Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring '25
2025-April-23

**Introduction:**

In this lab we interfaced the LCD and the Keypad with the Nucleo Board and printed "EE 329 A3 TIMER; *=SET #=GO 00:00". When we press the '*' key on the Keypad, we set the LCD minutes and seconds values and check whether they are greater than 59. If either minutes or seconds are greater than 59, then we update them to 59. Then, the LCD counts down when we press GO (#) and checks for a reset button (*) press. When the LCD reaches zero, the timer resets.

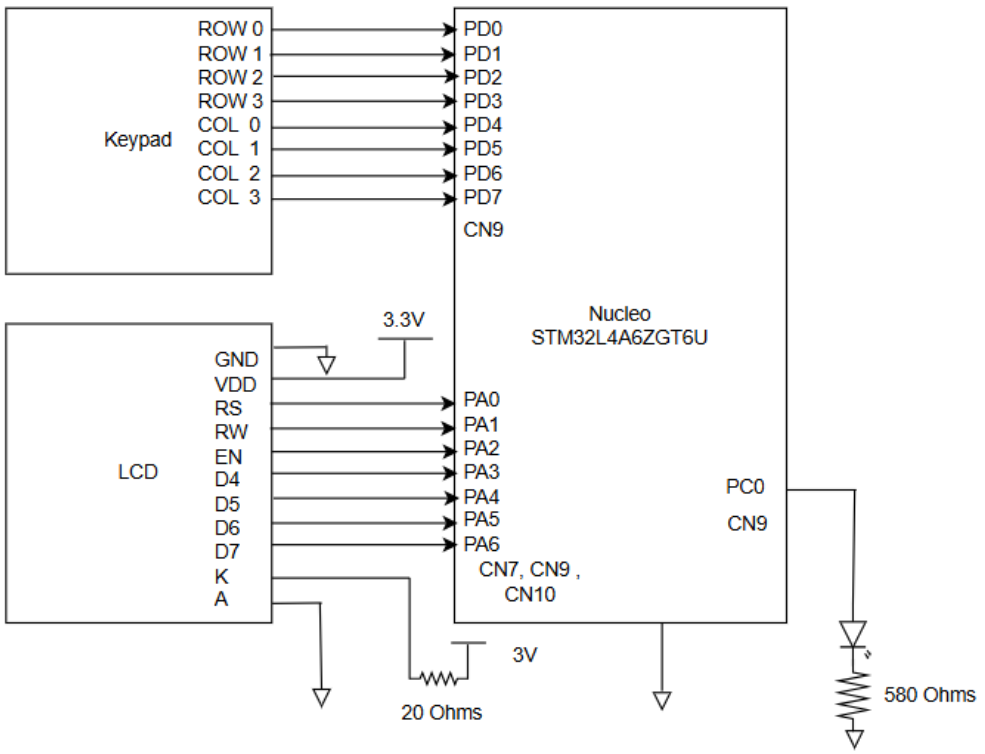
**Deliverables:**

Link to Youtube Video: https://youtu.be/hrnTioXvKFo

Timing Calibration Results:

| Seconds Per Tick | Actual Time | Measured Time | Accuracy |
|---|---|---|---|
| 1s | 0:30 | 00:30.15 | (30.15-30)/30 = 0.5% Error |
| 1s | 1:30 | 01:30.30 | (90.3-90)/90 = 0.33% Error |
| 1s | 3:00 | 03:00.57 | (180.57-180)/180 = 0.32% Error |

Wiring Diagram:



Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

Flowchart:

