California Polytechnic State University

Electrical Engineering Department
EE 329-05-2254 Microcontroller-Based System Design
Spring 2025
Professor: John Penvenne

Assignment A8
Analog to Digital Converter (ADC)

Written By:

Brayden Daly

Preston Mavady

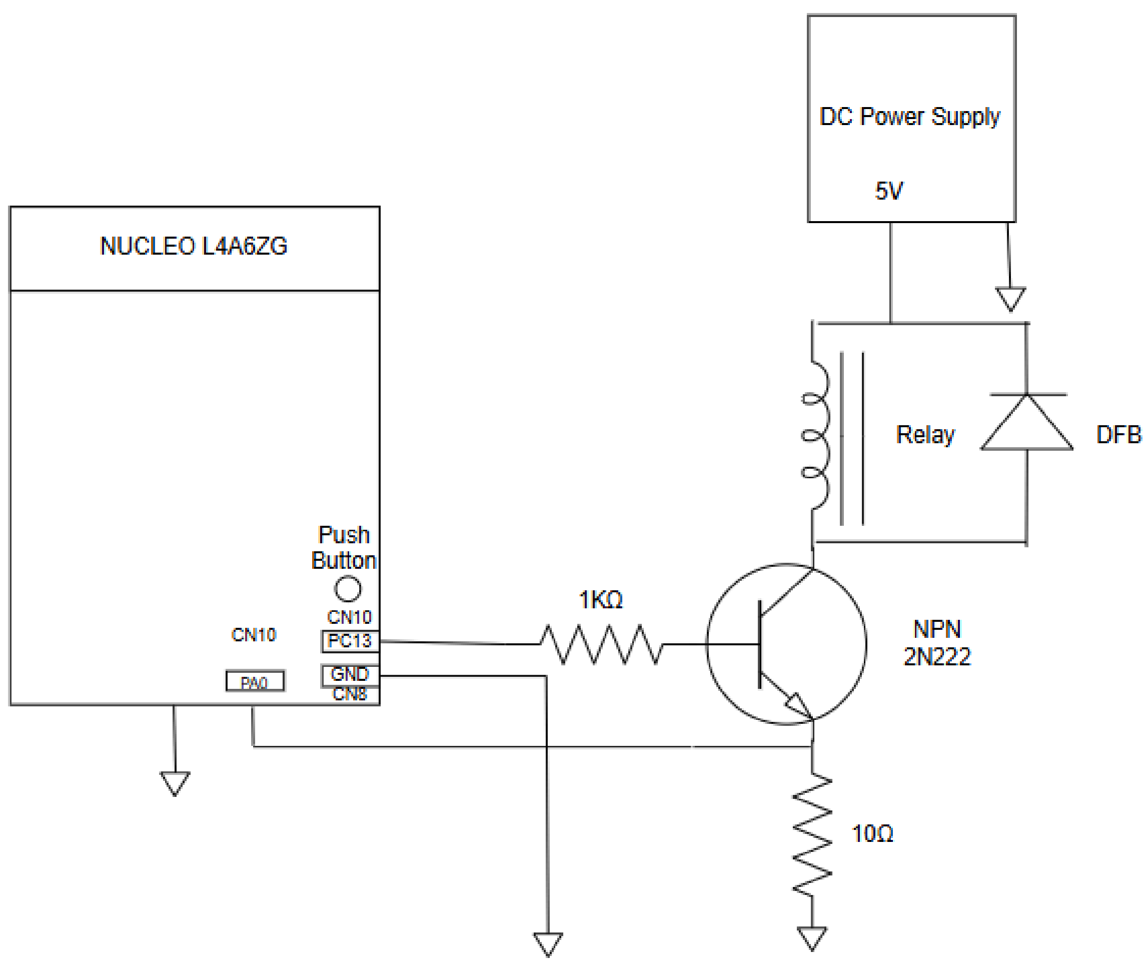
May 28, 2025

Video Link: https://youtu.be/3TDPQf7mZB8

Brief Intro:

In this lab we used the onboard ADC as well as a relay and a 2N222 BJT to turn a relay on and off. We configured the ADC by using the line of best fit with the multiple steps of DC voltages and the ADC ended up being accurate within a few millivolts. We then pulled up a relay and activated it with PA0 connected to the base of the BJT. Finally, we configured the Push Button on the board to turn the relay on and off.

