

LABOraToRY 2

Embedded System Design and Implementation – EEET2481



Group *Group 19*

Students *Tuan Phung – s3761489*

*Nghia Huu Le - s3654028*

*Trung Nguyen - s3633958*

Lecturer *Andrew Smith*

Table of Contents

[Introduction 6](#_Toc7306356)

[Exercise 1 - Analyzing the Bouncing of a Push-Button 7](#_Toc7306357)

[Question 3 7](#_Toc7306358)

[Question 4 8](#_Toc7306359)

[Exercise 2 – Controlling an LED using a Timer and Interrupt 11](#_Toc7306360)

[Question 1 11](#_Toc7306361)

[Question 2 11](#_Toc7306362)

[Question 3 11](#_Toc7306363)

[Question 4 12](#_Toc7306364)

[The highest user configurable priority is denoted as “0” and the lowest priority is denoted as “3”. 13](#_Toc7306365)

[Question 5 13](#_Toc7306366)

[Toggled LED every 1/16 of a second 13](#_Toc7306367)

[Toggled LED every 1/256 of a second 14](#_Toc7306368)

[Toggled LED every 1/4096 of a second 15](#_Toc7306369)

[Relationship between the values 1, 1/16, 1/256, and 1/4096 15](#_Toc7306370)

[Bitwise operation could be employed to change between the TCMP values 15](#_Toc7306371)

[Exercise 3 - Combinational Logic Design 16](#_Toc7306372)

[Question 1 16](#_Toc7306373)

[Flowchart to display a 4-digit hexadecimal number 16](#_Toc7306374)

[Are the segments in a 7-segment display (A to H) high or low acting? 16](#_Toc7306375)

[Does each segment from all 7-segment displays share a common connection to the NUC140VE3CN? 16](#_Toc7306376)

[Q4 - Q7 are transistors which are configured to behave like a switch. In view of this, what is the purpose of Q4 - Q7 and is SC1 - SC4 high or low acting? 16](#_Toc7306377)

[Question 2 17](#_Toc7306378)

[State table which accounts for the hexadecimal digits 0x0 to 0xF 17](#_Toc7306379)

[Use Karnaugh maps (K-maps) to determine a minimized Boolean equation for each output in the state table 18](#_Toc7306380)

[Exercise 4 - Display characters on one 7-Segment Display 28](#_Toc7306381)

[Question 1 28](#_Toc7306382)

[Question 2 29](#_Toc7306383)

[Question 3 30](#_Toc7306384)

[Question 4 30](#_Toc7306385)

[Question 5 30](#_Toc7306386)

[Exercise 5: Creating a 4x7-Segment Display Timer 32](#_Toc7306387)

[Question 1 32](#_Toc7306388)

[Question 2 32](#_Toc7306389)

[Question 3 32](#_Toc7306390)

[Conclusion 35](#_Toc7306391)

[Lab work and contribution 36](#_Toc7306392)

[References 37](#_Toc7306393)

[Figure 1. Initial signal of the switch 7](#_Toc7306394)

[Figure 2. Signal upon pressing 8](#_Toc7306395)

[Figure 3.VirtualBench confirmation for LED toggling every 1 second 13](#_Toc7306396)

[Figure 4. VirtualBench confirmation for LED toggling every 1/16 second 14](#_Toc7306397)

[Figure 5. VirtualBench confirmation for LED toggling every 1/256 second 14](#_Toc7306398)

[Figure 6. VirtualBench confirmation for LED toggling every 1/4096 second 15](#_Toc7306399)

[Figure 7. Flowchart to display a 4 digit hexadecimal number 16](#_Toc7306400)

[Figure 8. Division of 7-segment display 17](#_Toc7306401)

[Figure 9. Segment map for each output's bit 17](#_Toc7306402)

[Figure 10. State table which accounts for the hexadecimal digits 0x0 to 0xF 18](#_Toc7306403)

[Figure 11. Flowchart to convert decimal to binary 29](#_Toc7306404)

[Figure 12. First digit from left to right 33](#_Toc7306405)

[Figure 13. Second digit from left to right 33](#_Toc7306406)

[Figure 14. Third digit from left to right 34](#_Toc7306407)

[Figure 15. Fourth digit from left to right 34](#_Toc7306408)

[Table 1: Truth table for first bit output 19](#_Toc7306327)

[Table 2: Karnaugh map for first bit output 19](#_Toc7306328)

[Table 3: Karnaugh map location for each cell for first bit output 19](#_Toc7306329)

[Table 4: Largest K-map group for first bit output 19](#_Toc7306330)

[Table 5: Truth table for second bit output 20](#_Toc7306331)

[Table 6: Truth table for third bit output 20](#_Toc7306332)

[Table 7: Karnaugh map for third bit output 21](#_Toc7306333)

[Table 8: Karnaugh map location for each cell for third bit output 21](#_Toc7306334)

[Table 9: Largest K-map group for third bit output 21](#_Toc7306335)

[Table 10: Truth table for fourth bit output 22](#_Toc7306336)

[Table 11: Karnaugh map for fourth bit output 22](#_Toc7306337)

[Table 12: Karnaugh map location for each cell for fourth bit output 22](#_Toc7306338)

[Table 13: Largest K-map group for fourth bit output 22](#_Toc7306339)

[Table 14: Truth table of fifth bit output 23](#_Toc7306340)

[Table 15: Karnaugh map for fifth bit output 23](#_Toc7306341)

[Table 16: Karnaugh map location for each cell for fifth bit output 23](#_Toc7306342)

[Table 17: Largest K-map group for fifth bit output 23](#_Toc7306343)

[Table 18: Truth table for sixth bit output 24](#_Toc7306344)

[Table 19: Karnaugh map for sixth bit output 24](#_Toc7306345)

[Table 20: Karnaugh map location for each cell for sixth bit output 24](#_Toc7306346)

[Table 21: Largest K-map group for sixth bit output 24](#_Toc7306347)

[Table 22: Truth table for seventh bit output 25](#_Toc7306348)

[Table 23: Karnaugh map for seventh bit output 25](#_Toc7306349)

[Table 24: Karnaugh map location for each cell for seventh bit output 25](#_Toc7306350)

[Table 25: Largest K-map group for seventh bit output 26](#_Toc7306351)

[Table 26: Truth table for eighth bit output 26](#_Toc7306352)

[Table 27: Karnaugh map for eighth bit output 26](#_Toc7306353)

[Table 28: Karnaugh map location for each cell for eighth bit output 27](#_Toc7306354)

[Table 29: Largest K-map group for eighth bit output 27](#_Toc7306355)

# Introduction

Most microcontrollers are supplied with one or more precision timer whose purpose is to accomplish several precision timer functions including generating events and interrupts at specific times, determining the duration between two events, or counting events.

In this laboratory, we needed to apply our knowledge on NUC140 microprocessor to create applications that are timer-based. We were also required to make use of our familiarity with VirtualBench[9] and Keil Debugging tool to confirm the correctness of our applications.