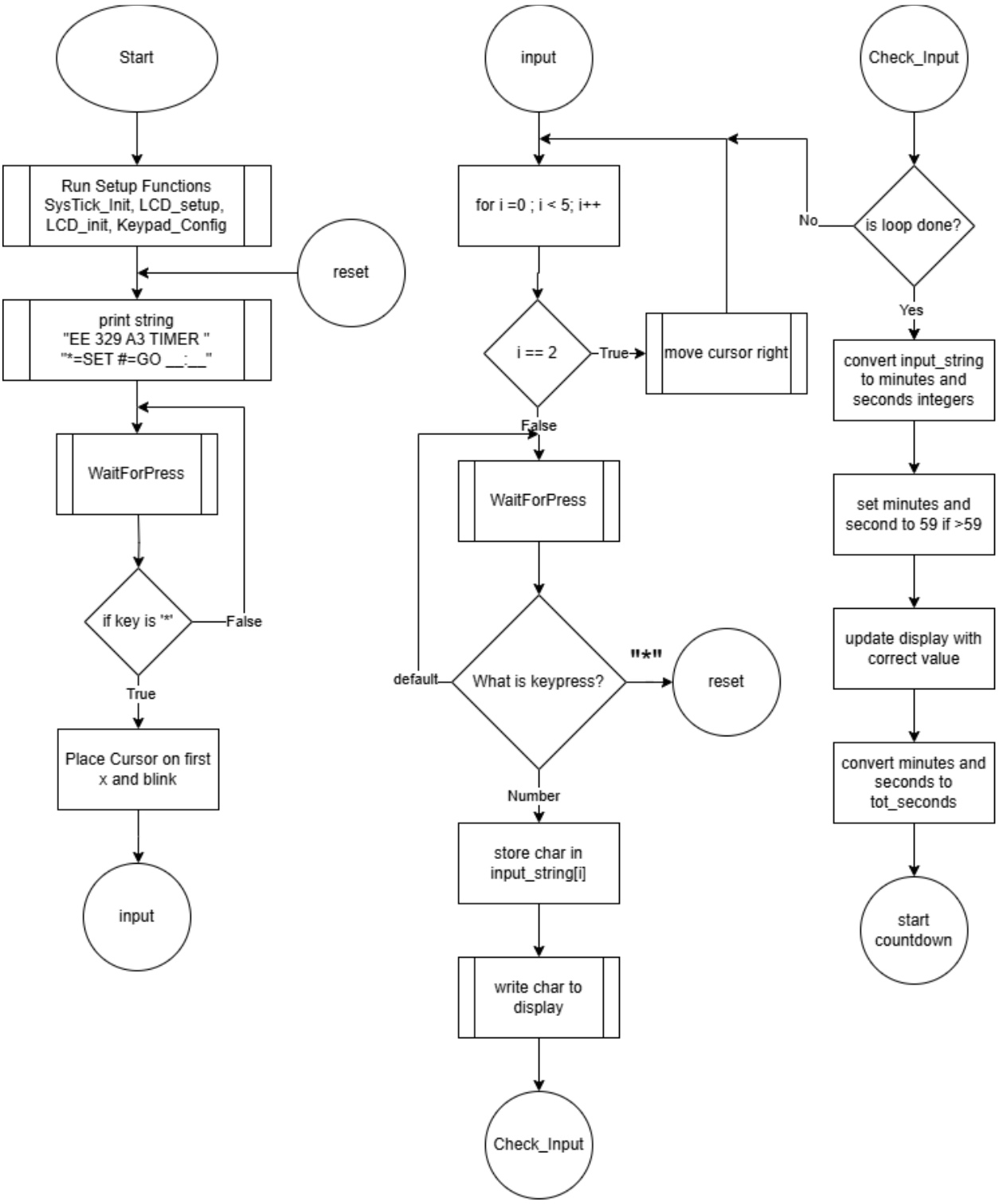



*Figure 1: Countdown Flowchart*

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

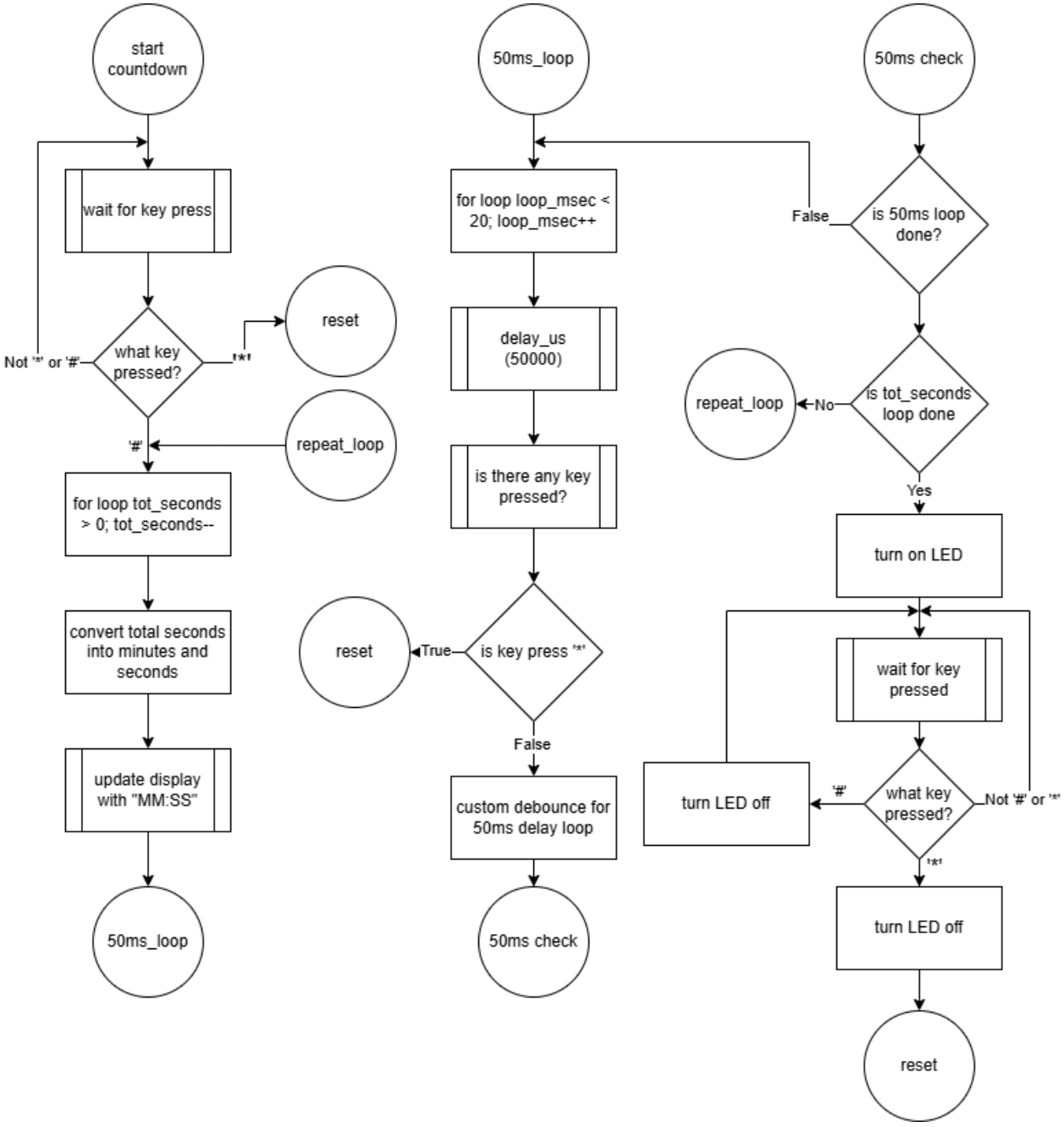



*Figure 2: Countdown Flowchart Cont.*

