UCSD Embedded C

Course Number: ECE-40291
Section ID: 142618

Final Assignment

Date: 11/17/2019

Chris Isabelle

christopher.j.isabelle@gmail.com
SID: U01136665

Contents

Main Project

Project Requirements

Content that is *italicized* are deviations or derived requirements.

Requirement ID	Description	Compliance
R1	Tool Requirements	
R1-1	Use STM32CubeMX to generate the initial code for this final assignment.	✓
R1-2	Use TrueStudio to edit/build/debug/run the code. *Request requirements deviation to use STM32CubeIDE.*	✓
R2	Use the UART that is connected to the Arduino connector to display output from this assignment onto PuTTy (or similar) terminal emulator.	✓
R3	Use the Blue Button on the STM board to cycle through different demos.	✓
R3-1	Each time the Blue button is pressed, the title of the demo will be sent to the UART and displayed on PuTTY.	✓
R3-2	Pressing the Blue Button should generate an interrupt and change a "demo count" so that the main looping code changes to the next demo on every button press.	✓
R3-3	On power-up "Demo 1" will auto start.	✓
R3-4	When Blue Button is pressed "Demo 1" ends and "Demo 2" starts.	✓
R3-5	This continues until the last demo is reached, then cycles back to Demo 1.	✓
R4	List of Demos	
R4-1	Demo1: LL APIs	
R4-1.1	Get flash size LL_GetFlashSize().	✓
R4-1.2	Get the device unique ID, LL_GetUID_Wordn().	✓
R4-1.3	Toggle the LED, LL_GPIO_TogglePin() at a 1 second rate.	✓
R4-1.4	Display the Flash Size and GUID only when demo begins.	✓
R4-1.5	Keep flashing the LED every 1 second until the Blue Button is pressed to advance to next demo.	✓
R4-2	Demo2: HAL APIs	
R4-2.1	Get the device ID, HAL_GetDEVID().	✓
R4-2.2	Read the device unique ID, HAL_GetUIDwn().	✓
R4-2.3	Toggle the LED, HAL_GPIO_TogglePin() at a 2 second rate.	✓
R4-2.4	Use HAL_Delay() to sleep for the 2 seconds.	X
R4-2.5	Display the Dev ID info only when demo 2 begins.	✓
R4-2.6	Keep flashing the LED every 2 second until the Blue Button is pressed to advance to next demo.	✓
R4-3	Demo3: BSP APIs	
R4-3.1	Read the temperature, BSP_TSENSOR_ReadTemp().	✓
R4-3.2	Turn the LED on every 3 seconds with BSP_LED_On().	✓
R4-3.3	Turn LED off, every 3 seconds with BSP_LED_Off().	✓
R4-3.4	LED should blink on/off at a 3-second rate (3 seconds on, 3 seconds off).	✓
R4-3.5	*Display temperature each time the LED is turned on, (update rate = 6 seconds).*	✓
R4-4	*Demo4: Develop MB85RS64 FRAM SPI Driver for STM32 HAL*	Separate project
R4-4.1	*Port/rewrite MBED MB85RSxx_SPI SPI driver code (developed by APS LAB) to STM32 HAL SPI.*	Separate project
R4-4.2	*Port FRAM test code developed on MBED for ESHD_L475VG-IOT01 to STM32.*	Separate project
R4-4.3	*Verify SPI messages as needed during unit testing.*	Separate project
R4-4.4	*Run test code and demonstrate PASS and FAIL condition*	Separate project

Code

Main Demo Loop

```
  while (1)
  {
        switch(demo_count & 0x3)
        {
              case(0):
                                printf("\n\n\nDemo 1 is running\n");
                        printf("\n~~~~~~~~~~~~~~~~~\n");
                                uint32_t flash_size = LL_GetFlashSize();
                                printf("flash_size: 0x%lx\n", flash_size);

                                //read and print unique ID
                                uid[0] = LL_GetUID_Word0();
                                uid[1] = LL_GetUID_Word1();
                                uid[2] = LL_GetUID_Word2();
                                printf("uid: 0x%08lx 0x%08lx 0x%08lx\n", uid[0], uid[1], uid[2]);

                                LL_Init1msTick(80000000);

                                while ((demo_count & 0x3) == 0)
                                {
                                      //toggle LED every 1 second
                                      LL_GPIO_TogglePin(GPIOB,  LED2_Pin);
                                      LL_mDelay(1000);
                                      LL_GPIO_TogglePin(GPIOB,  LED2_Pin);
                                      LL_mDelay(1000);
                                }
                                break;
```