Wiring Diagram:



Calculation of 12-bit ADC resolution for FSR = 3.3 V:

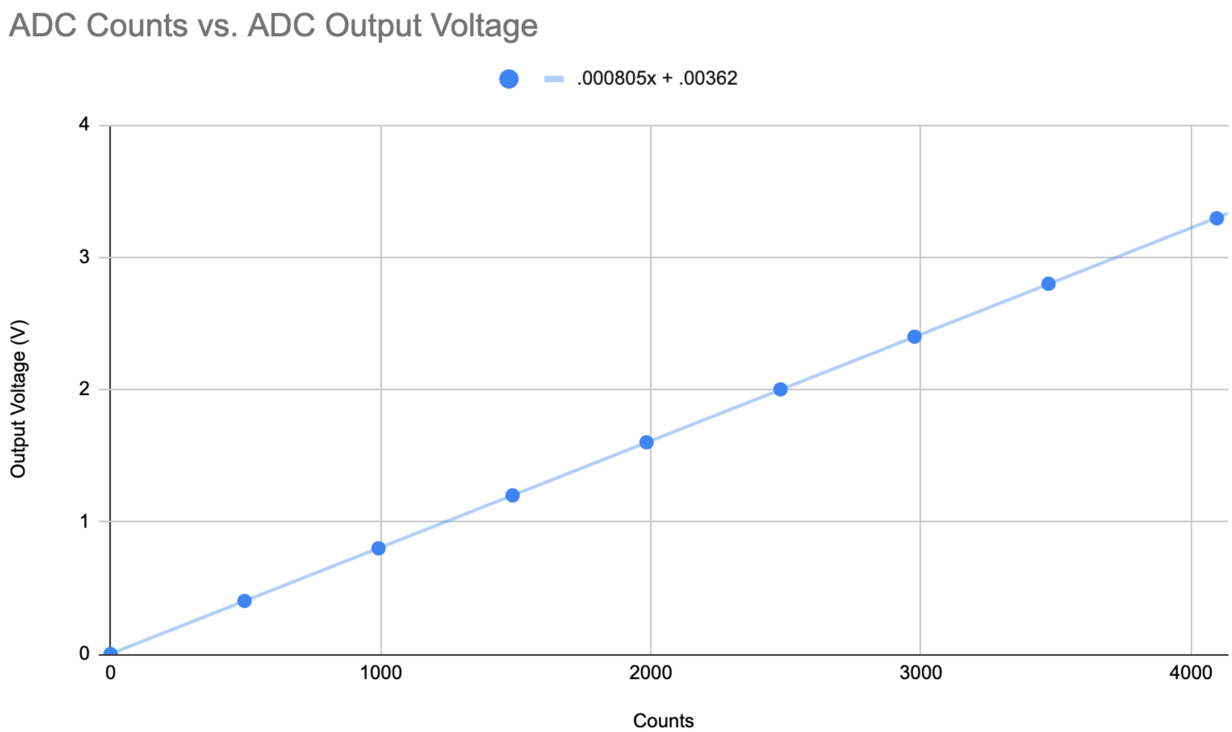
$$Resolution = \frac{FSR}{2^n - 1} = \frac{3.3V}{2^{12} - 1} = 0.806\ mV\ per\ count$$

Table and plot of raw ADC counts vs DC input voltage including linear regression with equation and R2 value *(including calibrated conversion from raw counts to volts)***:**

Table 1. Raw ADC Counts vs DC Input Voltage

Vin	ADC Counts	ADC Voltage
0	0	0.002
0.4	496	0.403
0.8	992	0.801
1.2	1488	1.201
1.6	1984	1.602
2	2480	2.002
2.4	2976	2.4
2.8	3472	2.8
3.3	4095	3.295

Figure 1. Raw ADC Counts vs DC Input Voltage



This graph's linear fit produced the following calibrated conversion equation (counts to volts):

```
return (805 * (uint32_t) adc_counts + 3620) / 1000;
```

Table of ADC min, max, and average voltages for sample time 47.5 and 640.5 clocks:

Table 2. ADC Min, Max, and Average Voltages

Sample Rate	Min [V]	Max [V]	Average [V]
47.5 Clocks	1.497	1.518	1.500
640.5 Clocks	1.492	1.503	1.498

Tabular summary of measured VCEsat, VBE, VRE, IB, VCOIL, and IC (= ICOIL).

