Tyler Wong, Brayden Daly

Lab Group G

EE 329 - 05 S’25

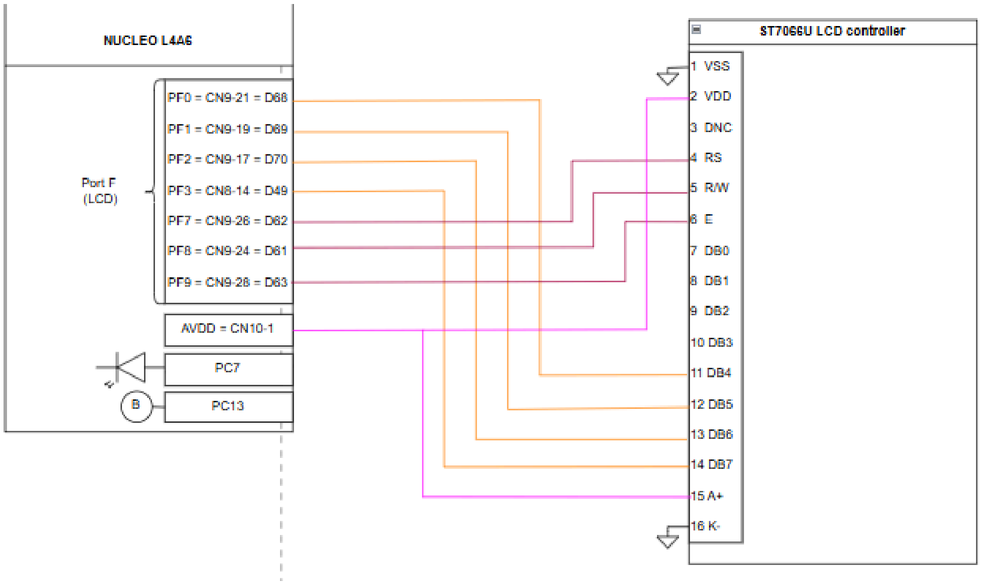
2025-Apr-30

Youtube Video: https://youtube.com/shorts/v21K_XSaoH0

Introduction:

This lab used STM32L4 timers and interrupts to generate precise 5 kHz PWM signals (25%/50% duty cycles), measured ISR timing via GPIO toggling, and built a reaction timer with LCD output. The code for the reaction timer is fully operational without any known bugs.

Wiring Diagram:



a.) Calculations

4 MHz clock period T_C = 1/(4M) = 250 ns

5 KHz square wave period T_S = 1/(5k) = 200 us

ARR count = T_S / T_C -1 = 200u / 250n - 1 = 800 -1 = 0x320 - 1

CCR1 (25% duty cycle) count = 0.25 * ARR count -1 = 200 -1 = 0xC8 -1

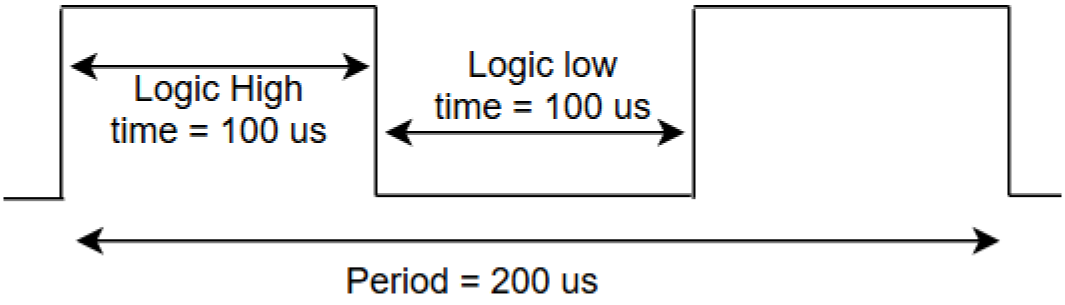


Figure 1. 5 kHz square wave sketch

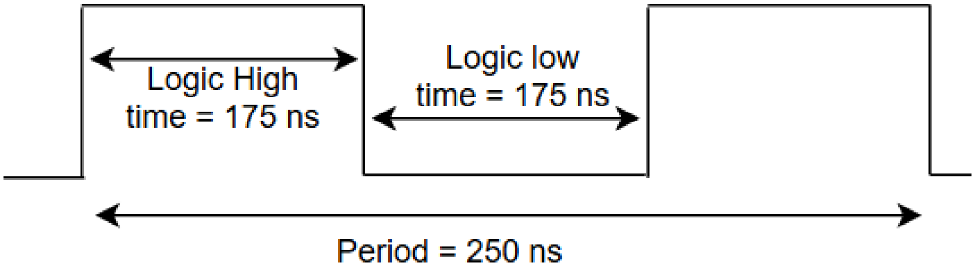
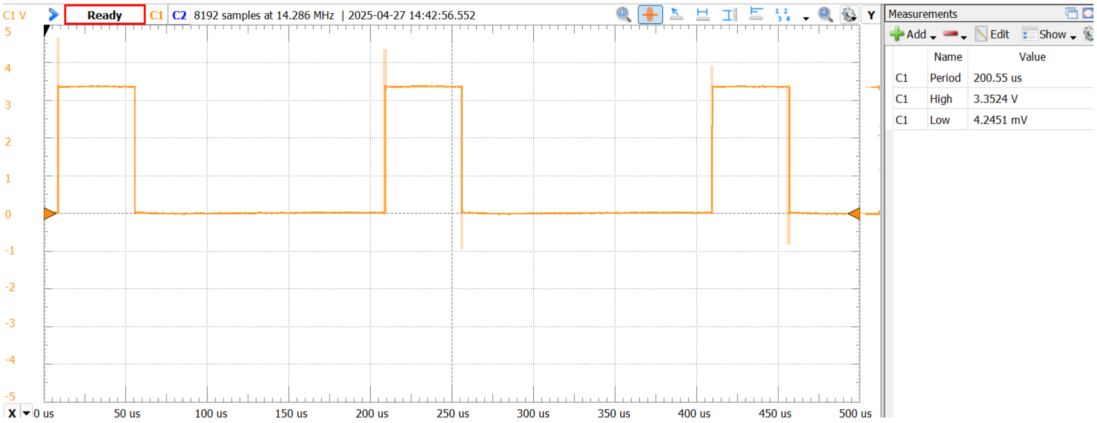