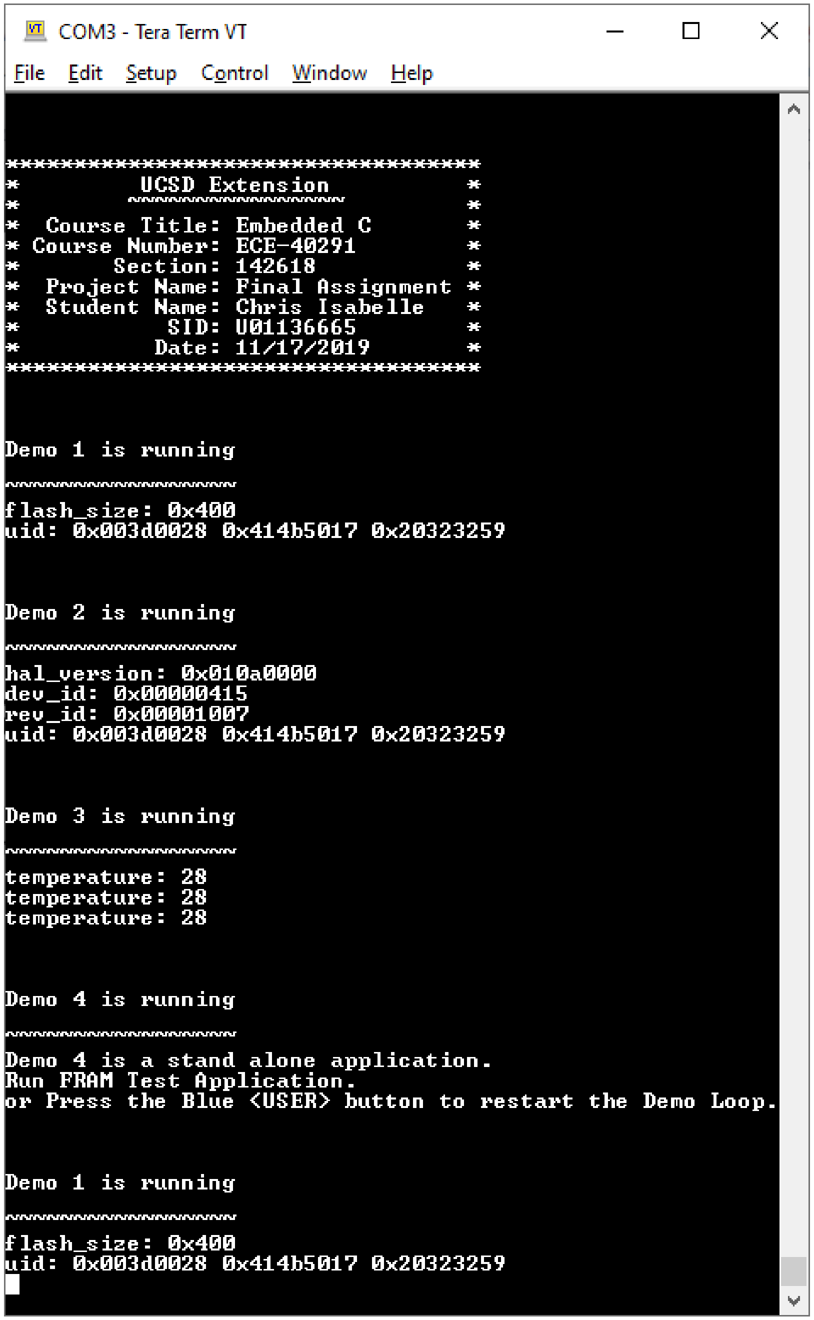
```
case(1):
                  printf("\n\n\nDemo 2 is running\n");
          printf("\n~~~~~~~~~~~~~~~~~~\n");
          //read and print HAL version
          uint32_t version = HAL_GetHalVersion();
          printf("hal_version: 0x%08lx\n", version);

          //read and print device ID
          uint32_t dev_id = HAL_GetDEVID();
          printf("dev_id: 0x%08lx\n", dev_id);

          //read and print revision ID
          uint32_t rev_id = HAL_GetREVID();
          printf("rev_id: 0x%08lx\n", rev_id);

          //read and print unique ID
          uid[0] = HAL_GetUIDw0();
          uid[1] = HAL_GetUIDw1();
          uid[2] = HAL_GetUIDw2();
          printf("uid: 0x%08lx 0x%08lx 0x%08lx\n", uid[0], uid[1], uid[2]);

                  while ((demo_count & 0x3) == 1)
                  {
                        //toggle LED every 2 seconds
                        HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
                        LL_mDelay(2000);
                        HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
                        LL_mDelay(2000);
                  }
                  break;
case(2):
                  printf("\n\n\nDemo 3 is running\n");
               printf("\n~~~~~~~~~~~~~~~~~~\n");
                  while ((demo_count & 0x3) == 2)
                  {
                        //toggle LED every 3 seconds
                        float temperature = BSP_TSENSOR_ReadTemp();
                        printf("temperature: %i\n", (int)temperature);
                        BSP_LED_On(LED_GREEN);
                        LL_mDelay(3000);
                        BSP_LED_Off(LED_GREEN);
                        LL_mDelay(3000);
                  }
                  break;
```

```
      case(3):
                               printf("\n\n\nDemo 4 is running\n");
                           printf("\n~~~~~~~~~~~~~~~~~\n");
                               printf("Demo 4 is a stand alone application.\n");
                               printf("Run FRAM Test Application.\n");
                               printf("or Press the Blue <USER> button to restart the Demo Loop.\n");
                               while ((demo_count & 0x3) == 3);
                           break;
}
```

Interrupt Handler

```
  if (LL_EXTI_IsActiveFlag_0_31(LL_EXTI_LINE_13) != RESET)
  {
    LL_EXTI_ClearFlag_0_31(LL_EXTI_LINE_13);
    /* USER CODE BEGIN LL_EXTI_LINE_13 */
    extern int demo_count;
    ++demo_count;
    /* USER CODE END LL_EXTI_LINE_13 */
  }
```

Screen shot of Final Project running on a serial terminal.



```


***********************************
*         UCSD Extension          *
*        ~~~~~~~~~~~~~~~~         *
*  Course Title: Embedded C       *
* Course Number: ECE-40291        *
*       Section: 142618           *
*  Project Name: Final Assignment *
*  Student Name: Chris Isabelle   *
*           SID: U01136665        *
*          Date: 11/17/2019       *
***********************************



Demo 1 is running

~~~~~~~~~~~~~~~~~
flash_size: 0x400
uid: 0x003d0028 0x414b5017 0x20323259



Demo 2 is running

~~~~~~~~~~~~~~~~~
hal_version: 0x010a0000
dev_id: 0x00000415
rev_id: 0x00001007
uid: 0x003d0028 0x414b5017 0x20323259



Demo 3 is running

~~~~~~~~~~~~~~~~~
temperature: 28
temperature: 28
temperature: 28



Demo 4 is running

~~~~~~~~~~~~~~~~~
Demo 4 is a stand alone application.
Run FRAM Test Application.
or Press the Blue <USER> button to restart the Demo Loop.