Include % error in the coil current calculated by the code relative to the benchtop DMM measurement. Describe how the code calculates coil current.

Table 3. Measured VCEsat, VBE, VRE, IB, VCOIL, and IC (ICOIL)

VCEsat	VBE	VRE	IB	VCOIL	ICOIL
0.15 V	0.72 V	0.26 V	2.58 mA	4.88 V	28.3 mA

Our code measured our current, ICOIL, to be around 31 mA. This means our code vs. measured error was **8.7%**.

The code calculates coil current using ohms law across the known resistor value. We use an analog voltage input to the ADC and convert that to the actual voltage across the emitter resistor. Since we know that the emitter resistor is 10 ohms, we can do I=V/R to calculate the coil current (since it's in series).

The following code shows how this process was implemented in the *print_current* function in LPUART.c.

```
Step 1: Get Emitter Voltage in uV:
uint32_t uV = calibrate_voltage(adc_val) * 1000;
Step 2: divide by Re (defined as MOHMS) to get the current
#define MOHMS 10000 // 10 Ohms RE
uint32_t uA = uV / MOHMS;
```

Appendices:

Appendix (main.h)

```c
/******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
******************************************************************************
* @file           : main.h
* @brief          : Header for main application
*
* project         : EE 329 S'25 Assignment 8
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
******************************************************************************/
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
#include "stm32l4xx_hal.h"
void SystemClock_Config(void);
void format_uint_to_str(uint16_t value, char* buffer, uint8_t width);
void format_voltage(uint32_t voltage_mV, char* buffer);
#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
```

```c
#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 */
```

Appendix (main.c)

```c
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : main.c
* @brief          : Main application code
*
* project         : EE 329 S'25 Assignment 8
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
*******************************************************************************/
#include "main.h"
#include <DELAY.h>
#include <LPUART.h>
#include <ADC.h>
void SystemClock_Config(void);
#define MILLIOHMS 10000 // 10 ohms in milliohms
/* -----------------------------------------------------------------------------
* function : main
* INs      : none
* OUTs     : none
* action   : display voltage and currents read from DMM
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
int main(void)
{
   //initialize variables for later use
   uint8_t idx = 0;
```

```c
uint8_t fullarr = 0;
uint16_t adc_result[32];
HAL_Init();
SystemClock_Config();
// Enable GPIO clocks
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOC_CLK_ENABLE();
// Configure PA1 (SET) and PC3 (RESET) as output push-pull
GPIO_InitTypeDef GPIO_InitStruct = {0};
GPIO_InitStruct.Pin = GPIO_PIN_1;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
GPIO_InitStruct.Pin = GPIO_PIN_3;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
// Configure PC13 as input (USER button)
GPIO_InitStruct.Pin = GPIO_PIN_13;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
// Initialize software delay, UART, and ADC
