DEPARTMENT OF COMPUTER SCIENCE AND ELECTRONIC ENGINEERING

ASSIGNMENT COVER SHEET

Module Number	CE865
Module Supervisor	Dongbing Gu
Assignment Number	1
Deadline date and time	Please see the deadline on FASER
Assignment Weighting	20%

You should submit your assignment using the **Electronic Submission System (FASER)** by the above deadline.

You must complete your assignment independently. Please refer to the Postgraduate Students' Handbook for details of the Departmental policy regarding submission and University regulations regarding plagiarism.

The assignment mark and feedback sheet will be returned via the Electronic Submission System **no later than three weeks** after the assignment submission date.

CE865 Assignment 1

Autumn Term 2021

1. Objectives

The objectives of this assignment are to demonstrate an understanding of:

1. Programming a microcontroller using the C programming language.
2. The use of the parallel I/O port of a microcontroller.
3. The use of the interrupt system of a microcontroller.

2. ARM Development Boards and Software Libraries

There are a number of laboratory sessions (Labs 1 to 5) you need to attend before attempting this assignment. These laboratory sessions help you to use the ARM development boards and the associated software. Extra time may be needed for some laboratory sessions. In many cases example source code is provided for the laboratory tasks. You should familiarise yourself with them before you start any programming.
You will need to read the ARM Development Board and Software Library manuals first. These can be found on the CE865 course webpage on Moodle under Downloads.

3. The Task

The objective of this assignment is to implement a reaction test timer using the ARM development board.

3.1. Operation

The reaction timer displays the time it takes for the user to react to an LED being illuminated. It displays the time in seconds to an accuracy of one thousandth of a second.

1. On start-up or after a reaction test has completed, LED 1 should flash to indicate Button 1 is to be pressed to start the next reaction test.
2. When Button 1 is pressed, the display is zeroed; all the LEDs are turned RED and begin to count down (turn off) at the rate of one per second. The count down can be from left to right or right to left.

3. When the last LED turns off a random LED is lit (turned GREEN) indicating the button that should be pressed to stop the count on the display. At the same time the display begins counting, showing the elapsed time.
4. When the indicated button is pressed the display stops counting. The timer is ready for the next reaction test (returns to step 1).

If the user presses any button other than the button indicated by the LED the count should continue. If the user fails to react, the count stops at 9.999 seconds.

If the user presses one or more buttons before the countdown of LEDs finishes, the countdown is suspended until all the buttons are released.

Video of the whole operation: Look *Assignment 1 – video* under *Coursework* on Moodle to have a better understanding of what you need to achieve.

3.2. Method

The reaction timer should use the Programmable Interval Timer (PIT) to ensure an accurate time period and the SevenSegmentDisplay library to turn the OLED display into a 4-digit seven-segment display device. *Lab 5 has introduced these concepts so would be good to revisit it if needed.*

You will need to create or use functions to read the state of the Buttons and set the colour of the LEDs. If you wish you could use the provided Button and LED libraries. *You have used these libraries through various labs (Lab 1 to Lab 5) so would be good to revisit them if needed.*

3.3. Assessment

You will be assessed on the source or pseudo code you submit. Before the deadline, you are asked to demonstrate your code to the modular supervisor or GLA in the lab for on campus students. For remote students, you may be asked to explain your work as part of the assessment. When you are ready for demonstration, just let us know.

Marks will be awarded as follows:

- Q1: Well Commented Code (10%)

  All the functions and other components in the code should be well commented, describing their purpose and operation. Specially you need to provide detailed

comments on how to implement each part of Display Functionality (Q3) and Button/LED Interface (Q4)

- Q2: Well Structured Code (10%)

  All the code (source or pseudo) produced should be well structured and use appropriate layout and control statements. Use of 'cut and paste' to replicate statement rather using a more recognised approach will result in a **lower** mark.

- Q3: Display Functionality (30%)

  The marks for the Display Functionality logics are made up as follows:

  Use of PIT interrupts (5%)

  Display of elapsed time (5%)

  Reaction Time count stops at 9.999 (10%)

  Accuracy of Reaction Time count (10%)

- Q4: Button/LED Interface (40%)

  The marks for the Button/LED Interface logics are made up as follows:

  Implementation of Button functions (10%)

  Implementation of LED functions (10%)

  Flashing LED start indicator (10%)

  'False Start' detection stops countdown (10%)

- Q5: Implementation Quality (10%)

  This will evaluate your ways to implement the functionality. A higher mark will be awarded for novel implementations.

3.4. Sample Example

Problem: Design a program for the following functionality. A single RED led should be running backwards and forwards across the row of LEDs towards the bottom of the board. The LEDs running should stop when Button 8 is pressed.