Demo 1 is running

~~~~~~~~~~~~~~~~~
flash_size: 0x400
uid: 0x003d0028 0x414b5017 0x20323259
█
```

FRAM Test (Demo 4 – Extra Credit)

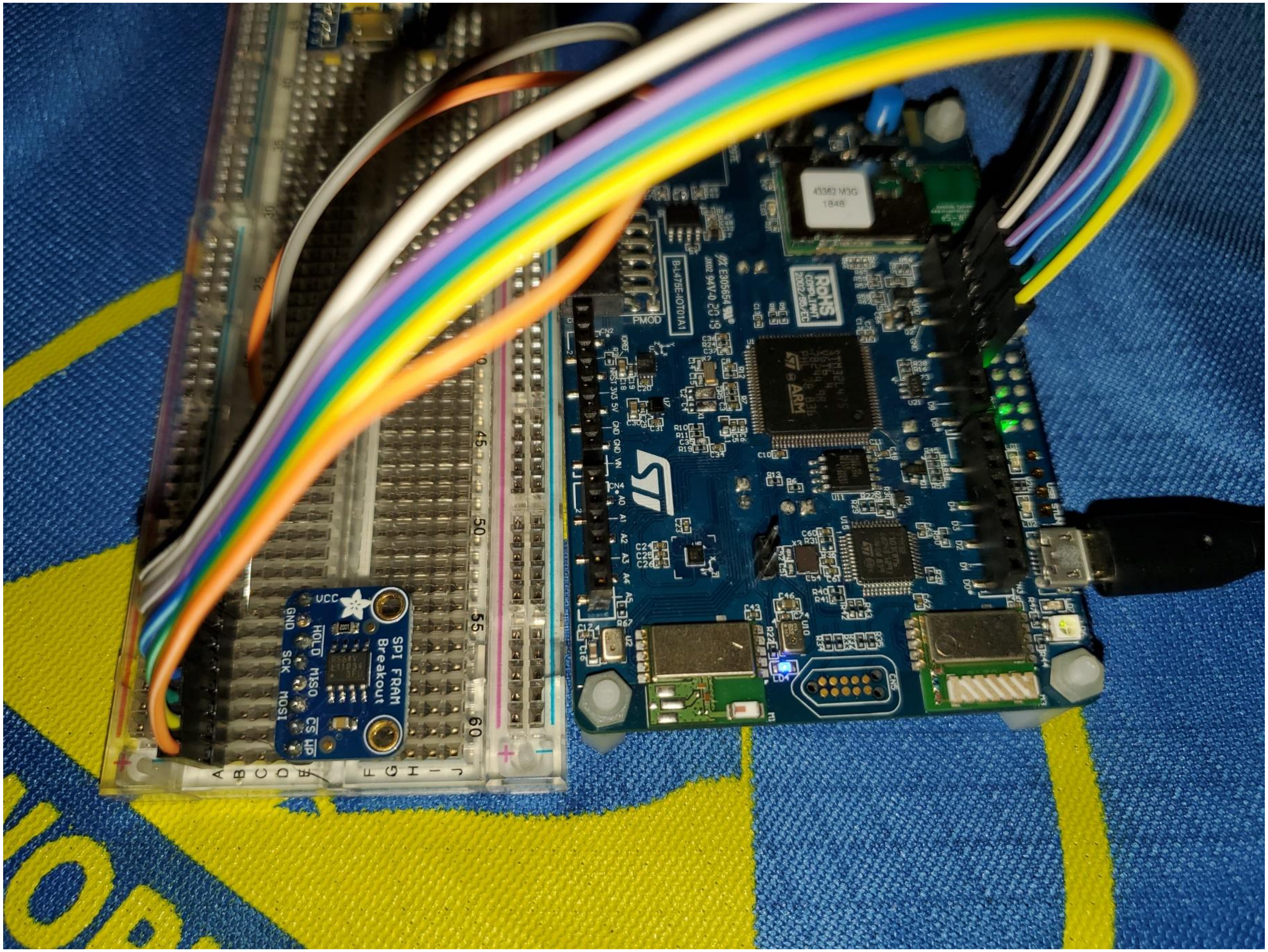
Abstract

- Adapted a Adafruit SPI FRAM Breakout board for use with SPI1 on the STM IOT Discovery board B-L475-IOT01A.
- FRAM device is an MB85RS64V 64K (8Kx8) bit SPI FRAM device
- Test and driver code ported from the MBED project:
  - https://os.mbed.com/users/stillChris/code/ESHD_L475VG_IOT01-Sensors-BSP/
- Enabled SPI1 using STMCubeMX
- Had to change the following from their defaults:
  - `hspi1.Init.DataSize = SPI_DATASIZE_8BIT;`
  - `hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;`
- Arrived at the 32 bit clock prescaler based on observing SCLK on an O'Scope. SCLK is ~2.5MHz. The bread board jumper wires significantly impact signal integrity.
- Driver code is for hardware testing only:
  - Reduced SCK clock rate.
  - FRAM requires a refresh following write for normal operation.
  - Driver not optimized for performance.

