Wyatt Tack
Bernardo Flores
EE 329-01 F'24
Group F
2024-Oct-7

EE 329 A3

This code is designed to validate the use of a simple LCD screen, in which the communication timing was critical. The code in the LCD module works through sending a nibble (half of a byte), with an extra RS bit, of data out on the communication lines, while the enable pin is pulsed to signal the LCD controller that data is being sent through. The timing is critical, as the LCD must be given enough time to read and implement the data before more is sent through. This calls for another module, the delay module, which uses one of the STM32L4 internal clocks to set microsecond timers as delays in the program. This exercise helped with timing-critical bus analyzation to ensure a working device.

Link to YouTube Presentation:

https://youtu.be/KA2iyTTzBKA

Obtained Data:

Figure A3(a): 3x4 Keypad and 16x2 LCD Wiring

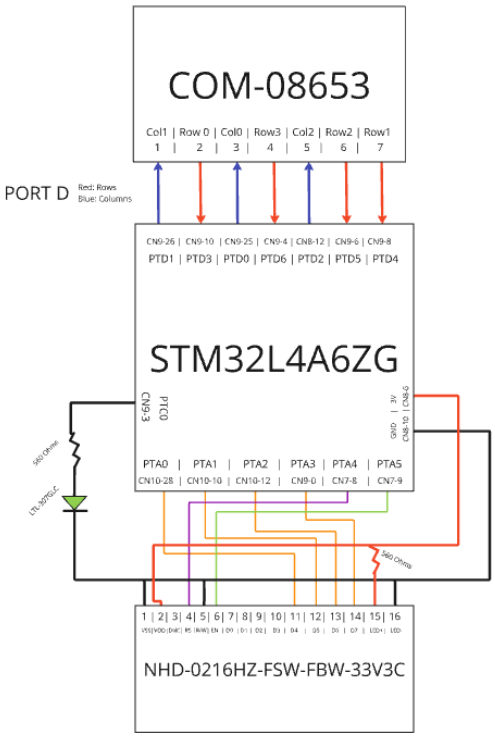
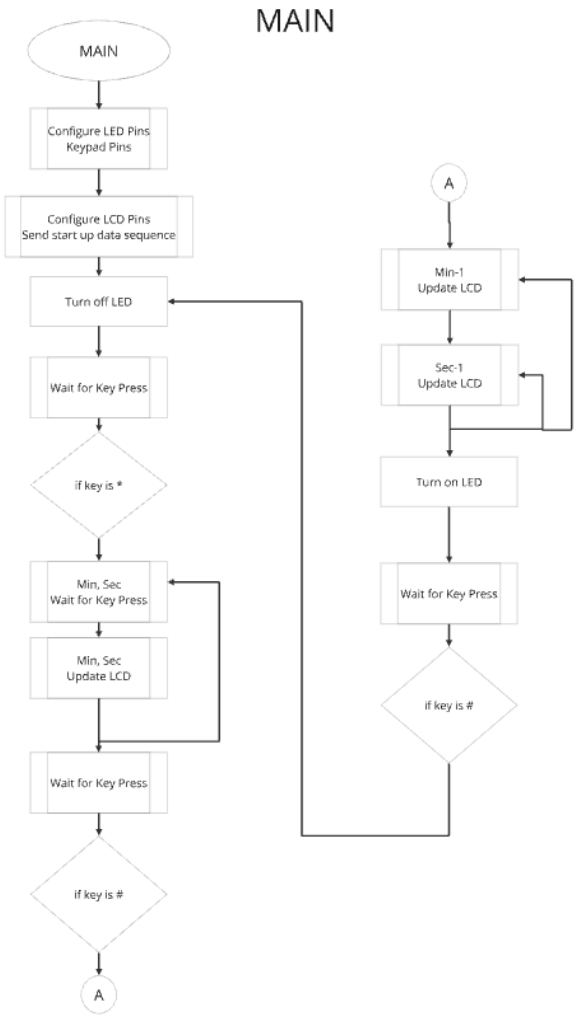


Table A3(a): Calibrated Delay Loop Timing

Entered In Time	**Actual Time**	**Loop Delay Length**	**Delay Loop Iterations**
00:30	00:30.30	930 uS	1000
01:30	1:30.21	930 uS	1000
03:00	2:59.18	930 uS	1000

