**ECE - 4097 Embedded Controller programing for Real Time Systems**

**Final Exam - Total points 35**

**Name:**

**Follow these steps to complete this assignment.**

**Final Exam prerequisite:**

1. Use Cube32 to generate code without UART interrupt
2. Enable UART in **Interrupt** mode.
3. Enable GPIO PIN 13 interrupt, used for blue user switch and handler is

EXTI15\_10\_IRQHandler ()

**Note: This is same as mid-term**

1. Enable Timer 2 and Timer 3, Timer 6 interrupts
2. Enable both watchdog, IWDG and WWDG
3. Write additional code in C language to do the followings. **1 Point**
   1. implement logMsg() method to display the string on terminal using interrupt
   2. Implement logGetMsg () method to receive character from terminal using interrupt
   3. If you enter char ‘g’ on the terminal
      1. Print the received char
      2. And toggle green LED after 1 sec
   4. If you enter char ‘b’ on the terminal
      1. Print the received char
      2. And toggle blue LED after 1 sec
   5. For any other character, print “unknown character received”

/\* **Here is the code template for your reference only. Feel free to use the methods**

**parameter differently but keep the method name same. We will be using the same name**

**for future assignments \*/**

**Note: Make sure MX\_IWDG\_Init(), MX\_WWDG\_Init(), MX\_RTC\_Init() is commented out to start with**

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include "string.h"

#include <stdio.h>

/\* USER CODE END Includes \*/

// logMsg method

void logMsg(UART\_HandleTypeDef \*huart, uint8\_t \_out[])

{

// fill in here

// Use HAL\_UART\_Transmit\_IT()

}

// logGetMsg method

uint8\_t logGetMsg(UART\_HandleTypeDef \*huart){

// fill in here

}

// Main starts here

Void main()

{

/\* USER CODE BEGIN 2 \*/

logMsg(&huart1, "welcome to Embedded controller programming \r\n");

logMsg(&huart1, "Enter g for toggling Green LED \r\n");

logMsg(&huart1, "Enter b for toggling Blue LED \r\n");

logMsg(&huart1, "Enter s to generate SW interrupt \r\n");

logMsg(&huart1, "Enter t to start timer 3 \r\n");

logMsg(&huart1, "Enter w to trigger watchdog reset\r\n");

// Rest of the code goes here

/\* USER CODE END 2 \*/

}

Note: This is same as Mid-term.

1. Create a software interrupt and use one of the non-used IRQ – **5 points**
   1. You could any one but let’s use 48 for this assignment.

FMC\_IRQn = 48, /\*!< FMC global Interrupt

* 1. Enable the FMC\_IRQn in **MX\_GPIO\_Init()**
  2. Create another menu with character ‘s’, which will generate the software interrupt and you should print the message “ SW Interrupt detected”. Here are steps involved.
     1. When the key is pressed, you should enable STIR using FMC\_IRQn
     2. This will trigger the interrupt and callback method will be called
     3. Set a flag or do something else to print the message “ SW Interrupt detected”.

1. Create a method myDelay1() using timer2 and should take input in MilliSec – **5 Points** 
   1. Replace the existing HAL\_Delay() with myDelay1() for Blue LED

// Here is the method signature and feel free to implement the way you want to

void myDelay1(uint32\_t val) {

}

1. Create a method myDelay2() using SysTick and should take input in Millisecond – **6 Points**
   1. Replace the existing HAL\_Delay() with myDelay2() for Green LED

// Here is the method signature and feel free to implement the way you want to

void myDelay2(uint32\_t val) {

}

1. Use Timer 3 to count events – **6 points**
   1. Program the timer in **MX\_TIM3\_Init()** with values , which will expire every second.
   2. Create a menu with character ‘t’ to start the timer3
   3. You will need to implement HAL\_TIM\_PeriodElapsedCallback() method and set a flag here to know when 1 sec has lapsed.
   4. Count for 10 (equivalent to 10 sec) in main loop and stop the timer3 when count reaches 10.
   5. Print the log message - "Total counted timer3 event = %d\r\n"
2. Uncomment MX\_IWDG\_Init() code to test the watchdog – **6 Points**
   1. Program the prescaler, window and reload value for timeout of 0.5 sec.
   2. Pet the watchdog in main () code – to avoid board reset
      1. Make sure you don’t have any delay in main() code.
   3. There should be no reset and software should be working properly.
   4. Now simulate a failure by introducing a delay more than 0.5 sec to miss the watchdog pet.
   5. Create a menu with character ‘w’ and delay of 1 sec. Since 1 sec is more that 0.5 sec, SW will miss the pet and unit will reset.
3. Uncomment MX\_RTC\_Init() to test RTC Alarm – **6 Points**
   1. Set the alarm for hour 0, minute 1
   2. Implement callback and set the flag to detect the alarm
   3. 1 minute after start of the board , alarm should be set.
   4. Display a message “ RTC alarm A detected”