I/O planning

ADA Fruit Daughter Card			Functional Description	Design & Implementation Specifics	Arduino Connector Option			
Pin No	Pin Name	Breadboard jumper wire color			CN1 PIN	Arduino PIN	Board Signal	STM32L475 pin
1	nWP	Orange	Write Protect	Not used on this design. Tie to 3.3VDC through a pull-up resister. This sets the active low input, (nWP) high, which disables Write Protection.	N/A	N/A	N/A	N/A
2	nCS	Yellow	Chip Select	This is driver by STM32L475 GPIO bank D, port 5. This active low net is driven low when the SPI2 clock and data are targeted to the FRAM device.	3	D10	SPI1_SSN	PA2
3	MOSI	Green	Serial Data Input	This is serial data output form the STM32L475 to the FRAM slave device.	4	D11	SPI1_MOSI	PA7
4	MISO	Blue	Serial Data Output	This is serial data output form the FRAM slave device to the STM32L475 master.	5	D12	SPI1_MISO	PA6
5	SCK	Violet	Serial Clock	This is a clock output from the STM32L475 to the FRAM.	6	D13	SPI1_SCK	PA5
6	nHOLD	Grey	Hold	Not used on this design. Tie to 3.3VDC through a pull-up resister. This sets the active low input, (nHOLD) high, which disables Hold.	N/A		N/A	N/A
7	GND	White	Ground	Tied to system ground	7	D14	GND	N/A
8	VCC	Black	Supply Voltage	Tied to 3.3VDC	8	D15	3V3	N/A

Breadboard Photo



Code

github repo

https://github.com/stillChris/MB85RS64_FRAM_Test_STM32_HAL.git

Defines

```
/* USER CODE BEGIN PD */
#define FRAM_WREN    0x06
#define FRAM_WRDI    0x04
#define FRAM_RDSR    0x05
#define FRAM_WRSR    0x01
#define FRAM_READ    0x03
#define FRAM_WRITE   0x02
#define FRAM_RDID    0x9f
#define FRAM_SR_WPEN 0x80
#define FRAM_SR_BP0  0x08
#define FRAM_SR_BP1  0x04
#define FRAM_SR_WEL  0x02
#define FRAM_NULL    0x00

#define FRAM_NUM_BYTES (8 * 1024) //8KBytes
#define FRAM_TEST_DATA (((testAddr * 0x51)+0x17)&0xff)
#define FRAM_TEST_ERROR_INSERT 0 //set one bit in the ESHD_FRAM_TEST_ERROR_INSERT byte to intentionally induce write errors
#define FRAM_TEST_BLOCK_SIZE 512
#define FRAM_TEST_BLOCK_MASK ((FRAM_NUM_BYTES/FRAM_TEST_BLOCK_SIZE)-1)
//defines for FRAM Chip Select
#define FRAM_CS_Pin ARD_D10_Pin
#define FRAM_CS_Port ARD_D10_GPIO_Port
//Chip Select is active low.  So Enable drives the GPIO low (Reset)
#define FRAM_CS_ENABLE HAL_GPIO_WritePin(FRAM_CS_Port, FRAM_CS_Pin, 0);
#define FRAM_CS_DISABLE HAL_GPIO_WritePin(FRAM_CS_Port, FRAM_CS_Pin, 1);

/* USER CODE END PD *
```

Driver Code

```c
void FRAM_init(void)
{
      uint8_t spiCMD;

    //make sure FRAM chip select is disabled - Active low so disable drives GPIO high
    FRAM_CS_DISABLE;

      spiCMD = FRAM_RDID;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_WRDI;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_WREN;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_WRSR;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_SR_WEL;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_RDSR;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_NULL;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;
}
```

```c
void FRAM_write(uint16_t address, uint8_t byte)
{
      uint8_t spiCMD;
      uint8_t spiAddrByte;

      spiCMD = FRAM_WREN;
    FRAM_CS_ENABLE;
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    FRAM_CS_DISABLE;

    spiCMD = FRAM_WRITE;
    //enable Chip Select
    FRAM_CS_ENABLE;
    //send WRITE command
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    //send upper 8 bits of address
    spiAddrByte = ((address&0x3f00)>>8);
    HAL_SPI_Transmit(&hspi1, &spiAddrByte, sizeof(spiAddrByte), HAL_MAX_DELAY);
    //send lower 8 bits of address
    spiAddrByte = (address&0x00ff);
    HAL_SPI_Transmit(&hspi1, &spiAddrByte, sizeof(spiAddrByte), HAL_MAX_DELAY);
    //sent data byte
    HAL_SPI_Transmit(&hspi1, &byte, sizeof(byte), HAL_MAX_DELAY);
    //disable Chip Select
    FRAM_CS_DISABLE;
}
```

```
uint8_t FRAM_read(uint16_t address)
{
	uint8_t spiCMD;
	uint8_t spiAddrByte;
	uint8_t byte;

	spiCMD = FRAM_READ;
    //enable Chip Select
    FRAM_CS_ENABLE;
    //send WRITE command
    HAL_SPI_Transmit(&hspi1, &spiCMD, sizeof(spiCMD), HAL_MAX_DELAY);
    //send upper 8 bits of address
    spiAddrByte = ((address&0x3f00)>>8);
    HAL_SPI_Transmit(&hspi1, &spiAddrByte, sizeof(spiAddrByte), HAL_MAX_DELAY);
    //send lower 8 bits of address
    spiAddrByte = (address&0x00ff);
    HAL_SPI_Transmit(&hspi1, &spiAddrByte, sizeof(spiAddrByte), HAL_MAX_DELAY);
    //receive data byte
    HAL_SPI_Receive(&hspi1, &byte, sizeof(byte), HAL_MAX_DELAY);
    //disable Chip Select
    FRAM_CS_DISABLE;

    return(byte);
}
```