Psuedocode Flow Chart:



Formatted Source Code main.h:

```
/**
  ******************************************************************************
  * @file               : main.h
  * project             : EE329 Lab A3
  * author              : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                : 10/5/2024
  * firmware            : ST-Link V1
  * @attention          : Copyright (c) 2024 STMicroelectronics. All rights
  *                                                                    reserved.
  ******************************************************************************
  *
  *     main header for defines for C and stm32 headers/hal
  *
  ******************************************************************************
  */

#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

#define STAR_KEY (0x0A)
#define POUND_KEY (0x0C)


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

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


#ifdef __cplusplus
}
#endif

#endif
```

Formatted Source Code main.c:

```
/**
  ******************************************************************************
  * @file           : main.c                                                    : EE329 Lab A3
  * @project                                                                    : Wyatt Tack (wwt) - wtack@calpoly.edu
  * @author                                                                     : 10/5/2024
  * @date                                      : ST-Link V1
  * @firmware                                  : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  * @attention
  ******************************************************************************
  *
  *                     Timer device, uses Keypad to enter numbers for timer, and on pressing #
  *                     will count down until timer reached, signaling LED until # is pressed again.
  *                     pressing * at any time will go to timer enter mode.
  *                     Keypad defined at GPIOD Col PTD0-2, Row PTD3-6
  *                     LED defined at GPIOC PTC0
  *                     LCD defined at GPIOA PTA0-3 (4 bit data), PTA4-5 (RS and EN)
  *
  ******************************************************************************
  */

#include "main.h"
#include "keypad.h"
#include "lcd.h"
#include "delay.h"


int main(void)
{
                      //Initialize clock and delay configs
                      HAL_Init();
                      SystemClock_Config();
                      SysTick_Init();
                      //Initialize keypad and LCD
                      Keypad_Config();
                      LCD_init();
                      Led_Config();

//display:
//"EE 329 A3 TIMER "
//"*=SET #=GO 00:00"
                      uint8_t origin[2] = {0,0};
                      uint8_t line2[2] = {0,1};
                      uint8_t openingLine1[16] = "EE 329 A3 TIMER ";
                      uint8_t openingLine2[16] = "*=SET #=GO 00:00";
//Initialize variables and cursor locations for each digit
                      uint8_t clockMin10[2] = {11,1};
                      uint8_t clockMin1[2] = {12,1};
                      uint8_t clockSec10[2] = {14,1};
                      uint8_t clockSec1[2] = {15,1};
                      uint8_t currentKeyValue;
                      uint8_t min10, min1;
                      uint8_t sec10, sec1;


  while (1)
  {
// reset LED and poll keypad until * pressed
                      currentKeyValue = 0;
                                          LCD_set_cursor(origin);
                                          LCD_write_string(openingLine1);
                                          LCD_set_cursor(line2);
                                          LCD_write_string(openingLine2);
                        GPIOC->BRR = (GPIO_PIN_0);
                        while(Key_Poll() != STAR_KEY);
// set time for each digit on clock
                                          min10= 0xF; min1= 0xF;    //reset values
                                          sec10= 0xF; sec1= 0xF;
                      //Set 10*min
                                          LCD_set_cursor(clockMin10);
                                          while(min10 < 0 || min10 > 5) {
                                                              if (currentKeyValue == STAR_KEY) break;
                                                              currentKeyValue = Key_Poll();
                                                              min10 = currentKeyValue;
                                          }
                                          LCD_write_char(min10+ 0x30);
                      //Set 1*min
                                          LCD_set_cursor(clockMin1);
                                          while(min1 < 0 || min1 > 9){
                                                              if (currentKeyValue == STAR_KEY) break;
                                                              currentKeyValue = Key_Poll();
                                                              min1 = currentKeyValue;
                                          }
                                          LCD_write_char(min1+ 0x30);
                      //Set 10*sec
                                          LCD_set_cursor(clockSec10);
                                          while(sec10 < 0 || sec10 > 5){
                                                              if (currentKeyValue == STAR_KEY) break;
                                                              currentKeyValue = Key_Poll();
                                                              sec10 = currentKeyValue;
                                          }
                                          LCD_write_char(sec10+ 0x30);
                      //Set 1*sec
                                          LCD_set_cursor(clockSec1);
                                          while(sec1 < 0 || sec1 > 9){
                                                              if (currentKeyValue == STAR_KEY) break;
                                                              currentKeyValue = Key_Poll();
                                                              sec1 = currentKeyValue;
                                          }
                                          LCD_write_char(sec1+ 0x30);
//wait for # start key
                                          while(currentKeyValue != POUND_KEY){
                                                              if (currentKeyValue == STAR_KEY) break;
                                                              currentKeyValue = Key_Poll();
                                          }
//clock loop
                                          //min 10s
                                          for (min10++; min10>0; min10--){
                                                                LCD_set_cursor(clockMin10);
                                                                LCD_write_char(min10-1 + 0x30);
                                                                //min 1s
                                                                for (min1++; min1>0; min1--){
                                                                                    LCD_set_cursor(clockMin1);
                                                                                    LCD_write_char(min1-1 + 0x30);
                                                                                    //sec 10s
                                                                                    for (sec10++; sec10>0; sec10--){
                                                                                                        LCD_set_cursor(clockSec10);
                                                                                                        LCD_write_char(sec10-1 + 0x30);
                                                                                                        //sec 1s
                                                                                                        for (sec1++; sec1>0; sec1--){
                                                                                                                            LCD_set_cursor(clockSec1);
                                                                                                                            LCD_write_char(sec1-1 + 0x30);
                                                                                                                            //1s loop that polls keypad if * pressed
                                                                                                                            for (uint32_t timeDelay = 0; timeDelay < 1000; timeDelay++){
                                                                                                                                                delay_us(930);          //adjust delay based on calibration
                                                                                                                                                if(currentKeyValue == STAR_KEY) break;
                                                                                                                                                currentKeyValue = Keypad_WhichKeyIsPressed();
                                                                                                                            }
                                                                                                        }//roll overs from 10s place
                                                                                                        sec1 = 9;
                                                                                                        if(currentKeyValue == STAR_KEY) break;
                                                                                    }
                                                                                    sec10 = 5;
                                                                                    if(currentKeyValue == STAR_KEY) break;
                                                                }
                                                                min1 = 9;
                                                                if(currentKeyValue == STAR_KEY) break;
                                          }
//timer done turn on LED
                                          GPIOC->BSRR |= (GPIO_PIN_0);
//wait till # or * pressed to reset
                                          while (!(currentKeyValue == STAR_KEY || currentKeyValue == POUND_KEY))currentKeyValue = Key_Poll();

  }
}
```

