Brayden Daly, Alex Nguyen

Lab A9

EE 329-05

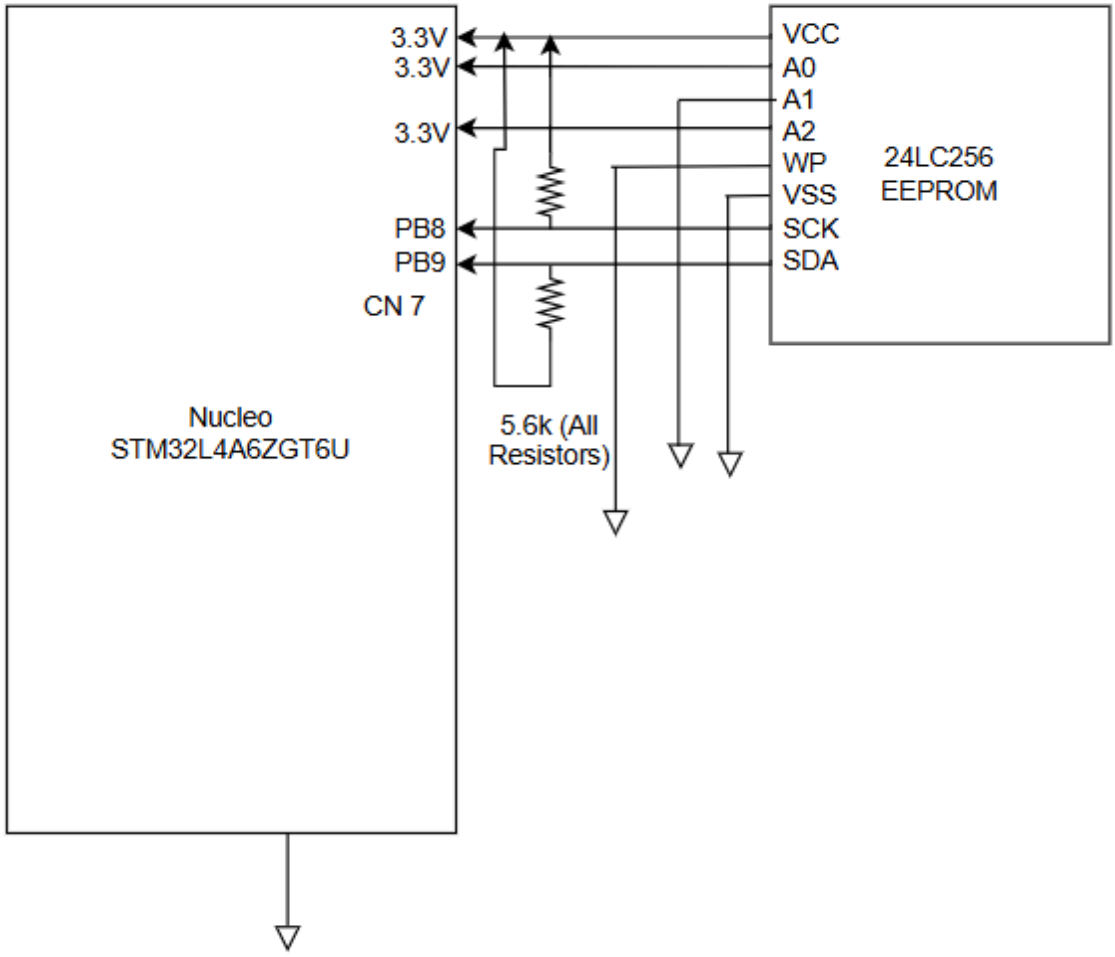
2025 May 22

A9 - I2C

Introduction:

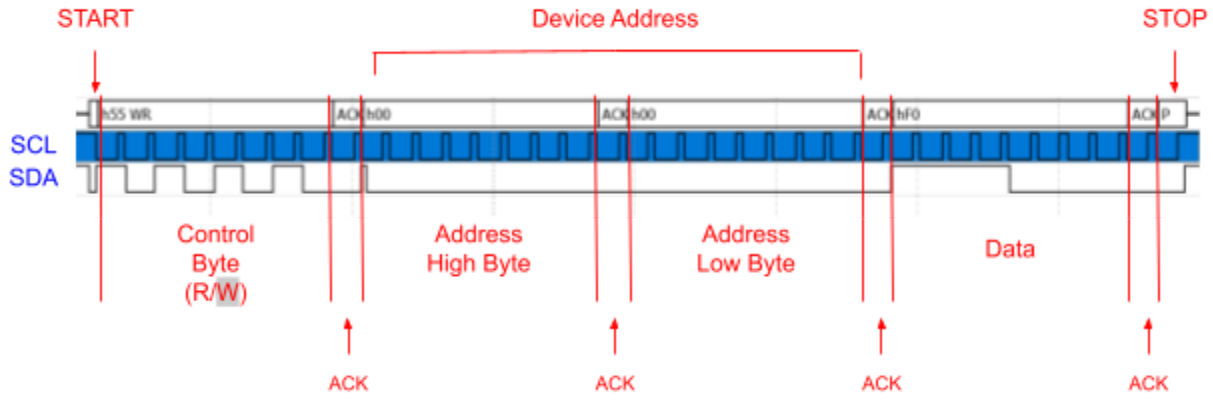
In this lab we set up I2C communication to read and write from an EEPROM. We wrote code to initialize the I2C bus (AF MODE, NO PUPDR, HIGH SPEED, OPEN DRAIN MODE, AF4 using I2C1), and wrote functions to write random data at a random address in the EEPROM then read that data. No further suggestions.

Wiring Diagram:

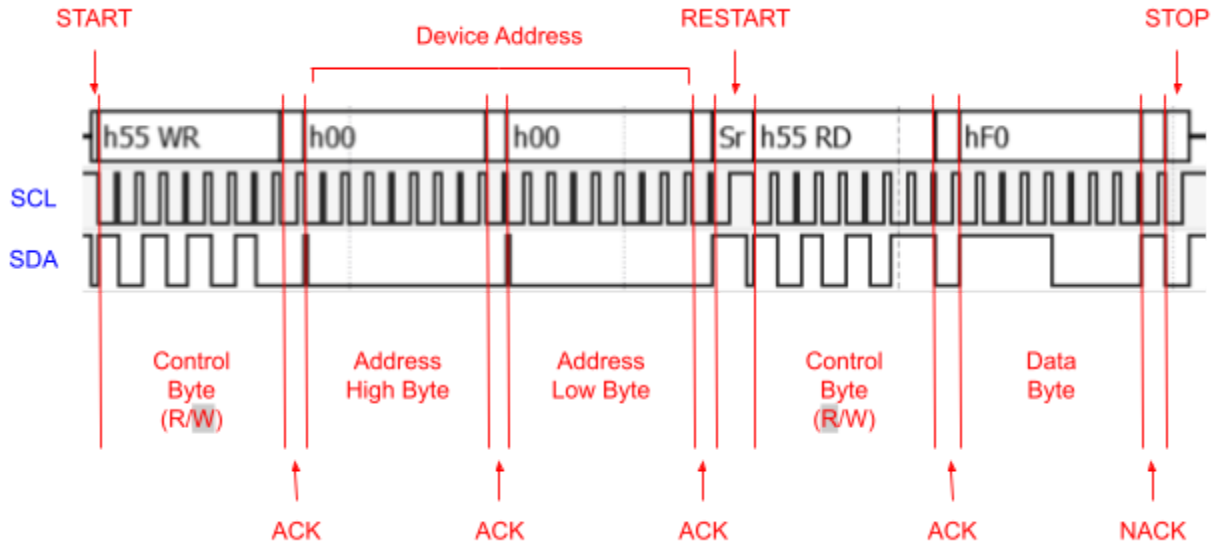


Annotated Logic Analyzer Images:

WRITE



READ



As shown in the Logic Analyzer outputs, we wrote 0xF0 to address 0x0000 and the EEPROM's address was 0x55. Then when we read the data from the EEPROM, we read from 0x0000 and the address of the chip was 0x55 as expected, and the data there was 0xF0 as expected.