# Exercise 1 - Analyzing the Bouncing of a Push-Button

## Question 3

Press SW\_INT1 about 15 to 20 times while observing the signal on the oscilloscope. Discuss your observations, keeping in mind the time base setting

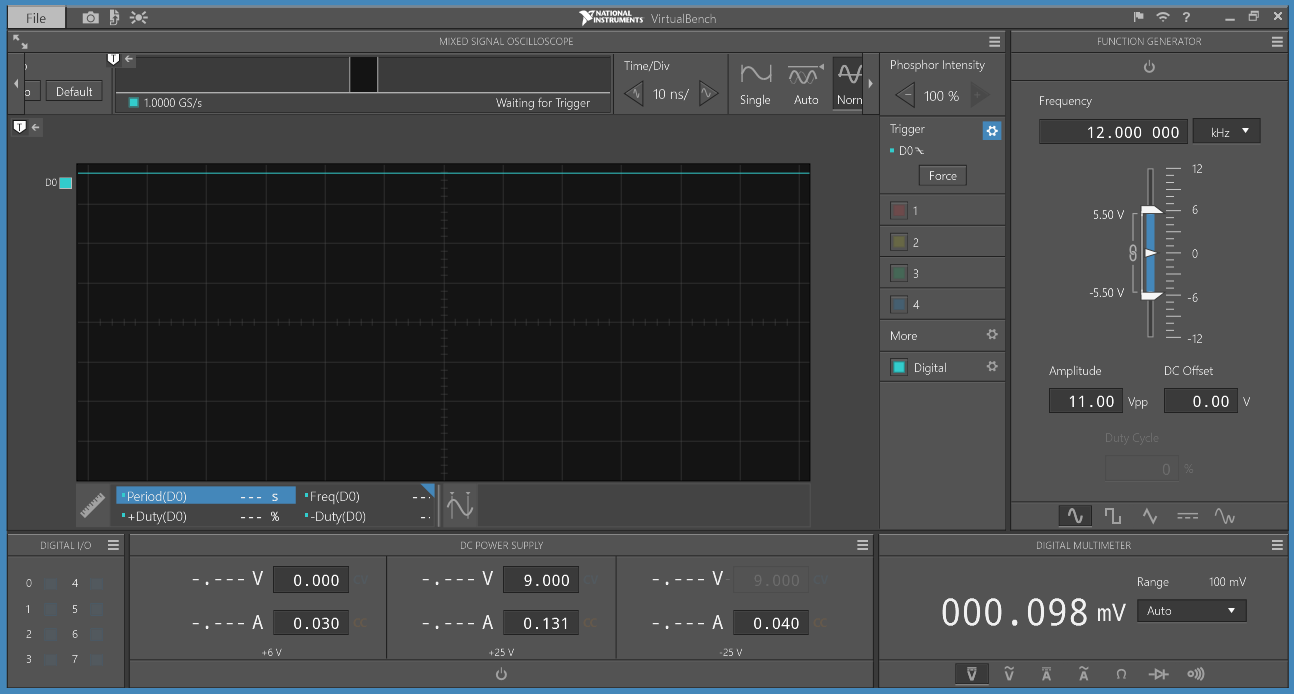


Figure 1. Initial signal of the switch

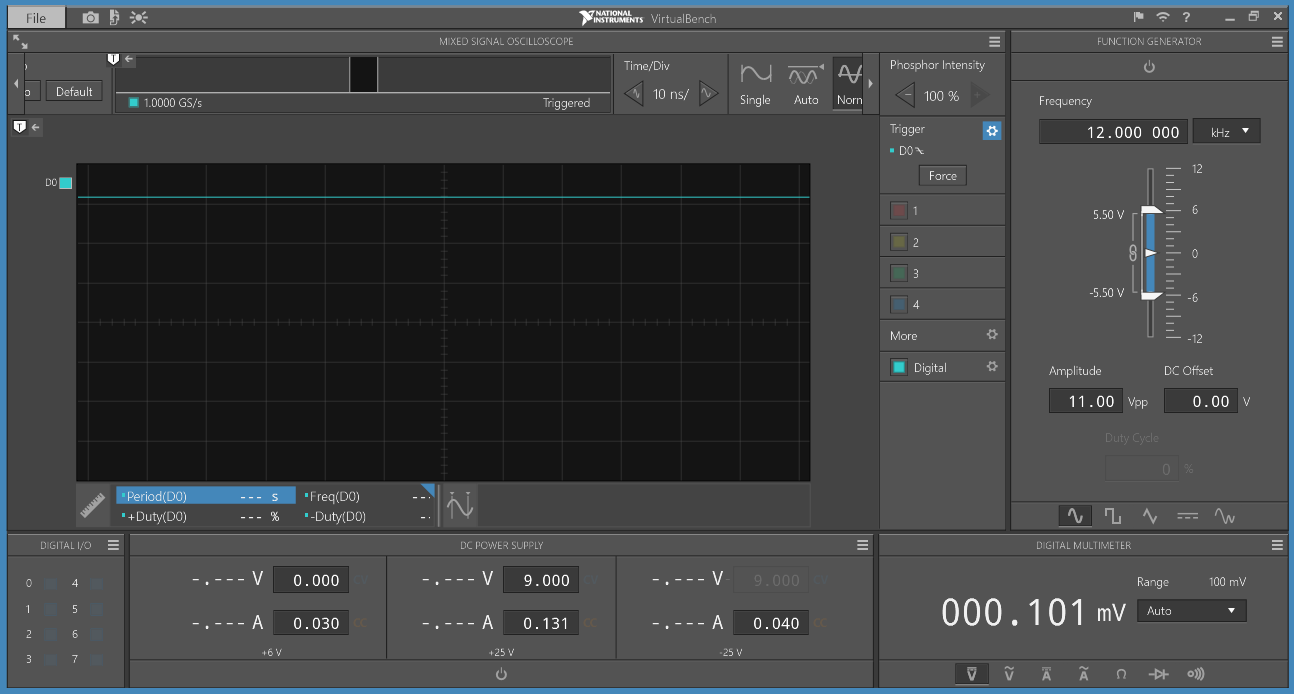


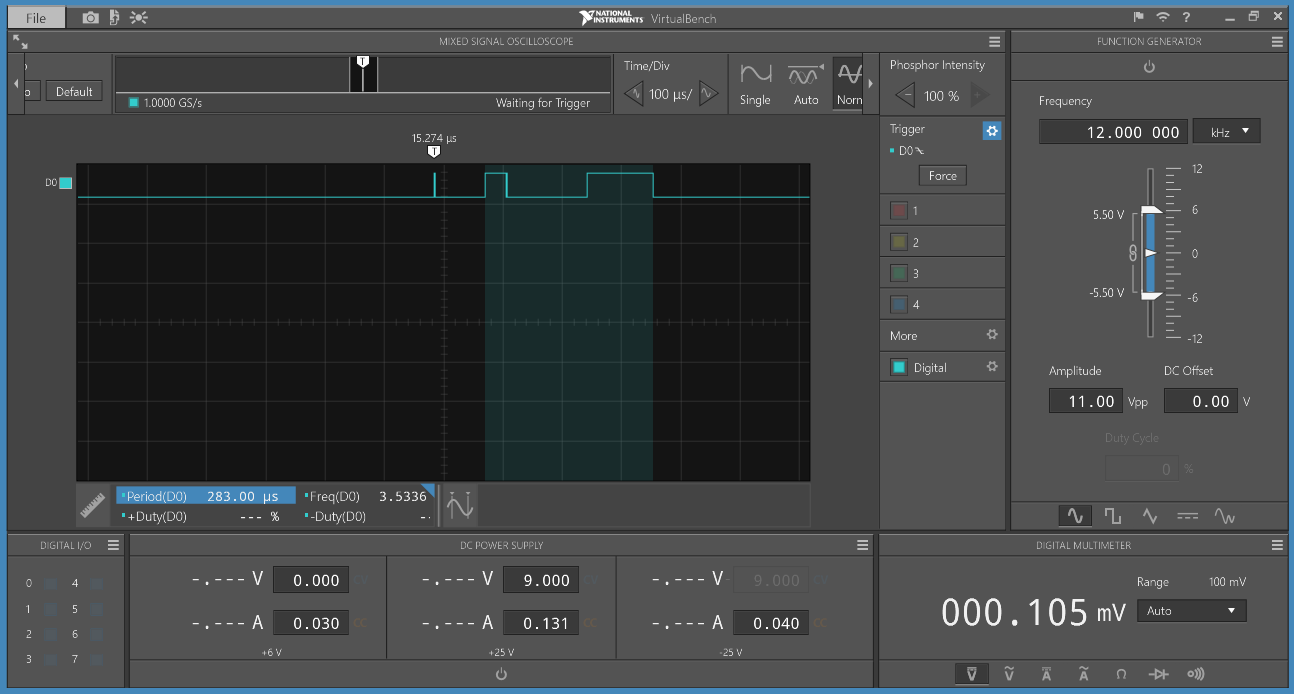
Figure 2. Signal upon pressing

The pushing of PUSH button makes the signal go and down, which is expected from a low-acting switch. If the button is not pressed, the signal stays at **HIGH**.