Figure 2. 4 MHz clock sketch

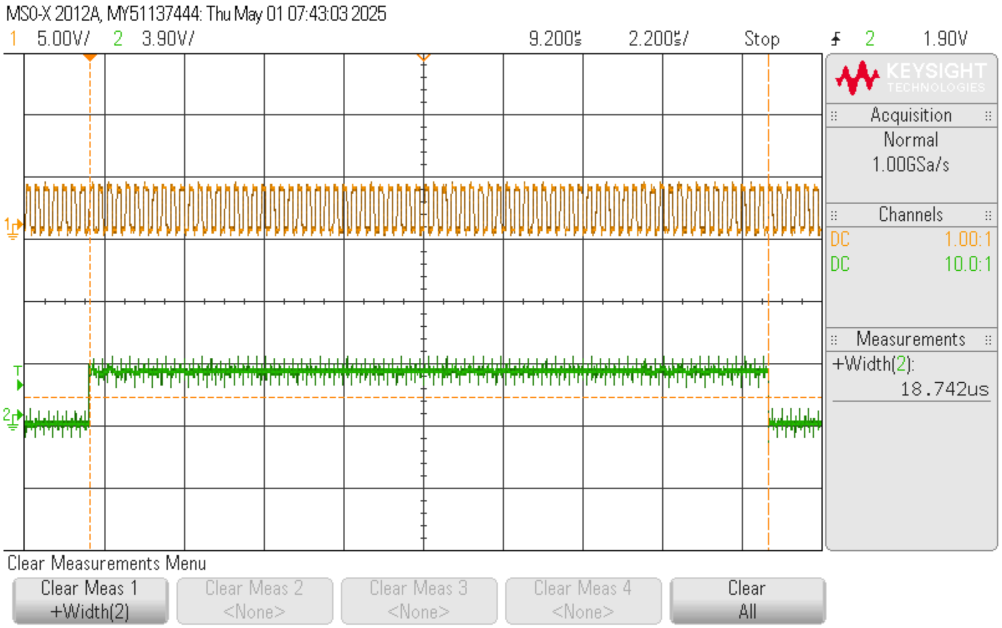
b.) Screen captures



8192 samples at 14.286 MHz | 2025-04-27 14:42:56.552

	Name	Value
C1	Period	200.55 us
C1	High	3.3524 V
C1	Low	4.2451 mV

Figure 3. 25% duty cycle square wave at 5 kHz.



MSO-X 2012A, MY51137444: Thu May 01 07:43:03 2025

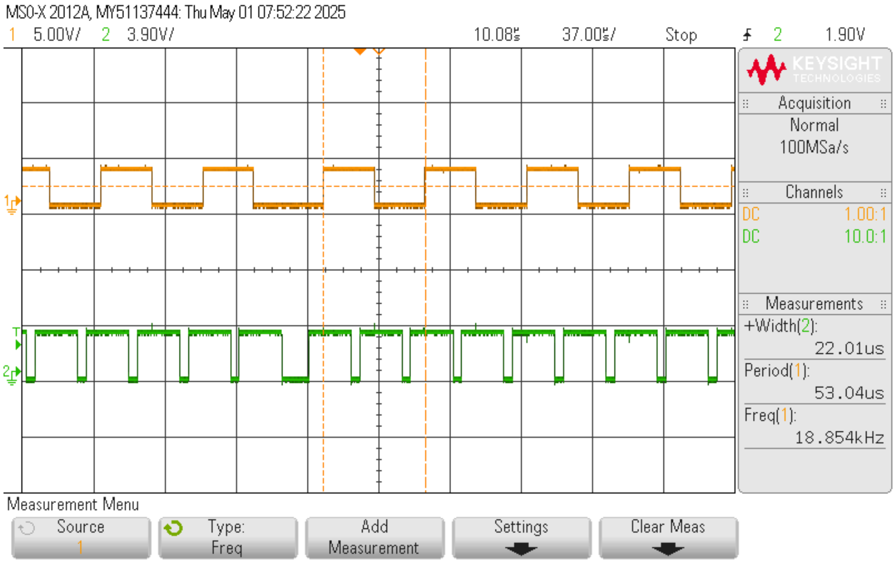
1 5.00V/ 2 3.90V/ 9.200₿ 2.200₿/ Stop ⨍ 2 1.90V

Acquisition
Normal
1.00GSa/s

Channels
DC 1.00:1
DC 10.0:1

Measurements
+Width(2):
18.742us

Figure 4. 4 Mhz Clock (yellow) and ISR execution timing (green).



MSO-X 2012A, MY51137444: Thu May 01 07:52:22 2025

1 5.00V/ 2 3.90V/ 10.08₿ 37.00₿/ Stop ⨍ 2 1.90V

Acquisition
Normal
100MSa/s

Channels
DC 1.00:1
DC 10.0:1

Measurements
+Width(2):
22.01us
Period(1):
53.04us
Freq(1):
18.854kHz

Figure 5. 50% duty cycle square wave at 5 kHz (yellow) and ISR execution timing (green). (Shortest CCR1 Pulse)

c.) observations

There were 75 MCO clock cycles within a single ISR execution time, as seen in Figure 2.

The lowest CCR1 value before ultimate failure was determined to be a count of 16. This does not correlate with the ISR execution time.