Appendix

---

main.h

```
/** EE 329 A9 I2C EEPROM
 ******************************************************************************
 * @file           : main.h
 * @brief          : Header for main.c file.
 *                   This file contains the common defines of the application.
 * @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
 ******************************************************************************
 *45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#ifndef __MAIN_H
#define __MAIN_H
#ifdef __cplusplus
extern "C" {
#endif
/* Includes ------------------------------------------------------------------*/
#include "stm32l4xx_hal.h"
void Error_Handler(void);
#ifdef __cplusplus
}
#endif
#endif /* __MAIN_H */
```

---

main.c

```
/******************************************************************************
* EE 329 A9 I2C EEPROM
******************************************************************************
* @file           : main.c
* @brief          : write data to EEPROM and read to verify
* project         : EE 329 S'25 Assignment 9
* authors         : Brayden Daly (bmd) - bdaly01@calpoly.edu
* version         : 0.1
* date            : 250522
* compiler        : STM32CubeIDE v.1.18.0 Build: 24413_20250227_1633 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
******************************************************************************
* MAIN PROGRAM Plan :
* Get random byte of data
```

```
* write data to EEPROM
* wait for 5 milliseconds
* read data from EEPROM
*******************************************************************************
* REVISION HISTORY
* 0.1 250522 bmd  completed functions for EEPROM
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#include "main.h"
#include "EEPROM.h"
#include "DELAY.h"
#include "RANDOM.h"
void SystemClock_Config(void);
uint16_t Get_Ran_Num(void);
void On_Board_LED(void);
/* ----------------------------------------------------------------------------
* function : main()
* INs      : none
* OUTs     : int
* action   : randomly generates address data and writes to EEPRROM
*            wait 5 ms then reads data from EEPROM
* authors  : Brayden Daly, Alex Nguyen
* version  : 0.3
* date     : 250522
* ---------------------------------------------------------------------------- */
int main(void) {
   ///initialize HAL and Systemclock
   HAL_Init();
   SystemClock_Config();
   //initialize EEPROM for I2C
   EEPROM_INIT();
   //initialize on board LED
   On_Board_LED();
   //initialize random number generator
   RNG_Init();
   GPIOC->ODR &= ~(1U << 7); //set LED low
   uint8_t random_data = Get_Ran_Num(); //random byte of data
   uint16_t random_address = Get_Ran_Num(); //random address
   while (1) {
      //write a byte to EEPROM address 0x55
      EEPROM_WriteByte(random_address, random_data); //write to 0x55
      //delay 5 seconds
      delay_us(5000);
```

```
      //read a byte from EEPROM
      //read from same address
      uint8_t data_check = EEPROM_ReadByte(random_address, 0);
      //check if data sent is the same data read
      if (random_data == data_check) { //check
         GPIOC->ODR |= (1U << 7);  // LED ON
      }
   }
}
```

EEPROM.h

```
/** EE 329 A9 I2C EEPROM
******************************************************************************
* @file           : EEPROM.h
* @brief          : Header for EEPROM.c file.
*                   This file contains the common defines of the application.
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
******************************************************************************
*45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#ifndef EEPROM_H_
#define EEPROM_H_
#include "main.h"
#define I2CBUS I2C1           //using I2C1
#define EEPROM_ADDR 0x55      //using Address 0x55
#define TIMINGVAL 0x00E0257A  //Timing value for 4MHz Clock
#define I2CPORT GPIOB
#define SDA 9
#define SCK 8
void EEPROM_INIT(void);
uint8_t EEPROM_WriteByte(uint16_t address, uint16_t data);
uint8_t EEPROM_ReadByte(uint16_t address, uint16_t *data);
#endif /* INC_EEPROM_H_ */
```

