# **IE4-MC LAB SESSION #2: PREPARATION SHEET**

|  |  |
| --- | --- |
| **Surname, First Name** | **Group Team** |
| **1.**  **2.**  **3.** | 1 ◻ A B C D E F  2 ◻  3 ◻ |

This preparation sheet must be filled and uploaded to EMIL by every student before the lab session. Cooperation within the lab team (of 2-3 students) is possible (same solution). However, no cooperation across lab teams.

## QUESTION 1:

## Give the 8 x n-matrix and the corresponding hexadecimal value which you want to display on the LED pendulum.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LED7 | ■ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED5 | ■ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED2 | ■ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LED0 | ■ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| HEX CODE | 0xA5 | 0x00 | 0x70 | 0x08 | 0x04 | 0x02 | 0x04 | 0x08 | 0x70 | 0x00 | 0x00 | 0x7e | 0x00 | 0x00 | 0x70 | 0x08 | 0x04 | 0x02 | 0x04 | 0x08 | 0x70 | 0x00 | 0x00 | 0x3e | 0x48 | 0x48 | 0x48 | 0x48 | 0x3e | 0xx  0  0 |

## QUESTION 2:

In order to realize a stable pattern on the LED pendulum, the character string has to be aligned (synchronized) to the turning points indicated by an edge of the -signal.

Give C code that detects the left and the right turning point of the pendulum. Assume that the -signal is connected to PD(0) and that the PORT D has been correctly configured (clock activated, port enabled and directions set).

|  |
| --- |
| IE4, MC\_Lab2//-------------------------------------------------------/  #include <stdio.h>  #include<stdlib.h>  #include "inc/lm3s9b92.h" // defines all registers  void main() {  Port Clock Gating Control  SYSCTL\_RCGC2\_R |= ((1 << 3)); // enabling clock for Port D  GPIO\_PORTD\_DEN\_R = 0x01; // Enabling Port D  GPIO\_PORTD\_DIR\_R = 0x00; // set bits D(0) of Port D as input  while (1){  if((GPIO\_PORTD\_DATA\_R &0x01)==1){  //we are on the right side  }  else if(!(GPIO\_PORTD\_DATA\_R &0x01)==1){  //we are on the Left side  }  } |

## QUESTION 3:

Develop a function **timerConfig(void)** that configures TIMER0A as a 16 bit periodic timer that can represent times up to 10 ms (i.e. does not overflow within 10ms). Give the C-Code:

|  |
| --- |
| timerconfig(void ){  SYSCTL\_RCC\_R = ((SYSCTL\_RCC\_R | 0x00000540) & ~0x000002B1);// activate main oscillator at 16 MHz w/ external XTAL  waitcycle ++;  TIMER0\_CTL\_R &= ~0x0001; // disablenable Timer 0A  TIMER0\_CFG\_R = 0x04; // 2 x 16-bit mode  TIMER0\_TAMR\_R = 0x22; // periodic mode + match enable  TIMER0\_TAPR\_R = 3-1;//prescalar =(t\*16MH)/65536  TIMER0\_TAILR\_R= 65574-1;//loadvalue =(16\*0.5)\123  TIMER0\_TAMATCHR\_R = 32785-1; //matchregister value (16Mhz\*5ms)/2.44 THIS MATCH VALUE IS AFTER 5 MS //not asked  } |

## QUESTION 4:

Develop a function **timerWait(unsigned short usec)** that realizes a delay by Timer 0A and uses the configuration of timerConfig(). The function sets the interval load value, enables the timer, waits for the time-out of the timer, clears the interrupt flags and disables the timer. usec is the value in µs after after which the timer reaches time-out.

|  |
| --- |
| **timerWait(**input**)//load value !!!!!!!!!!!!!**  **public void timerWait(unsigned short usec)**  **{**  TIMER0\_TAILR\_R=(((16\*10^6)\*input)**/216)-1;**  TIMER0\_CTL\_R |= 0x0001; // enable Timer 0  while((TIMER0\_RIS\_R & (1<<0))==0);  TIMER0\_ICR\_R|= (1<<0)  TIMER0\_CTL\_R &= ~0x0001; // disablenable Timer 0A  **}** |
|  |