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

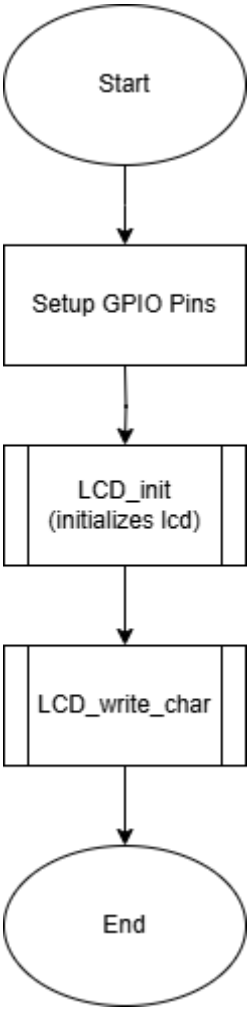



*Figure 3: Code to write one char*

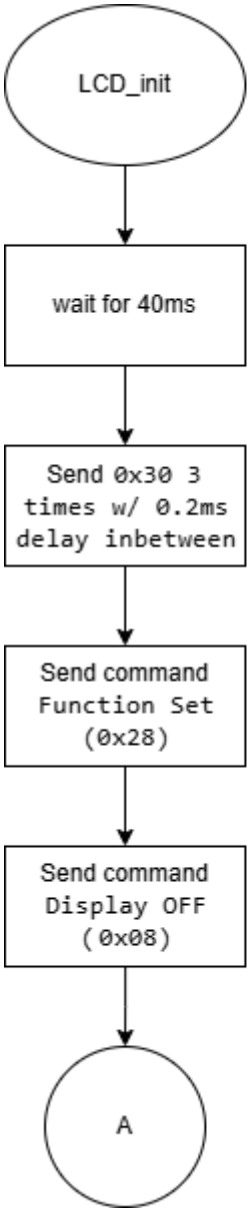
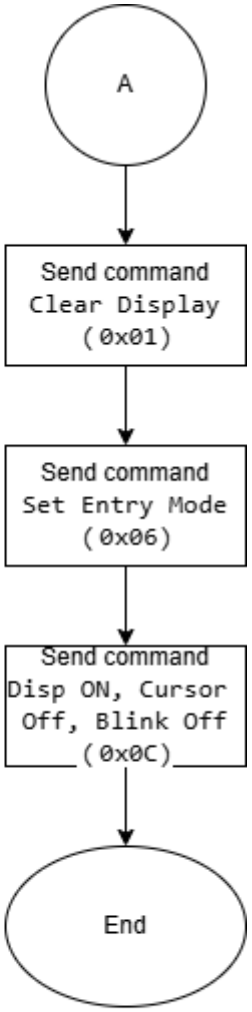



