```
/*****************************************************************************
 Module
   ShiftRegisterWrite.c

 Revision
   1.0.1

 Description
   This module acts as the low level interface to a write only shift register.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 10/11/15 19:55 jec     first pass

*****************************************************************************/
// the common headers for C99 types
#include <stdint.h>
#include <stdbool.h>

// the headers to access the GPIO subsystem
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"

// the headers to access the TivaWare Library
#include "driverlib/sysctl.h"
#include "driverlib/pin map.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"

#include "BITDEFS.H"
#include "ES_Port.h"

// readability defines
#define DATA GPIO_PIN_0

#define SCLK GPIO_PIN_1
#define SCLK_HI BIT1HI
#define SCLK_LO BIT1LO

#define RCLK GPIO_PIN_2
#define RCLK_LO BIT2LO
#define RCLK_HI BIT2HI

#define GET_MSB_IN_LSB(x) ((x & 0x80)>>7)

// an image of the last 8 bits written to the shift register
static uint8_t LocalRegisterImage=0;

// Create your own function header comment
void SR_Init(void){

  // set up port B by enabling the peripheral clock, waiting for the
  // peripheral to be ready and setting the direction
  // of PB0, PB1 & PB2 to output

      // Initialize Port B
      HWREG(SYSCTL_RCGCGPIO) |= BIT1HI;
```

```
      // Wait for clock to get ready
      while ((HWREG(SYSCTL_PRGPIO) & BIT1HI) != BIT1HI)
      {
      }

      // Set data direction and assign digital port on Port B
      HWREG(GPIO_PORTB_BASE+GPIO_O_DEN) |= (BIT0HI | BIT1HI | BIT2HI);
      HWREG(GPIO_PORTB_BASE+GPIO_O_DIR) |= (BIT0HI | BIT1HI | BIT2HI);

  // start with the data & sclk lines low and the RCLK line high
      HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) &= (BIT0LO & BIT1LO);
      HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) |= BIT2HI;
}

// Create your own function header comment
uint8_t SR_GetCurrentRegister(void){
  return LocalRegisterImage;
}

// Create your own function header comment
void SR_Write(uint16_t NewValue){

  uint8_t BitCounter;
  LocalRegisterImage = NewValue; // save a local copy

      // lower the register clock
      HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) &= BIT2LO;

      // shift out the data while pulsing the serial clock
      for (BitCounter = 0; BitCounter < 16; BitCounter++) {

            // Isolate the MSB of NewValue
            if (NewValue & BIT15HI) {
                  // Bit 0 high if Bit 7 is high
                  HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) |= BIT0HI;
            } else {
                  // Set Bit 0 low if Bit 7 is low
                  HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) &= BIT0LO;
            }

            // lower SCLK
            HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) &= BIT1LO;

            // raise SCLK
            HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) |= BIT1HI;

            // Left shift NewValue by 1
            NewValue = NewValue << 1;

            // finish looping through bits in NewValue
      }

      // raise the register clock to latch the new data
      HWREG(GPIO_PORTB_BASE+(GPIO_O_DATA + ALL_BITS)) |= BIT2HI;

}
```