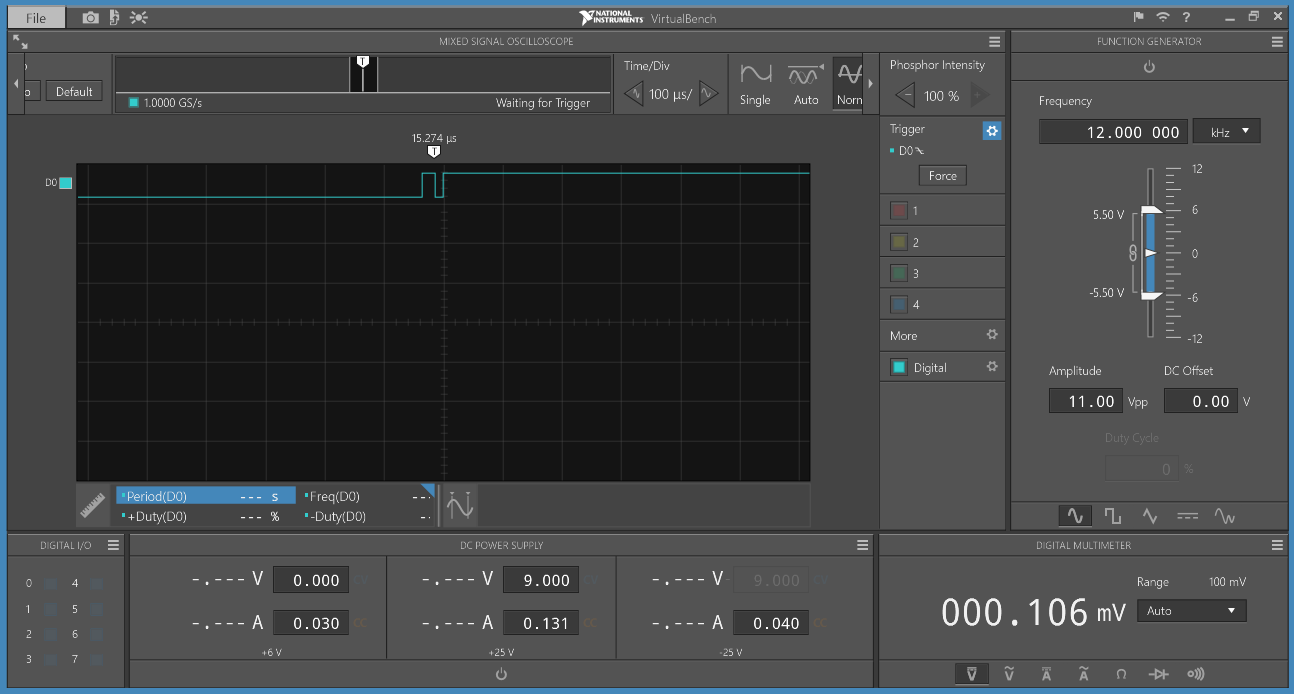
## Question 4

At what time base parameter is it possible to see the entire activity of one press of the push-button?

We tested various time base parameters from task 4 and it is only possible to see the entire activity of one press when the parameter is 100 microseconds.



From your observations in steps 3 & 4, how much time is required for the push-button to settle from bouncing?



The button settled from bouncing in about **40** microseconds.

Two methods on how the bouncing of the push-button can be eliminated in an embedded system[10]

* **Adding a small delay after the first bounce**. The microcontroller will have to wait for a short duration of time for the bouncing to stop and then continue with the program. This forced the microcontroller unable to do anything within that time span.
* **Using an interrupt for handling the switch bounce.** Within the button interrupt routine, we would do a time checking to ensure that the last bounce of the button is not shorter than the predefined debouncing time.

# Exercise 2 – Controlling an LED using a Timer and Interrupt

In the last laboratory, intermediate control of LEDs by manipulating GPIOs was carried out. It is evident that the amount of code was quite substantial, e.g. cycling through the colors of an RGB LED, and there was no way in which the timing events (in seconds) could be predicted. It is possible to precisely control events, with respect to time, by using timers. To demonstrate this, you are required to toggle an LED using a timer and the nested vector interrupt controller (NVIC) module. Specifically, the LED connected to GPC12 and Timer 0 (TMR0) must be used. The LED on GPC12 must toggle every second

## Question 1

Configure GPC12 to be in an appropriate output mode to control the LED connected to it[2]

|  |
| --- |
| void Set\_output\_mode(int gpcNum){      PC->PMD &= ~(0b11 << gpcNum \* 2);                   // Change pin at gpioNum to OUTPUT mode, whose value should be from 12-15      PC->PMD |= (0b01 << gpcNum \* 2);  }  Set\_output\_mode(12ul); |

## Question 2

Setup the NUC140VE3CN clock source to the external 12 MHz crystal and route it to TMR0

Which clock source does the NUC140VE3CN default to?

Default clock source: External 32.768KHz Low Speed (TRM page 135)[1]

## Question 3

What value needs to be entered into the timer control and status register (TCSR) to pre-scale the clock by 4?

The value needs to be entered is 3 because the actual value equaled to the value in the register + 1.[1]  
The timeout period can be calculated intuitively by knowing the clock period, prescaling factor and TCMP value:

Provide a logical explanation on the validity of the formula above

Clock period is the time it takes for a period of cycle. TDR (Counter register) starts from 0 and continually increases towards TCMP [3]. Each comparison took 1 cycle, thus, the timeout period is equal to the product of clock period, prescaling factor and compare value.

## Question 4

What does the interrupt priority value of 1 mean and how might it work with other

interrupts having different priority values?

The lower the priority is, the more important the task is.

All of the interrupts and most of the system exceptions can be configured to different priority levels. When an interrupt occurs, the NVIC will compare the priority of the new interrupt to the current running one’s priority. If the priority of the new interrupt is higher than the current one, the new interrupt handler will override the current handler

The default priority of all the user-configurable interrupts is “0”.

## The highest user configurable priority is denoted as “0” and the lowest priority is denoted as “3”. [1]

## Question 5

Verify with the Virtual Bench Oscilloscope that the LED is toggling exactly at every second.