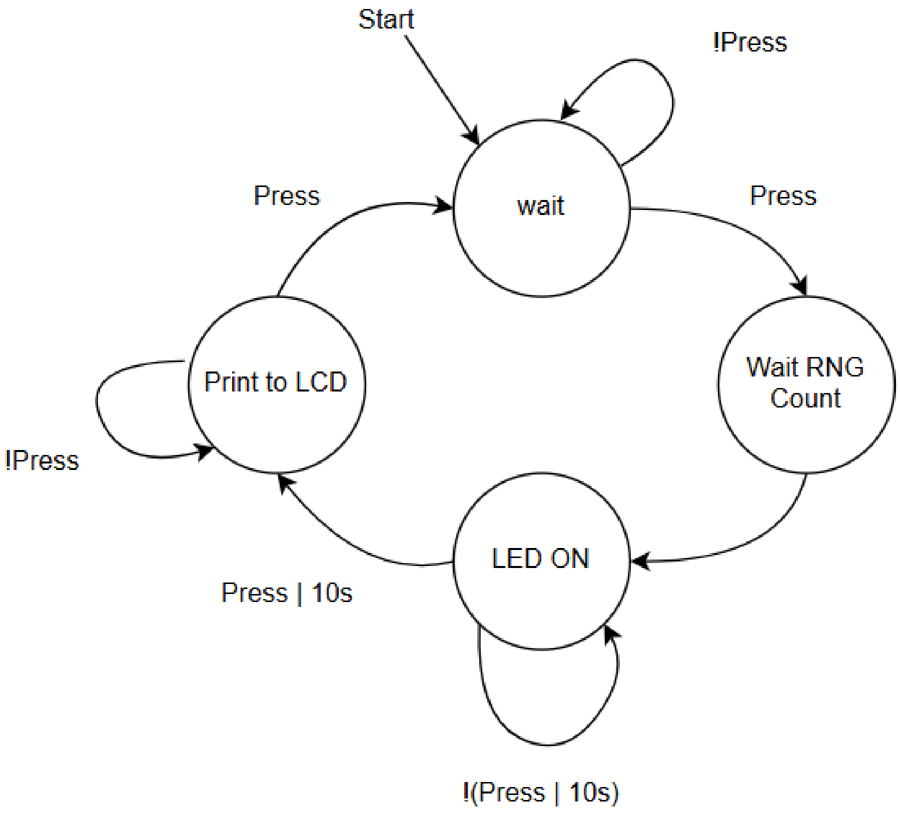


Figure 4. State transition diagram

Part B Code:

```
------------------------------------------------------------------------------

main.h
------------------------------------------------------------------------------

/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.h
 * @brief          : Header for main.c file.
 *                   This file contains the common defines of the application.
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
#include "stm32l4xx_hal.h"
void Error_Handler(void);
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define LD3_Pin GPIO_PIN_14
#define LD3_GPIO_Port GPIOB
#define USB_OverCurrent_Pin GPIO_PIN_5
#define USB_OverCurrent_GPIO_Port GPIOG
#define USB_PowerSwitchOn_Pin GPIO_PIN_6
#define USB_PowerSwitchOn_GPIO_Port GPIOG
#define STLK_RX_Pin GPIO_PIN_7
#define STLK_RX_GPIO_Port GPIOG
#define STLK_TX_Pin GPIO_PIN_8
#define STLK_TX_GPIO_Port GPIOG
#define USB_SOF_Pin GPIO_PIN_8
#define USB_SOF_GPIO_Port GPIOA
#define USB_VBUS_Pin GPIO_PIN_9
#define USB_VBUS_GPIO_Port GPIOA
#define USB_ID_Pin GPIO_PIN_10
#define USB_ID_GPIO_Port GPIOA
#define USB_DM_Pin GPIO_PIN_11
#define USB_DM_GPIO_Port GPIOA
#define USB_DP_Pin GPIO_PIN_12
#define USB_DP_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
#define LD2_Pin GPIO_PIN_7
#define LD2_GPIO_Port GPIOB
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
------------------------------------------------------------------------------

main.c
------------------------------------------------------------------------------
```

```c
/* USER CODE BEGIN Header */
/*******************************************************************************
* EE 329 A4 Interrupts & Timers
*******************************************************************************
* @file           : main.c
* @brief          :
* project         : EE 329 S'25 Assignment 4
* authors         : Tyler Wong (TW) - twong103@calpoly.edu ;
* version         : 0.1
* date            : 250416
* compiler        : STM32CubeIDE v.1.18.0
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
* REVISION HISTORY
* 0.1 230318 (TW) created, written
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
/* USER CODE END Header */
#include "main.h"
#include "LCD.h"
#include "Delay.h"
#include "Timer.h"
void SystemClock_Config(void);
int main(void) {
 HAL_Init();
 SystemClock_Config();

 // configure GPIO output pins
     RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
     GPIOC->MODER   &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1);
     GPIOC->MODER   |=  (GPIO_MODER_MODE0_0| GPIO_MODER_MODE1_0);
     GPIOC->OTYPER  &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1);
     GPIOC->PUPDR   &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1);
     GPIOC->OSPEEDR |=  (3 << GPIO_OSPEEDR_OSPEED0_Pos |
                                   3 << GPIO_OSPEEDR_OSPEED1_Pos);
// setup timer and clock
setup_TIM2(400);
setup_MCO_CLK();
 while (1) {
 }                                             // end of while (1)
}                                              // end of main
/* ---------------------------------------------------------------------------*/
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 /** Configure the main internal regulator output voltage
 */
 if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
 {
   Error_Handler();
 }
 /** Initializes the RCC Oscillators according to the specified parameters
 * in the RCC_OscInitTypeDef structure.
 */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
 RCC_OscInitStruct.MSIState = RCC_MSI_ON;
 RCC_OscInitStruct.MSICalibrationValue = 0;
 RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
 {
   Error_Handler();
 }
 /** Initializes the CPU, AHB and APB buses clocks
 */
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
 {
   Error_Handler();
 }
}
void Error_Handler(void)
{
 /* USER CODE BEGIN Error_Handler_Debug */
 /* User can add his own implementation to report the HAL error return state */
 __disable_irq();
 while (1)
 {
 }
 /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
 /* USER CODE BEGIN 6 */
 /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
 /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
------------------------------------------------------------------------------

Timer.h