Test Code

```
int FRAM_test(uint16_t addrOffset, uint16_t addrRange)
{
    uint8_t testData=0;
    uint16_t testAddr=0;
    int rtnVal = 0;

    for(testAddr=0; testAddr<addrRange; testAddr++)
    {
        testData = FRAM_TEST_DATA | FRAM_TEST_ERROR_INSERT;
        FRAM_write(testAddr, testData);
    }

    for(testAddr=0; testAddr<addrRange; testAddr++)
    {
       testData = FRAM_read(testAddr);
        if(testData != FRAM_TEST_DATA)
        {
            printf(">>>FRAM test failure - memory = 0x%04x, expected value = 0x%02x, data read = 0x%02x\n",
               testAddr, FRAM_TEST_DATA, testData);
            rtnVal = -1;
        }
    }

    return (rtnVal);
}
```

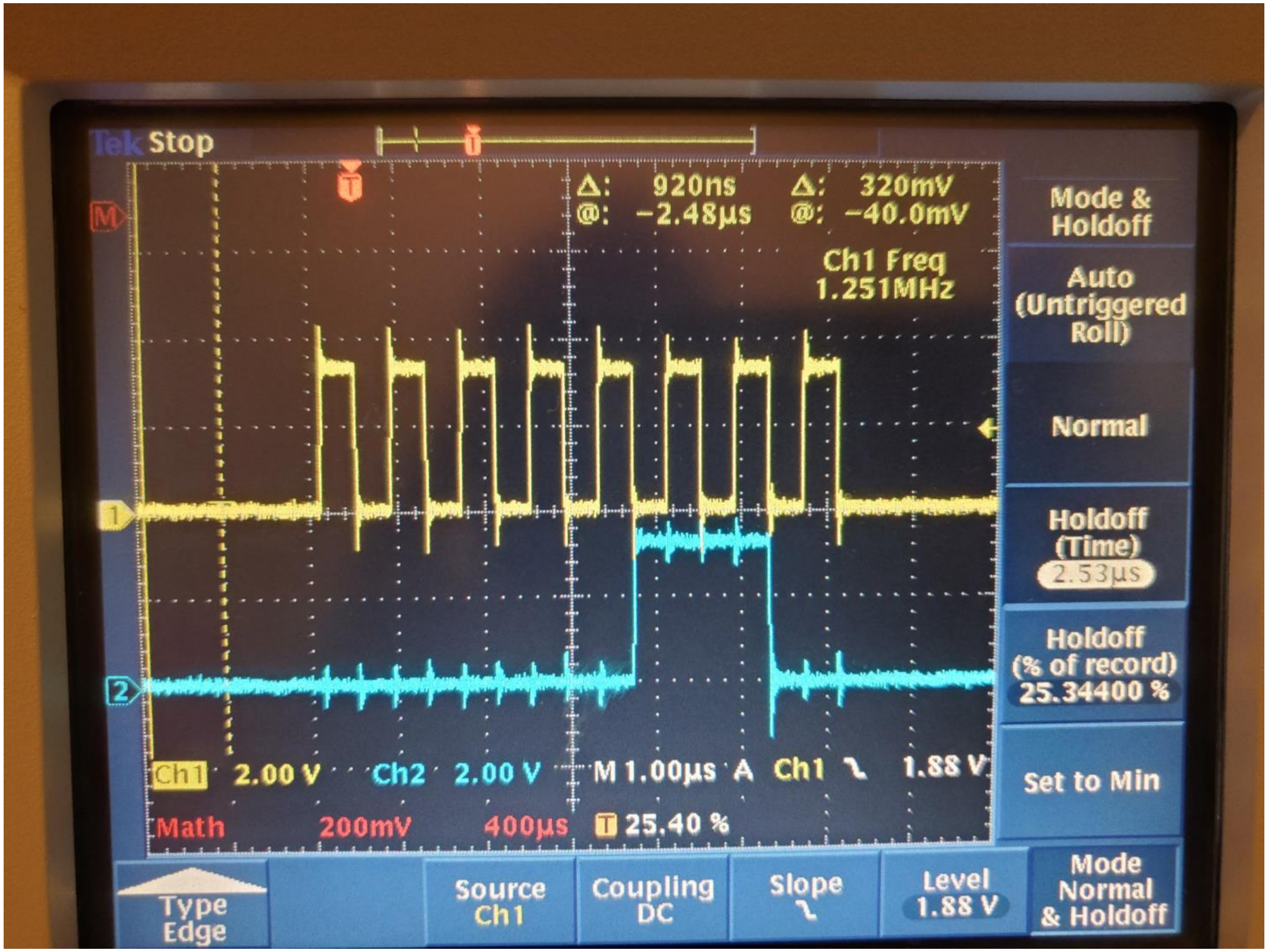
Test Executive

```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
      uint16_t FRAM_testAddr = FRAM_TEST_BLOCK_SIZE * (FRAM_testBlock & FRAM_TEST_BLOCK_MASK);
      printf("FRAM_test: addr=0x%08x, range=0x%08x\n", FRAM_testAddr, FRAM_TEST_BLOCK_SIZE);
      FRAM_test(FRAM_testAddr, FRAM_TEST_BLOCK_SIZE);
      ++FRAM_testBlock;
  }
  /* USER CODE END 3 */
```

