

```

1  #include "stm32f3xx.h"
2  #include "spi.h"
3  #include "ioexpander.h"
4
5
6  void writeReg(uint8_t reg, uint8_t val)
7  {
8      beginTransaction();
9      transfer8(OPCODE_WRITE); /*Transfer OpCode*/
10     transfer8(reg); /*Transfer Register address*/
11     transfer8(val); /*Transfer value*/
12     endTransaction();
13 }
14 uint8_t readReg(uint8_t reg)
15 {
16     uint8_t returnValue;
17     beginTransaction();
18     transfer8(OPCODE_READ); /*Transfer OpCode*/
19     transfer8(reg); /*Transfer Register address*/
20     returnValue = transfer8(0x55); /*Transfer junk*/
21     endTransaction();
22     return returnValue;
23 }
24 void setBitInReg(uint8_t reg, uint8_t bit)
25 {
26     uint8_t CurrentRegValue;
27     CurrentRegValue = readReg(reg); /*Read the previous value*/
28     CurrentRegValue |= 1<<bit; /*Set the bit required*/
29     writeReg(reg,CurrentRegValue); /*Update the value*/
30 }
31
32 void clearBitInReg(uint8_t reg, uint8_t bit)
33 {
34     uint8_t CurrentRegValue;
35     CurrentRegValue = readReg(reg); /*Read the previous value*/
36     CurrentRegValue &= ~(1<<bit); /*Clear the bit required*/
37     writeReg(reg,CurrentRegValue); /*Update the value*/
38 }
39
40
41 void pinMode(port p,uint8_t pin, mode type)
42 {
43     uint8_t addressIODIR;
44     uint8_t addressPULLUP;
45     if(p == PORTA) /*Set port A addresses if port A is selected*/
46     {
47         addressIODIR = IODIRA;
48         addressPULLUP = GPPUA;
49     }
50     else /*Else set port B addresses*/
51     {
52         addressIODIR = IODIRB;
53         addressPULLUP = GPPUB;
54     }
55     switch(type)
56     {
57         case OUTPUT:
58             clearBitInReg(addressIODIR,pin); /*Output requires the bit to be cleared in
59             the IODIR register*/
60             break;
61
62         case INPUT :
63             setBitInReg(addressIODIR,pin); /*Input requires the bit to be set in the IODIR
64             register*/
65             break;
66
67         case INPUT_PULLUP:
68             setBitInReg(addressIODIR,pin);
69             setBitInReg(addressPULLUP,pin); /*Input_Pullup requires an additional

```

```

        configuration of the pull up register*/
68
69         break;
70
71         default:
72
73         break;
74
75     }
76 }
77
78 void digitalWrite(port p,uint8_t pin,uint8_t state)
79 {
80     uint8_t addressGPIO;
81     if(p == PORTA) /*Set port A address if port A is selected*/
82         addressGPIO = GPIOAEXT;
83     else /*Else set port B address*/
84         addressGPIO = GPIOBEXT;
85
86     if(state) /*Set bit to 1, indicating high output*/
87         setBitInReg(addressGPIO,pin);
88     else
89         clearBitInReg(addressGPIO,pin); /*Clear bit indicating low output*/
90 }
91
92 uint8_t readBits(port p)
93 {
94     uint8_t readValue;
95     uint8_t addressGPIO;
96     if(p == PORTA) /*Set port A address if port A is selected*/
97         addressGPIO = GPIOAEXT;
98     else /*Else set port B address*/
99         addressGPIO = GPIOBEXT;
100
101     readValue = readReg(addressGPIO); /*Read the register value*/
102     return readValue; /*Return the read value*/
103 }
104
105

```