```c
#include "stm32f3xx.h"
#include "spi.h"
#include "ioexpander.h"


void writeReg(uint8_t reg, uint8_t val)
{
    beginTransaction();
    transfer8(OPCODE_WRITE); /*Transfer OpCode*/
    transfer8(reg);  /*Transfer Register address*/
    transfer8(val);  /*Transfer value*/
    endTransaction();
}
uint8_t readReg(uint8_t reg)
{
    uint8_t returnValue;
    beginTransaction();
    transfer8(OPCODE_READ); /*Transfer OpCode*/
    transfer8(reg);  /*Transfer Register address*/
    returnValue = transfer8(0x55);  /*Transfer junk*/
    endTransaction();
    return returnValue;
}
void setBitInReg(uint8_t reg, uint8_t bit)
{
    uint8_t CurrentRegValue;
    CurrentRegValue = readReg(reg);  /*Read the previous value*/
    CurrentRegValue |= 1<<bit;  /*Set the bit required*/
    writeReg(reg,CurrentRegValue);  /*Update the value*/
}

void clearBitInReg(uint8_t reg, uint8_t bit)
{
    uint8_t CurrentRegValue;
    CurrentRegValue = readReg(reg);  /*Read the previous value*/
    CurrentRegValue &= ~(1<<bit);    /*Clear the bit required*/
    writeReg(reg,CurrentRegValue);   /*Update the value*/
}


void pinMode(port p,uint8_t pin, mode type)
{
    uint8_t addressIODIR;
    uint8_t addressPULLUP;
    if(p == PORTA)   /*Set port A addresses if port A is selected*/
    {
        addressIODIR = IODIRA;
        addressPULLUP = GPPUA;
    }
    else  /*Else set port B addresses*/
    {
        addressIODIR = IODIRB;
        addressPULLUP = GPPUB;
    }
    switch(type)
    {
        case OUTPUT:
        clearBitInReg(addressIODIR,pin);  /*Output requires the bit to be cleared in
        the IODIR register*/
        break;

        case INPUT :
        setBitInReg(addressIODIR,pin);  /*Input requires the bit to be set in the IODIR
        register*/
        break;

        case INPUT_PULLUP:
        setBitInReg(addressIODIR,pin);
        setBitInReg(addressPULLUP,pin); /*Input_Pullup requires an additional
```

```
            configuration of the pull up register*/

            break;

            default:

            break;

        }
    }

    void digitalWrite(port p,uint8_t pin,uint8_t state)
    {
        uint8_t addressGPIO;
        if(p == PORTA) /*Set port A address if port A is selected*/
            addressGPIO = GPIOAEXT;
        else  /*Else set port B address*/
            addressGPIO = GPIOBEXT;

        if(state)  /*Set bit to 1, indicating high output*/
            setBitInReg(addressGPIO,pin);
        else
            clearBitInReg(addressGPIO,pin);  /*Clear bit indicating low output*/
    }

    uint8_t readBits(port p)
    {
        uint8_t readValue;
        uint8_t addressGPIO;
        if(p == PORTA)  /*Set port A address if port A is selected*/
            addressGPIO = GPIOAEXT;
        else  /*Else set port B address*/
            addressGPIO = GPIOBEXT;

        readValue = readReg(addressGPIO);  /*Read the register value*/
        return readValue;  /*Return the read value*/
    }


```