Screen shots

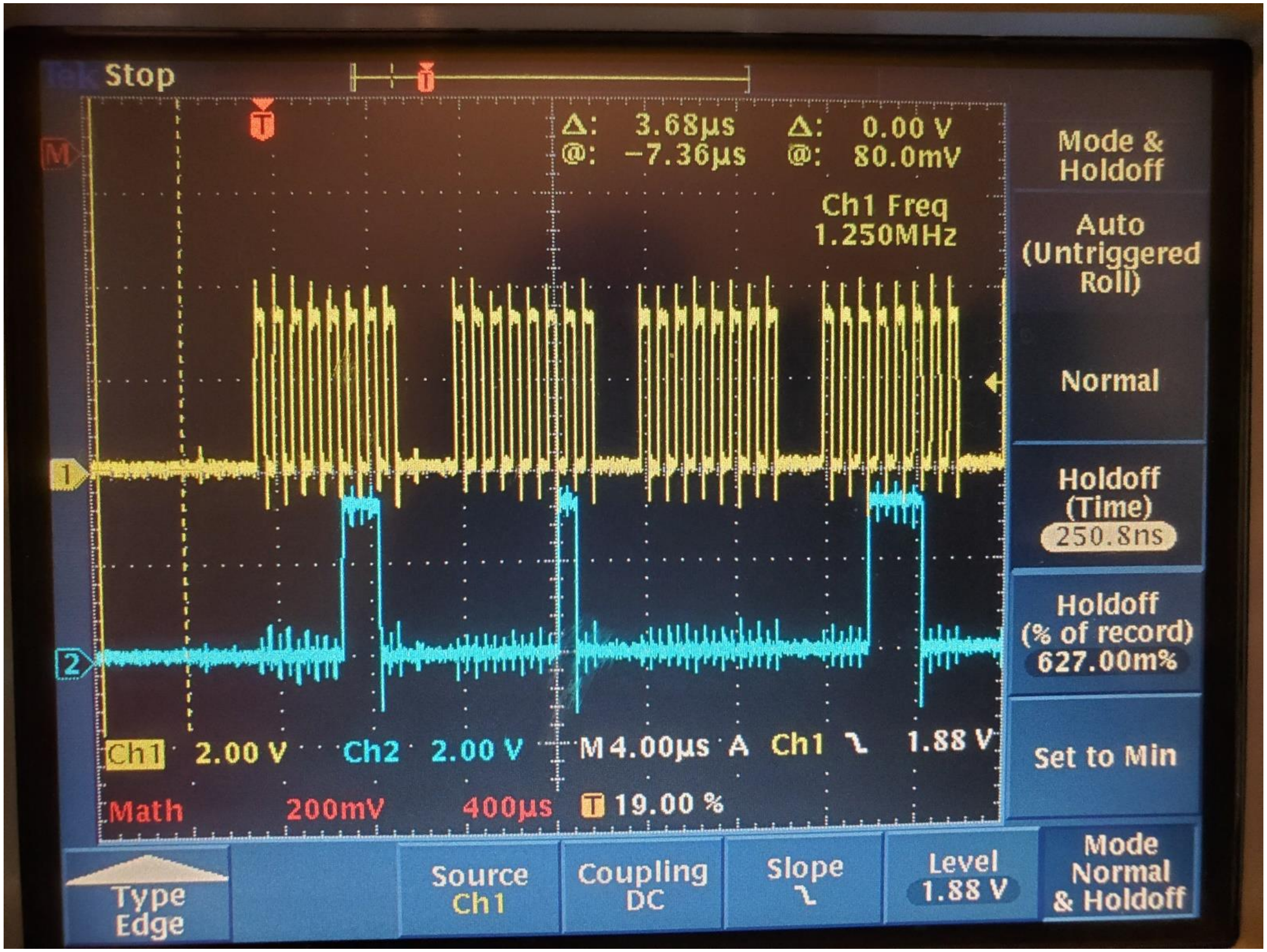
SPI Command Screen shot

To confirm proper SPI Polarity and Phase



SPI Write Sequence

To confirm proper multi SPI sequence. Chip Select timing evaluated seperately



Stop

Δ: 3.68µs
@: −7.36µs

Δ: 0.00 V
@: 80.0mV

Ch1 Freq
1.250MHz

Ch1 2.00 V Ch2 2.00 V M 4.00µs A Ch1 ⤵ 1.88 V

Math 200mV 400µs T 19.00 %

Mode & Holdoff

Auto (Untriggered Roll)