(CTD')

Formatted Source Code main.c (CTD’):

```
//-----------------------------------------------------------------------
void Led_Config(void)
{
                                                  // configure GPIO pins PC0-3 for:
                                                  // output mode, push-pull, no pull up or pull down, high speed
                                RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOCEN);
                                GPIOC->MODER   &= ~(GPIO_MODER_MODE0);
                                GPIOC->MODER   |=  (GPIO_MODER_MODE0_0);
                                GPIOC->OTYPER  &= ~(GPIO_OTYPER_OT0);
                                GPIOC->PUPDR   &= ~(GPIO_PUPDR_PUPD0);
                                GPIOC->OSPEEDR |=  (3 << GPIO_OSPEEDR_OSPEED0_Pos);
                                GPIOC->BRR |= (GPIO_PIN_0);
}

// System
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_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

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

void Error_Handler(void)
{

  __disable_irq();
  while (1)
  {
  }

}

#ifdef  USE_FULL_ASSERT

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif
```

Formatted Source Code keypad.h:

```
/**
  ******************************************************************************
  * @file       : keypad.h
  * project     : EE329 Lab A3
  * author      : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date        : 9/27/2024
  * firmware    : ST-Link V1
  * @attention : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *     main header for defines for keypad.h
  *
  ******************************************************************************
  */

#ifndef INC_KEYPAD_H_
#define INC_KEYPAD_H_
#include "stm32l4xx_hal.h"

//-------------------------------- KEYPAD Defines -------------------------------
#define COL_PORT GPIOD
#define COL_PINS (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2)
#define ROW_PORT GPIOD
#define ROW_PINS (GPIO_PIN_3 | GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6)
#define SETTLE 1900 //defined as time for 20ms loop instruction (*5.5us)

#define BIT0_COL GPIO_PIN_0 //defined as first bit for columns
#define BIT0_ROW GPIO_PIN_3 //defined as first bit for rows
#define NUM_COLS 3
#define NUM_ROWS 4
#define NO_KEYPRESS 0x0
#define KEY_ZERO 11
#define CODE_ZERO 0xF

//-------------------------------- function prototypes ------------------------
void Keypad_Config(void);
int Keypad_IsAnyKeyPressed(void);
int Keypad_WhichKeyIsPressed(void);
uint8_t Key_Poll(void);

#endif /* INC_KEYPAD_H_ */
```