*Figure 4: Function to initilize lcd*



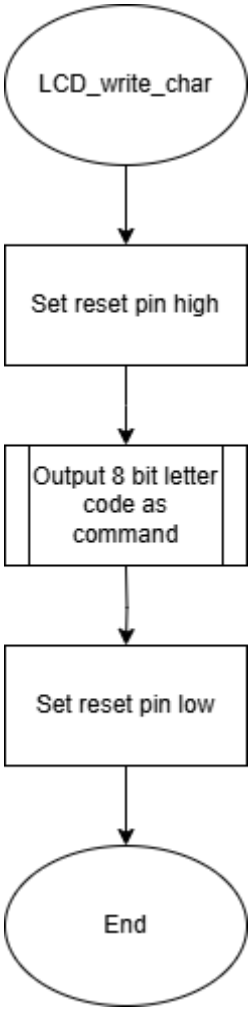




*Figure 5: Function to write char*

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

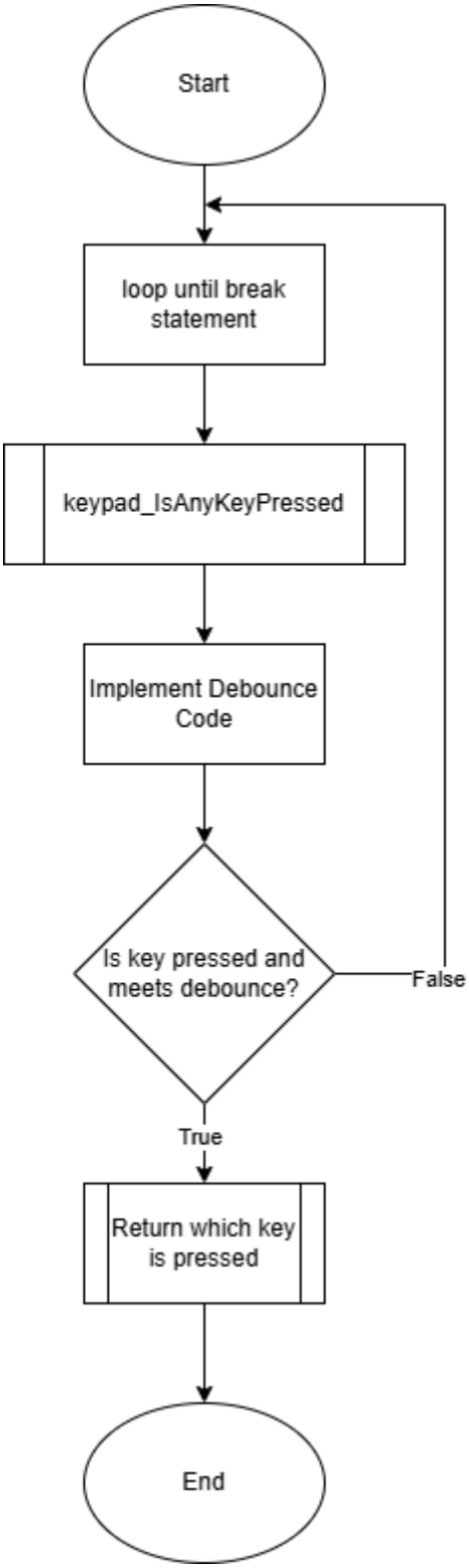



*Figure 6: Function that waits till key is pressed*

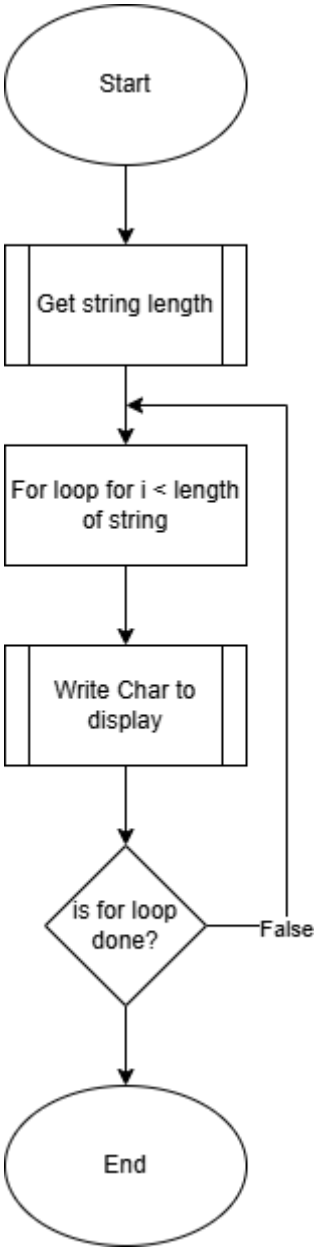



*Figure 7: Function to print string*

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

**Appendix:**

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

/* Includes ------------------------------------------------------------------*/
#include "stm32l4xx_hal.h"

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

Main.c

```
/* USER CODE BEGIN Header */
/******************************************************************************
 * EE 329 A3 <Name>
 ******************************************************************************
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
 * @file           : main.c
 * @brief          : <Description>
 * project         : EE 329 S'25 Assignment 3
 * authors         : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * target          : NUCLEO-L4A6ZG
 * @attention      : (c) 2023 STMicroelectronics.  All rights reserved.
 ******************************************************************************
 * LCD Plan :
 *
 ******************************************************************************
 * LCD Wiring 4 DATA PINS 1 ENABLE 1 READ/WRITE 1 REG SELECT
 * NOTE: 15 pins total pin1 will be ground and pin 15 will be the backlight neg
 *      peripheral - Nucleo I/O
 * LCD 1
 ******************************************************************************
 * REVISION HISTORY
 * 0.1 4/18/25   Formated code headers and imported keypad files
 ******************************************************************************
 * TODO
 *
 ******************************************************************************/
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "keypad.h"
#include "lcd.h"
#include "string.h"
#include "stdio.h"
#include "LED.h"

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */



/* ---------------------------------------------------------------------------
 * function : main( )
 * INs      : none
 * OUTs     : int
 * action   : initialize LCD, Keypad, and LED, run timer on LCD, take I/O from
 * keypad
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------- */
int main(void) {
    HAL_Init();
    SystemClock_Config();


    //initilize delay module
    SysTick_Init();
    // initilize LCD GPIO pins and send init commands
    LCD_Setup();
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring '25
2025-April-23

```
    LCD_Init();
    // initilize Keypad gpio pins
    Keypad_Config();
    // initilize gpio pins for LED
    LED_Config();

    //initialize timer for LCD with string
    char time_string[] = "__:__";
    //Variables to hold minutes and seconds values
    int minutes = 0, seconds = 0;
    //Variables to hold values for reset, inputs, key, and values
    int8_t reset = 0, input = 0xff, update_input = 0, last_value = 0xff,
           value = 0xff, key = 0xff, input_char = ' ';
    int32_t tot_sec = 0, consec_reads = 0;

    // print string to line 1
    LCD_Command(LCD_LINE1);
    //print first line to lcd
    LCD_Print_String( "EE 329 A3 TIMER " );
    // print string to line 2
    LCD_Command(LCD_LINE2);
    LCD_Print_String( "*=SET #=GO __:__" );

    // wait for * input
    while (1) {
        //check if key is pressed and store result in input
        input = keypad_WaitForPress();
        //if * is pressed, break out of loop
        if ( input == 0xe )
            break;
    }

    //constantly change lcd to update timer
    while (1) {
        strcpy(time_string, "__:__");
        //move cursor to lcd
        LCD_Command( MOVE_CURSOR(2, 11) );
        LCD_Print_String(time_string);
        //unflag reset
        reset = 0;

        // set cursor to begining xx:xx and turn on cursor blink
        LCD_Command( MOVE_CURSOR(2, 11) );
        LCD_Command ( LCD_DISPLAY_CTRL(1, 0, 1) );