------------------------------------------------------------------------------
```

```
/*
* Timer.h
*
*  Created on: Apr 30, 2025
*      Author: wongb
*/
#ifndef INC_TIMER_H_
#define INC_TIMER_H_
#endif /* INC_TIMER_H_ */
------------------------------------------------------------------------------

Timer.c
------------------------------------------------------------------------------

/* USER CODE BEGIN Header */
/********************************************************************************
* EE 329 A4 Timer INTERFACE
*******************************************************************************
* @file             : Timer.c
* @brief            : Timer and clock configuration
* project           : EE 329 S'23 Assignment 4
* authors           : Tyler Wong
*                   : Brayden Daly
* version           : 0.1
* date              : 250423
* compiler          : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target            : NUCLEO-L4A6ZG
* clocks            : 4 MHz MSI to AHB2
* @attention        : (c) 2023 STMicroelectronics.  All rights reserved.
*******************************************************************************
/* USER CODE END Header */
#include "main.h"
#include "Timer.h"
/* -----------------------------------------------------------------------------
* function : setup_TIM2( int iDutyCycle )
* INs      : CCR1 interrupt count
* OUTs     : none
* action   : Initializes TIM2 with infinite ARR and an input CCR1 count
* authors  : Tyler Wong (TW) - twong103@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void setup_TIM2( int iDutyCycle ) {
   RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;           // enable clock for TIM2
   TIM2->DIER |= (TIM_DIER_CC1IE | TIM_DIER_UIE);  // enable event gen, rcv CCR1
   TIM2->ARR = 0xFFFFFFFF;                         // ARR = T = counts @4MHz
   TIM2->CCR1 = iDutyCycle - 1;                    // ticks for duty cycle
   TIM2->SR &= ~(TIM_SR_CC1IF | TIM_SR_UIF);       // clr IRQ flag in status reg
   NVIC->ISER[0] |= (1 << (TIM2_IRQn & 0x1F));     // set NVIC interrupt: 0x1F
   __enable_irq();                                 // global IRQ enable
   TIM2->CR1 |= TIM_CR1_CEN;                       // start TIM2 CR1
}
/* -----------------------------------------------------------------------------
```

```
* function : setup_MCO_CLK(void)
* INs      : none
* OUTs     : none
* action   : Assigns the MCO clock to PA8 output
* authors  : Tyler Wong (TW) - twong103@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void setup_MCO_CLK(void) {
  // enable MCO, MCOSEL = 0b0001 to select SYSCLK = MSI (4 MHz source)
  RCC->CFGR = ((RCC->CFGR & ~(RCC_CFGR_MCOSEL)) | (RCC_CFGR_MCOSEL_0));
  // configure MCO output on PA8
  RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOAEN);
  GPIOA->MODER   &= ~(GPIO_MODER_MODE8);       // clear MODER bits
  GPIOA->MODER   |=  (GPIO_MODER_MODE8_1);         // MODE = alternate function
  GPIOA->OTYPER  &= ~(GPIO_OTYPER_OT8);        // push-pull output
  GPIOA->PUPDR   &= ~(GPIO_PUPDR_PUPD8);       // pullup & pulldown OFF
  GPIOA->OSPEEDR |=  (GPIO_OSPEEDR_OSPEED8);   // high speed
  GPIOA->AFR[1]  &= ~(GPIO_AFRH_AFSEL8);       // select MCO (AF0) on PA8 (AFRH)
}
/* -----------------------------------------------------------------------------
* function : TIM2_IRQHandler(void)
* INs      : none
* OUTs     : PC0 for 5 kHz, 50% duty cycle square wave & PC1 for IQR execution
*                 : time
* action   : Where the interrupts and executes GPIO outputs
* authors  : Tyler Wong (TW) - twong103@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void TIM2_IRQHandler(void) {
       GPIOC->BSRR = (GPIO_PIN_1);                // IQR execution time start
  if (TIM2->SR & TIM_SR_CC1IF) {      // triggered by CCR1 event
     TIM2->SR &= ~(TIM_SR_CC1IF);     // manage the flag
     GPIOC->ODR ^= (GPIO_PIN_0);      // Toggle PC0;
     TIM2->CCR1 += 400;                    // assign next CCR1 interrupt
    if(TIM2->CCR1 <= TIM2->CNT)      // check if CCR1 is in the past
     TIM2->CCR1 = TIM2->CNT + 400;         // reschedule the future
  }
  if (TIM2->SR & TIM_SR_UIF) {        // triggered by ARR event
     TIM2->SR &= ~(TIM_SR_UIF);       // manage the flag
     ;                                                   // does nothing
  }
  GPIOC->BRR = (GPIO_PIN_1);               // IQR execution time stop
}
```