Normal

Holdoff (Time) 250.8ns

Holdoff (% of record) 627.00m%

Set to Min

Type Edge

Source Ch1

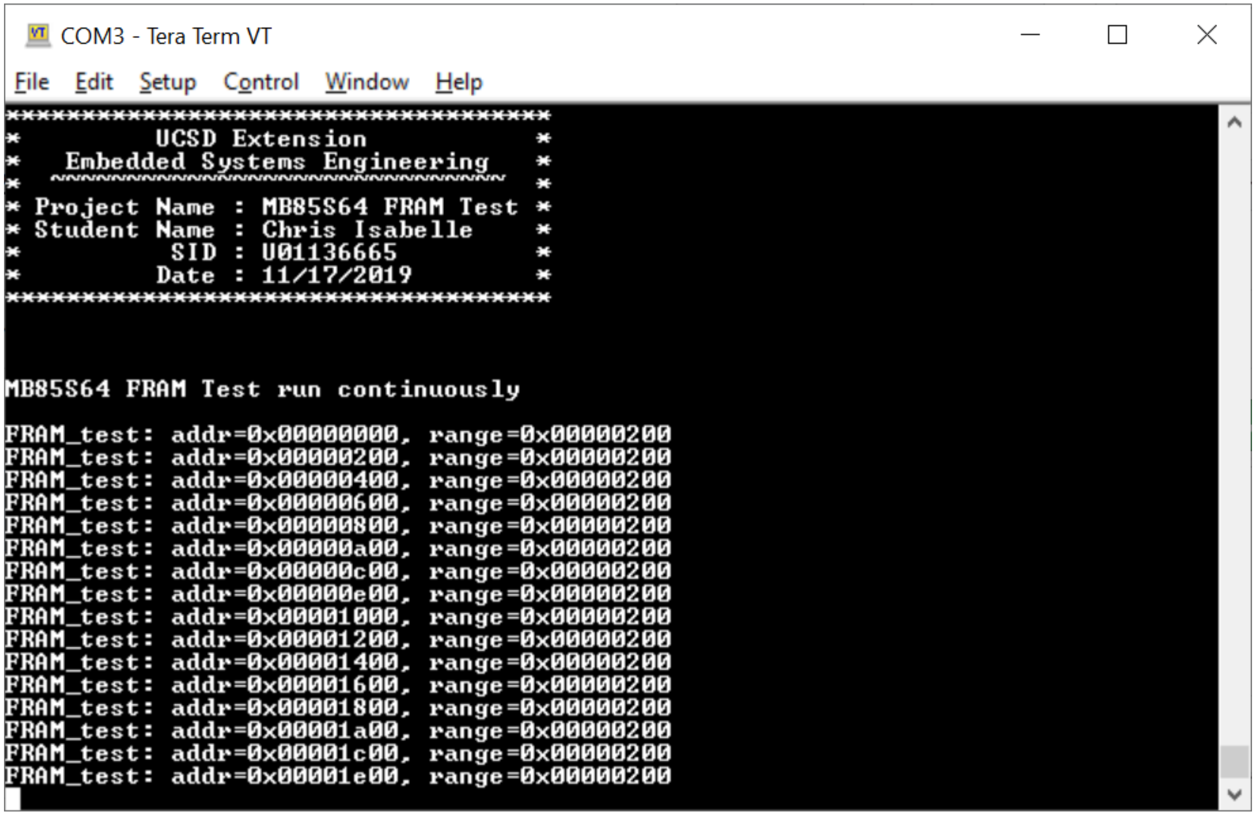
Coupling DC

Slope ⤵

Level 1.88 V

Mode Normal & Holdoff

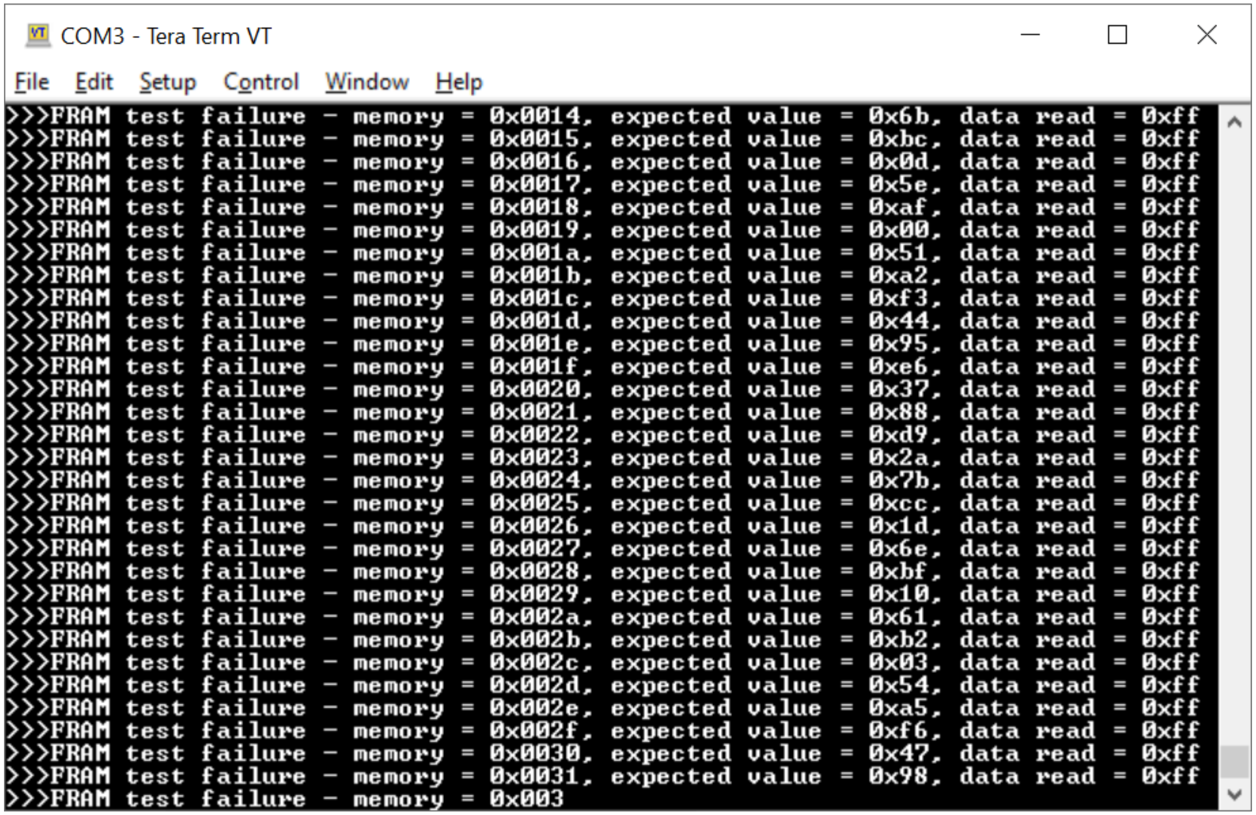
Passing Test



```
************************************
*         UCSD Extension           *
*   Embedded Systems Engineering   *
*  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  *
* Project Name : MB85S64 FRAM Test *
* Student Name : Chris Isabelle    *
*          SID : U01136665         *
*         Date : 11/17/2019        *
************************************



MB85S64 FRAM Test run continuously

FRAM_test: addr=0x00000000, range=0x00000200
FRAM_test: addr=0x00000200, range=0x00000200
FRAM_test: addr=0x00000400, range=0x00000200
FRAM_test: addr=0x00000600, range=0x00000200
FRAM_test: addr=0x00000800, range=0x00000200
FRAM_test: addr=0x00000a00, range=0x00000200
FRAM_test: addr=0x00000c00, range=0x00000200
FRAM_test: addr=0x00000e00, range=0x00000200
FRAM_test: addr=0x00001000, range=0x00000200
FRAM_test: addr=0x00001200, range=0x00000200
FRAM_test: addr=0x00001400, range=0x00000200
FRAM_test: addr=0x00001600, range=0x00000200
FRAM_test: addr=0x00001800, range=0x00000200
FRAM_test: addr=0x00001a00, range=0x00000200
FRAM_test: addr=0x00001c00, range=0x00000200
FRAM_test: addr=0x00001e00, range=0x00000200
```

Failing Test

Pulled MISO pin on breadboard



```
>>>FRAM test failure - memory = 0x0014, expected value = 0x6b, data read = 0xff
>>>FRAM test failure - memory = 0x0015, expected value = 0xbc, data read = 0xff
>>>FRAM test failure - memory = 0x0016, expected value = 0x0d, data read = 0xff
>>>FRAM test failure - memory = 0x0017, expected value = 0x5e, data read = 0xff
>>>FRAM test failure - memory = 0x0018, expected value = 0xaf, data read = 0xff
>>>FRAM test failure - memory = 0x0019, expected value = 0x00, data read = 0xff