        // input for loop
        for (uint8_t i = 0; i < 5; i++) {
            // skip ':' position
            if (i == 2) {
                LCD_Command(SHIFT_CURSOR_RIGHT);
                continue;
            }
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring '25
2025-April-23

```
        else {
            while (1){
                input = keypad_WaitForPress();
                // break out of while if valid key detected
                if ( (input < 10) || (input == 0xe) )
                    break;
            }
            // break out of input loop if *
            if ( input == 0xe ) {
                reset = 1;
                break;
            }
            // update time_string and write character to display
            else {
                input_char = keypad_char_table[input];
                time_string[i] = input_char;
                LCD_Write_Char( input_char );
                // wait for key release to loop
                keypad_WaitForRelease();
            }
        }
    }

    //print the
    LCD_Command ( LCD_DISPLAY_CTRL(1, 0, 0) );
    sscanf(time_string, "%2d:%2d", &minutes, &seconds);

    //if minutes or seconds greater than 59, update both to 59
    //(make sure cannot have value greater than 59)
    if ( minutes > 59 ) {
        minutes = 59;
        update_input = 1;
    }
    if ( seconds > 59 ) {
        seconds = 59;
        update_input = 1;
    }

    //if there is a number larger than 59, update to 59
    //and print to LCD
    if (update_input) {
        LCD_Command( MOVE_CURSOR(2, 11) );
        sprintf(time_string, "%02d:%02d", minutes, seconds);
        LCD_Print_String(time_string);
    }

    //calculate the minutes and seconds into seconds
    tot_sec = (minutes * 60) + seconds;
    //loop and check if keypad is pressed
    //if # or * pressed, break out of loop
    while(1) {
        input = keypad_WaitForPress();
        // break out of while if valid key detected
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
            if ( (input == 0xf) || (input == 0xe) )
                break;
        }
        //if # pressed loop every second and count down timer
        if ( input == 0xf ) {
            // for loops every second
            for ( ; tot_sec >= 0; tot_sec-- ) {
                // update lcd with minutes and seconds
                minutes = tot_sec / 60;
                seconds = tot_sec % 60;
                //print minutes and seconds to lcd
                LCD_Command( MOVE_CURSOR(2, 11) );
                sprintf(time_string, "%02d:%02d", minutes, seconds);
                LCD_Print_String(time_string);

                // loops every 50ms
                //every 50 ms, check if reset key is pressed
                for ( uint8_t loop_msec = 0; loop_msec < 20; loop_msec++) {
                    delay_us(50000);
                    if( keypad_IsAnyKeyPressed() ) {
                        // update last value before overwriting value
                        last_value = value;
                        value = 1;
                        // check if value is the same as the last value
                        //debounce for reset key
                        if (value == last_value) {
                            if (consec_reads < 200)
                                consec_reads++;
                        }
                        else
                            consec_reads = 0;
                        //if 4 consecutive reads detected,
                        //check if it is the reset key then
                        //break if it is the reset key
                        if ( consec_reads == 4 ) {
                            key = keypad_WhichKeyPressed();
                            if ( key == 0xe ) {
                                reset = 1;
                                break;
                            }
                        }
                    }
                    else {
                        // update last value before overwriting value
                        last_value = value;
                        value = 0;
                        //if value equals last value
                        //subtract one from consecutive reads
                        if (value == last_value) {
                            if (consec_reads > -200)
                                consec_reads--;
                        }
                        else
                            consec_reads = 0;
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
                }
            }
            //break out of loop if reset is caught
            if ( reset == 1 )
                break;
        }
//          LCD_Command( MOVE_CURSOR(2, 11) );
//          sprintf(time_string, "%02d:%02d", minutes, seconds);
//          LCD_Print_String(time_string);
          // turn on LED
          GPIOC->BSRR = GPIO_PIN_0;
          // wait until * or # are pressed
          while(1) {
            input = keypad_WaitForPress();
            if ( input == 0xf )
                GPIOC->BRR = GPIO_PIN_0;
            else if ( input == 0xe ) {
                GPIOC->BRR = GPIO_PIN_0;
                reset = 1;
                break;
            }
          }
       }
       else
          reset = 1;
   }
}


/**
 * @brief System Clock Configuration
 * @retval None
 */
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();
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
    }

    /** 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_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK) {
       Error_Handler();
    }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
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 */
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

## Lcd.h

```
#ifndef INC_LCD_H_
#define INC_LCD_H_

#include "stm32l4xx_hal.h"
#include "delay.h"
#include "string.h"

#define LCD_LINE1   (0x80)
#define LCD_LINE2 (0xC0)
#define MOVE_CURSOR(row, col) (0x80 | ((row) == 0 ? (col) : (0x40 + (col))))
#define SHIFT_CURSOR_RIGHT (0x14)
#define LCD_DISPLAY_CTRL(D, C, B) (0x08 | ((D) << 2) | ((C) << 1) | (B))



#define LCD_PORT (GPIOA)
#define LCD_RS   (GPIO_PIN_0)
#define LCD_RW   (GPIO_PIN_1)
#define LCD_EN   (GPIO_PIN_2)

// D4-D7 connected to PA3-PA6
#define LCD_D4      (GPIO_PIN_3)
#define LCD_D5      (GPIO_PIN_4)
#define LCD_D6      (GPIO_PIN_5)
#define LCD_D7      (GPIO_PIN_6)

// Mask to clear/set LCD data bits quickly
#define LCD_DATA_BITS   (LCD_D4 | LCD_D5 | LCD_D6 | LCD_D7)

void LCD_Print_String( char string[] );
void LCD_Clear(void);
void LCD_Setup( void );
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 );

#endif /* INC_LCD_H_ */
```

## Lcd.c

```
#include "lcd.h"

/* ------------------------------------------------------------------------
 * function : LCD_print_string( )
 * INs      : none
 * OUTs     : none
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
 * action   : print the string (character by character) to lcd
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_Print_String( char string[] ) {
   for ( uint8_t i = 0; i < strlen(string); i++ )
      LCD_write_char( string[i] );
}


/* ------------------------------------------------------------------------
 * function : LCD_clear( )
 * INs      : none
 * OUTs     : none
 * action   : clear the LCD and delay after
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_Clear(void) {
    LCD_command(0x01);
    Delay_Us(2000); // clear takes longer than usual
}


/* ------------------------------------------------------------------------
 * function : LCD_home( )
 * INs      : none
 * OUTs     : none
 * action   : go home on LCD and delay after
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_Home(void) {
    LCD_command(0x02);
    Delay_Us(2000); // clear takes longer than usual
}


