FALL 2022 ELEC335 MICROPROCESSORS LAB

Prepared By
1) 1901022038 – Selen Erdoğan
2) 200102002043 – Senanur Ağaç
3) 1901022050 - Merve Tutar

INTRODUCTION

The purpose of this lab is to enable us to communicate from PC to MCU and use bidirectional data transmission and parsing. We used C language for the problems, unless some parts require inline assembly.

PROBLEM 1

Problem 1. In this problem, you will connect your board to the PC using UART protocol. For this you will need to create an initialization routine for UART, and create send and receive functions. **You should not use interrupts for this assignment.**

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());
}
```

SOLUTION 1

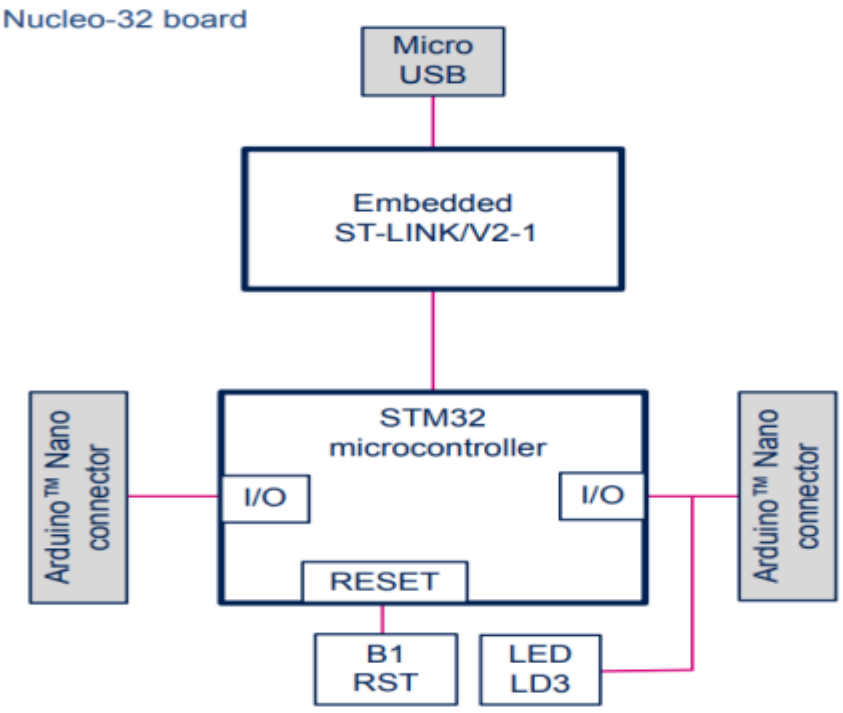

Figure 1. Block Diagram

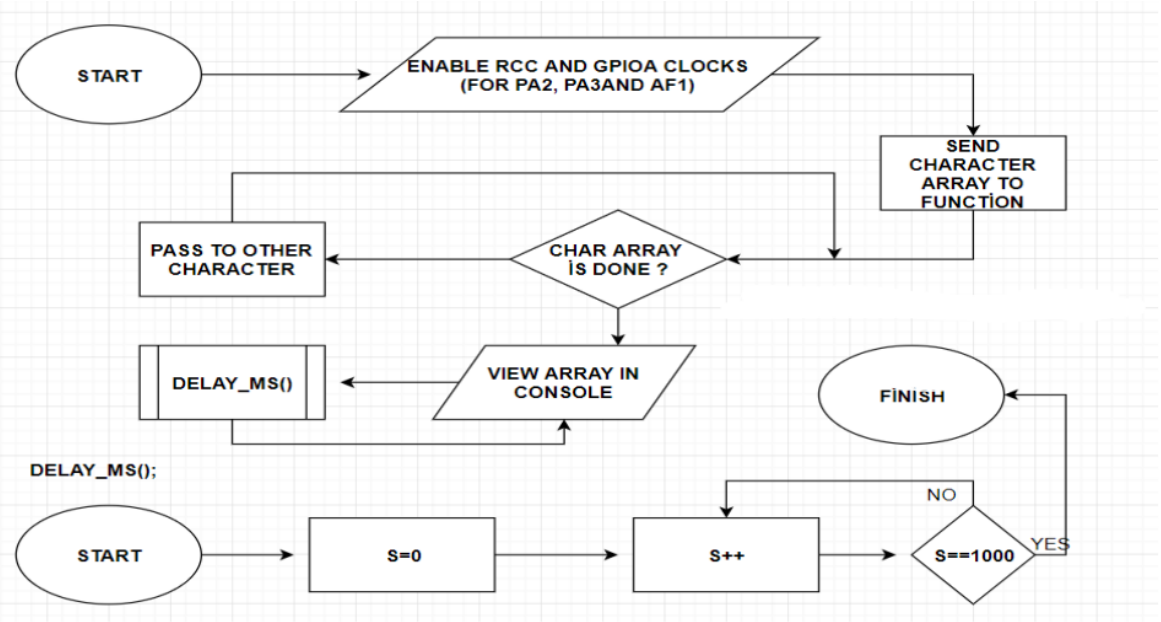

Figure 2. Flowchart

