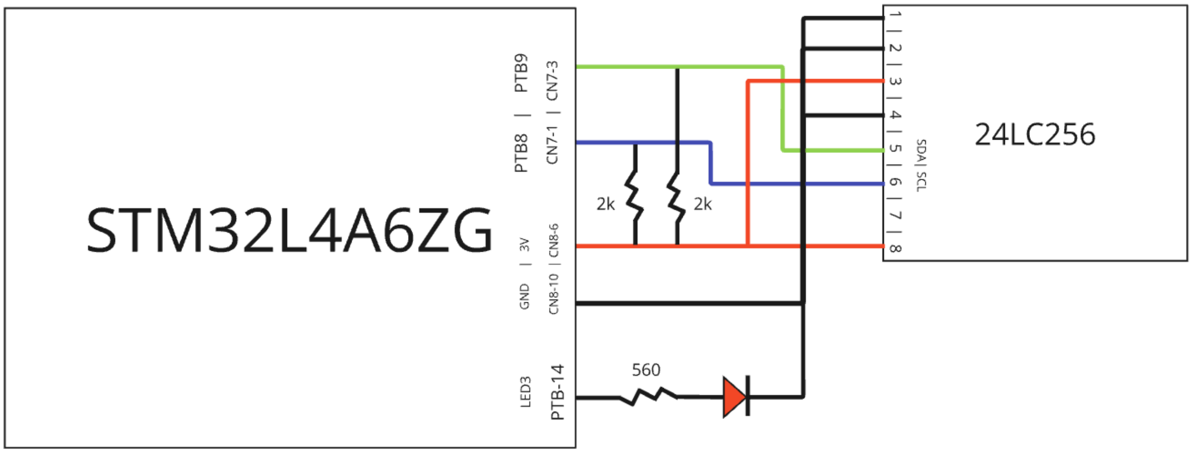
Wyatt Tack
Jonas Thyssen
EE 329-01 F'24
Group D
2024-Oct-28

EE 329 A9

This code is designed to use I2C to communicate to a 256k bit memory. I2C uses a 2 line communication that consists of 2 wires: a data bus and a clock bus. The device works through selecting the peripheral on the data bus (through messaging the peripherals address on the bus), and then writing the data address of the device wanted to read or write. The data is either then written, or the communication is started in read mode, and the data is read. This device works through selecting a random data address, writing a random data byte, then waits 5 seconds before reading that same address. If the data written is the same as the data read, turn the LED on, else turn it off. The device currently works in good condition, seen as how the LED stays on consistently.

Device Wiring Diagram:



Captures:

Figure A9.a: Sample Write Communication

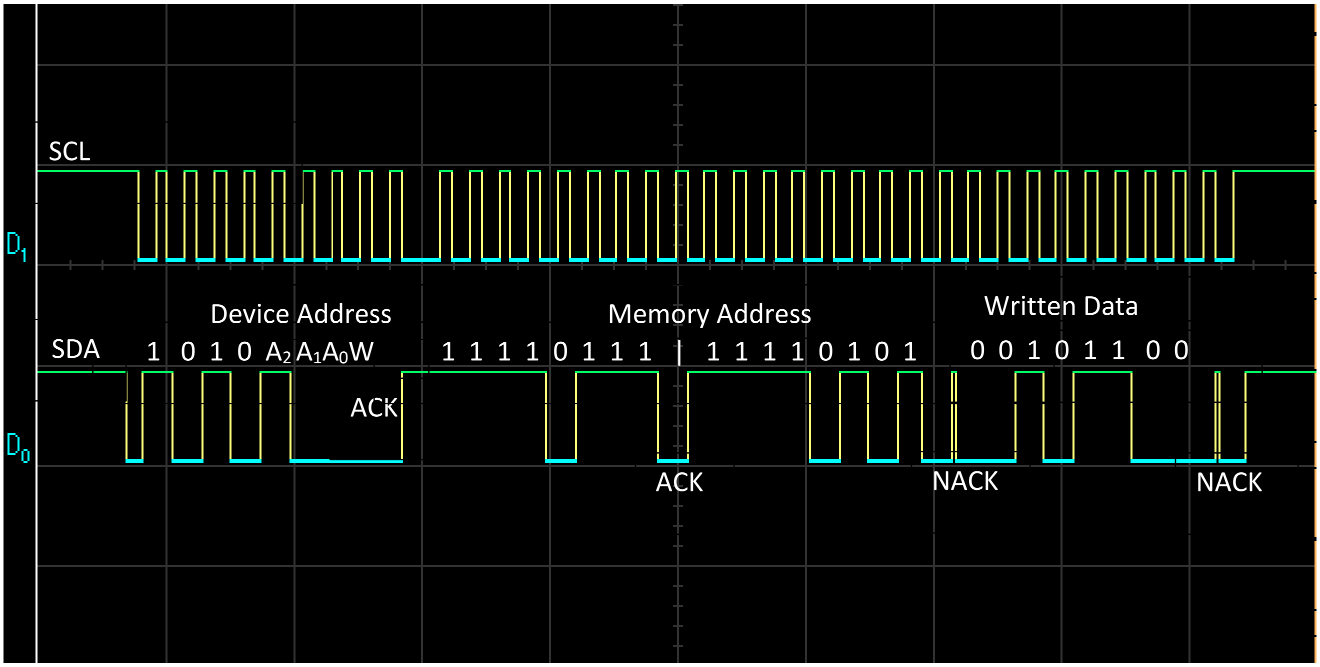
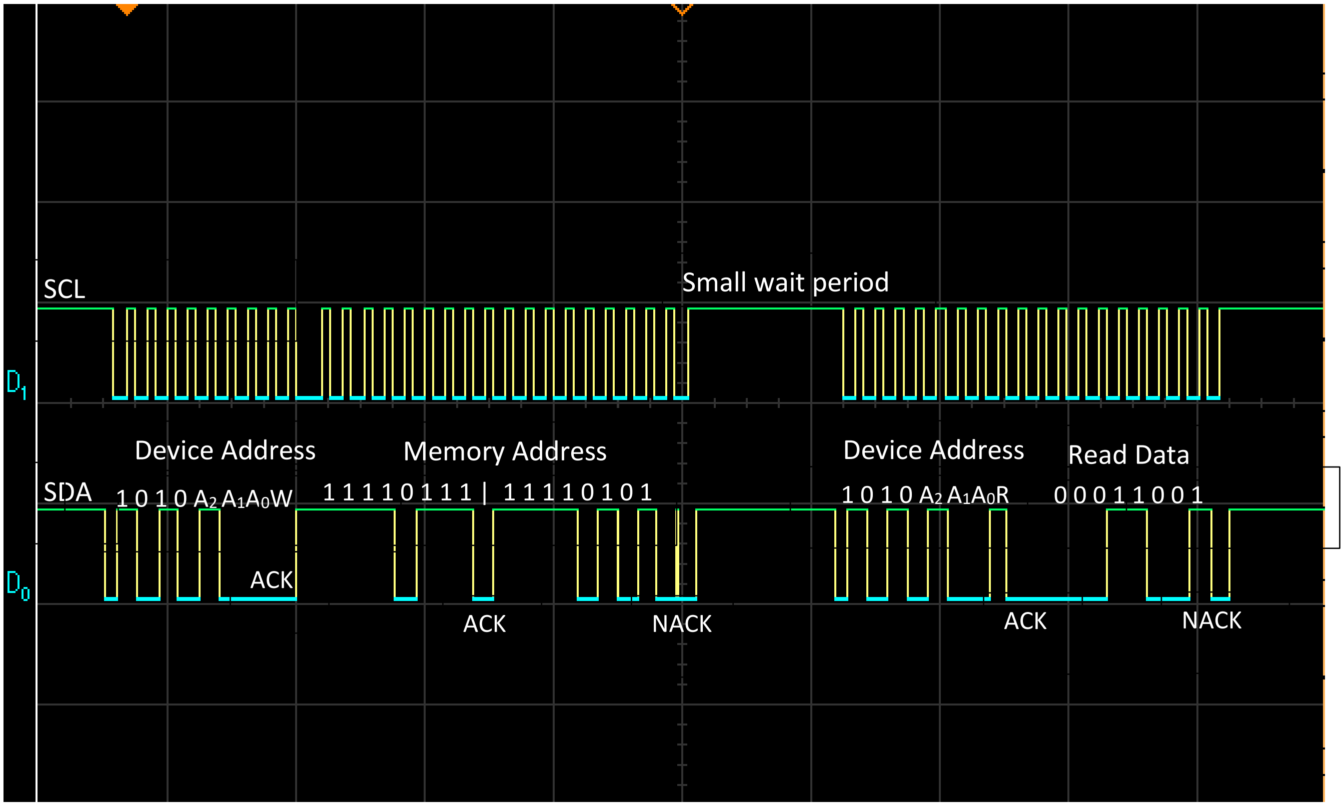