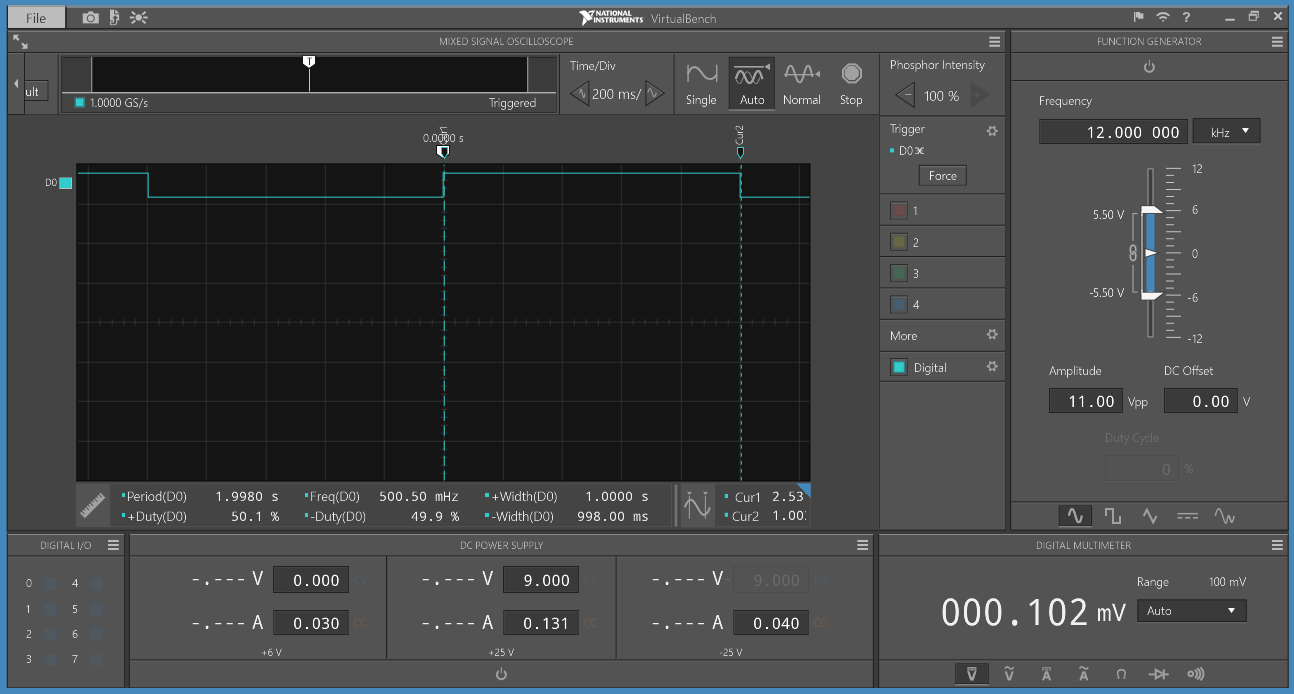


Figure 3.VirtualBench confirmation for LED toggling every 1 second

As we can see from the screenshot, the width is almost exactly at 1 second, which confirmed our value.

### Toggled LED every 1/16 of a second

Calculated TCMP value = 187500

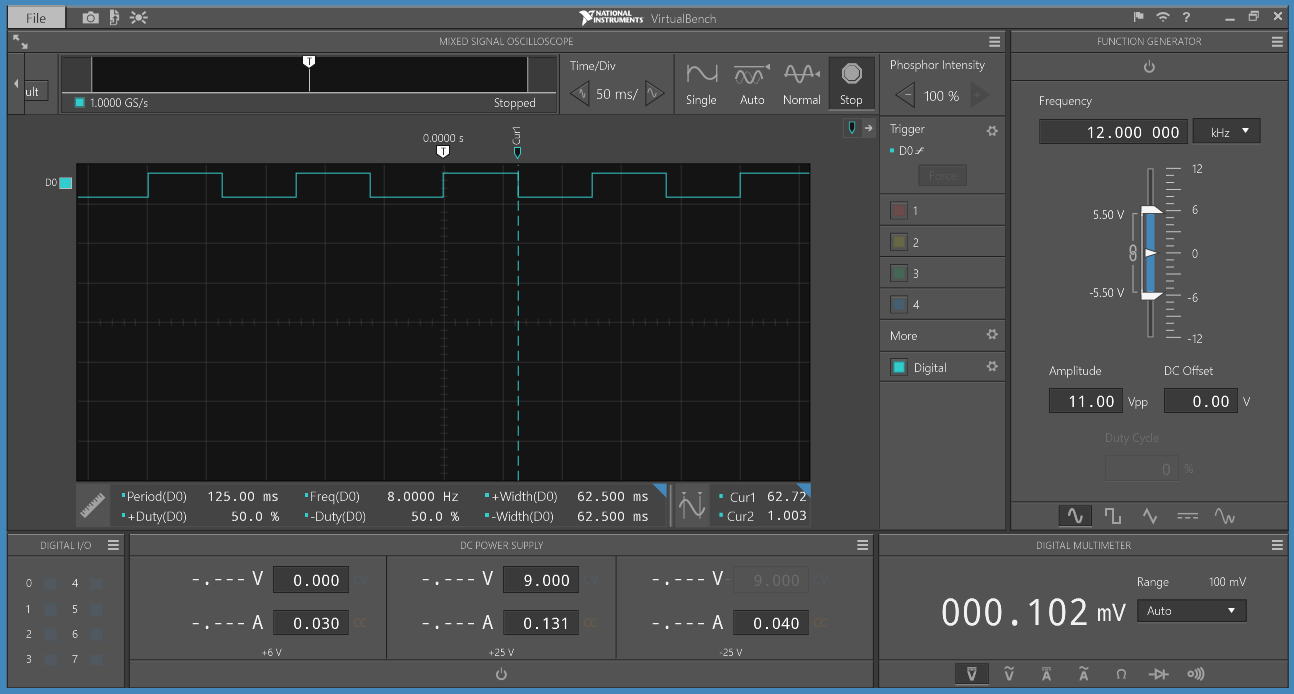


Figure 4. VirtualBench confirmation for LED toggling every 1/16 second

The width is at 62.5 milliseconds, which is equal to 1/16 of a second. This confirmed our TCMP value