EEPROM.c

```
/******************************************************************************
* EE 329 A9 I2C EEPROM
******************************************************************************
* @file           : EEPROM.c
* @brief          : functions to initialize, read, and write to EEPROM
* project         : EE 329 S'25 Assignment 9
* authors         : Brayden Daly (bmd) - bdaly01@calpoly.edu
```

```
* version         : 0.1
* date            : 250522
* compiler        : STM32CubeIDE v.1.18.0 Build: 24413_20250227_1633 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
* EEPROM Plan :
* function to initialize EEPROM
*  Configure I2C protocol
*  Configure AF pins for SCL and SDA
* function to write to EEPROM
* function to read from EEPROM
*******************************************************************************
* EEPROM WIRING (NUCLEO-L4A6ZG = L496ZG)
* 3V3 - VCC - CN8 -7 - OUT
* 3V3 - A0  - CN8 -7 - OUT
* GND - A1  - CN8-11 - OUT
* 3V3 - A2  - CN8 -7 - OUT
* GND - WP  - CN8-11 - OUT
* GND - VSS - CN8-11 - OUT
* PB8 - SCL - CN7 -2 - AF
* PB9 - SDA - CN7 -4 - AF
*******************************************************************************
* REVISION HISTORY
* 0.1 250522 bmd  completed functions for EEPROM
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#include "main.h"
#include "EEPROM.h"
/* ------------------------------------------------------------------------
* function : EEPROM_INIT()
* INs      : none
* OUTs     : none
* action   : initializes AF pins for SCL and SDA
*          configures the I2C protocol for interfacing with EEPROM
* authors  : Brayden Daly, Alex Nguyen
* version  : 0.3
* date     : 250522
* ------------------------------------------------------------------------ */
void EEPROM_INIT(void) {
   // Enable I2C1 and GPIO clocks
   RCC->AHB2ENR |= RCC_AHB2ENR_GPIOBEN;
```

```
    RCC->APB1ENR1 |= RCC_APB1ENR1_I2C1EN;
    // Configure PB8 (SCL) and PB9 (SDA) for AF4 (I2C1)
    GPIOB->MODER &= ~((3 << (8 * 2)) | (3 << (9 * 2)));      // Clear MODER
    GPIOB->MODER |= (2 << (8 * 2)) | (2 << (9 * 2));         // Set to AF mode
    GPIOB->OTYPER |= (1 << 8) | (1 << 9);                          // Open-drain
    GPIOB->OSPEEDR |= (3 << (8 * 2)) | (3 << (9 * 2));       // High speed
    GPIOB->AFR[1] |= (4 << ((8 - 8) * 4)) | (4 << ((9 - 8) * 4));  // AF4
    // Reset and configure I2C1
    I2C1->CR1 &= ~I2C_CR1_PE;        // Disable I2C
    I2C1->CR1 &= ~I2C_CR1_ANFOFF;    // Enable analog filter
    I2C1->CR1 &= ~I2C_CR1_DNF;       // Disable digital filter
    I2C1->TIMINGR = TIMINGVAL;       // Set timing from CubeMX
    I2C1->CR1 |= I2C_CR1_PE;         // Enable I2C
}
/* ------------------------------------------------------------------------
* function : EEPROM_WriteByte()
* INs      : 16 bit address, 16 bit data
* OUTs     : int for error
* action   : uses I2C protocol to send address and data to EEPROM
* authors  : Brayden Daly, Alex Nguyen
* version  : 0.3
* date     : 250522
* ------------------------------------------------------------------------ */
uint8_t EEPROM_WriteByte(uint16_t address, uint16_t data) {
    I2C1->CR1 &= ~I2C_CR1_PE;
    I2C1->CR1 |= I2C_CR1_PE;
    // Set up 7-bit address write mode for 3 bytes
    I2C1->CR2 = 0;
    I2C1->CR2 |= (EEPROM_ADDR << 1);          // Slave address
    I2C1->CR2 |= (3 << I2C_CR2_NBYTES_Pos);   // 2 addr bytes + 1 data
    I2C1->CR2 &= ~I2C_CR2_RD_WRN;             // Write mode
    I2C1->CR2 |= I2C_CR2_START | I2C_CR2_AUTOEND;
    // Transmit: MSB address
    //check ACK and SEND msb of Address
    while (!(I2CBUS->ISR & (I2C_ISR_TXIS | I2C_ISR_NACKF)))
        ;
    if (I2CBUS->ISR & I2C_ISR_NACKF) {
        I2CBUS->ICR |= I2C_ICR_NACKCF;
        return ERROR;
    }
    I2C1->TXDR = address >> 8;
    // Transmit: LSB address
    //check ACK and SEND msb of Address
```

```
    while (!(I2CBUS->ISR & (I2C_ISR_TXIS | I2C_ISR_NACKF)))
        ;
    if (I2CBUS->ISR & I2C_ISR_NACKF) {
        I2CBUS->ICR |= I2C_ICR_NACKCF;
        return ERROR;
    }
    I2C1->TXDR = address & 0xFF;
    // Transmit: data
    //check ACK and SEND msb of Address
    while (!(I2CBUS->ISR & (I2C_ISR_TXIS | I2C_ISR_NACKF)))
        ;
    if (I2CBUS->ISR & I2C_ISR_NACKF) {
        I2CBUS->ICR |= I2C_ICR_NACKCF;
        return ERROR;
    }
    I2C1->TXDR = data;
    // Wait for STOP
    while (!(I2C1->ISR & I2C_ISR_STOPF))
        ;
    I2C1->ICR |= I2C_ICR_STOPCF;  // Clear STOP flag
}
/* ------------------------------------------------------------------------
* function : EEPROM_ReadByte()
* INs      : 16 bit address, 16 bit data
* OUTs     : int for data or error
* action   : uses I2C protocol to read data at certain address from EEPROM
* authors  : Brayden Daly, Alex Nguyen
* version  : 0.3
* date     : 250522
* ------------------------------------------------------------------------ */
uint8_t EEPROM_ReadByte(uint16_t address, uint16_t *data) {
    I2C1->CR1 &= ~I2C_CR1_PE;
    I2C1->CR1 |= I2C_CR1_PE;
    I2C1->CR2 = 0;
    I2C1->CR2 |= (EEPROM_ADDR << 1);
    I2C1->CR2 |= (2 << I2C_CR2_NBYTES_Pos);     // 2 bytes addr
    I2C1->CR2 &= ~I2C_CR2_RD_WRN;               // Write
    I2C1->CR2 |= I2C_CR2_START;
    //check ACK and SEND msb of Address
    while (!(I2CBUS->ISR & (I2C_ISR_TXIS | I2C_ISR_NACKF)))
        ;
    if (I2CBUS->ISR & I2C_ISR_NACKF) {
        I2CBUS->ICR |= I2C_ICR_NACKCF;
```

```
      return ERROR;
   }
   I2C1->TXDR = address >> 8;
   //check ACK and SEND msb of Address
   while (!(I2CBUS->ISR & (I2C_ISR_TXIS | I2C_ISR_NACKF)))
      ;
   if (I2CBUS->ISR & I2C_ISR_NACKF) {
      I2CBUS->ICR |= I2C_ICR_NACKCF;
      return ERROR;
   }
   I2C1->TXDR = address & 0xFF;
   while (!(I2C1->ISR & I2C_ISR_TC))
      ; // Wait for transfer complete
   // Second: read 1 byte from set address
   I2C1->CR2 = 0;
   I2C1->CR2 |= (EEPROM_ADDR << 1);
   I2C1->CR2 |= (1 << I2C_CR2_NBYTES_Pos);
   I2C1->CR2 |= I2C_CR2_RD_WRN | I2C_CR2_START | I2C_CR2_AUTOEND;
   while (!(I2C1->ISR & I2C_ISR_RXNE))
      ;
   data = I2C1->RXDR;
   while (!(I2C1->ISR & I2C_ISR_STOPF))
      ;
   I2C1->ICR |= I2C_ICR_STOPCF;
   return data;
}
```