Figure A9.b: Sample Read Communication



Formatted Source Code main.h:

```
/**
  ******************************************************************************
  * @file                  : main.h
  * project                : EE329 Lab A9
  * author                 : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                   : 10/21/2024
  * 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

/* Created defines and function prototypes ----------------------------------*/


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

/* Exported functions prototypes ---------------------------------------------*/
#define EEPROM_ADDRESS 0x54
#define LED_PORT GPIOB

void Led_Config(void);
void SystemClock_Config(void);
void Error_Handler(void);


#ifdef __cplusplus
}
#endif

#endif
```

Formatted Source Code main.c:

```c
/**
  ******************************************************************************
  * @file           : main.c
  * project                                   : EE329 Lab A9
  * author                                    : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                                      : 10/21/2024
  * firmware                   : ST-Link V1
  * @attention                 : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *             Device uses I2C to write to an EEPROM connected at:
  *                            PTB-8: SCL
  *                            PTB-9: SDA
  * The device writes a byte to an address, then reads the byte at that address.
  * If data read is the same as data written, then the on board LED will turn on,
  * else the LED will turn off.
  ******************************************************************************
  */

#include "main.h"
#include "delay.h"
#include "i2c.h"



int main(void)
{
               //Initialize clock, I2C config
               HAL_Init();
               SystemClock_Config();
               SysTick_Init();
               I2C_Init();
               Led_Config();
               uint8_t rngData;
               uint16_t rngAddr;
               while (1){
                              rngData = 0x15;                //8 bit data
                              rngAddr = 0x1515;                              //16 bit address
                              LED_PORT->BRR |= (GPIO_PIN_14);               //reset light
                              I2C_Write (EEPROM_ADDRESS, rngAddr, rngData); //write
                              delay_us(5000);
                              //wait for data set
                              if (I2C_Read (EEPROM_ADDRESS, rngAddr) == rngData)
                                             LED_PORT->BSRR |= (GPIO_PIN_14);              //if data set, set LED
                              delay_us(10000);                                                              //delay between samples
               }
}

//--------------------------- GPIO -----------------------------------------
void Led_Config(void)
{                              // configure GPIO pin PB14 for:
                               // output mode, push-pull, no pull up or pull down, high speed
  RCC->AHB2ENR   |=  (RCC_AHB2ENR_GPIOBEN);
  LED_PORT->MODER   &= ~(GPIO_MODER_MODE14);
  LED_PORT->MODER   |=  (GPIO_MODER_MODE14_0);
  LED_PORT->OTYPER  &= ~(GPIO_OTYPER_OT14);
  LED_PORT->PUPDR   &= ~(GPIO_PUPDR_PUPD14);
  LED_PORT->OSPEEDR |=  (3 << GPIO_OSPEEDR_OSPEED14_Pos);
  LED_PORT->BRR |= (GPIO_PIN_14);
}

//----------------------------- 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 i2c.h:

```
/**
  *******************************************************************************
  * @file                 : i2c.h
  * project               : EE329 Lab A9
  * author                : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                  : 10/21/2024
  * firmware              : ST-Link V1
  * @attention            : Copyright (c) 2024 STMicroelectronics. All rights
reserved.
  *******************************************************************************
  *
  *      i2c header for defines and function prototypes
  *
  *******************************************************************************
  */

#ifndef INC_I2C_H_
#define INC_I2C_H_

/* Created defines and function prototypes ------------------------------------*/
#include "delay.h"
#define EEPROM_ADDRESS 0x54
#define EEPROM_MEMORY_ADDR 0xf7f5

void I2C_Init (void);
uint8_t I2C_Read (uint8_t devAddr, uint16_t addr);
void I2C_Write (uint8_t devAddr, uint16_t addr, uint8_t data);

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

#endif /* INC_I2C_H_ */
```

Formatted Source Code i2c.c:

```
               /**
  ******************************************************************************
  * @file                     : i2c.c
  * project                   : EE329 Lab A9
  * author                    : Wyatt Tack (wwt) - wtack@calpoly.edu
  * date                      : 10/21/2024
  * firmware                  : ST-Link V1
  * @attention                : Copyright (c) 2024 STMicroelectronics. All rights reserved.
  ******************************************************************************
  *
  *            Device uses I2C on GPIO Port B. Meant for an EEPROM data transfer, in an
  *            Device, Add1, Add2, Data. Write format, and a Device, Add1, Add2...
  *            Device, Data format for reading.
  *
  *            PTB-8: SCL
  *            PTB-9: SDA
  *
  ******************************************************************************
  */
#include "i2c.h"

void I2C_Init (void){
//initialize GPIO pins to have AF set to I2C functions
               //Power and Clock
               RCC->AHB2ENR |= (RCC_AHB2ENR_GPIOBEN);   // enable GPIOG clock
               //GPIO Ports - AF8, no PU/PD, open drain, fast
               GPIOB->MODER   &= ~(GPIO_MODER_MODE8 | GPIO_MODER_MODE9);
               GPIOB->MODER   |=  (GPIO_MODER_MODE8_1 | GPIO_MODER_MODE9_1);
               GPIOB->OTYPER  |= (GPIO_OTYPER_OT8 | GPIO_OTYPER_OT9);
               GPIOB->PUPDR   &= ~(GPIO_PUPDR_PUPD8 | GPIO_PUPDR_PUPD9);
               GPIOB->OSPEEDR |=  ((3 << GPIO_OSPEEDR_OSPEED8_Pos) | (3 << GPIO_OSPEEDR_OSPEED9_Pos));
               GPIOB->AFR[1] &= ~((0x000F << GPIO_AFRH_AFSEL8_Pos) | (0x000F << GPIO_AFRH_AFSEL9_Pos));
               GPIOB->AFR[1] |=  ((0x0004 << GPIO_AFRH_AFSEL8_Pos) | (0x0004 << GPIO_AFRH_AFSEL9_Pos));
//configure I2C functions and specific EEPROM data transactions
               // Configure I2C
               RCC->APB1ENR1 |= RCC_APB1ENR1_I2C1EN;  // enable I2C bus clock
               I2C1->CR1   &= ~( I2C_CR1_PE );        // put I2C into reset (release SDA, SCL)
               I2C1->CR1   &= ~( I2C_CR1_ANFOFF );    // filters: enable analog
               I2C1->CR1   &= ~( I2C_CR1_DNF );       // filters: disable digital
               I2C1->TIMINGR = 0x00000509;            // 16 MHz SYSCLK timing from CubeMX
               I2C1->CR2   |=  ( I2C_CR2_AUTOEND );   // auto send STOP after transmission
               I2C1->CR2   &= ~( I2C_CR2_ADD10 );     // 7-bit address mode
               I2C1->CR1   |=  ( I2C_CR1_PE );        // enable I2C
               //set address for EEPROM to be continually used
               }