Formatted Source Code keypad.c:

```
/**
  ******************************************************************************
  * @file           : keypad.c
  * project         : EE329 Lab A3
  * author          : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date            : 10/7/2024
  * firmware        : ST-Link V1
  * @attention      : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *                Keypad pulling function source file provided on behalf of the EE329 lab
  *                manual. Adapted from EE329 lab manual. Currently attached to GPIO PORT D
  *
  ******************************************************************************
  */
#include "keypad.h"

// ------------------------------------ modified from excerpt from keypad.c ---
void Keypad_Config(void){//must be manually changed if separate GPIO port is used

                     //set for port D as current config
                    //Port clock initialize
                    RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIODEN);
                    //Column pin initialize ~ Push Pull, no PU/PD, high speed
                    COL_PORT->MODER    &= ~(GPIO_MODER_MODE0 | GPIO_MODER_MODE1
                                      | GPIO_MODER_MODE2);
                    COL_PORT->MODER    |=  (GPIO_MODER_MODE0_0 | GPIO_MODER_MODE1_0
                                      | GPIO_MODER_MODE2_0);
                    COL_PORT->OTYPER  &= ~(GPIO_OTYPER_OT0 | GPIO_OTYPER_OT1
                                      | GPIO_OTYPER_OT2 | GPIO_OTYPER_OT3);
                    COL_PORT->PUPDR    &= ~(GPIO_PUPDR_PUPD0 | GPIO_PUPDR_PUPD1
                                      | GPIO_PUPDR_PUPD2 | GPIO_PUPDR_PUPD3);
                    COL_PORT->OSPEEDR |=  ((GPIO_OSPEEDR_OSPEED0) |
                                        (GPIO_OSPEEDR_OSPEED1) |
                                        (GPIO_OSPEEDR_OSPEED2));
                    //Row pin initialize ~ Input, pull down
                    ROW_PORT->MODER   &= ~(GPIO_MODER_MODE3 | GPIO_MODER_MODE4
                                      | GPIO_MODER_MODE5 | GPIO_MODER_MODE6);
                    ROW_PORT->PUPDR    &= ~(GPIO_PUPDR_PUPD3 | GPIO_PUPDR_PUPD4
                                      | GPIO_PUPDR_PUPD5 | GPIO_PUPDR_PUPD6);
                    ROW_PORT->PUPDR    |= (GPIO_PUPDR_PUPD3_1 | GPIO_PUPDR_PUPD4_1
                                      | GPIO_PUPDR_PUPD5_1 | GPIO_PUPDR_PUPD6_1);
}

// ---------------------------------------------------------------------------
int Keypad_IsAnyKeyPressed(void) {
// drive all COLUMNS HI; see if any ROWS are HI
// return true if a key is pressed, false if not
   COL_PORT->BSRR = COL_PINS;                // set all columns HI
   for ( uint16_t idx=0; idx<SETTLE; idx++ )              // let it settle
      ;
   if ((ROW_PORT->IDR & ROW_PINS) != 0 ) {              // got a keypress!
                     for ( uint16_t idx=0; idx < SETTLE; idx++ ){
                                    if ((ROW_PORT->IDR & ROW_PINS) == 0 ) return( 0 );
                                    }// if key held for 20ms then return 1 (debounce)
                     return( 1 );
                     }
   else
      return( 0 );                           // nope.
}

// ---------------------------------------------------------------------------
int Keypad WhichKeyIsPressed(void) {
// detect and encode a pressed key at (row,col)
// assumes a previous call to Keypad_IsAnyKeyPressed() returned TRUE
// verifies the Keypad_IsAnyKeyPressed() result (no debounce here),
// determines which key is pressed and returns the encoded key ID

   int8_t iRow=0, iCol=0, iKey=0;  // keypad row & col index, key ID result
   int8 t bGotKey = 0;             // bool for keypress, 0 = no press

   COL PORT->BSRR = COL PINS;                              // set all columns HI
   for ( iRow = 0; iRow < NUM ROWS; iRow++ ) {             // check all ROWS
      if ( ROW_PORT->IDR & (BIT0_ROW << iRow) ) {          // keypress in iRow!!
         COL_PORT->BRR = ( COL_PINS );                     // set all cols LO
         for ( iCol = 0; iCol < NUM_COLS; iCol++ ) {    // 1 col at a time
            COL_PORT->BSRR = ( BIT0_COL << (iCol) );     // set this col HI
            if ( ROW_PORT->IDR & (BIT0_ROW << iRow) ) {     // keypress in iCol!!
               bGotKey = 1;
               break;                                    // exit for iCol loop
            }
         }
         if ( bGotKey )
            break;
      }
   }
   //              encode (iRow,iCol) into LED word : row 1-3 : numeric, '1'-'9'
   //                                                 row 4   : '*'=10, '0'=15, '#'=12
   //                                  no press: send NO_KEYPRESS
   if ( bGotKey ) {
      iKey = ( iRow * NUM_COLS ) + iCol + 1;  // handle numeric keys ...
                  if ( iKey == KEY_ZERO )                 //    works for '*', '#' too
         iKey = CODE_ZERO;
                  return( iKey );                         // return encoded keypress
   }
   return( NO_KEYPRESS );                     // unable to verify keypress
}

// ---------------------------------------------------------------------------
uint8_t Key_Poll(void) {
                  // polls key and returns value once key is inputed and let go
                  uint8_t currentKeyValue;
                  while(!Keypad_IsAnyKeyPressed());
                                    if (Keypad_WhichKeyIsPressed() > 0 && Keypad_WhichKeyIsPressed() < 16){
                                                      currentKeyValue = Keypad_WhichKeyIsPressed();
                                                      if (currentKeyValue == 0xF ) currentKeyValue = 0; //zero position
                                    }
                                    else
                                                      currentKeyValue = (0xF);//error flag
     while(Keypad_IsAnyKeyPressed());
                                    return currentKeyValue;
                                    //wait till key is let go
}
```