Part D Code:

main.c

```
/* USER CODE BEGIN Header */
/******************************************************************************
```

```c
* EE 329 A4 Interrupts & Timers
*******************************************************************************
* @file           : main.c
* @brief          :
* project         : EE 329 S'25 Assignment 4
* authors         : Tyler Wong (TW) - twong103@calpoly.edu, Brayden Daly ;
* version         : 0.1
* date            : 250416
* compiler        : STM32CubeIDE v.1.18.0
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
* LED WIRING
*      peripheral – Nucleo I/O
* LED 0  PC0 = CN9  - 3
* LED 1  PC1 = CN9  - 7
* LED 2  PC2 = CN10 - 9
* LED 3  PC3 = CN9  - 5
*******************************************************************************
* REVISION HISTORY
* 0.1 230318 (TW) created, written
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
/* USER CODE END Header */
#include "main.h"
#include "LCD.h"
#include "Delay.h"
#include "string.h"
#include "TIMER.h"
#include "stdio.h"
//declare functions below main
void SystemClock_Config(void);
uint32_t get_random_number(void);
//variables to hold strings and total time after flash
uint32_t time_elapse = 0;
char stringtime[30];
char fullstring[30];
/* -----------------------------------------------------------------------------
* function : main
* INs      : none
* OUTs     : int
* action   : runs game and writes to LCD
*
* authors  : Brayden Daly, Tyler Wong
* version  : 0.3
* date     : 253004
* -------------------------------------------------------------------------- */
int main(void) {
        HAL_Init();
        SystemClock_Config();
        //enable clock for LED
        RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
        // reset output mode
        LED_GPIO->MODER   &= ~(GPIO_MODER_MODE7);
        // set mode to output (set PC13 to Input)
        LED_GPIO->MODER   |=  (GPIO_MODER_MODE7_0);
        //
        LED_GPIO->OTYPER  &= ~(GPIO_OTYPER_OT7);
        // set all pins to no pull up or pull down
        LED_GPIO->PUPDR   &= ~(GPIO_PUPDR_PUPD7);
        // set all pin speeds to high
        LED_GPIO->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED7_Pos));
        // reset all pins to 0
        LED_GPIO->BRR = (GPIO_PIN_7);
        //set pin 13 to input mode
        GPIOC->MODER &= ~(GPIO_MODER_MODE13);  // sets to input mode
        //Enable Random number generating clock
        RCC->AHB2ENR   |=  (RCC_AHB2ENR_RNGEN);
        //Enable HSI48 clock
        RCC->CRRCR |= RCC_CRRCR_HSI48ON;
        //wait until the HSI48 clock is stable
        while((RCC->CRRCR & RCC_CRRCR_HSI48RDY)==0);
        //clear the clock select bits
        RCC->CCIPR &= ~RCC_CCIPR_CLK48SEL;
        //select HSI48 as CLK38 source
        RCC->CCIPR |= (0b11 << RCC_CCIPR_CLK48SEL_Pos);
        //Enable the RNG peripheral
        RNG->CR |= RNG_CR_RNGEN;
        //initialize Systick for delay_us
         SysTick_Init();
         //configure the LCD
         LCD_config();
         //Initialize the LCD
         LCD_Init();
         //clears the LCD
         LCD_command(0x01);
         //wait 2000 seconds
         delay_us(2000);
         //set the cursor to first line first index
         LCD_Set_Cursor(0,0);
         //write this string to the first line
```

```
LCD_write_string("EE 329 A4 REACT ");
//set the cursor to first line
LCD_Set_Cursor(1,0);
//write this string to the second line
LCD_write_string("PUSH SW TO TRIG ");
//turns on display and sets cursor
LCD_command(0x0F);
//set the GPIOC to input mode for PC13
GPIOC->MODER &= ~(GPIO_MODER_MODE13);
//initialize and declare statenum and randomint to hold values
//for the state
// and random integer returned
uint8_t statenum = 0;
uint32_t randomint = 0;
//loop forever to run game forever
while (1)
{
        switch (statenum)
        {
                //first state -> write title screen and delay until
                //button is pressed
                case 0:
                        //turn off LED
                        LED_GPIO->BRR = (GPIO_PIN_7);
                        //Clear LCD
                        LCD_Clear();
                        //Set cursor to first index first line
                        LCD_Set_Cursor(0,0);
                        //write string to line 1
                        LCD_write_string("EE 329 A4 REACT ");
                        //set cursor line 1 first index
                        LCD_Set_Cursor(1,0);
                        //write string
                        LCD_write_string("PUSH SW TO TRIG ");
                        //delay for unwanted bounce
                        delay_us(1000000);
                        //wait until button pressed
                       while ((GPIOC->IDR & GPIO_IDR_ID13) == 0)
                       {
                                          continue;
                       }
                       //increment state num (go to next state)
                       statenum ++;
                       //break out of switch statement
                       break;
               //state 2 -> generate random integer
```

```
 case 1:
                 //generate random int and store into
                 //variable
                 randomint = get_random_number();
                 //go to next state
                 statenum ++;
                 //break out of switch statment
                 break;
//stage 3 -> turn on LED and wait for reaction
 case 2:
                 //delay for random amount of time
                 delay_us(randomint);
                 //turn on led
                 LED_GPIO->BSRR = (GPIO_PIN_7);
                 //start ARR timer
                 setup_TIM2(100);
                 // wait for button press
                 while (((GPIOC->IDR & GPIO_IDR_ID13) == 0))
                 {
                         //if button not pressed within
                         //10 sec, reset
                         if (time_elapse >= 10000)
                         {
                                 //set statenum to -1 since it auto
                                 //accum at end
                                 statenum = -1;
                                 //restart timer
                                 time_elapse = 0;
                                 //break out of while loop
                                 break;
                          }
                 }
                 //turn off all leds
                LED_GPIO->BRR = (GPIO_PIN_7);
                // go to next state
                //(unless 10 seconds violation)
                 statenum ++;
                 //break out of switch
                 break;
//stage 4 -> calculate the seconds and write to LCD
 case 3:
         //calculate milliseconds into seconds
                 uint32_t seconds = time_elapse / 1000;
                 uint32_t millis_remainder = time_elapse % 1000;
                 //turn seconds into a string to write
                 sprintf(fullstring, "TIME: %lu.%03lu",
                                                           seconds, millis_remainder);
                                          //disable the interrupt
                                           __disable_irq();
                                          //clear the LCD
                                          LCD_Clear();
                                          //Set the cursor
                                          LCD_Set_Cursor(1,2);
                                          //write the seconds to the LCD
                                          LCD_write_string(fullstring);
                                          //go to next state
                                          statenum ++;
                                          //break out of switch statement
                                          break;
                         //stage 5 -> wait for button press to restart
                          case 4:
                                  //small delay for debounce
                                         delay_us(1000000);
                                         //wait for button press
                                         while ((GPIOC->IDR & GPIO_IDR_ID13) == 0)
                                         {
                                                 continue;
                                         }
                                         //if button pressed,reset the timer
                                         //and state number
                                         time_elapse = 0;
                                         statenum = 0;
                                         //break out of switch statment
                                         break;
                 }
         }
}
/* ---------------------------------------------------------------------------*/
void SystemClock_Config(void)
{
 RCC_OscInitTypeDef RCC_OscInitStruct = {0};
 RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
 /** Configure the main internal regulator output voltage
 */
 if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
 {
   Error_Handler();
 }
 /** Initializes the RCC Oscillators according to the specified parameters
 * in the RCC_OscInitTypeDef structure.
 */
 RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
 RCC_OscInitStruct.MSIState = RCC_MSI_ON;
 RCC_OscInitStruct.MSICalibrationValue = 0;
 RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
 RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
 if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
 {
   Error_Handler();
 }
 /** Initializes the CPU, AHB and APB buses clocks
 */
 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
 {
   Error_Handler();
 }
}
void Error_Handler(void)
{
 /* USER CODE BEGIN Error_Handler_Debug */
 /* User can add his own implementation to report the HAL error return state */
 __disable_irq();
 while (1)
 {
 }
 /* USER CODE END Error_Handler_Debug */
}
/* -----------------------------------------------------------------------------
* function : get_random_number
* INs      : none
* OUTs     : uintn32_t
* action   : calculates a random number and returns it
*
* authors  : Brayden Daly, Tyler Wong
* version  : 0.3
* date     : 253004
* -------------------------------------------------------------------------- */
uint32_t get_random_number(void) {
        // Wait until data is ready
  while ((RNG->SR & RNG_SR_DRDY) == 0);
  // Read 32-bit random value
  return RNG->DR;
```

```c
}
#ifdef  USE_FULL_ASSERT
/**
 * @brief  Reports the name of the source file and the source line number
 *         where the assert_param error has occurred.
 * @param  file: pointer to the source file name
 * @param  line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
 /* USER CODE BEGIN 6 */
 /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
 /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

main.h

```c
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.h
 * @brief          : Header for main.c file.
 *                   This file contains the common defines of the application.
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
```

