the elapsed time for the respective direction   * counting down from the maximum time down to Zero * when the direction light is RED, then the 7segment is OFF | **U11** |
| **7seg\_ws** | **U12** |
| **7seg\_s** | **U13** |
| **bt\_s** | Button used to replace the presence sensor, which detects vehicles travelling from the South | **~~K1 from button matrix~~**  **K9 from button matrix** |
| **bt\_w** | Button used to replace the presence sensor, which detects vehicles travelling from the West turning right to South | **~~K9 from button matrix~~**  **K1 from button matrix** |

You can also check out the following photo for more information



For timing purpose, you must use SysTick Timer code from the Exercise 3 earlier ( 1 second delay) as your way to keep track of time for traffic from all directions.

Submission for this Exercise includes:

1. Complete Keil MDK project which I can rebuild and re-run from my end.
2. A state diagram for the controller
3. A short clip (not longer than 5 minutes) in which one of you will present the system and explain the system operation.