```
#include <unistd.h>                     //Needed for I2C port
#include <fcntl.h>                      //Needed for I2C port
#include <sys/ioctl.h>              //Needed for I2C port
#include <linux/i2c-dev.h>          //Needed for I2C port
int file_i2c;
int length;
unsigned char buffer[60] = {0};

//----- OPEN THE I2C BUS -----
char *filename = (char*)"/dev/i2c-1";
if ((file_i2c = open(filename, O_RDWR)) < 0)
{
    //ERROR HANDLING: you can check errno to see what went wrong
    printf("Failed to open the i2c bus");
    return;
}

int addr = 0x5a;          //<<<<<The I2C address of the slave
if (ioctl(file_i2c, I2C_SLAVE, addr) < 0)
{
    printf("Failed to acquire bus access and/or talk to slave.\n");
    //ERROR HANDLING; you can check errno to see what went wrong
    return;
}


//----- READ BYTES -----
length = 4;             //<<< Number of bytes to read
if (read(file_i2c, buffer, length) != length)        //read() returns the number
of bytes actually read, if it doesn't match then an error occurred (e.g. no
response from the device)
{
    //ERROR HANDLING: i2c transaction failed
    printf("Failed to read from the i2c bus.\n");
}
else
{
    printf("Data read: %s\n", buffer);
}

//----- WRITE BYTES -----
buffer[0] = 0x01;
buffer[1] = 0x02;
length = 2;             //<<< Number of bytes to write
if (write(file_i2c, buffer, length) != length)       //write() returns the number
of bytes actually written, if it doesn't match then an error occurred (e.g. no
response from the device)
{
    /* ERROR HANDLING: i2c transaction failed */
    printf("Failed to write to the i2c bus.\n");
}
//------------------------------
// stusb4500_reset resets the STUSB4500. This also results in loss of
// power to the entire board while STUSB4500 boots up again, effectively
// resetting the uC as well.
bool stusb4500_reset() {
  enum { addr = 0x28 };
  return i2c_reg_write(addr, 0x23, 0x01);
} // stusb4500_reset
```