/* ------------------------------------------------------------------------
 * function : LCD_setup( )
 * INs      : none
 * OUTs     : none
 * action   : set GPIOA pins 0-7 to being outputs
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
 void LCD_Setup( void ) {
   // enable GPIOA for LCD
   RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOAEN);
   // reset output mode
   LCD_PORT->MODER   &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2 |
          GPIO_MODER_MODE3 | GPIO_MODER_MODE4 | GPIO_MODER_MODE5 |
          GPIO_MODER_MODE6);
   // set mode to output
   LCD_PORT->MODER   |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0 |
          GPIO_MODER_MODE2_0 | GPIO_MODER_MODE3_0 | GPIO_MODER_MODE4_0 |
          GPIO_MODER_MODE5_0 | GPIO_MODER_MODE6_0);
   //
   LCD_PORT->OTYPER  &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1 | GPIO_OTYPER_OT2 |
          GPIO_OTYPER_OT3 | GPIO_OTYPER_OT4 | GPIO_OTYPER_OT5 | GPIO_OTYPER_OT6);
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
    // set all pins to no pull up or pull down
    LCD_PORT->PUPDR   &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2 |
           GPIO_PUPDR_PUPD3 | GPIO_PUPDR_PUPD4 | GPIO_PUPDR_PUPD5 |
           GPIO_PUPDR_PUPD6);
    // set all pin speeds to high
    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_OSPEED4_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED5_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED6_Pos));
    // reset all pins to 0
    LCD_PORT->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 |
           GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6);
 }


// ------------------------------------------------------ excerpt from lcd.c ---
 /* ------------------------------------------------------------------------
  * function : LCD_Init( )
  * INs      : none
  * OUTs     : none
  * action   : initialize the LCD and set correct 4 bit mode
  * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
  * ------------------------------------------------------------------------ */
void LCD_Init( void )  {   // RCC & GPIO config removed - leverage A1, A2 code
   Delay_Us( 40000 );                     // power-up wait 40 ms
   for ( int idx = 0; idx < 3; idx++ ) {  // wake up 1,2,3: DATA = 0011 XXXX
      LCD_4b_command( 0x30 );             // HI 4b of 8b cmd, low nibble = X
      Delay_Us( 200 );
   }
   LCD_4b_command( 0x20 ); // fcn set #4: 4b cmd set 4b mode - next 0x28:2-line

   LCD_command(0x28); // Function Set: 4-bit, 2-line, 5x8 font

   LCD_command(0x08); // Display OFF

   LCD_clear();

   LCD_command(0x06); // Entry Mode Set: increment, no shift

   LCD_command(0x0C); // Display ON, cursor off, blink off
}


/* ------------------------------------------------------------------------
 * function : LCD_pulse_ENA( )
 * INs      : none
 * OUTs     : none
 * action   : Enable the LCD
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
void LCD_Pulse_ENA( void )  {
   // ENAble line sends command on falling edge
   // set to restore default then clear to trigger
   LCD_PORT->ODR   |= ( LCD_EN );         // ENABLE = HI
   Delay_Us( 5 );                         // TDDR > 320 ns
   LCD_PORT->ODR   &= ~( LCD_EN );        // ENABLE = LOW
   Delay_Us( 5 );                         // low values flakey, see A3:p.1
}

/* ------------------------------------------------------------------------
 * function : LCD_4b_command( )
 * INs      : uint8_t
 * OUTs     : none
 * action   : enable the LCD into 4 bit mode
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_4b_Command( uint8_t command )  {
   // LCD command using high nibble only - used for 'wake-up' 0x30 commands
   LCD_PORT->ODR   &= ~( LCD_DATA_BITS );  // clear DATA bits
   LCD_PORT->ODR   |= ( (command >> 4) << 3 );   // DATA = command
   Delay_Us( 5 );
   LCD_pulse_ENA( );
}

/* ------------------------------------------------------------------------
 * function : LCD_command( )
 * INs      : uint8_t
 * OUTs     : none
 * action   : write a comand to LCD in 4 bit mode
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_Command( uint8_t command )  {
   // send command to LCD in 4-bit instruction mode
   // HIGH nibble then LOW nibble, timing sensitive
   LCD_PORT->ODR   &= ~( LCD_DATA_BITS );  // isolate cmd bits
   // HIGH shifted low
   LCD_PORT->ODR   |= ( ((command >> 4) << 3) & LCD_DATA_BITS );
   Delay_Us( 5 );
   // latch HIGH NIBBLE
   LCD_pulse_ENA( );
   // isolate cmd bits
   LCD_PORT->ODR   &= ~( LCD_DATA_BITS );
   // LOW nibble
   LCD_PORT->ODR   |= ( ((command & 0x0F) << 3) & LCD_DATA_BITS );
   Delay_Us( 5 );
   // latch LOW NIBBLE
   LCD_pulse_ENA( );
   Delay_Us( 40 );
}

/* ------------------------------------------------------------------------
 * function : LCD_write_char( )
 * INs      : none
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
 * OUTs     : uint8_t
 * action   : Write a character to the LCD
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
void LCD_Write_Char( uint8_t letter )  {
   // calls LCD_command() w/char data; assumes all ctrl bits set LO in LCD_init()
   LCD_PORT->ODR   |= ( LCD_RS );      // RS = HI for data to address
   Delay_Us( 5 );
   LCD_command( letter );               // character to print
   LCD_PORT->ODR   &= ~( LCD_RS );     // RS = LO
   Delay_Us( 40 ); // ~40us required per datasheet
}
```

## Keypad.h

```
#ifndef INC_KEYPAD_H_
#define INC_KEYPAD_H_

#include "stm32l4xx_hal.h"