```
#include "stm32l4xx_hal.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */
/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */
/* USER CODE END ET */
/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */
/* USER CODE END EC */
/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
/* USER CODE END EM */
/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);
/* USER CODE BEGIN EFP */
/* USER CODE END EFP */
/* Private defines -----------------------------------------------------------*/
#define B1_Pin GPIO_PIN_13
#define B1_GPIO_Port GPIOC
#define LD3_Pin GPIO_PIN_14
#define LD3_GPIO_Port GPIOB
#define USB_OverCurrent_Pin GPIO_PIN_5
#define USB_OverCurrent_GPIO_Port GPIOG
#define USB_PowerSwitchOn_Pin GPIO_PIN_6
#define USB_PowerSwitchOn_GPIO_Port GPIOG
#define STLK_RX_Pin GPIO_PIN_7
#define STLK_RX_GPIO_Port GPIOG
#define STLK_TX_Pin GPIO_PIN_8
#define STLK_TX_GPIO_Port GPIOG
#define USB_SOF_Pin GPIO_PIN_8
#define USB_SOF_GPIO_Port GPIOA
#define USB_VBUS_Pin GPIO_PIN_9
#define USB_VBUS_GPIO_Port GPIOA
#define USB_ID_Pin GPIO_PIN_10
#define USB_ID_GPIO_Port GPIOA
#define USB_DM_Pin GPIO_PIN_11
#define USB_DM_GPIO_Port GPIOA
#define USB_DP_Pin GPIO_PIN_12
#define USB_DP_GPIO_Port GPIOA
#define TMS_Pin GPIO_PIN_13
#define TMS_GPIO_Port GPIOA
#define TCK_Pin GPIO_PIN_14
#define TCK_GPIO_Port GPIOA
#define SWO_Pin GPIO_PIN_3
#define SWO_GPIO_Port GPIOB
```

```
#define LD2_Pin GPIO_PIN_7
#define LD2_GPIO_Port GPIOB
/* USER CODE BEGIN Private defines */
/* USER CODE END Private defines */
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

delay.c

```
#include "main.h"
#include "Delay.h"
#include "stm32l4xx.h"
#include <stdint.h>
/* -----------------------------------------------------------------------------
* function : SysTick_Init(void);
* INs      : none
* OUTs     : none
* action   : Configures the ARM Cortex-M SysTick timer for microsecond delays.
*            Disables interrupts and sets it to use the processor clock.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.3
* date     : 253004
* -------------------------------------------------------------------------- */
void SysTick_Init(void) {
  SysTick->CTRL |= (SysTick_CTRL_ENABLE_Msk |      // Enable SysTick
                    SysTick_CTRL_CLKSOURCE_Msk);    // Use processor clock
  SysTick->CTRL &= ~(SysTick_CTRL_TICKINT_Msk);     // Disable SysTick interrupt
}
/* -----------------------------------------------------------------------------
* function : delay_us(uint32_t time_us);
* INs      : time_us - number of microseconds to delay
* OUTs     : none (blocking delay)
* action   : Uses SysTick countdown to delay for specified number of microseconds.
*            Note: small values may result in longer-than-expected delay.
* authors  : Brayden Daly
* version  : 0.3
* date     : 253004
* -------------------------------------------------------------------------- */
void delay_us(const uint32_t time_us) {
  // Calculate number of clock cycles for the desired delay
  SysTick->LOAD = (uint32_t)((time_us * (SystemCoreClock / 1000000)) - 1);
  SysTick->VAL = 0;                                     // Reset SysTick counter
  SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);       // Clear count flag
  while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)); // Wait for countdown
}
```

delay.h

```
/*
* delay.h
*
*  Created on: Apr 22, 2025
*      Author: wongb
*/
#ifndef INC_DELAY_H_
#define INC_DELAY_H_
void SysTick_Init(void);
void delay_us(const uint32_t time_us);
#endif
```

timer.c

```
/* USER CODE BEGIN Header */
/*******************************************************************************
* EE 329 A4 Timer INTERFACE Part D
*******************************************************************************
* @file                : Timer.c
* @brief               : Timer configuration
* project              : EE 329 S'23 Assignment 4
* authors              : Tyler Wong
*               : Brayden Daly
* version              : 0.1
* date                 : 250423
* compiler             : STM32CubeIDE v.1.12.0 Build: 14980_20230301_1550 (UTC)
* target               : NUCLEO-L4A6ZG
* clocks               : 4 MHz MSI to AHB2
* @attention           : (c) 2023 STMicroelectronics.  All rights reserved.
*******************************************************************************
```

```
/* USER CODE END Header */
#include "timer.h"
#include "main.h"
/* -----------------------------------------------------------------------------
* function : init_GPIO
* INs      : none
* OUTs     : none
* action   : Initializing the output ports
* authors  : Brayden Daly (TW) - bdaly01@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void init_GPIO(void)
{
   RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
   GPIO_PORT->MODER &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2 |
GPIO_MODER_MODE3);
   GPIO_PORT->MODER |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0 | GPIO_MODER_MODE2_0
| GPIO_MODER_MODE3_0);
   GPIO_PORT->OTYPER &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1 | GPIO_OTYPER_OT2 |
GPIO_OTYPER_OT3);
   GPIO_PORT->PUPDR &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2 |
GPIO_PUPDR_PUPD3);
   // Set very high output speed for PC0, PC1, PC2, and PC3
   GPIO_PORT->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED0_Pos) |
                        (3 << GPIO_OSPEEDR_OSPEED1_Pos) |
                        (3 << GPIO_OSPEEDR_OSPEED2_Pos) |
                        (3 << GPIO_OSPEEDR_OSPEED3_Pos));
   GPIO_PORT->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3); // preset
PC0, PC1, PC2, PC3 to 0
}
/* -----------------------------------------------------------------------------
* function : setup_TIM2( int iDutyCycle )
* INs      : CCR1 interrupt count
* OUTs     : none
* action   : Initializes TIM2 with infinite ARR and an input CCR1 count
* authors  : Brayden Daly (TW) - bdaly01@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void setup_TIM2( int iDutyCycle ) {
  RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;           // enable clock for TIM2
  TIM2->DIER |= (TIM_DIER_CC1IE | TIM_DIER_UIE);  // enable event gen, rcv CCR1
  TIM2->ARR = 0xFFFFFFFF;                             // ARR = T = counts @4MHz
  TIM2->CCR1 = iDutyCycle;                        // ticks for duty cycle
  TIM2->SR &= ~(TIM_SR_CC1IF | TIM_SR_UIF);       // clr IRQ flag in status reg
  NVIC->ISER[0] |= (1 << (TIM2_IRQn & 0x1F));     // set NVIC interrupt: 0x1F
  __enable_irq();                                 // global IRQ enable
  TIM2->CR1 |= TIM_CR1_CEN;                       // start TIM2 CR1
}
/* -----------------------------------------------------------------------------
* function : TIM2_IRQHandler(void)
* INs      : none
* OUTs     : none
* action   : Where the interrupts and executes GPIO outputs
* authors  : Brayden Daly (TW) - bdaly01@calpoly.edu
* version  : 0.1
* date     : 250430
* -------------------------------------------------------------------------- */
void TIM2_IRQHandler(void) {
      GPIO_PORT->BSRR = GPIO_PIN_3;
  if (TIM2->SR & TIM_SR_CC1IF) {      // triggered by CCR1 event ...
     TIM2->SR &= ~(TIM_SR_CC1IF);     // manage the flag
     GPIO_PORT->ODR ^= (GPIO_PIN_0);  //toggle GPIO
     TIM2->CCR1 += (PERIOD / 2);               //add half a period to ccr1
  }
  if (TIM2->SR & TIM_SR_UIF) {        // triggered by ARR event ...
     TIM2->SR &= ~(TIM_SR_UIF);       // manage the flag
     //GPIO_PORT->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
// <-- manage GPIO pin here
     TIM2->CCR1 = (PERIOD / 2);        // <-- set CCR1 halfway through next cycle
  }
      GPIO_PORT->BRR = GPIO_PIN_3;
}
```