SysTick_Init();
LPUART_init();
LPUART_Print("\x1b[2J"); // Clear terminal screen
ADC_Init();
while (1) {
    LPUART_Print("\x1b[2J"); // Clear terminal screen
    LPUART_Print("\x1b[H"); //Set cursor upper left
    // Relay control via USER button on PC13 (active-low logic)
    if ((GPIOC->IDR & GPIO_PIN_13) == 0) {
        GPIOA->BSRR = GPIO_PIN_1; // Pulse SET coil (PA1)
        delay_us(10000);
        GPIOA->BRR = GPIO_PIN_1;
    } else {
        GPIOC->BSRR = GPIO_PIN_3; // Pulse RESET coil (PC3)
        delay_us(10000);
        GPIOC->BRR = GPIO_PIN_3;
    }
    // Process ADC sample when conversion is complete
    if (adc_ready) {
        adc_ready = 0; // Reset flag set by ADC interrupt
        // Store sample and update circular buffer idx
        adc_result[idx++] = adc_measurement;
        if (idx >= 20) {
            idx = 0;
            fullarr = 1;
        }
        // compute values if array is full
```

```c
            if (fullarr) {
                //set the sum to 0 and max low and min high
                uint32_t arraysum = 0;
                uint32_t min = 0xFFFF;
                uint32_t max = 0;
                //iterate through array and update mins and maxes
                for (int i = 0; i < 20; i++) {
                    arraysum += adc_result[i];
                    if (adc_result[i] < min) min = adc_result[i];
                    if (adc_result[i] > max) max = adc_result[i];
                }
                //calculate average of array
                uint16_t avg = arraysum / 20;
                // Display values in terminal
                LPUART_Print("ADC counts volts\r\n");
                LPUART_Print("MIN "); print_uint4(min); LPUART_Print(" ");
print_voltage(min);
                LPUART_Print("MAX "); print_uint4(max); LPUART_Print(" ");
print_voltage(max);
                LPUART_Print("AVG "); print_uint4(avg); LPUART_Print(" ");
print_voltage(avg);
                print_current(avg); // Display estimated coil current
                //small delay for readability
                delay_us(100000);
            }
            ADC1->CR |= ADC_CR_ADSTART; // Start next ADC conversion
        }
    }
    return 0;
}
/* -----------------------------------------------------------------------------
* function : SystemClock_Config()
* INs      : N/A
* OUTs     : N/A
* action   : configures system clocks using MSI as source with PLL to achieve
desired
*            SYSCLK, HCLK, PCLK1, and PCLK2 frequencies for STM32L4A6ZG
* authors  : STM-Generated
* version  : 0.1
* date     : May 28, 2025
* -------------------------------------------------------------------------- */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
    HAL_PWR_EnableBkUpAccess();
    __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSE |
RCC_OSCILLATORTYPE_MSI;
```

```
  RCC_OscInitStruct.LSEState = RCC_LSE_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 71;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV6;
  HAL_RCC_OscConfig(&RCC_OscInitStruct);
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                              | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4);
  HAL_RCCEx_EnableMSIPLLMode();
}
```