#define DEBOUNCE_LOOP 20
#define NO_KEYPRESS (0xff)

extern const char keypad_char_table[16];

void Keypad_Config(void);
uint8_t keypad_WaitForPress(void);
void keypad_WaitForRelease(void);
uint8_t keypad_IsAnyKeyPressed(void);
uint8_t keypad_WhichKeyPressed(void);

#endif /* INC_KEYPAD_H_ */
```

## Keypad.c

```
#include "keypad.h"

//table for mapping which key pressed
const char keypad_char_table[16] = {
    '0',  // 0x0
    '1',  // 0x1
    '2',  // 0x2
    '3',  // 0x3
    '4',  // 0x4
    '5',  // 0x5
    '6',  // 0x6
    '7',  // 0x7
    '8',  // 0x8
    '9',  // 0x9
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
    'A',  // 0xA
    'B',  // 0xB
    'C',  // 0xC
    'D',  // 0xD
    '*',  // 0xE
    '#'   // 0xF
};


//table for mapping which key pressed
const uint8_t keypad_decode_table[16] = {
    0x1,  // 0
    0x2,  // 1
    0x3,  // 2
    0xA,  // 3
    0x4,  // 4
    0x5,  // 5
    0x6,  // 6
    0xB,  // 7
    0x7,  // 8
    0x8,  // 9
    0x9,  // 10
    0xC,  // 11
    0xE,  // 12 (*)
    0x0,  // 13
    0xF,  // 14 (#)
    0xD   // 15
};



/* ------------------------------------------------------------------------------
 * function : Keypad_Config( )
 * INs      : none
 * OUTs     : none
 * action   : configure the clocks, GPIO I/O mode types, and PUPDR for keypad
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------------ */
void Keypad_Config(void) {
   // config pins for keypad
   // using PD0-7 for pins 0-7 on keypad
   // PD0-3 are inputs, PD4-7 outputs
   // pull down resistors on only PD4-7
   RCC->AHB2ENR  |= (RCC_AHB2ENR_GPIODEN);
   GPIOD->MODER  &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1 | GPIO_MODER_MODE2 |
          GPIO_MODER_MODE3 | GPIO_MODER_MODE4 | GPIO_MODER_MODE5 |
          GPIO_MODER_MODE6 | GPIO_MODER_MODE7);
   GPIOD->MODER  |= (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0 |
          GPIO_MODER_MODE2_0 | GPIO_MODER_MODE3_0);
   GPIOD->OTYPER    &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1 | GPIO_OTYPER_OT2 |
          GPIO_OTYPER_OT3 | GPIO_OTYPER_OT4 | GPIO_OTYPER_OT5 |
          GPIO_OTYPER_OT6 | GPIO_OTYPER_OT7);
   GPIOD->PUPDR  &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1 | GPIO_PUPDR_PUPD2 |
          GPIO_PUPDR_PUPD3 | GPIO_PUPDR_PUPD4 | GPIO_PUPDR_PUPD5 |
          GPIO_PUPDR_PUPD6 | GPIO_PUPDR_PUPD7);
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring '25
2025-April-23

```
    GPIOD->PUPDR |= (GPIO_PUPDR_PUPD4_1 | GPIO_PUPDR_PUPD5_1 |
           GPIO_PUPDR_PUPD6_1 | GPIO_PUPDR_PUPD7_1);
    GPIOD->OSPEEDR   |= ((3 << GPIO_OSPEEDR_OSPEED0_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED1_Pos) | (3 << GPIO_OSPEEDR_OSPEED2_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED3_Pos) | (3 << GPIO_OSPEEDR_OSPEED4_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED5_Pos) | (3 << GPIO_OSPEEDR_OSPEED6_Pos) |
           (3 << GPIO_OSPEEDR_OSPEED7_Pos));


    // preset PD0-3 to 0
    GPIOD->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
}



/* ---------------------------------------------------------------------------
 * function : keypad_WaitForPress( )
 * INs      : none
 * OUTs     : uint8_t
 * action   : Determine with debounce if key is actually pressed
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------- */
uint8_t keypad_WaitForPress(void) {
    uint8_t value = 0, last_value = 0xff;
    int32_t consec_reads = 0;
    // keep looping until keypress is detected
    while(1) {
        if(keypad_IsAnyKeyPressed()) {
            // update last value before overwriting value
            last_value = value;
            value = 1;
            // check if value is the same as the last value
            if (value == last_value) {
                if (consec_reads < 200)
                    consec_reads++;
            }
            else
                consec_reads = 0;


            if (consec_reads == DEBOUNCE_LOOP) {
                //return
                return keypad_WhichKeyPressed();
            }
        }
        else {
            // update last value before overwriting value
            last_value = value;
            value = 0;
            if (value == last_value) {
                if (consec_reads > -200)
                    consec_reads--;
            }
            else
                consec_reads = 0;
        }
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
    }
}



/* ------------------------------------------------------------------------------
 * function : keypad_WaitForRelease( )
 * INs      : none
 * OUTs     : none
 * action   : wait to see if key is released (debounce)
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------------ */
void keypad_WaitForRelease(void) {
    uint8_t value = 0, last_value = 0xff;
    int32_t consec_reads = 0;
    // keep looping until keypress is detected
    while(1) {
        if(keypad_IsAnyKeyPressed()) {
            // update last value before overwriting value
            last_value = value;
            value = 1;
            // check if value is the same as the last value
            if (value == last_value) {
                if (consec_reads < 200)
                    consec_reads++;
            }
            else
                consec_reads = 0;
        }
        else {
            // update last value before overwriting value
            last_value = value;
            value = 0;
            if (value == last_value) {
                if (consec_reads > -200)
                    consec_reads--;
            }
            else
                consec_reads = 0;
            if (consec_reads == -DEBOUNCE_LOOP) {
                // end loop
                break;
            }
        }
    }
}

/* ------------------------------------------------------------------------------
 * function : keypad_IsAnyKeyPressed( )
 * INs      : none
 * OUTs     : uint8_t
 * action   : Check if any key is pressed and return 1 if it is and 0 if not
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------------ */
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
uint8_t keypad_IsAnyKeyPressed(void) {
    // drive all COLUMNS HI; see if any ROWS are HI
    // return true if a key is pressed, false if not
    GPIOD->BSRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
    // give time for voltage to settle
    for (uint16_t time = 0; time < 1000; time++)
        ;
    // check if any port is 0
    if ((GPIOD->IDR & (GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7)) != 0)
        return (1);
    else
        return (0);
}

/* ------------------------------------------------------------------------
 * function : keypad_WhichKeyPressed( )
 * INs      : none
 * OUTs     : uint8_t
 * action   : check which key on keypad is pressed and return that value
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
uint8_t keypad_WhichKeyPressed(void) {
    // keep track if key is found
    uint8_t gotKey = 0;
    uint8_t row = 0, column = 0;
    // reset column pins to 1
    GPIOD->BSRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
    // find which row is pressed
    for (row = 0; row < 4; row++) {
        if (GPIOD->IDR & (GPIO_PIN_4 << row)) {
            // turn on each column 1 at a time
            for (column = 0; column < 4; column++) {
                // reset all columns and then turn on column
                GPIOD->BRR = (GPIO_PIN_0 | GPIO_PIN_1 |
                GPIO_PIN_2 | GPIO_PIN_3);
                GPIOD->BSRR = (GPIO_PIN_0 << column);
                // check if key is pressed
                if (GPIOD->IDR & (GPIO_PIN_4 << row)) {
                    // if key pressed break out
                    //of column loop
                    gotKey = 1;
                    break;
                }
            }
            if (gotKey)
                break;
        }
    }
    // decode row and column idx to overall idx
    uint8_t keypad_idx_pos = (row * 4) + column;
    //return keypad_idx_pos;
    // convert overall idx to values on keypad
    return keypad_decode_table[keypad_idx_pos];
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
}
```