### Toggled LED every 1/256 of a second

Calculated TCMP value = 11719

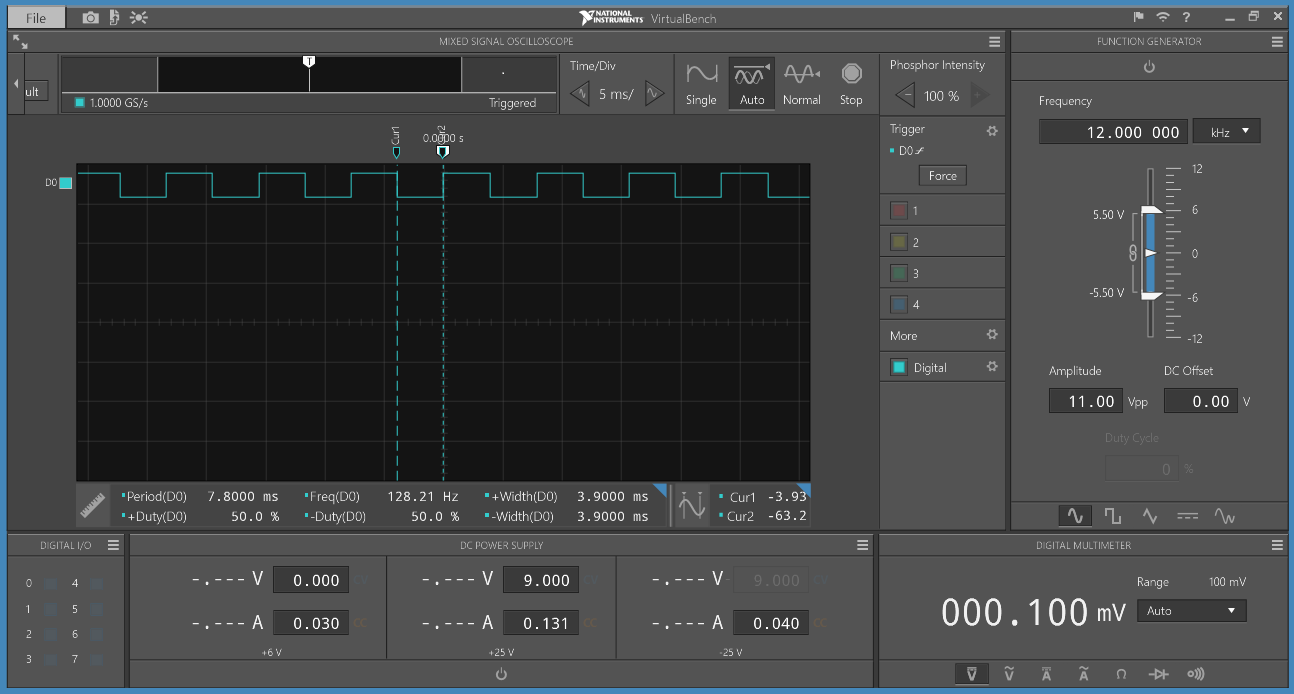


Figure 5. VirtualBench confirmation for LED toggling every 1/256 second

The width of an edge is 3.9 milliseconds, which verified the TCMP value

### Toggled LED every 1/4096 of a second

Calculated TCMP value = 732

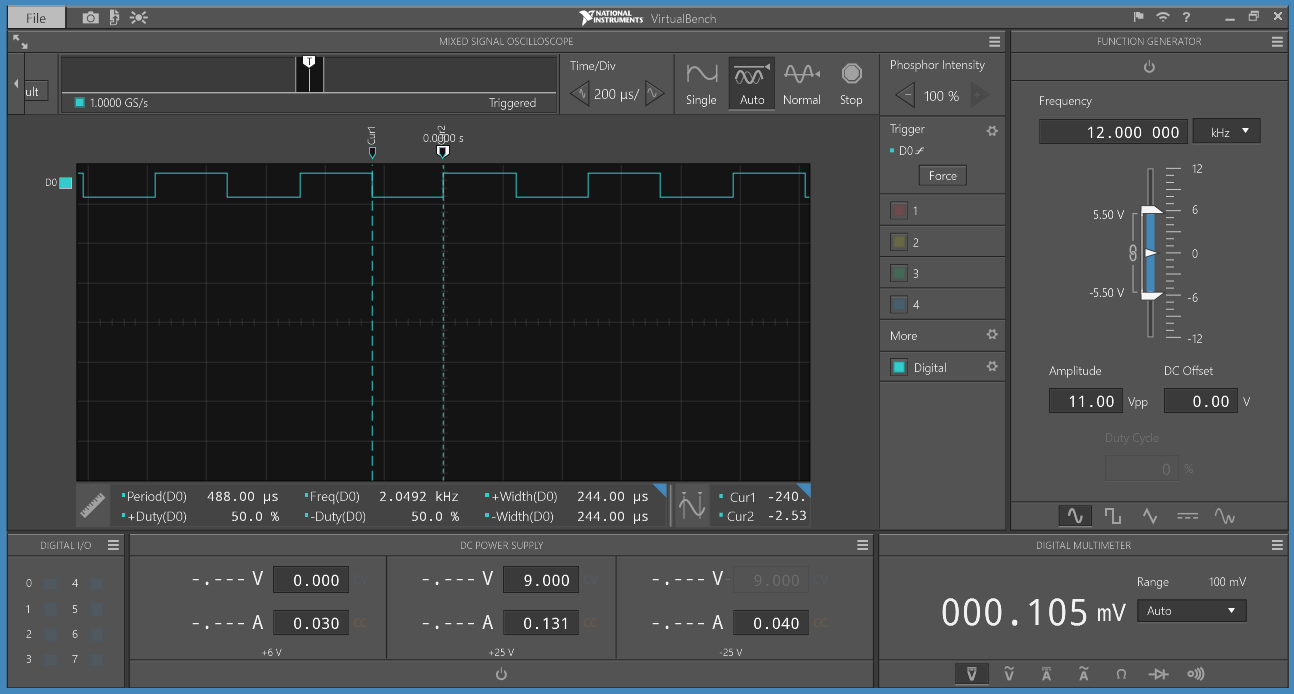


Figure 6. VirtualBench confirmation for LED toggling every 1/4096 second

An edge lasts exactly 488 microseconds, so it validated our TCMP value of 732.

### Relationship between the values 1, 1/16, 1/256, and 1/4096

The next TCMP value is exactly 16 times smaller than the previous TCMP value.

### Bitwise operation could be employed to change between the TCMP values

Shifting bit left and right 4 times could be used to change between the TCMP values. [7]

# Exercise 3 - Combinational Logic Design

## Question 1

Determine the connections between the 7-segment displays and the NUC140VE3CN

### Flowchart to display a 4-digit hexadecimal number

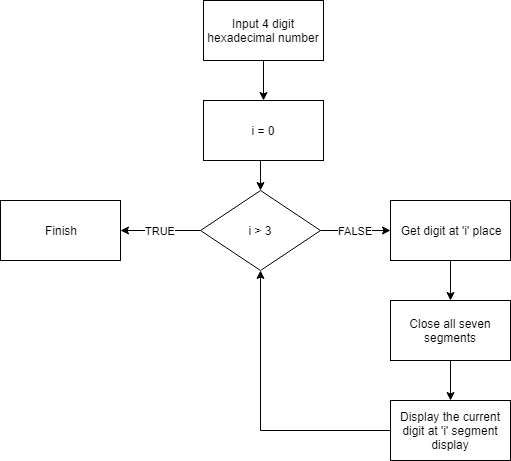


Figure 7. Flowchart to display a 4-digit hexadecimal number

### Are the segments in a 7-segment display (A to H) high or low acting?

All segments in 7-segment display is low acting.[1]

### Does each segment from all 7-segment displays share a common connection to the NUC140VE3CN?

Yes, they share the same GPIO PIN PORT E from 0 to 7. [1]

### Q4 - Q7 are transistors which are configured to behave like a switch. In view of this, what is the purpose of Q4 - Q7 and is SC1 - SC4 high or low acting?

Q4-7 acts as the source for 7-segment , SC1 - SC4 is high-acting.

## Question 2

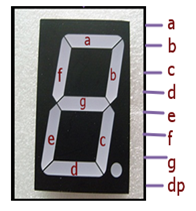


Figure 8. Division of 7-segment display

According to the specifications, we could divide the 7-segment display into 8 parts as above figure.

Because our input number ranged from 0 to 15, 4 bits were enough to cover the whole input.

And for the output, it contained 8 bits, and each of them mapped to the corresponding segment like below:

|  |  |
| --- | --- |
| Bit number | Map to |
| 0 | C |
| 1 | DP |
| 2 | F |
| 3 | A |
| 4 | B |
| 5 | D |
| 6 | E |
| 7 | G |

Figure 9. Segment map for each output's bit

### State table which accounts for the hexadecimal digits 0x0 to 0xF

Our team tested every single value for each segment of the display and come up with the table below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **8-bit output (Left to right)** | | | | | | | |
| **A** | **B** | **C** | **D** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

Figure 10.   
State table which accounts for the hexadecimal digits 0x0 to 0xF

### Use Karnaugh maps (K-maps) to determine a minimized Boolean equation for each output in the state table

We applied K-maps for each individual output bit and derived a Boolean equation for every single of them. [11]

#### First bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 0 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 1 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 0 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 0 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 1 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 1 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 1 |

Table 1: Truth table for first bit output

##### Karnaugh map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 0 | 0 | 1 |
|  | 0 | 0 | 0 | 0 |
|  | 1 | 0 | 1 | 1 |
|  | 0 | 0 | 0 | 0 |

Table 2: Karnaugh map for first bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 3: Karnaugh map location for each cell for first bit output

##### Karnaugh group

|  |  |
| --- | --- |
| (12, 14) |  |
| (14, 15) |  |
| (2 ) |  |

Table 4: Largest K-map group for first bit output

##### Derived equation

#### Second bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[1]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 1 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 1 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 1 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 1 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 1 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 1 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 1 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 1 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 1 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 1 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 1 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 1 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 1 |

Table 5: Truth table for second bit output

##### Derived equation

Every output for this case is 1, thus there is no need to apply K-map over this case. We can just deduce the output equation for this bit, which is:

#### Third bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[2]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 1 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 1 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 0 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 0 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 1 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 0 |

Table 6: Truth table for third bit output

##### Karnaugh map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 1 | 1 |
|  | 0 | 0 | 1 | 0 |
|  | 0 | 1 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |

Table 7: Karnaugh map for third bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 8: Karnaugh map location for each cell for third bit output

##### Karnaugh Groups

|  |  |
| --- | --- |
| (1, 3) |  |
| (2, 3) |  |
| (3, 7) |  |
| (13) |  |

Table 9: Largest K-map group for third bit output

##### Derived equation

#### Fourth bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[3]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 1 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 0 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 1 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 0 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 1 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 0 |

Table 10: Truth table for fourth bit output

##### Karnaugh Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 0 | 0 |
|  | 1 | 0 | 0 | 0 |
|  | 0 | 1 | 0 | 0 |
|  | 0 | 0 | 1 | 0 |