Appendix (ADC.h)

```
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : ADC.h
* @brief          : Header file for Analog-Digital Converter Routines
*
* project         : EE 329 S'25 Assignment 8
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
*******************************************************************************/
#ifndef INC_ADC_H_
#define INC_ADC_H_
// Global variables shared with main
extern volatile uint16_t adc_measurement;
extern volatile uint8_t adc_ready;
// Function prototypes
void ADC_Init(void);
void ADC1_2_IRQHandler(void);
#endif /* INC_ADC_H_ */
```

Appendix (ADC.c)

```c
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : ADC.c
* @brief          : ADC (Analog-Digital Converter) Function Definitions
*
* project         : EE 329 S'25 Assignment 8
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
*******************************************************************************/
#include <main.h>
#include <DELAY.h>
#include <ADC.h>
volatile uint16_t adc_measurement = 0;  // Global variable to store ADC result
volatile uint8_t adc_ready = 0;     // Global flag to indicate data is ready
/* -----------------------------------------------------------------------------
* function : ADC_Init
* INs      : none
* OUTs     : none
* action   : initialize ADC
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void ADC_Init(void)
{
      RCC->AHB2ENR |= RCC_AHB2ENR_ADCEN;         // turn on clock for ADC
      // power up & calibrate ADC
      ADC123_COMMON->CCR |= (1 << ADC_CCR_CKMODE_Pos); // clock source = HCLK/1
      ADC1->CR &= ~(ADC_CR_DEEPPWD);             // disable deep-power-down
      ADC1->CR |= (ADC_CR_ADVREGEN);             // enable V regulator - see RM
18.4.6
      delay_us(20);                              // wait 20us for ADC to power up
      ADC1->DIFSEL &= ~(ADC_DIFSEL_DIFSEL_5);    // PA0=ADC1_IN5, single-ended
      ADC1->CR &= ~(ADC_CR_ADEN | ADC_CR_ADCALDIF); // disable ADC, single-end calib
      ADC1->CR |= ADC_CR_ADCAL;                  // start calibration
      while (ADC1->CR & ADC_CR_ADCAL) {;}        // wait for calib to finish
      // enable ADC
      ADC1->ISR |= (ADC_ISR_ADRDY);              // set to clr ADC Ready flag
      ADC1->CR |= ADC_CR_ADEN;                   // enable ADC
      while(!(ADC1->ISR & ADC_ISR_ADRDY)) {;}    // wait for ADC Ready flag
      ADC1->ISR |= (ADC_ISR_ADRDY);              // set to clr ADC Ready flag
```

```c
    // configure ADC sampling & sequencing
    ADC1->SQR1  |= (5 << ADC_SQR1_SQ1_Pos);     // sequence = 1 conv., ch 5
    ADC1->SMPR1 |= (1 << ADC_SMPR1_SMP5_Pos);   // ch 5 sample time = 6.5 clocks
    ADC1->CFGR  &= ~( ADC_CFGR_CONT  |          // single conversion mode
                      ADC_CFGR_EXTEN |          // h/w trig disabled for s/w trig
                      ADC_CFGR_RES   );         // 12-bit resolution
    // configure & enable ADC interrupt
    ADC1->IER |= ADC_IER_EOCIE;                 // enable end-of-conv interrupt
    ADC1->ISR |= ADC_ISR_EOC;                   // set to clear EOC flag
    NVIC->ISER[0] = (1<<(ADC1_2_IRQn & 0x1F)); // enable ADC interrupt service
    __enable_irq();                             // enable global interrupts
    // configure GPIO pin PA0
    RCC->AHB2ENR  |= (RCC_AHB2ENR_GPIOAEN);     // connect clock to GPIOA
    GPIOA->MODER &= ~(GPIO_MODER_MODE0_Msk);
    GPIOA->MODER  |= (GPIO_MODER_MODE0);        // analog mode for PA0 (set MODER
last)
    ADC1->CR |= ADC_CR_ADSTART;                 // start 1st conversion
}
/* -----------------------------------------------------------------------------
* function : ADC1_2_IRQHandler
* INs      : none
* OUTs     : none
* action   : check for data read in adc
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* --------------------------------------------------------------------------- */
void ADC1_2_IRQHandler(void)
{
   if (ADC1->ISR & ADC_ISR_EOC) // Check if End of Conversion caused the interrupt
   {
       adc_measurement = ADC1->DR;   // Read result from data register (clears EOC
flag)
       adc_ready = 1;            // Set flag for main loop
   }
}
```

Appendix (LPUART.h)

```c
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : LPUART.h
* @brief          : Header file for LPUART Routines
*
* project         : EE 329 S'25 Assignment 8
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
```

```
* date              : May 28, 2025
* 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.
*******************************************************************************/
#ifndef INC_LPUART_H_
#define INC_LPUART_H_
#include "main.h"
#include "stm32l4xx.h"
#define UART_PORT GPIOG       // Define the UART port as GPIOG
#define LEFT_BORDER 20        //Define Left border position
#define RIGHT_BORDER 140//Define Right border position
#define UPPER_BORDER 35       //define upper border position
#define LOWER_BORDER 5        //define lower border position
// Function Prototypes
void LPUART_init(void);
void LPUART_Print(const char* message);
void LPUART1_IRQHandler(void);
void LPUART_ESC_Print(const char* esc_code, const char* message);
void Splash_Screen(void);
void Create_Border(void);
void print_current(uint16_t adc_val);
void print_voltage(uint16_t adc_val);
void print_uint4(uint16_t val);
#endif /* INC_LPUART_H_ */
```