>>>FRAM test failure - memory = 0x001a, expected value = 0x51, data read = 0xff
>>>FRAM test failure - memory = 0x001b, expected value = 0xa2, data read = 0xff
>>>FRAM test failure - memory = 0x001c, expected value = 0xf3, data read = 0xff
>>>FRAM test failure - memory = 0x001d, expected value = 0x44, data read = 0xff
>>>FRAM test failure - memory = 0x001e, expected value = 0x95, data read = 0xff
>>>FRAM test failure - memory = 0x001f, expected value = 0xe6, data read = 0xff
>>>FRAM test failure - memory = 0x0020, expected value = 0x37, data read = 0xff
>>>FRAM test failure - memory = 0x0021, expected value = 0x88, data read = 0xff
>>>FRAM test failure - memory = 0x0022, expected value = 0xd9, data read = 0xff
>>>FRAM test failure - memory = 0x0023, expected value = 0x2a, data read = 0xff
>>>FRAM test failure - memory = 0x0024, expected value = 0x7b, data read = 0xff
>>>FRAM test failure - memory = 0x0025, expected value = 0xcc, data read = 0xff
>>>FRAM test failure - memory = 0x0026, expected value = 0x1d, data read = 0xff
>>>FRAM test failure - memory = 0x0027, expected value = 0x6e, data read = 0xff
>>>FRAM test failure - memory = 0x0028, expected value = 0xbf, data read = 0xff
>>>FRAM test failure - memory = 0x0029, expected value = 0x10, data read = 0xff
>>>FRAM test failure - memory = 0x002a, expected value = 0x61, data read = 0xff
>>>FRAM test failure - memory = 0x002b, expected value = 0xb2, data read = 0xff
>>>FRAM test failure - memory = 0x002c, expected value = 0x03, data read = 0xff
>>>FRAM test failure - memory = 0x002d, expected value = 0x54, data read = 0xff
>>>FRAM test failure - memory = 0x002e, expected value = 0xa5, data read = 0xff
>>>FRAM test failure - memory = 0x002f, expected value = 0xf6, data read = 0xff
>>>FRAM test failure - memory = 0x0030, expected value = 0x47, data read = 0xff
>>>FRAM test failure - memory = 0x0031, expected value = 0x98, data read = 0xff
>>>FRAM test failure - memory = 0x003
```