timer.h

```
/*
* TIMER.h
*
*  Created on: Apr 30, 2025
*      Author: bdaly
*/
#ifndef TIMER_H_
```

```
#define TIMER_H_
#define PERIOD 4000
void setup_TIM2( int iDutyCycle );
void setup_MCO_CLK(void);
void TIM2_IRQHandler(void);
#endif /* INC_TIMER_H_ */
```

lcd.c

```
#include "LCD.h"
#include "main.h"
#include "Delay.h"
#include "string.h"
#include "stm32l4xx.h"     // ← Must be first for STM32 registers (RCC, GPIOx, etc.)
#include <stdint.h>        // ← Needed for uint32_t, int8_t, etc.
/* -----------------------------------------------------------------------------
* function : LCD-Clear(void);
* INs      : none
* OUTs     : none
* action   : Clears the LCD
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_Clear(void) {
   LCD_command(0x01);
   delay_us(2000); // clear takes longer than usual
}
/* -----------------------------------------------------------------------------
* function : LCD_Init(void);
* INs      : none
* OUTs     : none
* action   : Initializes LCD to operate in 4-bit mode using standard sequence.
*            Sends configuration commands and prepares display for use.
* authors  : Tyler Wong, Brayden Daly
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_Init(void) {
  delay_us(40000);  // Wait 40ms after power-up
  // Send wake-up command 3 times in 8-bit mode
  for (int idx = 0; idx < 3; idx++) {
     LCD_4b_command(0x30);
     delay_us(200);
  }
  LCD_4b_command(0x20);  // Set to 4-bit mode
  delay_us(40);
  // Configure function, display, entry mode
  LCD_command(LCD_function_set);     // Function set (4-bit, 2-line, 5x8)
  delay_us(40);
  LCD_command(0x10);                 // Cursor move settings
  delay_us(40);
  LCD_command(LCD_display_on);      // Display ON, cursor ON, blink ON
  delay_us(40);
  LCD_command(LCD_entry_mode_set);  // Increment cursor, no display shift
  delay_us(40);
  LCD_command(LCD_clear_display);   // Clear display
  delay_us(2000);
  LCD_command(LCD_return_home);     // Return home
  delay_us(40);
  LCD_command(0x80);                // Set DDRAM address to line 1
  delay_us(40);
}
/* -----------------------------------------------------------------------------
* function : LCD_pulse_ENA(void);
* INs      : none
* OUTs     : none
* action   : Generates enable pulse to latch data into LCD on falling edge.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_pulse_ENA(void) {
  LCD_PORT->ODR |= LCD_EN;   // Enable high
  delay_us(20);              // Wait for signal setup
  LCD_PORT->ODR &= ~LCD_EN;  // Enable low (falling edge)
  delay_us(20);              // Wait for signal hold
}
/* -----------------------------------------------------------------------------
* function : LCD_4b_command(uint8_t command);
* INs      : 8-bit command (only high nibble is used)
* OUTs     : none
* action   : Sends only the high nibble of a command (used during LCD wake-up).
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_4b_command(uint8_t command) {
  LCD_PORT->ODR &= ~LCD_DATA_BITS;       // Clear data pins
  LCD_PORT->ODR |= (command >> 4);       // Output high nibble
  delay_us(20);
  LCD_pulse_ENA();                       // Latch it
}
/* -----------------------------------------------------------------------------
* function : LCD_command(uint8_t command);
* INs      : 8-bit command
* OUTs     : none
* action   : Sends a full 8-bit command to the LCD in 4-bit mode.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_command(uint8_t command) {
  LCD_PORT->ODR &= ~LCD_DATA_BITS;
  LCD_PORT->ODR |= (command >> 4) & 0x0F;  // Send high nibble
  delay_us(20);
  LCD_pulse_ENA();
  delay_us(400);
  LCD_PORT->ODR &= ~LCD_DATA_BITS;
  LCD_PORT->ODR |= command & 0x0F;        // Send low nibble
  delay_us(20);
  LCD_pulse_ENA();
  delay_us(400);
}
/* -----------------------------------------------------------------------------
* function : LCD_write_char(uint8_t letter);
* INs      : 8-bit character
* OUTs     : none
* action   : Sends a single character to the LCD to display at the current cursor position.
* authors  : Tyler Wong, Brayden Daly
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_write_char(uint8_t letter) {
  LCD_PORT->ODR |= LCD_RS;        // RS = data mode
  LCD_PORT->ODR &= ~LCD_RW;       // RW = write mode
  delay_us(20);
  LCD_command(letter);            // Send character
  LCD_PORT->ODR &= ~LCD_RS;       // RS = command mode
}
/* -----------------------------------------------------------------------------
* function : LCD_write_string(const char* str);
* INs      : Pointer to null-terminated string
* OUTs     : none
* action   : Writes a full string to the LCD by sending characters one by one.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_write_string(const char* str) {
  for (int i = 0; i < strlen(str); i++) {
     LCD_write_char(str[i]);
     delay_us(50);
  }
}
/* -----------------------------------------------------------------------------
* function : LCD_Set_Cursor(uint8_t row, uint8_t col);
* INs      : row (0 or 1), column (0-15)
* OUTs     : none
* action   : Sets LCD cursor to specified row and column.
* authors  : Brayden Daly
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_Set_Cursor(uint8_t row, uint8_t col) {
  uint8_t address;
  switch (row) {
     case 0:
        address = 0x80 + col;  // Line 1 starts at 0x80
        break;
     case 1:
        address = 0xC0 + col;  // Line 2 starts at 0xC0
        break;
     default:
        return;  // Invalid row
  }
  LCD_command(address);
  delay_us(40);
}
/* -----------------------------------------------------------------------------
* function : LCD_config(void);
* INs      : none
* OUTs     : none
* action   : Configures GPIOF pins connected to LCD as output push-pull.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.5
* date     : 253004
* -------------------------------------------------------------------------- */
void LCD_config(void) {
  // Enable GPIOF peripheral clock
```

```
RCC->AHB2ENR |= RCC_AHB2ENR_GPIOFEN;
// Set PF0-PF3 and PF7-PF9 as general purpose outputs (MODER = 01)
LCD_PORT->MODER &= ~(GPIO_MODER_MODE0 |
                     GPIO_MODER_MODE1 |
                     GPIO_MODER_MODE2 |
                     GPIO_MODER_MODE3 |
                     GPIO_MODER_MODE7 |
                     GPIO_MODER_MODE8 |
                     GPIO_MODER_MODE9);
LCD_PORT->MODER |=  (GPIO_MODER_MODE0_0 |
                     GPIO_MODER_MODE1_0 |
                     GPIO_MODER_MODE2_0 |
                     GPIO_MODER_MODE3_0 |
                     GPIO_MODER_MODE7_0 |
                     GPIO_MODER_MODE8_0 |
                     GPIO_MODER_MODE9_0);
// Set output type to push-pull
LCD_PORT->OTYPER &= ~(GPIO_OTYPER_OT0 |
                      GPIO_OTYPER_OT1 |
                      GPIO_OTYPER_OT2 |
                      GPIO_OTYPER_OT3 |
                      GPIO_OTYPER_OT7 |
                      GPIO_OTYPER_OT8 |
                      GPIO_OTYPER_OT9);
// Disable pull-up/pull-down resistors
LCD_PORT->PUPDR &= ~(GPIO_PUPDR_PUPD0 |
                     GPIO_PUPDR_PUPD1 |
                     GPIO_PUPDR_PUPD2 |
                     GPIO_PUPDR_PUPD3 |
                     GPIO_PUPDR_PUPD7 |
                     GPIO_PUPDR_PUPD8 |
                     GPIO_PUPDR_PUPD9);
// Set GPIO speed to low for PF pins
LCD_PORT->OSPEEDR &= ~(3 << GPIO_OSPEEDR_OSPEED0_Pos |
                       3 << GPIO_OSPEEDR_OSPEED1_Pos |
                       3 << GPIO_OSPEEDR_OSPEED2_Pos |
                       3 << GPIO_OSPEEDR_OSPEED3_Pos |
                       3 << GPIO_OSPEEDR_OSPEED7_Pos |
                       3 << GPIO_OSPEEDR_OSPEED8_Pos |
                       3 << GPIO_OSPEEDR_OSPEED9_Pos);
// Reset all LCD control and data lines to LOW
LCD_PORT->BRR = (GPIO_MODER_MODE0 |
                 GPIO_MODER_MODE1 |
                 GPIO_MODER_MODE2 |
                 GPIO_MODER_MODE3 |
                 GPIO_MODER_MODE7 |
                   GPIO_MODER_MODE8 |
                   GPIO_MODER_MODE9);
}
```

