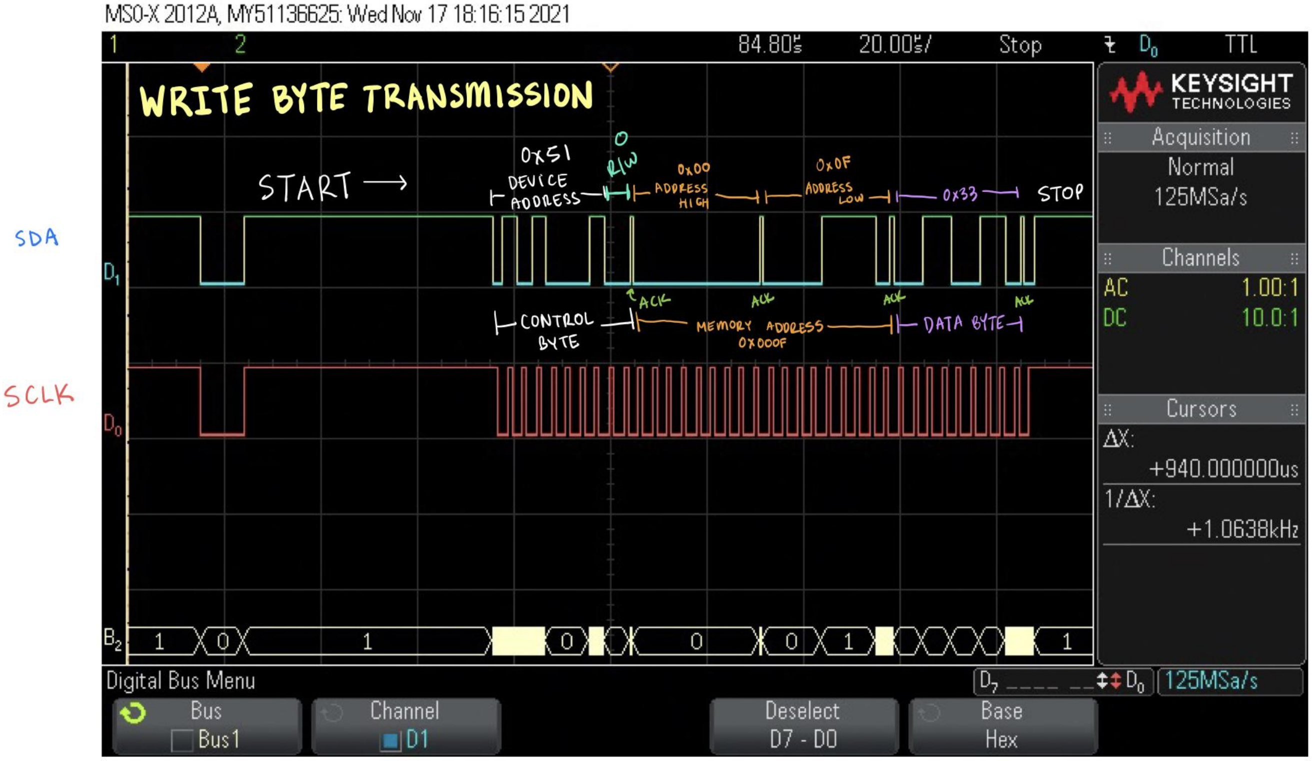
Benchohra, Sereen
Pierce, Haley
CPE 329

A8 - EEPROM/I2C

MSO-X 2012A, MY51136625: Wed Nov 17 18:16:15 2021



Acquisition
Normal
125MSa/s

Channels

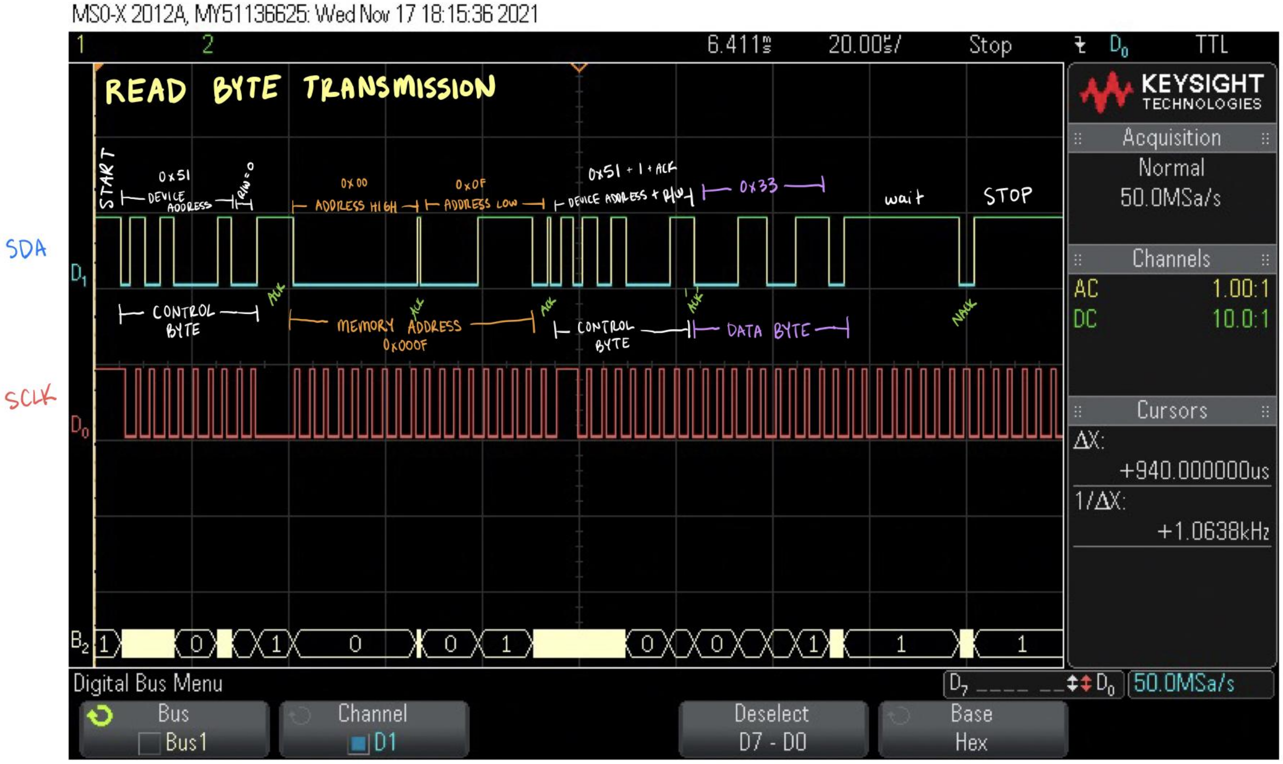
AC	1.00:1
DC	10.0:1

Cursors
ΔX: +940.000000us
1/ΔX: +1.0638kHz

Digital Bus Menu
D7 ____ ___ D0 125MSa/s

Bus: Bus1
Channel: D1
Deselect: D7 - D0
Base: Hex

MSO-X 2012A, MY51136625: Wed Nov 17 18:15:36 2021



Acquisition
Normal
50.0MSa/s

Channels

AC	1.00:1
DC	10.0:1

Cursors
ΔX: +940.000000us
1/ΔX: +1.0638kHz

Digital Bus Menu
D7 ____ ___ D0 50.0MSa/s

Bus: Bus1
Channel: D1
Deselect: D7 - D0
Base: Hex

MAIN.C

```c
#include "main.h"
#include "eeprom.h"
#include "delay.h"

uint16_t RANDOM_ADDRESS   = 0x000F;
uint8_t DATA = 0x33;

void SystemClock_Config(void);
void led_configure(void);
void led_on(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  eeprom_init();

  eeprom_write(RANDOM_ADDRESS, DATA);
  uint8_t data = eeprom_read(RANDOM_ADDRESS);

  if (data == DATA){
         led_on();
  }

  while (1)
  {
  }

}

void led_configure(void){
       // Configure PC0
       // GPIO output, push-pull, no pull-up/down resistors, low speed
       RCC->AHB2ENR       |=     (RCC_AHB2ENR_GPIOCEN);
       GPIOC->MODER       &=     ~(GPIO_MODER_MODE0);
       GPIOC->MODER       |=     (GPIO_MODER_MODE0);
       GPIOC->OTYPER      &=     ~(GPIO_OTYPER_OT0);
       GPIOC->PUPDR       &=     ~(GPIO_PUPDR_PUPD0);
       GPIOC->OSPEEDR     &=     ~(GPIO_OSPEEDR_OSPEED0);

       GPIOC->ODR   &=    ~(GPIO_ODR_OD0);
}

void led_on(void){
       GPIOC->ODR   &=    ~(GPIO_ODR_OD0);
       GPIOC->ODR   |=    (GPIO_ODR_OD0);
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
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();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
```

```
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

EEPROM.C

