```c
/*
 * MAX5402_verify.c
 *
 * Created: 3/14/2022 9:53:50 PM
 * Author : jason
 */


#include <avr/io.h>
#define F_CPU 4000000
#include <util/delay.h>

//All the header funnctions
void MAX5402_SPI0_init(void);
void MAX5402_SPI0_write(uint8_t);




int main(void)
{
    uint8_t i = -1;
    MAX5402_SPI0_init();  //Initialize the SPI0 module
    while (1)
    {
            if(i < 256){
                MAX5402_SPI0_write(i++);      //Write the data to the SPI of the
                  incremented value

            }
            else{
                //Reset the value to 0 if trying to exceed the value of 255
                i = 0;
                MAX5402_SPI0_write(i);
            }
        //  _delay_ms(1);
        //}
    }
}


//initializes the AVR128DB48's SPI0 module to communicate with the
//MAX5402. SPI0 must be configured in the Normal mode (no buffering).
//Thus, this function must check that the previous transfer is complete
//after writing each byte of data and then deselect the MAX5402 before
//returning. Note that there are two variations of the INTFLAGS register.
// One for use in Normal mode and the other for use in Buffered mode
void MAX5402_SPI0_init(void){

    PORTF_DIR |= PIN2_bm;   //Chose PF2 as the output to drive the /SS input
    PORTA_DIR |= PIN6_bm | PIN4_bm ;          //Configure PA6 and PA4 output where the
       pin level is controlled by the SPI
```

```
    PORTA_DIR &= ~PIN5_bm;   //Configure PA5 as the input

     //Enable the SPI and also Select the SPI master/slave operation by writing the
       Master/Slave Select
     //(MASTER) bit in the Control A (SPIn.CTRLA) register
    SPI0.CTRLA |= SPI_ENABLE_bm | SPI_MASTER_bm | SPI_PRESC_DIV4_gc;

    SPI0_CTRLB |= SPI_SSD_bm;   //Disable the Slave select since we don't need the
      actual /SS


    PORTF_OUT |= PIN2_bm;   //Initialize the /SS to be 1 meaning no data to be sent
      yet


}




//the value to be send to the MAX5402 to set the position of the its wiper.
//It is important that before this function returns, it must deselect the
//MAX5402. If this is not done, there could be a subsequent SPI bus
//conflict between the MAX5402 and other (future) SPI devices added to
//the system.

void MAX5402_SPI0_write(uint8_t data){

    PORTF_OUT &= ~PIN2_bm;   //Set to be 0 when transmission is happening


    SPI0_DATA = data;

    //Poll until the  transmit buffer register are empty
    //when they contain data that has not been moved to
    //transmit shift register
    while ((SPI0_INTFLAGS & SPI_IF_bm) == 0x00)
    {
        ;
    }

    PORTF_OUT |= PIN2_bm;

}
```