```
– main.c–
#include "stm32g0xx.h"
#include "bsp.h"
#define LEDDELAY 1600000
void delay(volatile uint32_t);
int main(void) {
//led_init();
//button_init();
//SysTick_Config(SystemCoreClock / 1000);
 BSP_UART_init(9600);
 while(1) {
//print("\e[1;1H\e[2J"); konsolu temizliyor?
 print("\n Sena Selen Merve interrupt ile çalışıyor ");
 delay_ms (50000);
 }
 return 0;
}

– bsh.h–
#ifndef BSP_H_
#define BSP_H_
void delay(volatile uint32_t s);
void BSP_UART_init(uint32_t baud);
void print(char *buf);
void _print(int fd, char *ptr, int len);
void printChar(uint8_t c);
```

```
void USART2_IRQHandler(void);

#endif

—bsh.c—
#ifndef BSP_H_
#define BSP_H_
void delay(volatile uint32_t s);
void BSP_UART_init(uint32_t baud);
void print(char *buf);
void _print(int fd, char *ptr, int len);
void printChar(uint8_t c);
void USART2_IRQHandler(void);

#endif
```

➔ Before starting the experiment, the datasheet of the microprocessor was examined. During the experiment, the part about the UART was read from the RM0444 file and the modules and registers that should be used were activated. In this way, Reset and Clock Control connections were made with the register assignments that should be made in accordance with the UART protocol. Usart2 module from UART protocols was used. Since the PA2 and PA3 pins are connected to the usart2 module of the chip, these pins are used in the code. The alternating function map was examined and the PA2 and PA3 pins to which it was connected were activated in the software.

PROBLEM 2

In this problem, you will implement a PWM signal and drive an external LED using varying duty cycles. Your LED should display a triangular pattern, meaning the brightness should linearly increase and decrease in 1 second.

Bonus - After you setup and get your PWM working with the LED,replace the LED with a speaker and see what happens. (Make sure to keep the series resister)

SOLUTION 2

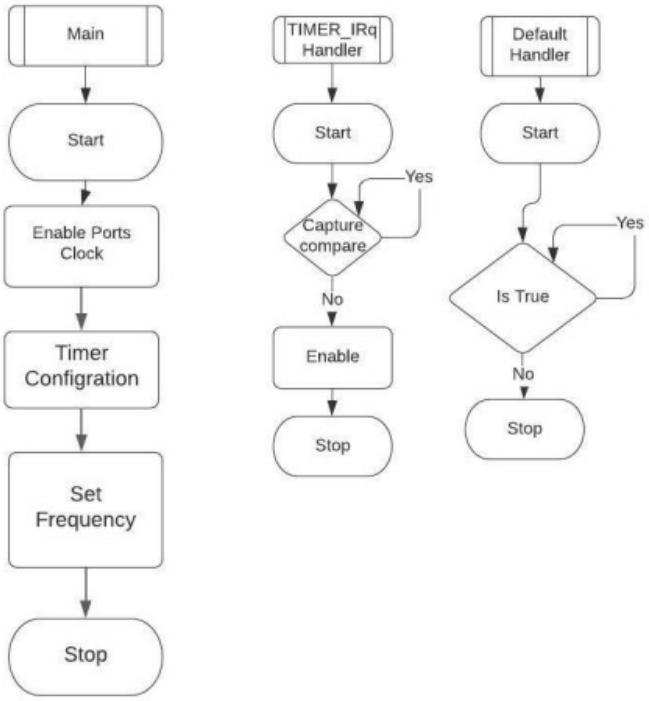

Figure 3. Flowchart

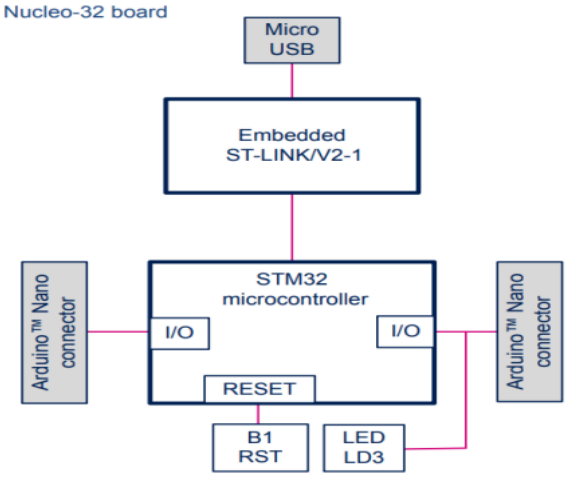

Figure 4. Block Diagram

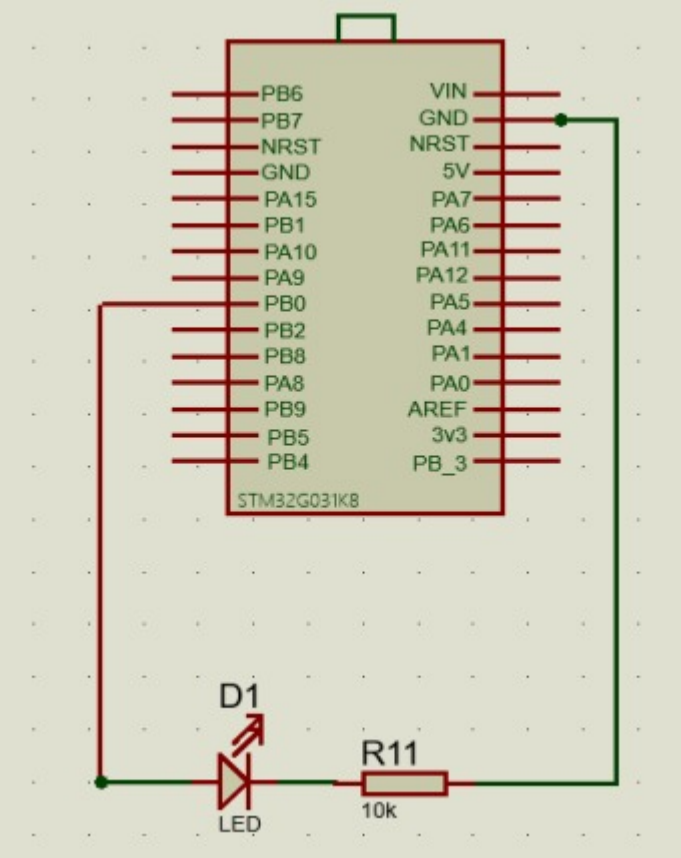

Figure 5. Schematic Diagrams