DELAY.h

```
/** EE 329 A9 I2C EEPROM
*******************************************************************************
* @file           : DELAY.h
* @brief          : Header for DELAY.c file.
*                   This file contains the common defines of the application.
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
*45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#ifndef INC_DELAY_H_
#define INC_DELAY_H_
void SysTick_Init(void);
void delay_us(const uint32_t time_us);
#endif
```

DELAY.c

```
/*******************************************************************************
* EE 329 A9 I2C EEPROM
*******************************************************************************
* @file           : DELAY.c
* @brief          : function for delay in microseconds
* project         : EE 329 S'25 Assignment 9
* authors         : Brayden Daly (bmd) - bdaly01@calpoly.edu
* version         : 0.1
* date            : 250522
* compiler        : STM32CubeIDE v.1.18.0 Build: 24413_20250227_1633 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
* REVISION HISTORY
* 0.1 250522 bmd  completed functions for EEPROM
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#include "main.h"
#include "Delay.h"
#include "stm32l4xx.h"
#include <stdint.h>
/* ------------------------------------------------------------------------------
* function : SysTick_Init(void);
* INs      : none
* OUTs     : none
* action   : Configures the ARM Cortex-M SysTick timer for microsecond delays.
*            Disables interrupts and sets it to use the processor clock.
* authors  : Brayden Daly, Tyler Wong
* version  : 0.3
* date     : 253004
* ------------------------------------------------------------------------------ */
void SysTick_Init(void) {
   SysTick->CTRL |= (SysTick_CTRL_ENABLE_Msk |       // Enable SysTick
         SysTick_CTRL_CLKSOURCE_Msk);    // Use processor clock
   SysTick->CTRL &= ~(SysTick_CTRL_TICKINT_Msk);    // Disable SysTick interrupt
}
/* ------------------------------------------------------------------------------
* function : delay_us(uint32_t time_us);
* INs      : time_us - number of microseconds to delay
* OUTs     : none (blocking delay)
```

```
* action   : Uses SysTick countdown to delay for specified number of microseconds.
*            Note: small values may result in longer-than-expected delay.
* authors  : Brayden Daly
* version  : 0.3
* date     : 253004
* ------------------------------------------------------------------------- */
void delay_us(const uint32_t time_us) {
   // Calculate number of clock cycles for the desired delay
   SysTick->LOAD = (uint32_t) ((time_us * (SystemCoreClock / 1000000)) - 1);
   SysTick->VAL = 0;                                          // Reset SysTick counter
   SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);       // Clear count flag
   while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk))
      ; // Wait for countdown
}
```