Formatted Source Code delay.h:

```
/**
  ******************************************************************************
  * @file               : delay.h
  * project             : EE329 Lab A3
  * author              : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                : 10/1/2024
  * firmware            : ST-Link V1
  * @attention          : Copyright (c) 2024 STMicroelectronics. All rights
reserved.
  ******************************************************************************
  *
  *   main header for defines for delay.h
  *
  ******************************************************************************
  */

#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_ */
```

Formatted Source Code delay.c:

```
/**
  ******************************************************************************
  * @file               : delay.c
  * project             : EE329 Lab A3
  * author              : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                : 10/1/2024
  * firmware            : ST-Link V1
  * @attention          : Copyright (c) 2024 STMicroelectronics. All rights
reserved.
  ******************************************************************************
  *
  *   Functions for using SysTick clock for software delays. Provided
  *   on behalf of EE329 lab manual.
  *
  ******************************************************************************
  */

#include "delay.h"

// ------------------------------------------------- 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
}

// delay in microseconds using SysTick timer to count CPU clock cycles
// 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
}
```

Formatted Source Code lcd.h:

```
/**
  ******************************************************************************
  * @file               : lcd.h
  * project             : EE329 Lab A3
  * author              : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                : 10/1/2024
  * firmware            : ST-Link V1
  * @attention          : Copyright (c) 2024 STMicroelectronics. All rights
reserved.
  ******************************************************************************
  *
  *   main header for defines for lcd.h
  *
  ******************************************************************************
  */

#ifndef INC_LCD_H_
#define INC_LCD_H_

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

#define LCD_MODER (0x03FFF)
#define LCD_MODER_0 (0x01555)
#define LCD_OTYPER (0x07F)
#define LCD_PUPDR (0x03FFF)
#define LCD_OSPEEDR (0x03FFF)
#define LCD_PORT GPIOA
#define LCD_EN GPIO_PIN_5     //Pin 5
#define LCD_RS GPIO_PIN_4     //Pin 4
#define LCD_DATA_BITS (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3)
                                        //Pins 11-14
                                        //pull down R/W (set as only outputs)
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 );
void LCD_set_cursor( uint8_t position[2]);
void LCD_write_string( uint8_t writeData[] );

#endif /* INC_LCD_H_ */
```

