

main.c

```
/* =====
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * =====
 */
#include <project.h>

void WREN();
void transmit_address(uint8_t hi, uint8_t med, uint8_t low);

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */

    SPIM_1_Start();
    LCD_Start();
    LCD_DisplayOn();

    uint8_t addr_hi = 0x00;
    uint8_t addr_med = 0x00;
    uint8_t addr_low = 0x00;
    //our starting address is going to be at the bottom of memory

    //first, lets disable write protection
    SS_Write(0); //SS must be 0

    low to select the chip
    SPIM_1_WriteTxData(0x06); //0x06 is the command
    code for the Write enable command
    while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));
    //Due to the way SPI works on the PSoC, we'll
    wait for the SPI Tx buffer to be sent
    SS_Write(1); //bring SS high
    back high to latch the command

    SS_Write(0);
    SPIM_1_WriteTxData(0x01); //this is the command
    write status register (WSRS) command
    SPIM_1_WriteTxData(0x00); //writing it
    to 0x00 disables all write protection
    while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));

    uint8_t num_strings = 0;
}
```

```

main.c

//We'll use this to keep track of which strings we've written to the chip

char * strings[] = {"6.115", " is", " cool!!"};
//These are the example strings we'll be writing to the chip

char read_string[16] = {0};
//This will hold the string we read back from the chip

for(;;)
{
    if(!SW2_Read()){
        //when we press SW2, we will write another string to the flash ↗
memory
        if (num_strings < 3){
            //write each string to the flash chip
            //if we have already written all three parts, do nothing.
            uint8_t length = strlen(strings[num_strings]);

            WREN();

            SS_Write(0);
            SPIM_1_WriteTxData(0x02); //this is the ↗
write command
            SPIM_1_WriteTxData(addr_hi); //we then ↗
transmit our 24-bit address
            SPIM_1_WriteTxData(addr_med); //from now on, we'↗
ll use transmit_address() for brevity
            SPIM_1_WriteTxData(addr_low);

            SPIM_1_PutArray((uint8_t *)strings[num_strings], length+1);
            //transmit the data we want to write
            //remember, our string is null-terminated, which also ↗
needs to be transmitted.
            while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));
            //now we have to wait for it all to actually be sent by ↗
the psoc

            SS_Write(1);

            addr_low += length;
            //increment our address counter by however many bytes we wrote↗
, not including the null byte.
            //this way, next time we write, we'll overwrite the null byte ↗
with a space to continue the string.
            num_strings++;
        }
        CyDelay(250);
    }
    else if(!SW3_Read()){ //when we press SW3, we will read from flash ↗
memory and display on the LCD
        SS_Write(0);
        SPIM_1_WriteTxData(0x03); //0x03 is the ↗
command to read

```

```

                                main.c
    transmit_address(0x00, 0x00, 0x00);           //start the read ↗
back at the base of the flash chip
    while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));
                                //we need to wait for the address to finish ↗
transmitting before clearing the buffer
    SPIM_1_ClearRxBuffer();                       //get junk data ↗
out of the buffer

    int i;
    for (i = 0; i < 16; i ++){                   //sequentially ↗
read the data from the flash chip
        SPIM_1_WriteTxData(0xAA);
                                //the chip only outputs while we're ↗
writing to it, so write dummy data
        while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));
                                //make sure each dummy ↗
byte gets transmitted before we read
        read_string[i] = SPIM_1_ReadRxData();    //put each ↗
received byte into the string buffer
        if (read_string[i] == 0) break;         //if we see our null ↗
termination, break the loop
    }
    SS_Write(1);

    LCD_ClearDisplay();
    LCD_PrintString(read_string);
    //Display the string on the LCD.
    CyDelay(250);
}

/* Place your application code here. */
}

void WREN (){
    SS_Write(0);
    SPIM_1_WriteTxData(0x06);
    while(!(SPIM_1_ReadTxStatus() & SPIM_1_STS_SPI_DONE));
    SS_Write(1);
}

void transmit_address (uint8_t hi, uint8_t med, uint8_t low){
    SPIM_1_WriteTxData(hi);
    SPIM_1_WriteTxData(med);
    SPIM_1_WriteTxData(low);
}

/* [] END OF FILE */

```