```
/*
 * eeprom.c
 *
 *  Created on: Nov 14, 2021
 *      Author: Haley
 */
#include "main.h"
#include "eeprom.h"
#include "delay.h"
```

```
void eeprom_init(void){
        // Set PB6 to SCL and PB7 to SDA
        // Alternate function mode, slow speed, open-drain, no pull-up/down res
        RCC->AHB2ENR |=     (RCC_AHB2ENR_GPIOBEN);
        GPIOB->MODER &=     ~(GPIO_MODER_MODE6 | GPIO_MODER_MODE7);
        GPIOB->MODER |=     (GPIO_MODER_MODE6_1 | GPIO_MODER_MODE7_1);
        GPIOB->OTYPER|=     (GPIO_OTYPER_OT6 | GPIO_OTYPER_OT7);
        GPIOB->OSPEEDR      &=     ~(GPIO_OSPEEDR_OSPEED6 | GPIO_OSPEEDR_OSPEED7);
        GPIOB->PUPDR &=     ~(GPIO_PUPDR_PUPD6 | GPIO_PUPDR_PUPD7);
        // choose alternate function 4 for I2C1_SDA and SCL
        GPIOB->AFR[0]|=     (GPIO_AFRL_AFSEL6_2 | GPIO_AFRL_AFSEL7_2);

        //Configure I2C1
        RCC->APB1ENR1|=     (RCC_APB1ENR1_I2C1EN);
        //clear PE bit
        I2C1->CR1           &=     ~(I2C_CR1_PE);
        //analog with no digital filter
        I2C1->CR1           &=     ~(I2C_CR1_ANFOFF);
        I2C1->CR1           &=     ~(I2C_CR1_DNF);
        // set timing to
        I2C1->TIMINGR=      0x00000004;
        // allow NOSTRETCH
        I2C1->CR1           &=     ~(I2C_CR1_NOSTRETCH);
        // set PE bit high
        I2C1->CR1           |=     (I2C_CR1_PE);
        // Enable RXIE and TXIE interrupts
        I2C1->CR1           |=     (I2C_CR1_TXIE | I2C_CR1_RXIE);

}

uint8_t eeprom_read(uint16_t address){
        // turn address from 2 bytes in two single byte nibbles
        uint8_t address_high = (address & 0xF0) >> 4;
        uint8_t address_low = (address & 0x0F);

        //// CONTROL BYTE
        I2C1->CR2           &=     ~(I2C_CR2_AUTOEND);
        I2C1->CR2           &=     ~(I2C_CR2_NBYTES_Pos);
        I2C1->CR2           |=     (2 << I2C_CR2_NBYTES_Pos);
        I2C1->CR2           &=     ~(I2C_CR2_ADD10);
        // Set the control bit with the device address SADD[7:1]
        I2C1->CR2           |=     (DEVICE_ADDRESS << (I2C_CR2_SADD_Pos + 1));
        // Set to write to send memory address
        I2C1->CR2           &=     ~(I2C_CR2_RD_WRN);
        // Send START bit
        I2C1->CR2           |=     (I2C_CR2_START);
        //// SEND MEMORY ADDRESS
        // Wait for TXE to be high to transmit memory address high byte
        while (!(I2C1->ISR & I2C_ISR_TXIS));
        I2C1->TXDR          =     (address_high);
        // Wait for TXE to be high to transmit memory address low byte
        while (!(I2C1->ISR & I2C_ISR_TXIS));
        I2C1->TXDR          =     (address_low);
        // Wait for transfer to complete
        while (!(I2C1->ISR & I2C_ISR_TXIS));
        I2C1->CR2           |=    (I2C_CR2_STOP);