uint8_t I2C_Read (uint8_t devAddr, uint16_t addr){
               //build start write address[2] mode
               I2C1->CR1      |=  ( I2C_CR1_PE );                         //enable I2C
               I2C1->CR2   &= ~( I2C_CR2_SADD );      // clear device address
               I2C1->CR2   |=  ( devAddr << (I2C_CR2_SADD_Pos+1) ); // device addr SHL 1
               I2C1->CR2   &= ~( I2C_CR2_RD_WRN );    // set WRITE mode
               I2C1->CR2   &= ~( I2C_CR2_NBYTES );    // clear Byte count
               I2C1->CR2   |=  ( 2 << I2C_CR2_NBYTES_Pos); // write 2 bytes (2 addr)
               I2C1->CR2   |=    I2C_CR2_START;
               // send data
               while(!(I2C1->ISR & I2C_ISR_TXIS)) ;          // wait for start condition to transmit
               I2C1->TXDR = (addr >> 8);                                                // xmit MSByte of address
               while(!(I2C1->ISR & I2C_ISR_TXE)) ;           // wait for start condition to transmit
               I2C1->TXDR = (addr & 0xFF);                                              // xmit LSByte of address
               while(!(I2C1->ISR & I2C_ISR_STOPF));          // wait for stop condition to transmit
               I2C1->CR1 &= ~( I2C_CR1_PE );                 //disable I2C
               delay_us(5);                                                                         //wait for 5000ms to cycle power
               I2C1->CR1 |=  ( I2C_CR1_PE );                 //enable I2C
               // build start read data[1] mode
               I2C1->CR2   |= ( I2C_CR2_RD_WRN );    // set READ mode
               I2C1->CR2   &= ~( I2C_CR2_SADD );      // clear device address
               I2C1->CR2   |=  ( devAddr << (I2C_CR2_SADD_Pos+1) ); // device addr SHL 1
               I2C1->CR2   &= ~( I2C_CR2_NBYTES );    // clear Byte count
               I2C1->CR2   |=  ( 1 << I2C_CR2_NBYTES_Pos); // read 1 byte
               I2C1->CR2   |=    I2C_CR2_START;
               // read data
               while(!(I2C1->ISR & I2C_ISR_RXNE)) ;                          // wait for received data to be copied in
               uint8_t data = I2C1->RXDR;
               while(!(I2C1->ISR & I2C_ISR_STOPF));
               I2C1->CR1      &= ~( I2C_CR1_PE );                         //disable I2C
               return data;
}

void I2C_Write (uint8_t devAddr, uint16_t addr, uint8_t data){
               //build start write address[2], data[1] mode
               I2C1->CR1      |=  ( I2C_CR1_PE );                         //enable I2C
                               I2C1->CR2   &= ~( I2C_CR2_SADD );      // clear device address
                               I2C1->CR2   |=  ( devAddr << (I2C_CR2_SADD_Pos+1) ); // device addr SHL 1
               I2C1->CR2   &= ~( I2C_CR2_RD_WRN );           // set WRITE mode
               I2C1->CR2   &= ~( I2C_CR2_NBYTES );           // clear Byte count
               I2C1->CR2   |=  ( 3 << I2C_CR2_NBYTES_Pos); // write 3 bytes (2 addr, 1 data)
               I2C1->CR2   |=    I2C_CR2_START;              //start data transfer
               //send data
               while(!(I2C1->ISR & I2C_ISR_TXIS)) ;          // wait for start condition to transmit
               I2C1->TXDR = (addr >> 8);                                                // xmit MSByte of address
               while(!(I2C1->ISR & I2C_ISR_TXE)) ;           // wait for start condition to transmit
               I2C1->TXDR = (addr & 0xFFFF);                                // xmit LSByte of address
               while(!(I2C1->ISR & I2C_ISR_TXE)) ;                          // wait for Txdata to transmit
               I2C1->TXDR = (data);
               while(!(I2C1->ISR & I2C_ISR_TXE)) ;           // wait for Txdata to transmit
               while(!(I2C1->ISR & I2C_ISR_STOPF));          // wait for stop condition to transmit
               I2C1->CR1      &= ~( I2C_CR1_PE );                         //disable I2C
}
```