Appendix (LPUART.c)

```
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file             : LPUART.c
* @brief            : LPUART Function Definitions
*
* project           : EE 329 S'25 Assignment 8
* authors           : Preston Mavady, Brayden Daly
* version           : 0.1
* date              : May 28, 2025
* 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.
*******************************************************************************/
#include <main.h>
#include <LPUART.h>
//global variables to hold the position and start game interrupt
uint32_t x = 0;
uint32_t y = 0;
```

```c
#define MOHMS 10000 // 10 Ohms RE
/* -----------------------------------------------------------------------------
* function : LPUART_init
* INs      : none
* OUTs     : none
* action   : initializes UART GPIO (PORT G 7,8), sets baud rate
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* ---------------------------------------------------------------------------- */
void LPUART_init(void)
{
      PWR->CR2 |= (PWR_CR2_IOSV);  // power avail on PG[15:2] (LPUART1)
      RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOGEN);  // enable UART_PORT clock
      RCC->APB1ENR2 |= RCC_APB1ENR2_LPUART1EN; // enable LPUART clock bridge
      /* USER: configure UART_PORT registers MODER/PUPDR/OTYPER/OSPEEDR then
       select AF mode and specify which function with AFR[0] and AFR[1] */
      //RECEIVER (RX) PG8
      UART_PORT->MODER &= ~(GPIO_MODER_MODE7 | GPIO_MODER_MODE8);
      //TRANSMITTER (TX) PG7
      UART_PORT->MODER |= (GPIO_MODER_MODE7_1 | GPIO_MODER_MODE8_1); // Set AF mode
(10)
      UART_PORT->OTYPER &= ~(GPIO_OTYPER_OT7);
      //pullups
      //UART_PORT->PUPDR |= (GPIO_PUPDR_PUPD7_0 | GPIO_PUPDR_PUPD8_0); // Enable
pull-up for PG7 and PG8
      UART_PORT->PUPDR &= 0; // Enable pull-up for PG7 and PG8
      // Set very high output speed for PC7, 8
      UART_PORT->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED7_Pos) |
                              (3 << GPIO_OSPEEDR_OSPEED8_Pos));
      UART_PORT->BRR = (GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3); //
preset PC0, PC1, PC2, PC3 to 0
      UART_PORT->AFR[0] &= ~(0x000F << GPIO_AFRL_AFSEL7_Pos); // clear PG7 nibble AF
      UART_PORT->AFR[0] |= (0x0008 << GPIO_AFRL_AFSEL7_Pos); // set PG7 AF =
LPUART1_TX
      UART_PORT->AFR[1] &= ~(0x000F << GPIO_AFRH_AFSEL8_Pos); // clear PG8 nibble AF
      UART_PORT->AFR[1] |= (0x0008 << GPIO_AFRH_AFSEL8_Pos); // set PG8 AF =
LPUART1_RX
      LPUART1->CR1 &= ~(USART_CR1_M1 | USART_CR1_M0); // 8-bit data
      LPUART1->CR1 |= USART_CR1_UE;                   // enable LPUART1
      LPUART1->CR1 |= (USART_CR1_TE | USART_CR1_RE);  // enable xmit & recv
      LPUART1->CR1 |= USART_CR1_RXNEIE;         // enable LPUART1 recv interrupt
      LPUART1->ISR &= ~(USART_ISR_RXNE);        // clear Recv-Not-Empty flag
      /* USER: set baud rate register (LPUART1->BRR) */
      LPUART1->BRR = 0x22B9; //8889
      NVIC->ISER[2] = (1 << (LPUART1_IRQn & 0x1F));   // enable LPUART1 ISR
      __enable_irq();
      //clear screen and move cursor to top left
```

```
      LPUART_Print( "\x1b[2J");
      LPUART_Print( "\x1b[H");
}
/* ----------------------------------------------------------------------------
* function : LPUART_Print
* INs      : string
* OUTs     : none
* action   : print a string to the screen
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void LPUART_Print( const char* message ) {
  uint16_t iStrIdx = 0;
  while ( message[iStrIdx] != 0 ) {
     while(!(LPUART1->ISR & USART_ISR_TXE)) // wait for empty xmit buffer
        ;
     LPUART1->TDR = message[iStrIdx];       // send this character
      iStrIdx++;                             // advance index to next char
  }
}
/* ----------------------------------------------------------------------------
* function : LPUART_IRQHandler
* INs      : none
* OUTs     : none
* action   : Interrupt Handler to check for ESC key presses
*
* authors  : Brayden Daly, Zachary Lee
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void LPUART1_IRQHandler( void  ) {
      //set variable to hold which character was received
  uint8_t charRecv;
  //check if there was an interrupt
  if (LPUART1->ISR & USART_ISR_RXNE) {
         //set charRecv as the input from the UART interrupt
     charRecv = LPUART1->RDR;
     //switch to character inputted
