   Wyatt Tack

Jonas Thyssen  
   EE 329-01 F'24

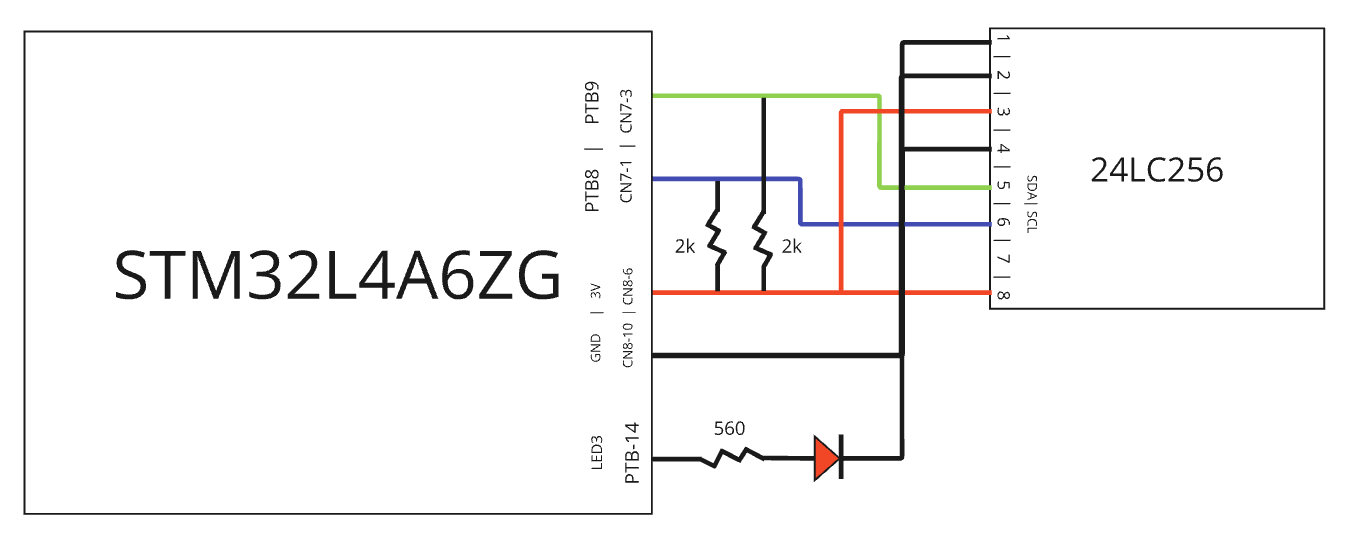
Group D

   2024-Oct-28

**EE 329 A9**

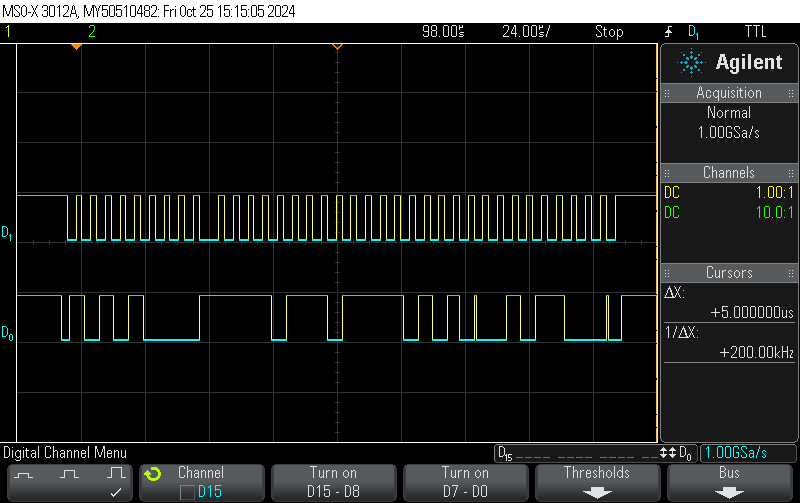
This code is designed to use I2C to communicate to a 256k bit memory. I2C uses a 2 line communication that consists of 2 wires: a data bus and a clock bus. The device works through selecting the peripheral on the data bus (through messaging the peripherals address on the bus), and then writing the data 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



SCL

0 0 1 0 1 1 0 0

Written Data

Memory Address

1 1 1 1 0 1 1 1 | 1 1 1 1 0 1 0 1

ACK

NACK

NACK

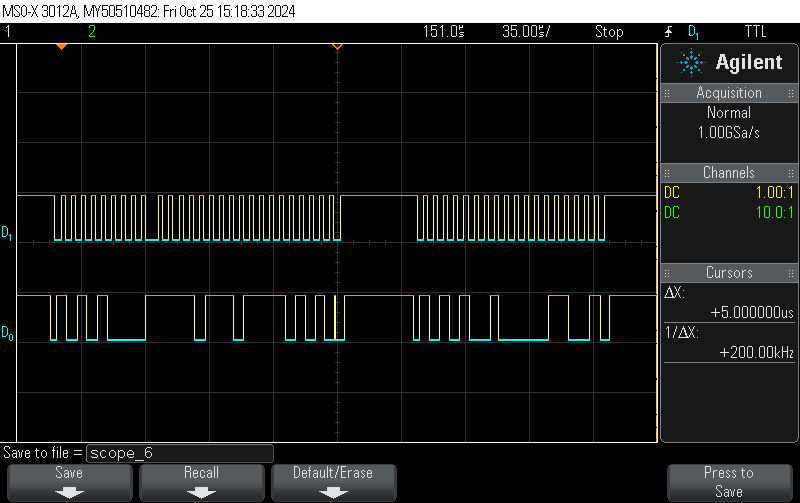
ACK

Device Address

SDA

1 0 1 0 A2 A1A0W

Figure A9.b: Sample Read Communication



Small wait period

Read Data

Device Address

0 0 0 1 1 0 0 1

NACK

ACK

1 0 1 0 A2 A1A0R

Memory Address

1 1 1 1 0 1 1 1 | 1 1 1 1 0 1 0 1

NACK

ACK

Device Address

1 0 1 0 A2 A1A0W

ACK

SCL

SDA

**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:**

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @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

}