```
#include "stm32g0xx.h"
volatile uint32_t counter = 0;
void delay(volatile uint32_t s);
uint32_t sample[256] = {

1000,1025,1049,1074,1098,1122,1147,1171,1195,1219,1243,1267,1290,1
314,1337,1360,1383,1405,1428,1450,1471,1493,1514,1535,1556,1576,15
96,1615,1634,1653,1672,1690,1707,1724,1741,1757,1773,1788,1803,181
8,1831,1845,1858,1870,1882,1893,1904,1914,1924,1933,
1942,1950,1957,1964,1970,1976,1981,1985,1989,1992,1995,1997,1999,2
000,2000,2000,1999,1997,1995,1992,1989,1985,1981,1976,1970,1964,19
57,1950,1942,1933,1924,1914,1904,1893,1882,1870,1858,1845,1831,181
8,1803,1788,1773,1757,1741,1724,1707,1690,1672,1653,
1634,1615,1596,1576,1556,1535,1514,1493,1471,1450,1428,1405,1383,1
360,1337,1314,1290,1267,1243,1219,1195,1171,1147,1122,1098,1074,10
49,1025,1000,975,951,926,902,878,853,829,805,781,757,733,710,686,6
63,640,617,595,572,550,529,507,
486,465,444,424,404,385,366,347,328,310,293,276,259,243,227,212,19
7,182,169,155,142,130,118,107,96,86,76,67,58,50,43,36,30,24,19,15,
11,8,5,3,1,0,0,0,1,3,5,8,11,15,19,24,30,36,43,50,58,67,76,86,96,10
7,118,130,142,155,169,182,197,212,227,243,259,276,293,310,328,347,
366,385,404,424,444,465,486,507,529,550,572,595,617,640,663,686,71
```

```
0,733,757,781,805,829,853,878,902,926,951,975
};
void setFrequency(uint32_t freq){
TIM3->PSC = ((4000)/freq)-1; //2000=ARR value 8_000_000 is Timer
clock and freq is pwm frequency
}

void GPIO_Config(void){
RCC->IOPENR |= (1U << 1); //Enable clock for port B
GPIOB->MODER &= ~(3U << 0);
GPIOB->MODER |= (2U << 2*0); //GPIOB alternate function mode //PB0
//Alternate function low register AF1 //TIM3_CH3
GPIOB->AFR[0] |= (1U << 4*0);
}

void TIM3_IRQHandler(){
           setFrequency(50);
TIM3->CCR3 = sample[counter];
           if(counter >= 255){
                    counter = 0;
            }
           else{
                     counter++;}
           }
void TIM3_Config(){

RCC->APBENR1 |= (1U << 1); //Enable TIM1 clock
TIM3->CR1 |= (0U << 4); //Direction --> Upcounter
//TIM3->CR1 |= (0U << 6);//Center-aligned mode /Edge-aligned
TIM3->CR1 |= (3U << 5);//Center aligned mode/Edge-aligned ==> 3U
yapınca TRIANGULAR ELDE ETTIK.
TIM3->CR1 |= (0U << 9); //Clock division=1
TIM3->CR1 |= (0U << 8); //Clock division=1
TIM3->DIER |= (1U << 0); //TIM3 interrupt enable
TIM3->CCMR1 |= (0U << 0);//CC1 channel is configured as output
TIM3->CCMR1 |= (0U << 1);//CC1 channel is configured as output
TIM3->CCMR1 |= (6U << 4); //PWM MODE1
TIM3->CCMR1 |= (0U << 9); //CC2 channel is configured as output
TIM3->CCMR1 |= (0U << 8); //CC2 channel is configured as output
TIM3->CCMR1 |= (6U << 12); //PWM MODE for channel2
//Capture/Compare 1&2 output selected
// Capture/Compare 1&2 PWM1 selected
TIM3->CCMR2 |= (0U << 0); //Capture compare 3 selection
TIM3->CCMR2 |= (0U << 1); //Capture compare 3 selection
TIM3->CCMR2 |= (6U << 4); //PWM MODE
TIM3->CCMR2 |= (0U << 8); //Capture compare 4 selection
TIM3->CCMR2 |= (0U << 9); //Capture compare 3 selection
TIM3->CCMR2 |= (6U << 12); //PWM MODE
// Capture/Compare 3&4 output selected
// Capture/Compare 3&4 PWM1 selected
TIM3->CCER |= (1U << 0); //Capture Compare 1 output enable
TIM3->CCER |= (1U << 4); //Capture Compare 2 output enable
```

