DESN2000 (Computer Engineering) 2026 T2

Lab sheet 2 (weeks 3 and 4)

Last edited: 10/06/2026

Make reasonable assumptions if not explicitly stated, and state such assumptions to your demonstrator when getting marked.

Task 1 (10%)

Follow the debugging tutorial available under lab 2 in WebCMS. Demonstrate to your tutor, how you will debug an example bug using debugging tools available in cubeIDE and answer any questions. Yes, this is an easy mark. But we want to ensure you all learn debugging methods so that life will become easier in the long run.

Task 2 (15%)

Write a programme that displays the following on the LCD.

Hello
World

Check the week 2 Thursday lecture on implementing functions such as `LCD_PutNibble` and `LCD_Pulse`. You may use the following function prototypes:

```
void LCD_PutNibble(uint8_t nibble);
void LCD_SendData(uint8_t c);
void LCD_SendCmd(uint8_t c);
void LCD_Pulse();
void LCD_SendStr(char *str);
```

You may use #define statements to define LCD controller instructions with sensible names. e.g.:

```
#define LCD_CLEAR_DISPLAY 0b00000001
```

Make sure that this exists: `__HAL_RCC_GPIOD_CLK_ENABLE()`

It would be good to have LCD code in a separate set of files (lcd.c and lcd.h, for example) so you can easily reuse them in your project later.

You may want to implement the LCD initialisation code from scratch if you are courageous. But if you need some guidance, refer to the following for initialising the LCD.

```
void LCD_init(){

  // 1. wait for enough time to stabilise
  HAL_Delay(50);

  // 2. send command 0011 (function set) and wait for >=4.1 ms (enough wait inside
the pulse)
  LCD_PutNibble(0b0011); LCD_Pulse();

  // 3. send command 0011 (function set) again and wait for >=100 us
  // fill this

  // 4. send command 0011 (function set) again
  // fill this

  // 5. send command 0010 to set to 4-bit bus mode
  // fill this

  // 6. send command 0010 1100 (function set: 4-bit mode, 2-lines, 5x8 font)
  LCD_SendCmd(0b00101100);

  // 7. Send command 0000 1000 to display ON/OFF
  LCD_SendCmd(0b00001000);

  // 8. Send command to clear the display
  // fill this

  // 9. Send command set entry mode (increment cursor, no display shift)
  // fill this

  // 10. send command 0000 1111 to display on, cursor on, blink on
  // fill this

}
```

Task 3 (20%)

Write a programme that displays the relevant character on the LCD when a key on the keypad is pressed. One key press should only produce one character – debounce! When the first row of the LCD is full, you should go to the second row. When the second row is full, stop updating the LCD. When SW1 is pressed, clear the display and start from the top left of the LCD.

It is recommended that you write the keypad scanning code inside a function like below (could be in a separate pair of files as well, keypad.c and keypad.h for example), so that you can reuse it across later exercises.

```
uint8_t scan_keypad(){

    return c;

}
```

Task 4 (25%)

Implement a programme that doubles a 16-bit signed integer counter when the SW1 button is pressed, subtracts 10 when the SW2 button is pressed, sets to the smallest possible value when the SW3 button is pressed and sets to the largest possible value when the SW4 button is pressed. The counter must be displayed on the LCD. For this task, you MUST use external interrupts for the buttons whenever possible. This should also saturate at the limits of a 16-bit signed integer.

Task 5 (30%)

Implement a simple calculator using the keypad and the LCD. The keys A, B, C and D should be used for +, -, x and / operations, respectively. The # key is to be used as the = operation. To clear the display use * key. Make sure the buttons are debounced appropriately.

See the following example:

1+2-3/2+2=
3.5

Assume that the expression on row 1 is not longer than 16 characters. Use the standard BODMAS order of operations. You can assume that the input numbers are integers (both positive and negative), but the outputs may be floats. Handle errors such as invalid syntax and division by 0, while giving an appropriate error message. When the answer is longer than 16 digits (i.e. cannot be fully shown on the LCD), show an error indicating that it is truncated.