RANDOM.h

```
/** EE 329 A9 I2C EEPROM
*****************************************************************************
* @file           : RANDOM.h
* @brief          : Header for RANDOM.c file.
*                   This file contains the common defines of the application.
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*****************************************************************************
*45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#ifndef RANDOM_H_
#define RANDOM_H_
void On_Board_LED(void);
uint16_t Get_Ran_Num(void);
void RNG_Init(void);
#endif /* INC_RANDOM_H_ */
```

RANDOM.c

```
/*****************************************************************************
* EE 329 A9 I2C EEPROM
*****************************************************************************
* @file           : RANDOM.c
* @brief          : functions to initialize and generate random number
* project         : EE 329 S'25 Assignment 9
* authors         : Brayden Daly (bmd) - bdaly01@calpoly.edu
* version         : 0.1
* date            : 250522
```

```
* compiler        : STM32CubeIDE v.1.18.0 Build: 24413_20250227_1633 (UTC)
* target          : NUCLEO-L4A6ZG
* clocks          : 4 MHz MSI to AHB2
* @attention      : (c) 2025 STMicroelectronics.  All rights reserved.
*******************************************************************************
* RANDOM Plan :
* generate random data to write to EEPROM
*******************************************************************************
* REVISION HISTORY
* 0.1 250522 bmd  completed functions for EEPROM
*******************************************************************************
* 45678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-234567 */
#include "main.h"
#include "delay.h"
void RNG_Init(void) {
   //init RNG reg
   RCC->CRRCR |= RCC_CRRCR_HSI48ON;
   while ((RCC->CRRCR & RCC_CRRCR_HSI48RDY) == 0)
      ; // wait until it's ready
   RCC->AHB2ENR |= RCC_AHB2ENR_RNGEN;
   RNG->CR |= RNG_CR_RNGEN;
}
/* ---------------------------------------------------------------------------
* function : Get_Ran_Num()
* INs      : none
* OUTs     : random 16 bit number
* action   : random value generator
* authors  : Brayden Daly
* version  : 0.1
* date     : 250428
* ------------------------------------------------------------------------- */
uint16_t Get_Ran_Num(void) {
   while ((RNG->SR & RNG_SR_DRDY) == 0)
      ; // Wait until a number is ready
   uint16_t rand_num = RNG->DR;
   return rand_num;
}
/* ---------------------------------------------------------------------------
* function : On_Board_LED()
* INs      : none
* OUTs     : none
* action   : initilizes on board LED 1
* authors  : Brayden Daly, Alex N
```

```
* version  : 0.1
* date     : 250428
* ------------------------------------------------------------------------- */
void On_Board_LED(void) {
   // Enable clock for GPIOC (for PC7)
   RCC->AHB2ENR |= RCC_AHB2ENR_GPIOCEN;
   // Set PC7 to output mode
   GPIOC->MODER &= ~(3U << (7 * 2));   // Clear bits
   GPIOC->MODER |= (1U << (7 * 2));   // Set to output
   GPIOC->OTYPER &= ~(1U << 7);       // Push-pull
   GPIOC->OSPEEDR |= (3U << (7 * 2)); // High speed
   GPIOC->PUPDR &= ~(3U << (7 * 2)); // No pull
}
-------------------------------------------------------------------------------------
```