```
TIM3->CCER |= (1U << 8); //Capture Compare 3 output enable
TIM3->CCER |= (1U << 12); //Capture Compare 4 output enable
TIM3->ARR = 2000;
TIM3->CR1 |= (1U << 0); //TIM3 enable
NVIC_SetPriority(TIM3_IRQn, 2);
NVIC_EnableIRQ(TIM3_IRQn);
}
int main(void) {
          TIM3_Config();
          GPIO_Config();
          while(1) {}
          return 0;
}
void delay(volatile uint32_t s) {
            for(; s>0; s--);
}
```

PROBLEM 3

In this problem, you will implement a PWM signal and drive an LED at di_erent speeds. You will use keypad to set the duty cycle. Pressing 10# will set the duty cycle to 10% and 90# will set the duty cycle to 90%. Any unusal presses should be ignored. (i.e. 9F#)

• The current duty cycle should print every 2 seconds using UART
• The new duty cycle should also print after a button presses.
• Your buttons should not bounce.

SOLUTION 3

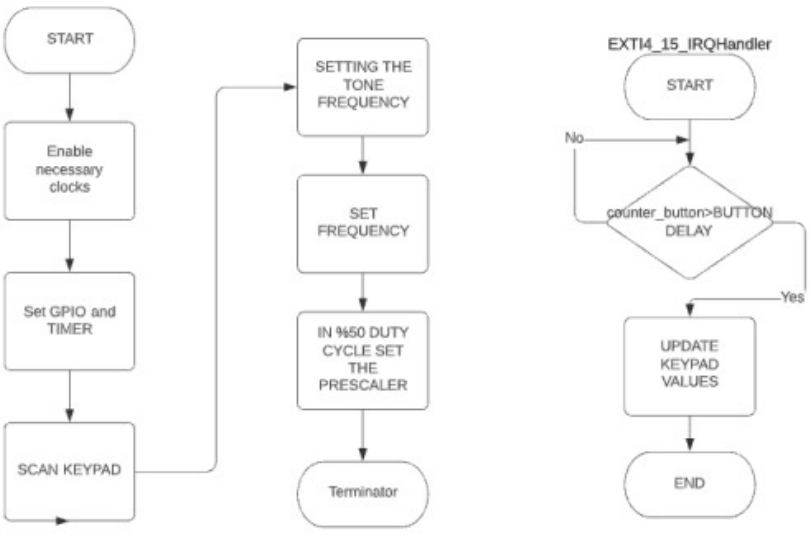

Figure 6. Flowchart

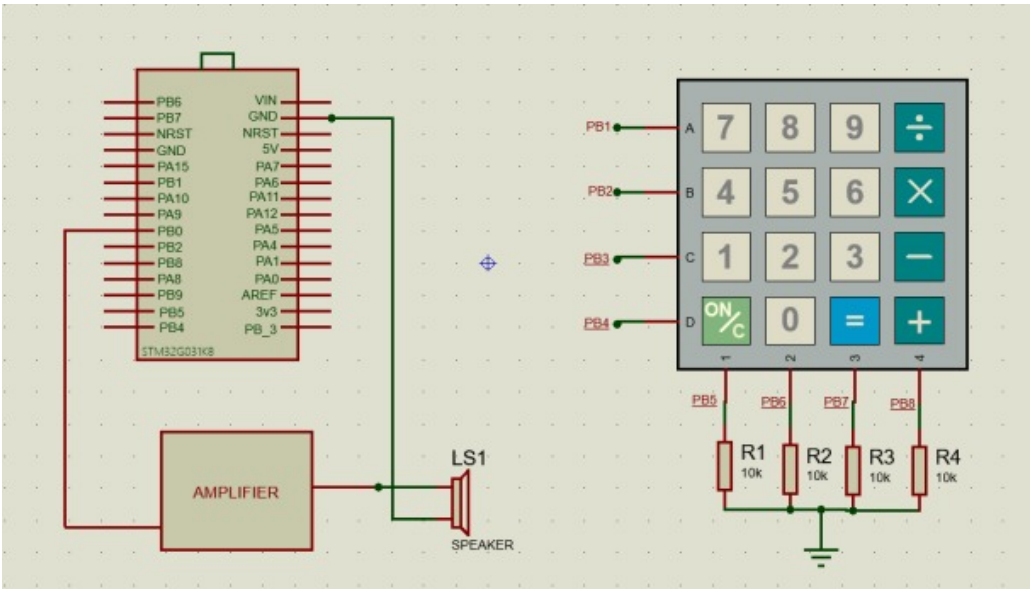

Figure 7. Schematic Diagrams