lcd.h

```
/*
* lcd.h
*
*  Created on: Apr 22, 2025
*      Author: wongb
*/
#include<stdint.h>
#ifndef INC_LCD_H_
#define INC_LCD_H_
//#define LCD_PORT (GPIOA)
//#define LCD_PORT_EN (RCC_AHB2ENR_GPIOAEN)
//// CONTROL BITS in HIGH NIBBLE
//#define LCD_RS (0x10)
//#define LCD_RW (0x20)
//#define LCD_EN (0x40)
//// DATA BITS in LOW NIBBLE
//#define LCD_DATA_BITS (0x0F)
//// FUNCTION PROTOTYPES
//void LCD_init( void );
//void LCD_pulse_ENA( void );
//void LCD_4b_command( uint8_t command );
//void LCD_command( uint8_t command );
//void LCD_write_char( uint8_t letter );
//uint8_t LCD_swapNibbles( uint8_t dataByte );
//
//// parameterized macro format
//#define NAME(param) (expression using param)
//#define SQUARE(x) ((x) * (x))
//// set Nth bit
//#define SET_BIT(x, pos) (x |= (1U << pos))
//// clear Nth bit
//#define CLR_BIT(x, pos) (x &= (~(1U<< pos)))
//// toggle Nth bit
//#define TOGGLE_BIT(x, pos) (x ^= (1U<< pos))
//
//#define CHK_BIT(x, pos) (x & (1UL << pos) )
//#define GET_BIT(x, pos) ((x & ( 1 << pos)) >> pos)
//// swap nibbles
```

```c
//#define SWAP_4B(x) (((x & 0x0F)<<4) | ((x & 0xF0)>>4))
#define LCD_PORT GPIOF
#define LCD_B4  GPIO_PIN_0
#define LCD_B5  GPIO_PIN_1
#define LCD_B6 GPIO_PIN_2
#define LCD_B7 GPIO_PIN_3
#define LCD_RS GPIO_PIN_7
#define LCD_RW GPIO_PIN_8
#define LCD_EN  GPIO_PIN_9
#define LCD_DATA_BITS (0x0F)
#define LCD_clear_display (0x01) // 0000x0001
//define LCD_setcursor (0x10)
#define LCD_return_home (0x02) // 0000x0010
#define LCD_entry_mode_set (0x06) // 0000x0111
#define LCD_display_on (0x0F) // 0000x1111
#define LCD_function_set (0x28) // 0010x1000
void LCD_config(void);
void LCD_Clear(void);
void LCD_4b_command(uint8_t command);
void LCD_Pulse_ENA(void);
void LCD_Init(void);
void LCD_command(uint8_t command);
void LCD_write_char(uint8_t letter);
void LCD_Set_Cursor(uint8_t row, uint8_t col);
void LCD_write_string(const char*str);
#endif /* INC_LCD_H_ */
```