

MB85RS64  
FRAM Test  
Using STM32L4xx\_HAL\_Driver  
HAL\_SPI

Chris Isabelle

[christopher.j.isabelle@gmail.com](mailto:christopher.j.isabelle@gmail.com)

SID: U01136665

## Contents

|                                    |    |
|------------------------------------|----|
| Abstract .....                     | 3  |
| I/O planning.....                  | 4  |
| Breadboard Photo.....              | 5  |
| Code .....                         | 6  |
| github repo .....                  | 6  |
| Defines.....                       | 6  |
| SPI Initialization Code.....       | 7  |
| Driver Code.....                   | 8  |
| Test Code .....                    | 11 |
| Test Executive .....               | 12 |
| O'Scope screen shots .....         | 13 |
| SPI Command Screen shot.....       | 13 |
| SPI Write Sequence .....           | 14 |
| UART Terminal Screen Captures..... | 15 |
| Passing Test .....                 | 15 |
| Failing Test.....                  | 16 |

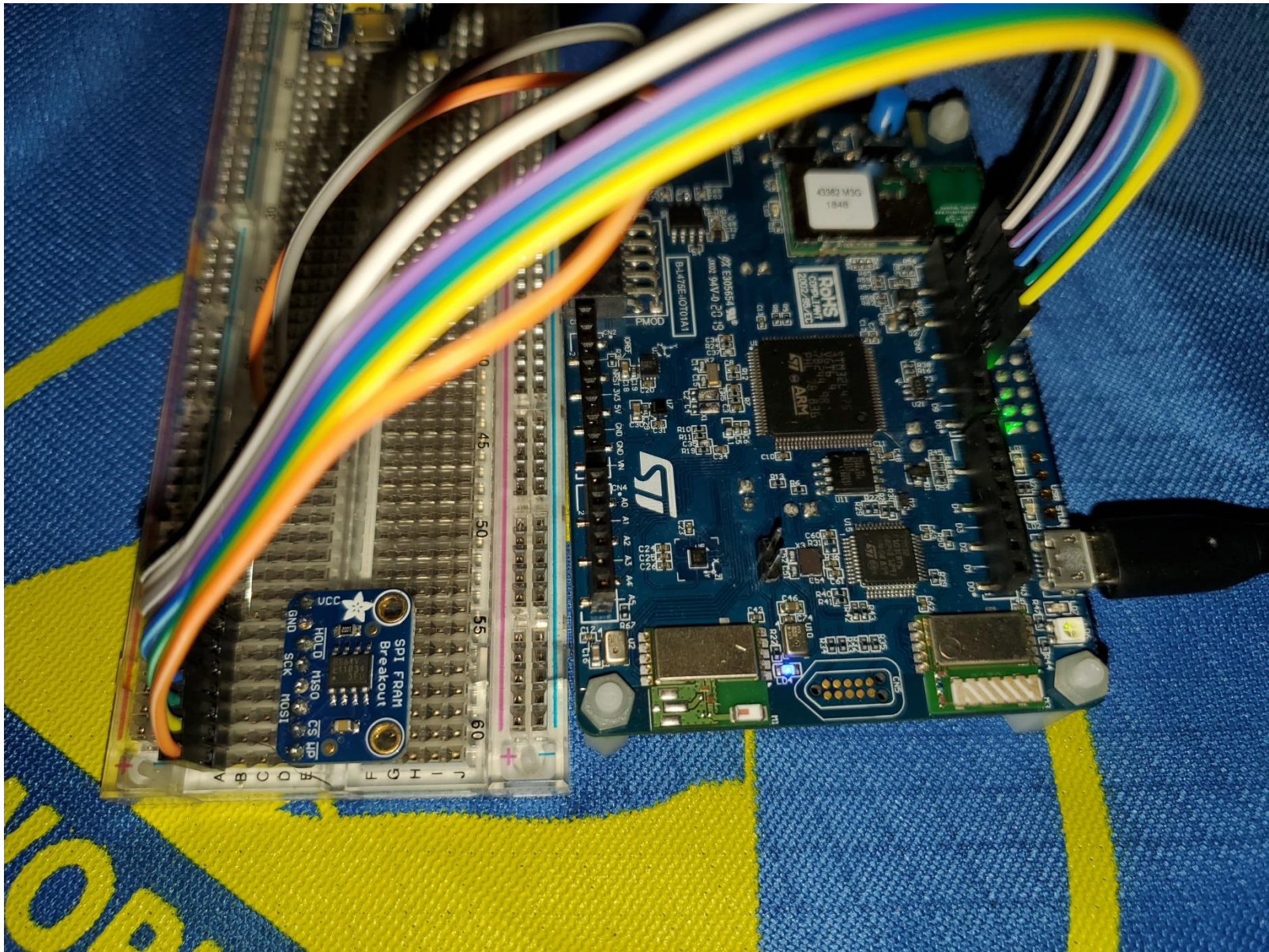
## Abstract

- 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/](https://os.mbed.com/users/stillChris/code/ESHD_L475VG_IOT01-Sensors-BSP/)
- Enabled SPI1 using STMCubeMX
- Changed the following from their defaults:
  - `hspi1.Init.DataSize = SPI_DATASIZE_8BIT;`
  - `hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;`
- Selected the 16 bit clock prescaler based on observing SCLK on an O'Scope. SCLK is ~5.0MHz.
- 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 resistor. 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 from the STM32L475 to the FRAM slave device.  | 4                        | D11         | SPI1_MOSI    | PA7           |