```
#include "stm32g0xx.h"
#define BUTTON_DELAY 100
volatile uint32_t counter_button = 0;
void delay(volatile uint32_t s) {
for(; s>0; s--);
}
void setFrequency(uint32_t freq){
TIM3->PSC = ((4000)/freq)-1; //2000=ARR value 8_000_000 is Timer
clock and freq is pwm frequency
}
void setTone(uint8_t tone){
switch(tone){
case 1:
setFrequency(16); //C0
break;
case 2:
setFrequency(21); //F0
break;
case 3:
setFrequency(30); //B0
break;
case 4:
setFrequency(41); //E1
break;
case 5:
setFrequency(61); //B1
break;
case 6:
setFrequency(164); //E3
```

```
break;
case 7:
setFrequency(329); //E4
break;
case 8:
setFrequency(1318); //E6
break;
case 9:
setFrequency(3951); //B7
break;
default:
setFrequency(0);
break;
}
}
void KeyPad(){
volatile uint32_t keypad_read = GPIOB->IDR;
keypad_read &= ~(1U << 0);
switch(keypad_read){
case 0x22 : //1
setTone(1);
break;
case 0x42 : //2
setTone(2);

break;
case 0x82 : //3
setTone(3);
break;
case 0x24 : //4
setTone(4);
break;
case 0x44 : //5
setTone(5);
break;
case 0x84 : //6
setTone(6);
break;
case 0x28 : //7
setTone(7);
break;
case 0x48 : //8
setTone(8);
break;
case 0x88 : //9
setTone(9);
break;
default:
setTone(0);
break;
}
}
```

```
void EXTI4_15_IRQHandler(){
if(counter_button >= BUTTON_DELAY)
{
KeyPad();
counter_button = 0;
}
EXTI->RPR1 = (0xF << 5);
}
void GPIO_Config(void){
RCC->IOPENR |= (1U << 1); //Enable clock for port B
GPIOB->MODER &= ~(3U << 0);
GPIOB->MODER |= (2U << 2*0); //GPIOB alternate function mode
//PB0
//Alternate function low register AF1 //TIM3_CH3
GPIOB->AFR[0] |= (1U << 4*0);
for(uint32_t i=1;i<9;i++){
if(i<5){
GPIOB->MODER &= ~(3U << 2*i);
GPIOB->MODER |= (1U << 2*i);
}
else{
GPIOB->MODER &= ~(3U << 2*i);
GPIOB->MODER |= (0U << 2*i);
GPIOB->PUPDR |= (2U << 2*i);
olatile uint32_t exticr_num;
if((i <= 3)){
exticr_num = 0;
}
else if((i >= 4) && (i <= 7)){
exticr_num = 1;
}
else if((i >= 8) && (i <= 11)){
exticr_num = 2;
}
else if((i >= 12) && (i <= 15)){
exticr_num = 3;
}
EXTI->EXTICR[exticr_num] |= (1U << 8*(i % 4));
EXTI->RTSR1 |= (1U << i);
EXTI->IMR1 |= (1U << i);
if((i <= 1)){
NVIC_EnableIRQ(EXTI0_1_IRQn);
}
else if((i >= 2) && (i <= 3)){
NVIC_EnableIRQ(EXTI2_3_IRQn);
}
else if((i >= 4) && (i <= 15)){
NVIC_EnableIRQ(EXTI4_15_IRQn);
}
}
}
}
```