Expected Solution:

SolutionWithSource.c:

```
/*
 * Header files giving access to addition ARM main board functionality
 */
#include <board/LED-lib.h> /* Provide access to LEDs */



#include <board/Button-lib.h> /* Provide access to buttons */

/*
 * Define the direction codes for the movement of the LEDs.
 */
#define LEFT  0
#define RIGHT 1

/*
 * Software routine to delay a number of milliseconds
 * All timings are approximate.
 */
void delay_ms(short ms)
{
    volatile short loop;

    while (ms-- > 0)
        for (loop = 0; loop < 2100; loop++);
}

/*
 * Main function - Program execution begins here!
 */

int main()
{
    /*
     * Declare a variable to store the number of an LED.
     */
    LEDnumber LEDpos, LeftEnd, RightEnd;

    short LeftRight;

    /*
     * First set the LED position variable to the first LED.
     */
    LEDpos = LED1;

    /*
     * Now set the end markers. This is the point at which the lit LED changes
     * direction.
     */
    LeftEnd = LED1;
    RightEnd = LED8;

    /*
```

```
     * Now set the direction of LED movement.
     */
    LeftRight = RIGHT;

    /*
     * This loop moves the lit LED along the row of LEDs until it hits the
end.
     * It then bounces back towards the other end.
     *
     * This loops continues until the right hand most button is pressed.
     */
    while(IsButtonReleased(BUTTON8))
    {

        /*
         * Display the lit LED for a short period of time
         */
        SetLEDcolor(LEDpos, RED);
        delay_ms(100);
        SetLEDcolor(LEDpos, OFF);

        /*
         * Has it reached the end yet? If so, change the direction of travel
indicator.
         */
        if (LEDpos == LeftEnd) LeftRight = RIGHT;
        if (LEDpos == RightEnd) LeftRight = LEFT;

        /*
         * Now move the LED to its new position.
         */
        if (LeftRight == RIGHT)
            LEDpos++;
        else
            LEDpos--;
    }

    return 0;
}
```

SolutionWithPseudo.c:

```
/*
 * Header files giving access to addition ARM main board functionality
 */
include LED-lib.h to provide access to LEDs
include Button-lib.h to provide access to buttons

/*
 * Define the direction codes for the movement of the LEDs.
 */
define LEFT as 0
define RIGHT as 1
```

```
/*
 * Software routine to delay a number of milliseconds
 * All timings are approximate.
 */
function delay_ms(argument ms)
{
    Put a pre-computed delay of ms milliseconds;
}

/*
 * Main function - Program execution begins here!
*/

In the main function
{
    /*
     * Declare a variable to store the number of an LED.
     */
    Variable declared: LEDpos;
    //Declare variables to hold left end, right end and direction

    Variables declared: LeftEnd, RightEnd, LeftRight;

    /*
     * First set the LED position variable to the first LED.
     */
    LEDpos = Led1;

    /*
     * Now set the end markers. This is the point at which the lit LED changes
     * direction.
     */
    LeftEnd = Led1;
    RightEnd = Led8;

    /*
     * Now set the direction of LED movement.
     */
    LeftRight = RIGHT;

    /*
     * This loop moves the lit LED along the row of LEDs until it hits the
end.
     * It then bounces back towards the other end.
     *
     * This loops continues until the right hand most button is pressed.
     */
    Loop until Button 8 is not pressed // use proper button library
    {
        /*
         * Display the lit LED for a short period of time
         */
        Set Led color of LEDpos to RED;
        Call delay_ms() to introduce a delay of 100 ms;
        Turn off the led at LEDpos;
```

```
        /*
         * Has it reached the end yet? If so, change the direction of travel
indicator.
         */
        if LEDpos is equivalent to LeftEnd
            Set the direction of led movement to right as LeftRight = RIGHT;
        if LEDpos is equivalent to RightEnd
            Set the direction of led movement to left as LeftRight = LEFT;

        /*
         * Now move the LED to its new position.
         */
        if LeftRight is equivalent to RIGHT



            Increment the current led position LEDpos;
        else
            Decremnet the current led position LEDpos;
    }
}
```

4. Submission

The deadline for submitting the code is on FASER. You should submit one file containing the source for on campus students or pseudo codes (mixing of source and pseudo is also allowed) for remote students.

No extensions of the deadlines will be given; if, for any reason, you do not submit your code there will be no record of the submission time and you will automatically get a zero mark.

I strongly encourage you to submit your work through the proper channel (electronic submission) and on time, even if it is incomplete. Some credit is always better than none.

This assignment is to be done individually, i.e. whatever you hand in must be your own individual work. Any software or any other materials that you use in this assignment, whether previously published or not, must be referred to and properly acknowledged.