Formatted Source Code lcd.c:

```
/**
  ******************************************************************************
  * @file                    : lcd.c
  * project                  : EE329 Lab A3
  * author                   : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                     : 10/1/2024
  * firmware                 : ST-Link V1
  * @attention               : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *            Functions for interfacing and communicating to LCD display through
  *            nibble mode. Provided on behalf of EE329 lab manual. Configured to
  *            be wired through GPIO PORT A Pin0-3 as D4-D7; Pin4 as RS; Pin5 as EN
  *
  ******************************************************************************
  */

#include "lcd.h"

// ------------------------------------------------------ excerpt from lcd.c ---
void LCD_init( void )  {
                //Port clock initialize
                RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOAEN);
                //LCD pin initialize - Push Pull, no PU/PD, high speed
                LCD_PORT->MODER   &= ~(LCD_MODER);
                LCD_PORT->MODER   |=  (LCD_MODER_0);
                LCD_PORT->OTYPER  &= ~(LCD_OTYPER);
                LCD_PORT->PUPDR   &= ~(LCD_PUPDR);
                LCD_PORT->OSPEEDR |=  (LCD_OSPEEDR);
              delay_us( 80000 );                        // power-up wait 80 ms
              LCD_PORT->ODR   &= ~(LCD_RS);                          // clear RS bit
              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( 5000 );
   }
   LCD_4b_command( 0x20 ); // fcn set #4: 4b cmd set 4b mode - next 0x28:2-line
   delay_us(3000);
   LCD_command( 0x28 );       // fcn set 4b mode and 2x28 line
   delay_us( 300 );
   LCD_command( 0x10 );       //Set cursor
   delay_us( 300 );
   LCD command( 0x0F ); // display, cursor, blink on
   delay_us( 300 );
   LCD command(0x06);         //Entry mode
   delay_us( 300 );
   LCD_command(0x01);         //clear display
   delay_us( 300 );
   LCD_command(0x80);         // set cursor home
   delay_us( 300 );

}

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( 25 );                          // TDDR > 320 ns
   LCD_PORT->ODR   &= ~( LCD_EN );        // ENABLE = LOW
   delay_us( 20 );                          // low values flakey, see A3:p.1
}

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 );   // DATA = command
   delay_us( 15 );
   LCD pulse ENA( );
}

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
   LCD_PORT->ODR   |= ( (command>>4) & LCD_DATA_BITS ); // HIGH shifted low
   delay_us( 15 );
   LCD_pulse_ENA( );                                   // latch HIGH NIBBLE

   LCD PORT->ODR   &= ~( LCD DATA BITS );              // isolate cmd bits
   LCD PORT->ODR   |= ( command & LCD DATA BITS );     // LOW nibble
   delay_us( 15 );
   LCD pulse ENA( );                                   // latch LOW NIBBLE
}

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( 15 );
   LCD_command( letter );             // character to print
   LCD_PORT->ODR   &= ~(LCD_RS);      // RS = LO
}

void LCD_set_cursor( uint8_t position[2])    {
//              calls LCD_command to change cursor position
                //position formatted as [row,col] (zero indexed)
                //sets ddram address for cursor set
                LCD_PORT->ODR   &= ~(LCD_RS);
                uint8_t ddRamAdd =  (40 * position[1]);
                ddRamAdd |= (0x80);
                LCD_command(ddRamAdd);
                delay_us( 500 );
                for (int col = 0; col < position[0]; col++){
                                LCD_command( 0x14 );
                                delay us( 300 );
                                }
                //set address as RS_Low, data as [0x80 | address]
                //address defined as 0x00-0x0F, 0x40-0x4F
}

void LCD_write_string( uint8_t              writeData[])   {
//              calls LCD_write_char in row long for loop
                for (uint8_t indexCol = 0; indexCol < 16; indexCol++ ){
                                                LCD_write_char(writeData[indexCol]);
                                                delay us(60);
                                }
                }
```