Table 11: Karnaugh map for fourth bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 12: Karnaugh map location for each cell for fourth bit output

##### Karnaugh Group

|  |  |
| --- | --- |
| (1) |  |
| (4) |  |
| (11) |  |
| (13) |  |

Table 13: Largest K-map group for fourth bit output

##### Derived Equation

#### Fifth bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[4]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 0 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 0 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 1 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 1 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 0 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 1 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 1 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 1 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 1 |

Table 14: Truth table of fifth bit output

##### Karnaugh Map

Equation for fifth bit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 0 | 1 |
|  | 1 | 0 | 1 | 1 |
|  | 0 | 0 | 1 | 0 |

Table 15: Karnaugh map for fifth bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 16: Karnaugh map location for each cell for fifth bit output

##### Karnaugh Groups

|  |  |
| --- | --- |
| (6,14) |  |
| (11,15) |  |
| (12,14) |  |
| (5) |  |

Table 17: Largest K-map group for fifth bit output

#### Sixth bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[5]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 1 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 1 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 0 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 1 |

Table 18: Truth table for sixth bit output

##### Karnaugh Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 0 | 0 |
|  | 1 | 0 | 1 | 0 |
|  | 0 | 0 | 1 | 0 |
|  | 0 | 0 | 0 | 1 |

Table 19: Karnaugh map for sixth bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 20: Karnaugh map location for each cell for sixth bit output

##### Karnaugh Groups

|  |  |
| --- | --- |
| (7,15) |  |
| (1) |  |
| (4) |  |
| (10) |  |

Table 21: Largest K-map group for sixth bit output

##### Derived Equation

#### Seventh bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[6]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 1 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 1 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 1 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 1 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 0 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 0 |

Table 22: Truth table for seventh bit output

##### Karnaugh Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 1 | 0 |
|  | 1 | 1 | 1 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 1 | 0 | 0 |

Table 23: Karnaugh map for seventh bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 24: Karnaugh map location for each cell for seventh bit output

##### Karnaugh Groups

|  |  |
| --- | --- |
| (1,3,5,7) |  |
| (1,9) |  |
| (4,5) |  |

Table 25: Largest K-map group for seventh bit output

##### Derived Equation

#### Eighth bit

##### Truth table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dec** | **Hex** | **Binary** | **4-bit input (ABCD)** | | | | **Output[7]** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0000 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0001 | 0 | 0 | 0 | 1 | 1 |
| 2 | 2 | 0010 | 0 | 0 | 1 | 0 | 0 |
| 3 | 3 | 0011 | 0 | 0 | 1 | 1 | 0 |
| 4 | 4 | 0100 | 0 | 1 | 0 | 0 | 0 |
| 5 | 5 | 0101 | 0 | 1 | 0 | 1 | 0 |
| 6 | 6 | 0110 | 0 | 1 | 1 | 0 | 0 |
| 7 | 7 | 0111 | 0 | 1 | 1 | 1 | 1 |
| 8 | 8 | 1000 | 1 | 0 | 0 | 0 | 0 |
| 9 | 9 | 1001 | 1 | 0 | 0 | 1 | 0 |
| 10 | A | 1010 | 1 | 0 | 1 | 0 | 0 |
| 11 | B | 1011 | 1 | 0 | 1 | 1 | 0 |
| 12 | C | 1100 | 1 | 1 | 0 | 0 | 1 |
| 13 | D | 1101 | 1 | 1 | 0 | 1 | 0 |
| 14 | E | 1110 | 1 | 1 | 1 | 0 | 0 |
| 15 | F | 1111 | 1 | 1 | 1 | 1 | 0 |

Table 26: Truth table for eighth bit output

##### Karnaugh Map

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 1 | 1 | 0 | 0 |
|  | 0 | 0 | 1 | 0 |
|  | 1 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |

Table 27: Karnaugh map for eighth bit output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 1 | 3 | 2 |
|  | 4 | 5 | 7 | 6 |
|  | 12 | 13 | 15 | 14 |
|  | 8 | 9 | 11 | 10 |

Table 28: Karnaugh map location for each cell for eighth bit output

##### Karnaugh Groups

|  |  |
| --- | --- |
| (0,1) |  |
| (7) |  |
| (12) |  |

Table 29: Largest K-map group for eighth bit output

##### Derived Equation

#### All derived equations

And finally, to sum up all the Boolean equations we obtained so far:

# Exercise 4 - Display characters on one 7-Segment Display

## Question 1

The steps which need to be taken to display the digits 0 - 15 (in hexadecimal) on a 7-segment display

* Set the GPIO port C from 4 to 7 to output mode
* Set the GPIO port E from 0 to 7 to quasi-bi-directional
* Close all segments by setting DOUT bit at port C port of each segment to 0
* Convert the desired number to display to binary
* Apply the Boolean equation we obtained from previous exercise to bit of GPIO port E from 0 to 7
* Set the GPIO port C value to High at the segment we want to display

## Question 2

A flowchart which describes your algorithm for converting a decimal number to binary

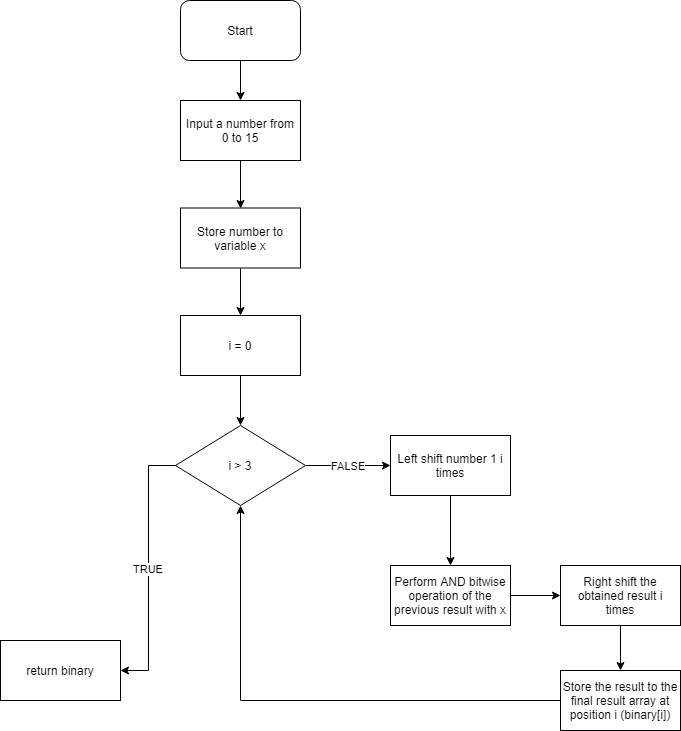


Figure 11. Flowchart to convert decimal to binary

## Question 3

You will need to perform bitwise operations on single bits. Discuss the issues in using the native C data types to perform such operations - is it possible?

There are 3 issues with using native C data types to perform such operations [7]:

* Bitwise operator only works on a limited number of types: **int** and **char**.
* Shifting does not change the variable value.
* To modify a bit in C, we need to perform a *masking*.

## Question 4

Define a datatype which can only accept the values 1 or 0

* What does this type define structure do? What is the purpose of using unsigned short in the structure element?
  + This type define structure contained a member x with type of **unsigned short** and had exactly 1 bit long. Therefore, x value could only be 0 or 1.
  + **unsigned short** was used to save memory
* How do you access an element in a structure?

