ELEC 335 - Lab #5

Objective

The objective of this lab is to
- practice C language, and write basic programs,
- work with interrupts and utilize external interrupts,
- work with pulse width modulation,
- work with serial communication protocols.

Setup

Install STM32CubeIDE and implement problems on the breadboard. For that, use the **`blinky`** project from the g0 project repo (`https://github.com/fcayci/stm32g0`) and follow `https://youtu.be/UYk_NAnDJlw`. Do not use **standalone** for any problem / lab. That is just for demonstration purposes. You can use one of the undergraduate laboratories for using the scope and measuring the necessary voltages if needed.
- **For each coding problem**, you are required to create a flowchart that shows software operation and a block/connection diagram that shows how things are connected and what modules are in use. You can use online tools to do that. One example is lucidchart (`https://lucid.app`)
- **All codes should be written in C.**
- If a specific pin number is not stated, you are free to pick any appropriate pin to connect your components. Make sure to include this in your block/connection diagram.

Submission

You should submit the following items organized in a folder:
- **Lab report** - Written in English. PDF file. Should include
  - a cover page
  - for each problem
    - a flow chart
    - a block/connection diagram
    - pictures/photos if any/requested
    - if oscilloscope measurements are requested, detailed explanation of results in addition to photographs or video links
    - code listing
    - at least one paragraph explanation/comment about that problem, code listing
  - final lab conclusion about what you've learned.
- **Source files** - should have proper comments and pin connections if there are any external components.
- **Elf files** - generated elf file with debug information.

Problems

Problem 1 [15 pts]. In this problem, you will connect your board to the PC using UART protocol and loopback all the data that is sent from the PC back. For this you will need to create an initialization routine for UART, then create send and receive functions.

- You should not use interrupts for this problem.
- Basically create two functions as given below

  ```
  void uart_tx(uint8_t c);
  uint8_t uart_rx(void);
  ```

  and in your main loop, call them like

  ```
  while(1) {
     uart_tx(uart_rx());
  }
  ```

- Observe and verify the packets and baud rate using an oscilloscope.

Problem 2 [15 pts]. In this problem, implement Problem 1 using interrupts. Setup UART interrupt line to receive the character.

- Observe and verify the packets and baud rate using an oscilloscope.
- Explaining the difference of using interrupt vs. no interrupt for UART.
- Bonus - Try to implement the interrupt for the UART tx line as well.

Problem 3 [20 pts]. In this problem, you will implement a PWM signal and drive an external LED using varying duty cycles. Your LED should display a sinusoidal pattern. The sinusoidal period should be 20 ms.

- Look at the oscilloscope to see the PWM signal.
  - What happens when you do AC coupling vs. DC coupling?
- After you set up and get your PWM working with the LED, replace the LED with a speaker and see what happens. (Make sure to keep the series resistor)

Problem 4 [50 pts]. In this problem, you will work on implementing a simple tone generator utilizing Timer, PWM and External Interrupt modules and use a keypad, a speaker, and SSDs.

- Connect a keypad to your microcontroller, pick a block of musical notes, and assign a tone for each key on the keypad.
- One of the keys should be silence (rest).
- Tones will be played using PWM by changing the period at 50% duty cycle.
- Design an amplifier, and connect a speaker with adjustable gain using a pot.
- When a key is pressed, the relevant tone should play indefinitely.
- Connect your SSD to display the tone that is being played. You can display the frequency being played. (440, 480, ...) OR you can display the tone being played (A4, B4, C4, ...).

Appendix A - Setting up UART and creating a print() function

Pins PA2 and PA3 are connected to the ST/Link Debugger IC on the back which gets passed to the PC over USB as a Virtual COM port. This can be seen from the schematic of the board. First, we need to determine the USART modules PA2 and PA3 are connected to. From the stm32g031k8 datasheet, we can see that we have two USART modules named USART1 and USART2.

Looking at Table 13 - Port A alternate function mapping, we can also see that PA2 and PA3 are connected to the USART2 module using Alternate Function 1 mode (AF1). Now that we have the relevant info, we can start initializing steps. Check the RCC, GPIO and USART sections from rm0444 reference manual for exact registers and relevant bits.

1. enable GPIOA and USART2 module clocks from RCC
2. enable and setup pins PA2 and PA3 as alternate function for USART2 module
3. enable receive and transmit from USART2 module
4. set baud rate to 9600 bps assuming clock is running at 16Mhz
5. enable UART module

At this point, if we open a serial port from PC, and send a character from MCU, we should be able to see. Let’s write some helpers for this purpose.

6. implement a `printChar` function given as below. Now you can just call `printChar('a');` from your main loop and this should be displayed on your PC terminal
7. implement a `_print` function that will call the `printChar` function for the given string in a for loop given as below. Now you can call `_print(0, "Hello\n", 6);` and it should be displayed on your PC terminal
8. implement a print function that will calculate the number of characters in a string and call `_print` function given as below. Now you can just call `print("Hello\n");` and it should be displayed on your PC terminal

```
 // step 6 skeleton
 void printChar(uint8_t c) {
   // write c to the transmit data register
   // wait until transmit is complete
 }

// step 7 skeleton
 int _print(int f, char *ptr, int len)
 {
   (void)f; // don't do anything with f
   // call printChar within a for loop len times
   return len; // return length
}

 // step 8 skeleton
 void print(char *s) {
```

```
// count number of characters in s string until a null byte comes `\0`
  _print(0, s, length);
}
```

In order to connect to your MC and see these, you can open a serial port from your PC. When you connect the device, it should create a COM port for Windows and /dev/tty* for Unix-like systems (ACM / USB). You can use STM32CubeIDE.

1. From the Console window, press the window icon with plus symbol, then choose Command Shell Console
2. For connection type, choose Serial Port
3. Hit new and create a connection. Give a connection name, choose the correct uart setup, and choose serial port. If your device is connected, a COM port should show up here. Choose that.
4. Choose encoding as UTF-8 and hit OK.

You should now have a serial connection to your board.

Appendix B - Setting up PWM

1. First, we need to pick a pin which is capable of timer functionality. From the stm32g031k8 datasheet pinout, pick one that has a TIMx_CHy functionality listed and from Table 13/14 find the AF number. In this case x is the timer module that we will use and y is the channel that we will activate for PWM.
2. Setup the TIMx module like you would normally do, but choose PWM mode 1 from the CCMR register for the relevant y channel. (CCMR1 has control over channels 1/2, and CCMR2 has control over channels 3/4)
3. Enable the Capture/Compare output from the relevant bit in the CCER register.
4. ARR register is the period for PWM and CCRy register is the duty cycle for the PWM.
5. Everytime full period happens, it should interrupt and you should decide on the next duty cycle to send.
6. In the interrupt routine, make sure to clear the status register of the timer