```
/* I2C Self made Library
Author: Guillermo J. Ramires
Student nr: 464852 */

#include <avr/io.h>
#include <stdio.h>
#include <util/twi.h>

#define ATTEMPT 50
#define I2C_START 0
#define I2C_DATA 1
#define I2C_STOP 2
#define I2C_DATA_ACK 3
#define I2C_DATA_NACK 4

#define MCP23017_ID 0x40        //--- MCP23017 Device Identifier
#define MCP23017_ADDRESS 0x00   //--- MCP23017 Device Address

#define MCP23017_IODIRA 0x00    //--- MCP23017 I/O Direction A
#define MCP23017_IODIRB 0x01    //--- MCP23017 I/O Direction B

void i2c_init(void)
{
    TWSR = 0x00; // Prescaler : 1
    TWBR = 0x48; // SCL = F_CPU / (16 + (2 * TWBR) * Prescaler) formula in
      datasheet
}

uint8_t i2c_transmit(uint8_t type)
{
    switch(type)
    {
        case I2C_START: // Send Start Condition
        TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN);
        break;
        case I2C_DATA:  // Send Data with No-Acknowledge
        TWCR = (1 << TWINT) | (1 << TWEN);
        break;
        case I2C_DATA_ACK: // Send Data with Acknowledge
        TWCR = (1 << TWEA) | (1 << TWINT) | (1 << TWEN);
        break;
        //case I2C_DATA_NACK: // // Send Stop Condition
        //TWCR = (1 << TWINT) | (1 << TWEN);
        //break;
        case  I2C_STOP:
        TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);
        return 0;
        default:
        return -1;
```

```
    }// Wait for TWINT flag set on Register TWCR
    while (!(TWCR & (1 << TWINT)));
    return (TWSR & 0xF8); // Return TWI Status Register, mask the pre-scaler bits
      (TWPS1,TWPS0)
}

uint8_t i2c_start(uint8_t dev_id, uint8_t dev_addr, uint8_t rw_type)
{
    unsigned char n = 0;
    unsigned char twi_status;
    char r_val = -1;
    i2c_retry:
    if (n++ >= ATTEMPT) return r_val;
    // Transmit Start Condition
    twi_status = i2c_transmit(I2C_START);

    // Check the TWI Status
    if (twi_status == TW_MT_ARB_LOST) goto i2c_retry;
    if ((twi_status != TW_START) && (twi_status != TW_REP_START)) goto i2c_quit;
    // Send slave address (SLA_W)
    TWDR = (dev_id & 0xF0) | (dev_addr & 0x0E) | rw_type;
    // Transmit I2C Data
    twi_status = i2c_transmit(I2C_DATA);
    // Check the TWSR status
    if ((twi_status == TW_MT_SLA_NACK) || (twi_status == TW_MT_ARB_LOST)) goto
      i2c_retry;
    if (twi_status != TW_MT_SLA_ACK) goto i2c_quit;
    r_val = 0;
    i2c_quit:
    return r_val;
}

void i2c_stop(void)
{
    unsigned char twi_status;
    // Transmit I2C Data
    twi_status=i2c_transmit(I2C_STOP);
}

uint8_t i2c_write(uint8_t data)
{
    unsigned char twi_status;
    char r_val = -1;
    TWDR = data;
    twi_status = i2c_transmit(I2C_DATA);
    if (twi_status != TW_MT_DATA_ACK) goto i2c_quit;
    r_val = 0;
    i2c_quit:
    return r_val;
```

```
}

void mcp_write(unsigned char reg_addr, unsigned char data)
{
    i2c_start(MCP23017_ID, MCP23017_ADDRESS, TW_WRITE);
    i2c_write(reg_addr);
    i2c_write(data);
    i2c_stop();
}
```