To access any member of a structure, we use the member access operator (.). If we have the pointer of that structure, we can use the (->) operator. [6]

## Question 5

Using the bitwise operators and being aware of their order of precedence, code each output equation for a 7-segment display

|  |
| --- |
| void ShowSevenSegment(uint8\_t no, uint8\_t number) {  short binA, binB, binC, binD;  bin \* binValue = decimal\_to\_binary(number); // Binary value of  CloseSevenSegment();  PE - > DOUT |= 0xFF;  binA = (binValue + 3) - > x;  binB = (binValue + 2) - > x;  binC = (binValue + 1) - > x;  binD = binValue - > x;  // [7]  Y   = A'B'C' + A'BCD + ABC'D'  PE - > DOUT &= ~(((~binA & ~binB & ~binC) |  (~binA & binB & binC & binD) |  (binA & binB & ~binC & ~binD)) << 7);  // [6] Y    = A'D + B'C'D + A'BC'  PE - > DOUT &= ~(((~binA & binD) |  (~binB & ~binC & binD) |  (~binA & binB & ~binC)) << 6);  // [5] Y = BCD + A'B'C'D + A'BC'D' + AB'CD'  PE - > DOUT &= ~(((binB & binC & binD) |  (~binA & ~binB & ~binC & binD) |  (~binA & binB & ~binC & ~binD) |  (binA & ~binB & binC & ~binD)) << 5);  // [4]  Y = BCD' + ACD + ABD' + A'BC'D  PE - > DOUT &= ~(((binB & binC & ~binD) |  (binA & binC & binD) |  (binA & binB & ~binD) |  (~binA & binB & ~binC & binD)) << 4);  // [3]  Y = A'B'C'D + A'BC'D' + AB'CD + ABC'D  PE - > DOUT &= ~(((~binA & ~binB & ~binC & binD) |  (~binA & binB & ~binC & ~binD) |  (binA & ~binB & binC & binD) |  (binA & binB & ~binC & binD)) << 3);  // [2]  Y = A'B'D + A'B'C + A'CD + ABC'D  PE - > DOUT &= ~(((~binA & ~binB & binD) |  (~binA & ~binB & binC) |  (~binA & binC & binD) |  (binA & binB & ~binC & binD)) << 2);  // [1]  Y = 1  PE - > DOUT &= ~(1 << 1);  // [0]  C = ABD' + ABC + A'B'CD'  PE - > DOUT &= ~(((binA & binB & ~binD) |  (binA & binB & binC) |  (~binA & ~binB & binC & ~binD)) << 0);  EnableSegment(no);  } |

# Exercise 5: Creating a 4x7-Segment Display Timer

In this final exercise, we are required to combine all the knowledge we acquired from previous tasks to create an incrementing hexadecimal timer.

## Question 1

Use the knowledge you learnt and information from Exercise 2 to configure SW\_INT1 to trigger an interrupt upon every button press.

To enable the SW\_INT1, we had to do the following to steps: [5]

|  |
| --- |
| //GPIO Interrupt configuration. GPIO-B15 is the interrupt source      PB->PMD &= (~(0x03ul << 30));      PB->IMD &= (~(1ul << 15));      PB->IEN |= (1ul << 15);        //NVIC interrupt configuration for GPIO-B15 interrupt source      NVIC->ISER[0] |= 1ul<<3;      NVIC->IP[0] &= (~(3ul<<30)); |

## Question 2

What value needs to be written to TCMPR such that the third hexadecimal digit (from the right) is incrementing once every second?

The value is 11719 (which is 3 000 000 >> 8).

How can a new TCMP value be determined immediately such that the fourth hexadecimal digit (from the right) is incrementing once every second?

It is 16 times smaller than the above value, which is **732** (3 000 000 >> 12)

TCMP value for the second digit to increment every second: **187500** (3 000 000 >> 4)

TCMP value for the first digit

## Question 3

Use the Virtual Bench oscilloscope to verify that the four TCMP values obtained are correct

To verify the TCMP value we obtain above, we set up the VirtualBench and our code as follow:

* D0 to D3 wires were connected to GPC4 to 7
* D31 wire was connected to GPC 12.

It was impossible to verify it via GPC4 to 7, thus, we add an extra line of code which toggled the LED 5 every time a digit increase.

And the result is as we expected, every interval for each digit is exactly 1 second.