```
//void TIM3_IRQHandler(){
//}
void TIM3_Config(){
RCC->APBENR1 |= (1U << 1); //Enable TIM1 clock
TIM3->CR1 |= (0U << 4); //Direction --> Upcounter
//TIM3->CR1 |= (0U << 6);//Center-aligned mode /Edge-aligned
TIM3->CR1 |= (3U << 5);//Center aligned mode/Edge-aligned ==> 3U
yapınca TRIANGULAR ELDE ETTIK.
TIM3->CR1 |= (0U << 9); //Clock division=1
TIM3->CR1 |= (0U << 8); //Clock division=1
//TIM3->DIER |= (1U << 0); //TIM3 interrupt enable
TIM3->CCMR1 |= (0U << 0);//CC1 channel is configured as output
TIM3->CCMR1 |= (0U << 1);//CC1 channel is configured as output
TIM3->CCMR1 |= (6U << 4); //PWM MODE1
TIM3->CCMR1 |= (0U << 9); //CC2 channel is configured as output
TIM3->CCMR1 |= (0U << 8); //CC2 channel is configured as output
TIM3->CCMR1 |= (6U << 12); //PWM MODE for channel2
/*
* Capture/Compare 1&2 output selected
* Capture/Compare 1&2 PWM1 selected
*/
TIM3->CCMR2 |= (0U << 0); //Capture compare 3 selection
TIM3->CCMR2 |= (0U << 1); //Capture compare 3 selection
TIM3->CCMR2 |= (6U << 4); //PWM MODE
TIM3->CCMR2 |= (0U << 8); //Capture compare 4 selection
TIM3->CCMR2 |= (0U << 9); //Capture compare 3 selection
TIM3->CCMR2 |= (6U << 12); //PWM MODE
/*
* Capture/Compare 3&4 output selected
* Capture/Compare 3&4 PWM1 selected
*/
TIM3->CCER |= (1U << 0); //Capture Compare 1 output enable
TIM3->CCER |= (1U << 4); //Capture Compare 2 output enable
TIM3->CCER |= (1U << 8); //Capture Compare 3 output enable
TIM3->CCER |= (1U << 12); //Capture Compare 4 output enable
TIM3->ARR = 2000;
TIM3->CR1 |= (1U << 0); //TIM3 enable
//NVIC_SetPriority(TIM3_IRQn, 2);
//NVIC_EnableIRQ(TIM3_IRQn);
}
int main(void) {
TIM3_Config();
GPIO_Config();
TIM3->CCR3 = 1000;
volatile uint32_t keypad_scan[] = {0x01,0x02,0x04,0x8};
volatile uint8_t counter_keypad = 0;
setTone(0);
while(1) {
GPIOB->ODR = 0;
GPIOB->ODR |= (keypad_scan[counter_keypad] << 1);
if(counter_keypad >= 3){
counter_keypad = 0;
```

```
}
else{
counter_keypad++;
}
if(counter_button <= BUTTON_DELAY){
counter_button++;
}
delay(1000);
}
return 0;
}
```