| 4                       | MISO     | Blue                         | Serial Data Output     | This is serial data output from 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 resistor. 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](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 */
```

## SPI Initialization Code

```
static void MX_SPI1_Init(void)
{
    /* USER CODE BEGIN SPI1_Init 0 */

    /* USER CODE END SPI1_Init 0 */

    /* USER CODE BEGIN SPI1_Init 1 */

    /* USER CODE END SPI1_Init 1 */
    /* SPI1 parameter configuration*/
    hspi1.Instance = SPI1;
    hspi1.Init.Mode = SPI_MODE_MASTER;
    hspi1.Init.Direction = SPI_DIRECTION_2LINES;
    hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
    hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
    hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
    hspi1.Init.NSS = SPI_NSS_SOFT;
    hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
    hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
    hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
    hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    hspi1.Init.CRCPolynomial = 7;
    hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
    hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
    if (HAL_SPI_Init(&hspi1) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN SPI1_Init 2 */

    /* USER CODE END SPI1_Init 2 */

}
```

## Driver Code

```
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;
}
```



```

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 */
```

## O'Scope screen shots

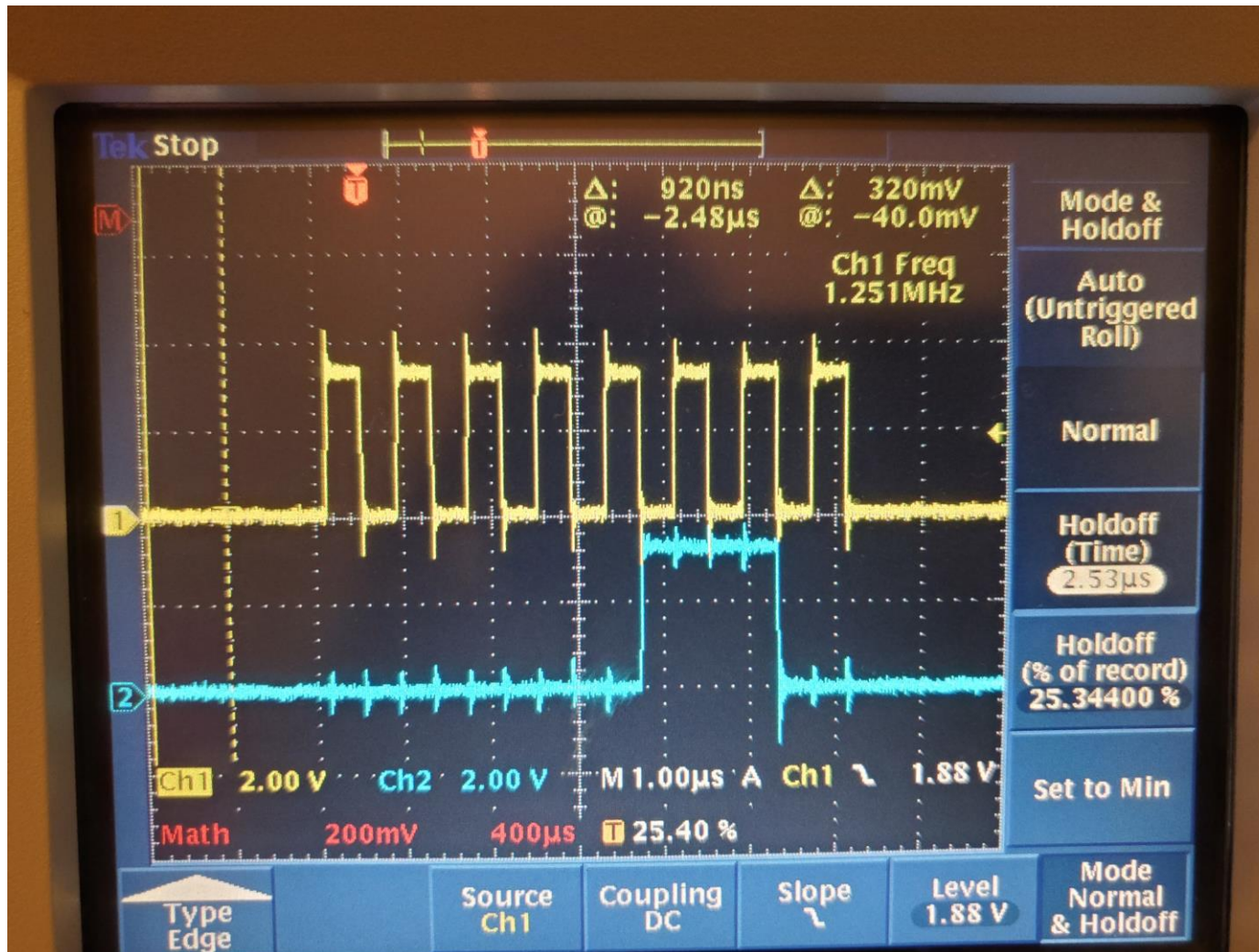
Breadboard jumpers induce significant noise with SCK = 5MHz. O'Scope screen shots taken with:

```
huart1.Init.OverSampling = UART_OVERSAMPLING_64;
```

for an SCK = 5MHz.

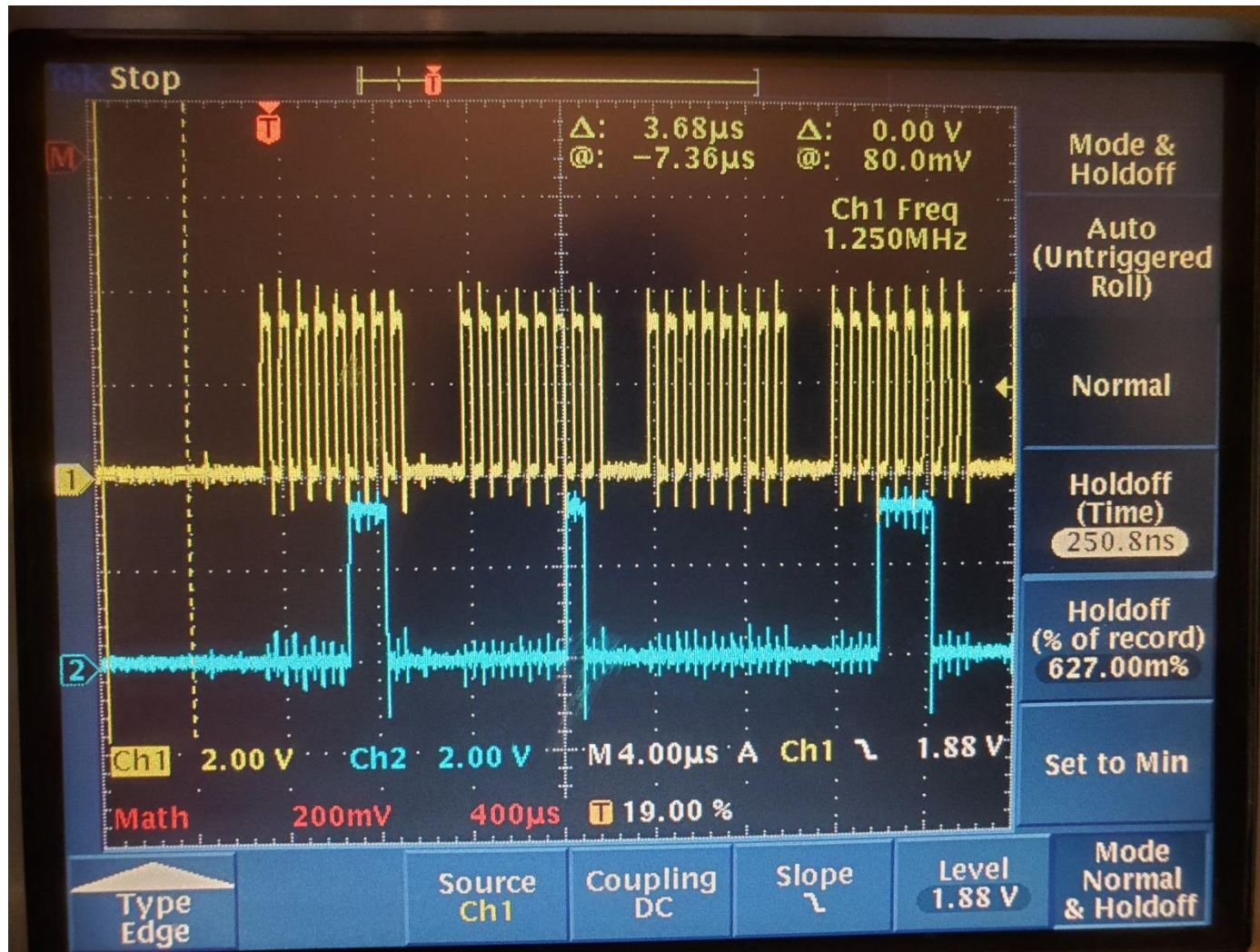
## SPI Command Screen shot

To confirm proper SPI Polarity and Phase



## SPI Write Sequence

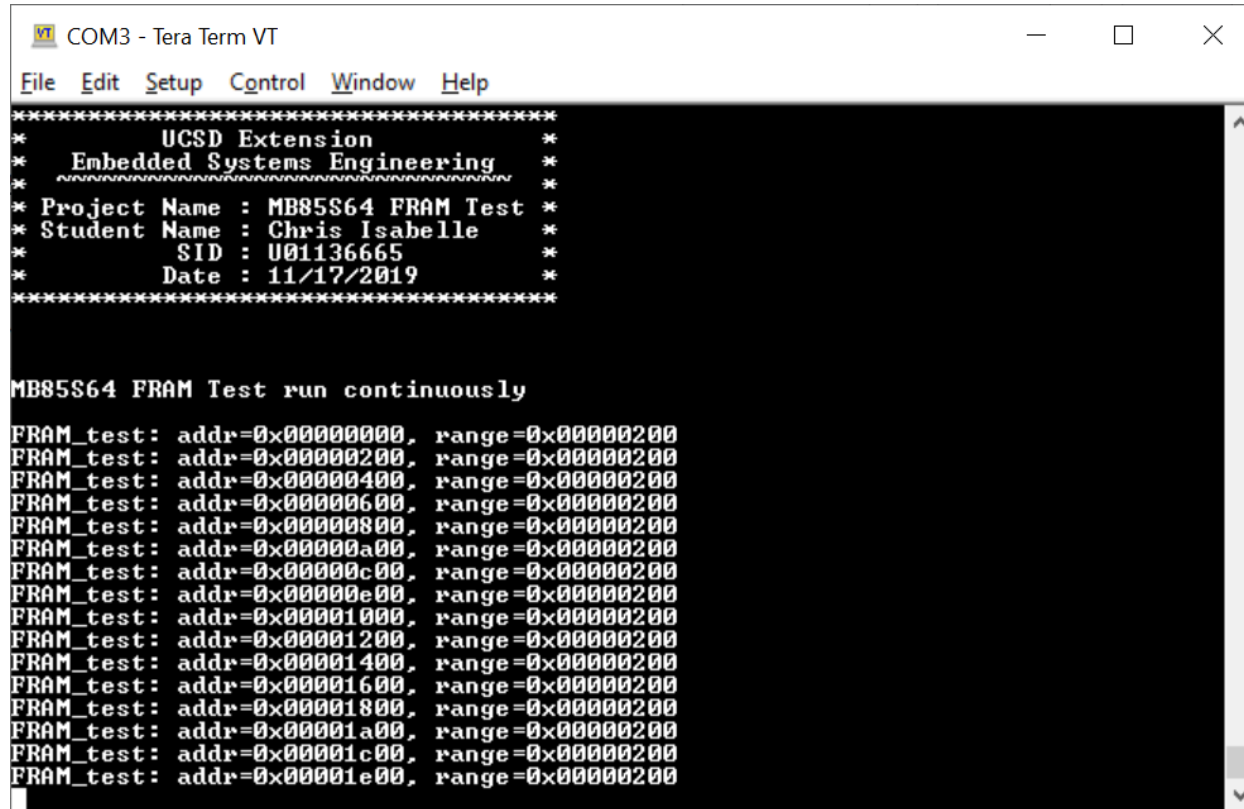
To confirm proper multi SPI sequence. Chip Select timing evaluated seperately





## UART Terminal Screen Captures

### Passing Test



```
COM3 - Tera Term VT
File Edit Setup Control Window Help
*****
* 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

```
COM3 - Tera Term VT
File Edit Setup Control Window Help
>>>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
```