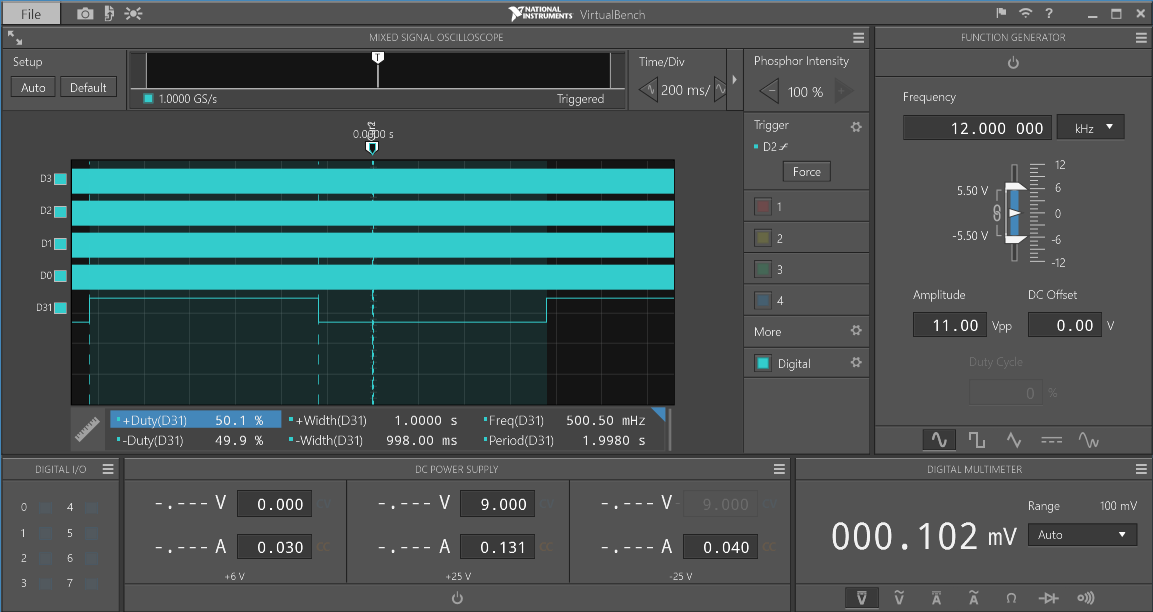


Figure 12. First digit from left to right

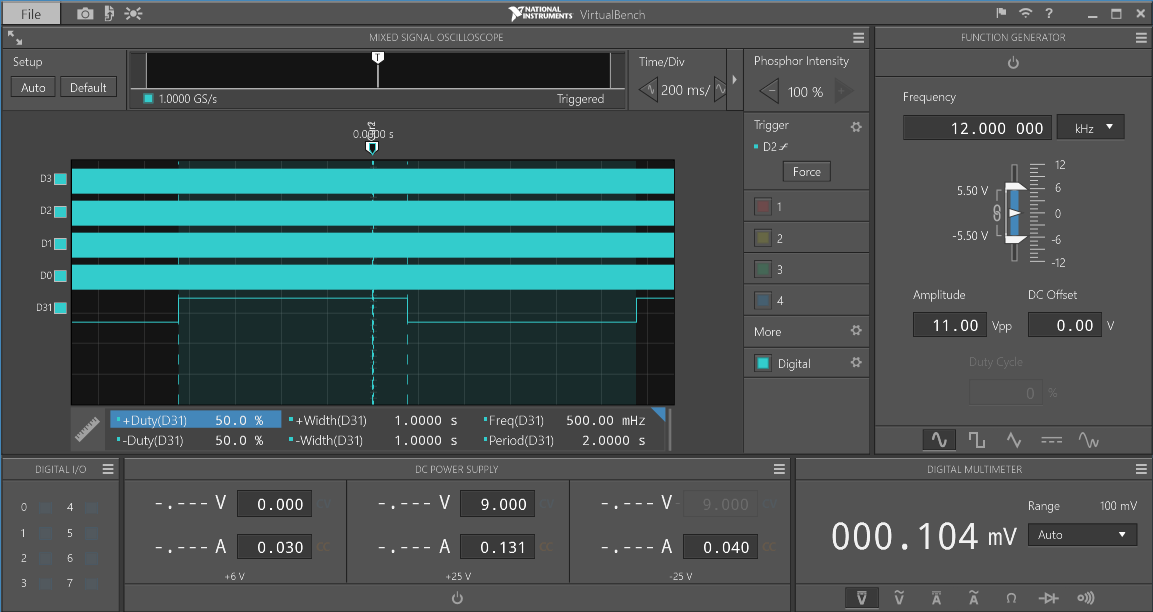


Figure 13. Second digit from left to right

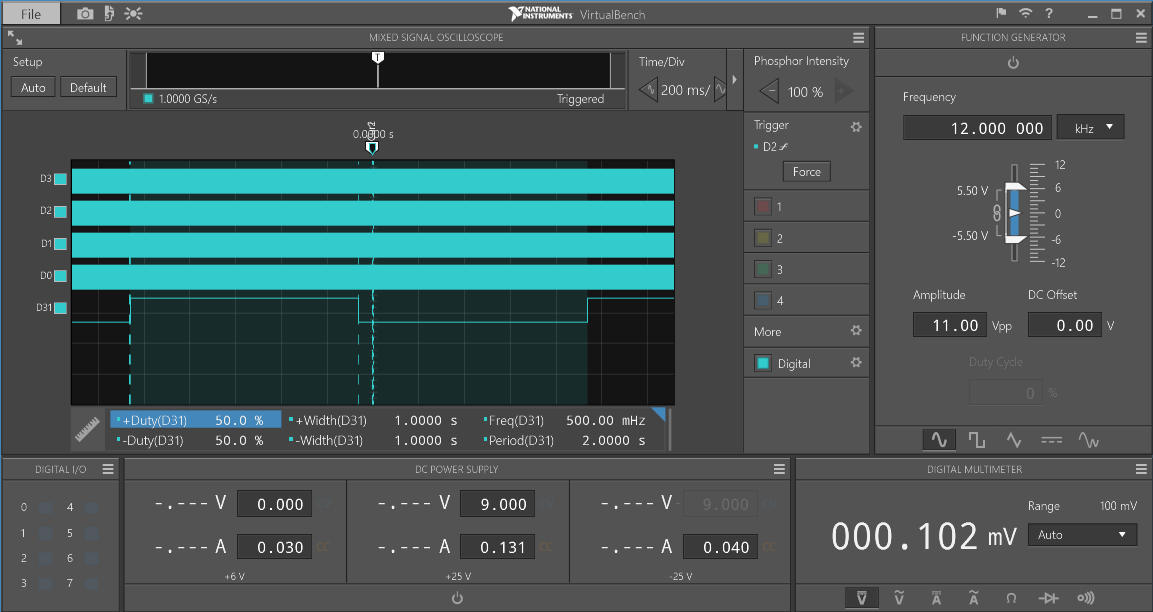


Figure 14. Third digit from left to right

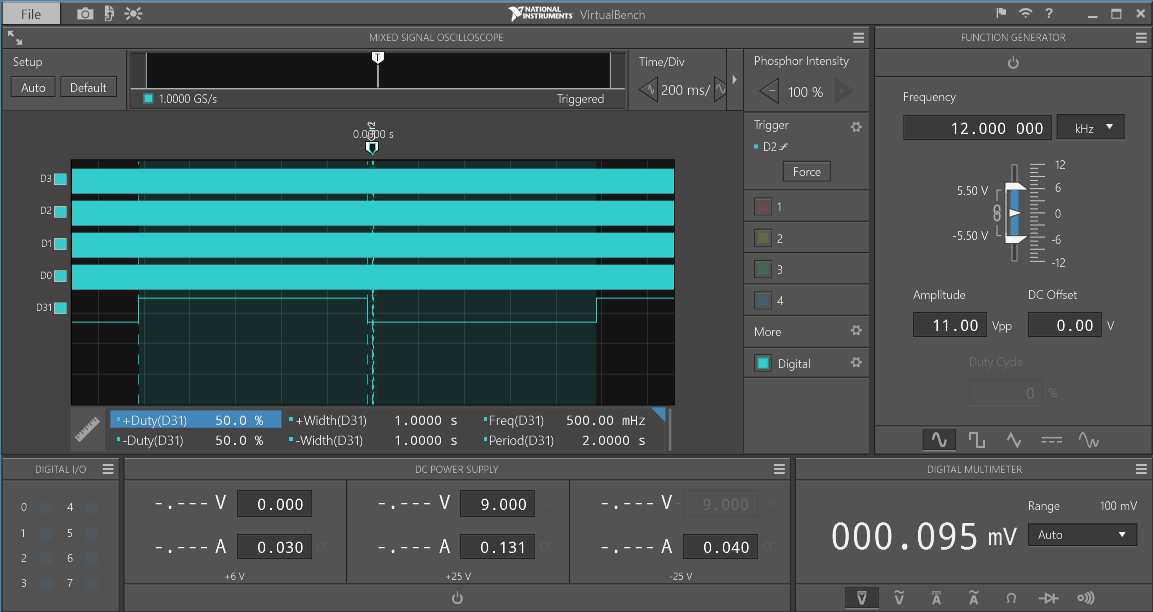


Figure 15. Fourth digit from left to right

What line of code will need to be added to the timer's interrupt service routine to monitor the timer's incremental rate?

Toggle LED 5, which is connected to GPC 12.

# Conclusion

By completing this laboratory, we are more confident on programming embedded system using C. We have learnt to deal with timer and interrupt register, how to control the 7-segment displays and handle the button pressing event. Another thing that is also improved is the knowledge on bitwise operation, K-map and debugging our application via Keil. Finally, it is important to validate the application’s correctness with VirtualBench.

# Lab work and contribution

|  |  |  |
| --- | --- | --- |
| **Member name** | **Work contribution** | **Work contributions in words** |
| Phung Minh Tuan | % | Tuan is doing half of exercise 4 and 5. Also, he writes for his own report for the exercise 1 and 5. He also helped with the layout and format of the report |
| Le Huu Nghia | 34% | Nghia’s part is to complete K-map for exercise 3 and half of exercise 4. Nghia also wrote the report for his work. |
| Trung Nguyen | 30% | Trung is responsible for doing and writing report for exercise 1 and 2. |

# References

[1] NUC130/NUC140 Technical Reference Manual. Available: https://rmit.instructure.com/courses/55465/files/5638356/download?verifier=LdJjAu4ogz8E1w8TUSrIAVuoX9E4FxotJIbIQgCT&wrap=1. [Accessed: 25- March- 2019].

[2] Andrew Smith. General Purpose Inputs Outputs (GPIOs). Available: <https://rmit.instructure.com/courses/59115/pages/t4-lecture-resources?module_item_id=1619028> [Accessed: 25- March- 2019].

[3] Andrew Smith. Memory Map & Clock System. Available: <https://rmit.instructure.com/courses/59115/pages/t5-lecture-resources?module_item_id=1536299>

[4] Andrew Smith. Timers & Counters. Available: <https://rmit.instructure.com/courses/59115/pages/t6-lecture-resources?module_item_id=1536306>

[5] Andrew Smith. Interrupts. Available: <https://rmit.instructure.com/courses/59115/pages/t7-lecture-resources?module_item_id=1536314>

[6] Tutorialspoint.com. (n.d.). Structures in C. Available: https://www.tutorialspoint.com/cprogramming/c\_structures.htm

[7] Hadar Binsky. Bitwise Operators in C. Available: <http://www2.mta.ac.il/~hbinsky/c%20content/Bits.pdf>

[8] Thampi, G. T. (n.d.). Decimal to binary using Bitwise operator. Available: <https://stackoverflow.com/a/49830262>

[9] Andrew Smith. Guide to Using the Virtual Bench. Available: <https://rmit.instructure.com/courses/59115/pages/t6-guide-to-using-the-virtual-bench?module_item_id=1635737>

[10] Christoffersen, J. (2015, September 03). Switch Bounce and How to Deal with It. Available: <https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it>.

[11] Logic circuit simplification (SOP and POS). (n.d.). Retrieved from http://www.32x8.com/