## delay.h

```
/*
 * delay.h
 *
 *  Created on: Apr 21, 2025
 *      Author: jeffl
 */


#ifndef INC_DELAY_H_
#define INC_DELAY_H_

#include "stm32l4xx_hal.h"

void SysTick_Init( void );
void Delay_Us( const uint32_t time_us );

#endif /* INC_DELAY_H_ */
```

## delay.c

```
#include "delay.h"
/* ------------------------------------------------------------------------
 * function : SysTick_Init( )
 * INs      : none
 * OUTs     : none
 * action   : Enable and Initialize Systick to implement delay function
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
// ---------------------------------------------------- delay.c w/o #includes ---
// configure SysTick timer for use with delay_us().
// warning: breaks HAL_delay() by disabling interrupts for shorter delay timing.
void SysTick_Init( void ) {
    SysTick->CTRL |= (SysTick_CTRL_ENABLE_Msk |       // enable SysTick Timer
                      SysTick_CTRL_CLKSOURCE_Msk);    // select CPU clock
    SysTick->CTRL &= ~(SysTick_CTRL_TICKINT_Msk);     // disable interrupt
}



/* ------------------------------------------------------------------------
 * function : delay_us( )
 * INs      : none
 * OUTs     : none
 * action   : this is a helper function that uses Systick to delay in microseconds
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ------------------------------------------------------------------------ */
// delay in microseconds using SysTick timer to count CPU clock cycles
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
// do not call with 0 : error, maximum delay.
// careful calling with small nums : results in longer delays than specified:
//    e.g. @4MHz, delay_us(1) = 10-15 us delay.
void Delay_Us( const uint32_t time_us ) {
   // set the counts for the specified delay
   SysTick->LOAD = (uint32_t)((time_us * (SystemCoreClock / 1000000)) - 1);
   SysTick->VAL = 0;                                          // clear timer count
   SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);        // clear count flag
   while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)); // wait for flag
}
```

## Led.h

```
#ifndef SRC_LED_H_
#define SRC_LED_H_

#include "stm32l4xx_hal.h"

void LED_Config( void );

#endif /* SRC_LED_H_ */
```

## Led.c

```
#include "LED.h"

/* -----------------------------------------------------------------------------
 * function : LED_Config( )
 * INs      : none
 * OUTs     : none
 * action   : initialize the clocks and output mode types and speeds for LED
 * keypad
 * authors  : Michael Brandon Lee (mbl) - mlee394@calpoly.edu
 * ----------------------------------------------------------------------------- */
void LED_Config( void ) {
   // configure GPIO pins PC0(A1), PC1(A3), PC2(A7), PC3(A2) for:
   // output mode, push-pull, no pull up or pull down, high speed
   RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
   GPIOC->MODER   &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1
          | GPIO_MODER_MODE2 | GPIO_MODER_MODE3);
   GPIOC->MODER   |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0
          | GPIO_MODER_MODE2_0 | GPIO_MODER_MODE3_0);
   GPIOC->OTYPER  &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1
          | GPIO_OTYPER_OT2 | GPIO_OTYPER_OT3);
   GPIOC->PUPDR   &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1
          | GPIO_PUPDR_PUPD2 | GPIO_PUPDR_PUPD3);
   GPIOC->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED0_Pos) |
          (3 << GPIO_OSPEEDR_OSPEED1_Pos) |
          (3 << GPIO_OSPEEDR_OSPEED2_Pos) |
          (3 << GPIO_OSPEEDR_OSPEED3_Pos));
   GPIOC->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2
```

Michael Lee, Brayden Daly
Lab Group H
EE 329-05 Spring ’25
2025-April-23

```
            | GPIO_PIN_3); // preset PC0, PC1 to 0
}
```