     switch ( charRecv ) {
     //If R pressed, make font color red
         case 'R':
               LPUART_Print( "\x1b[31m");
            break;
            //for default, wait for TX buffer
         default:
            while( !(LPUART1->ISR & USART_ISR_TXE) )
            {
```

```c
          ;    // wait for empty TX buffer
         }
         //reset start game
         LPUART1->TDR = charRecv;  // echo char to terminal
      }  // end switch
   }
}
/* ------------------------------------------------------------------------------
* function : print_uint4
* INs      : uint16_t
* OUTs     : none
* action   : print the value as a uint8_t
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void print_uint4(uint16_t val) {
      //initialize array and store the values
      char out[5];
      out[0] = '0' + (val / 1000) % 10;
      out[1] = '0' + (val / 100) % 10;
      out[2] = '0' + (val / 10) % 10;
      out[3] = '0' + (val % 10);
      out[4] = '\0';
      //print the values to terminal
      LPUART_Print((uint8_t*)out);
}
/* ------------------------------------------------------------------------------
* function : calibrate_voltage
* INs      : none
* OUTs     : none
* action   : use line of best fit to calibrate the adc
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
//calibration, V = 0.000805x + 0.00362
uint32_t calibrate_voltage(uint16_t adc_counts) {
      return (805 * (uint32_t) adc_counts + 3620) / 1000;
}
/* ------------------------------------------------------------------------------
* function : print voltage
* INs      : none
* OUTs     : none
* action   : print the voltage in volts
*
* authors  : Brayden Daly, Preston
```

```c
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void print_voltage(uint16_t adc_val) {
	//calibrate the voltage
	uint32_t mV = calibrate_voltage(adc_val);
	//initialze array and store the mv values in each index of array
	char out[10];
	out[0] = '0' + (mV / 1000);
	out[1] = '.';
	out[2] = '0' + (mV % 1000) / 100;
	out[3] = '0' + (mV % 100) / 10;
	out[4] = '0' + (mV % 10);
	out[5] = ' ';
	out[6] = 'V';
	out[7] = '\r';
	out[8] = '\n';
	out[9] = '\0';
	//print the voltage
	LPUART_Print((uint8_t*)out);
}
/* ----------------------------------------------------------------------------
* function : print_current
* INs      : none
* OUTs     : none
* action   : print the current in amps
*
* authors  : Brayden Daly, Preston
* version  : 0.3
* date     : 250507
* -------------------------------------------------------------------------- */
void print_current(uint16_t adc_val) {
	//calibrate voltage
	uint32_t uV = calibrate_voltage(adc_val) * 1000;
	//divide by Re to get the current
	uint32_t uA = uV / MOHMS;
	//initialize array and store microamps in indices
	char out[12];
	out[0] = '0' + (uA / 1000);
	out[1] = '.';
	out[2] = '0' + (uA % 1000) / 100;
	out[3] = '0' + (uA % 100) / 10;
	out[4] = '0' + (uA % 10);
	out[5] = ' ';
	out[6] = 'A';
	out[7] = '\r';
	out[8] = '\n';
	out[9] = '\0';
	//print the coil current and value of coil current
```

```
	LPUART_Print((uint8_t*)"coil current = ");
	LPUART_Print((uint8_t*)out);
}
```

Appendix (delay.h)

```
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : delay.h
* @brief          : Delay utility for microsecond timing
*
* project         : EE 329 S'25 Assignment 4
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
*******************************************************************************/
#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
```

Appendix (delay.c)

```
/*******************************************************************************
* EE 329 A8 Analog to Digital Converter (ADC)
*******************************************************************************
* @file           : delay.c
* @brief          : Microsecond delay function definitions
*
* project         : EE 329 S'25 Assignment 4
* authors         : Preston Mavady, Brayden Daly
* version         : 0.1
* date            : May 28, 2025
* 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.
*******************************************************************************/
#include "main.h"
#include "Delay.h"
#include "stm32l4xx.h"
#include <stdint.h>
/* -----------------------------------------------------------------------------
* function : SysTick_Init(void);
```

```c
* 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
}
```