        //// CONFIGURE TO READ DATA
        // Set autoend to stop after one byte of data
        I2C1->CR2           &=    ~(I2C_CR2_NBYTES_Pos);
        I2C1->CR2           |=    (1 << I2C_CR2_NBYTES_Pos);
        I2C1->CR2           |=    (I2C_CR2_AUTOEND);
        // Set the control bit with the device address SADD[7:1]
        I2C1->CR2           |=    (DEVICE_ADDRESS << (I2C_CR2_SADD_Pos + 1));
        // Set to read
        I2C1->CR2           |=    (I2C_CR2_RD_WRN);

        // Send START bit
        I2C1->CR2           |=    (I2C_CR2_START);


        // Wait for RX reg to contain data
        while (!(I2C1->ISR & I2C_ISR_RXNE));
        // Read the data from the EEPROM
        uint8_t data;
        data = (I2C1->RXDR);

        // Wait 5 ms after receiving
        delay_ms(5);

        return data;
}

void eeprom_write(uint16_t address, uint8_t data){
        // turn address from 2 bytes in two single byte nibbles
        uint8_t address_high = (address & 0xF0) >> 8;
        uint8_t address_low = (address & 0x0F);

        I2C1->CR2           &=    ~(I2C_CR2_ADD10);
        // Set to write
        I2C1->CR2           &=    ~(I2C_CR2_RD_WRN);
        // Set autoend for 3 byte of data
        I2C1->CR2           &=    ~(I2C_CR2_NBYTES_Pos);
        I2C1->CR2           |=     (3 << I2C_CR2_NBYTES_Pos);
        I2C1->CR2           |=     (I2C_CR2_AUTOEND);
        // Set the control bit with the device address SADD[7:1]
        I2C1->CR2           |=     (DEVICE_ADDRESS << (I2C_CR2_SADD_Pos + 1));

        // Send START bit
        I2C1->CR2           |=     (I2C_CR2_START);
        // Wait for TXE to be high to transmit memory address high byte
        while (!(I2C1->ISR & I2C_ISR_TXE));
        I2C1->TXDR          =      address_high;
        // Wait for TXE to be high to transmit memory address low byte
        while (!(I2C1->ISR & I2C_ISR_TXE));
        I2C1->TXDR          =      address_low;
        // Wait for TXE to be high to transmit data
        while (!(I2C1->ISR & I2C_ISR_TXE));
        I2C1->TXDR          =      data;
        // wait 5 ms for data to be saved
        delay_ms(5);


}
```

EEPROM.H

```
/*
 * eeprom.h
 *
 *  Created on: Nov 14, 2021
 *      Author: Haley
 */

#ifndef SRC_EEPROM_H_
#define SRC_EEPROM_H_

#define DEVICE_ADDRESS    0x51

void eeprom_init(void);
uint8_t eeprom_read(uint16_t address);
void eeprom_write(uint16_t address, uint8_t data);

#endif /* SRC_EEPROM_H_ */
```

DELAY.C

```
/*
 * delay.c
 *
 *  Created on: Oct 7, 2021
 *      Author: Haley
 */
#include "main.h"
#include "delay.h"

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, breaks HAL delay fcn
}

void delay_us(const uint16_t time_us) {
       // set the counts for the specified delay
       SysTick->LOAD = (uint32_t)((time_us * (SystemCoreClock / 1000000)) - 1);
       SysTick->VAL = 0;                                         // clear the timer
count
       SysTick->CTRL &= ~(SysTick_CTRL_COUNTFLAG_Msk);           // clear the count
flag
       while (!(SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk));    // wait for the
flag to be set
}

void delay_ms(const uint8_t loop_num) {
       uint8_t t = 0;
       while (t <= loop_num) {
              delay_us(1000);
              t++;
       }
       return;
}
```

DELAY.H

```
/*
 * delay.h
 *
 *  Created on: Oct 7, 2021
 *      Author: Haley
 */
```

```
#ifndef SRC_DELAY_H_
#define SRC_DELAY_H_

void SysTick_Init(void);
void delay_us(const uint16_t time_us);
void delay_ms(const uint8_t loop_num);

#endif /* SRC_DELAY_H_ */
```