```
//
//  i2c.c
//  i2c
//
//  Created by Michael Köhler on 09.10.17.
//
//

#include "i2c.h"
#include <util/delay.h>

//------------------------------------------------ I2C Status und Adresse
#define START 0x08
#define R_START 0x10
#define SLA_W 0b10001000        // Adresse: 100 1000 + Write 0
#define SLA_R 0b10001001        // Adresse: 100 1000 + Write 1
#define MT_SLA_ACK 0x18
#define MR_SLA_ACK 0x40
#define MR_SLA_NACK 0x48
#define MT_DATA_ACK 0x28
#define MR_DATA_ACK 0x50
#define MR_DATA_NACK 0x58

#define BIT_RATE 0x50
#define POINTER_ADDRESS 0x00
#define SCL_CLOCK  10000UL

uint8_t error1 = 0;


#if defined (__AVR_ATmega32__) || (__AVR_ATmega328__) || defined
  (__AVR_ATmega328P__) || \
defined(__AVR_ATmega168P__) || defined(__AVR_ATmega168PA__) || \
defined(__AVR_ATmega88P__) || \
defined(__AVR_ATmega48P__) || \
defined(__AVR_ATmega1284P__) || \
defined (__AVR_ATmega324A__) || defined (__AVR_ATmega324P__) || defined
  (__AVR_ATmega324PA__) || \
defined (__AVR_ATmega644__) || defined (__AVR_ATmega644A__) || defined
  (__AVR_ATmega644P__) || defined (__AVR_ATmega644PA__) || \
defined (__AVR_ATmega1284P__) || \
defined (__AVR_ATmega2560__)
#if PSC_I2C != 1 && PSC_I2C != 4 && PSC_I2C != 16 && PSC_I2C != 64
#error "Wrong prescaler for TWI !"
#elif SET_TWBR < 0 || SET_TWBR > 255
#error "TWBR out of range, change PSC_I2C or F_I2C !"
#endif

uint8_t I2C_ErrorCode;
/*************************************************
 Public Function: i2c_init

 Purpose: Initialise TWI/I2C interface

 Input Parameter: none
```

```
 Return Value: none
 ******************************************/
void i2c_init(void){
//  TWCR = 0;
//  TWSR = 0;
//  TWBR = ((F_CPU/SCL_CLOCK)-16)/2;
//  _delay_ms(50);


    // set clock
    switch (PSC_I2C) {
        case 4:
            TWSR = 0x1;
            break;
        case 16:
            TWSR = 0x2;
            break;
        case 64:
            TWSR = 0x3;
            break;
        default:
            TWSR = 0x00;
            break;
    }
    TWBR = (uint8_t)SET_TWBR;
    // enable
    TWCR = (1 << TWEN);
}
/*******************************************
 Public Function: i2c_start

 Purpose: Start TWI/I2C interface

 Input Parameter:
 - uint8_t i2c_addr: Adress of reciever

 Return Value: none
 *******************************************/
void i2c_start(uint8_t i2c_addr){
//  uint8_t timeout = 0;
//    TWCR = (1 << TWINT | 1 << TWSTA | 1 << TWEN);
//    while(!(TWCR & (1<<TWINT))){
//      if((timeout++) > 100) return 1;
//    }
//    TWDR = OLED_I2C_ADDR;
//    TWCR = (1 << TWINT | 1 << TWEN);
//  timeout = 0;
//    while(!(TWCR & (1<<TWINT))){
//      if((timeout++) > 100) return 1;
//    }
//  return 0;


    // i2c start
    TWCR = (1 << TWINT)|(1 << TWSTA)|(1 << TWEN);
    uint16_t timeout = F_CPU/F_I2C*2.0;
```

```
    while((TWCR & (1 << TWINT)) == 0 &&
        timeout !=0){
        timeout--;
        if(timeout == 0){
            I2C_ErrorCode |= (1 << I2C_START);
            return;
        }
    };
    // send adress
    TWDR = i2c_addr;
    TWCR = (1 << TWINT)|( 1 << TWEN);
    timeout = F_CPU/F_I2C*2.0;
    while((TWCR & (1 << TWINT)) == 0 &&
          timeout !=0){
        timeout--;
        if(timeout == 0){
            I2C_ErrorCode |= (1 << I2C_SENDADRESS);
            return;
        }
    };
}
/*********************************************
 Public Function: i2c_stop

 Purpose: Stop TWI/I2C interface

 Input Parameter: none

 Return Value: none
 *********************************************/
void i2c_stop(void){
    // i2c stop
    TWCR = (1 << TWINT)|(1 << TWSTO)|(1 << TWEN);
}
/*********************************************
 Public Function: i2c_byte

 Purpose: Send byte at TWI/I2C interface

 Input Parameter:
 - uint8_t byte: Byte to send to reciever

 Return Value: none
 *********************************************/
void i2c_byte(uint8_t byte){
    TWDR = byte;
    TWCR = (1 << TWINT)|( 1 << TWEN);
    uint16_t timeout = F_CPU/F_I2C*2.0;
    while((TWCR & (1 << TWINT)) == 0 &&
          timeout !=0){
        timeout--;
        if(timeout == 0){
            I2C_ErrorCode |= (1 << I2C_BYTE);
            return;
        }
    };
```

```
}
/*****************************************
 Public Function: i2c_readAck

 Purpose: read acknowledge from TWI/I2C Interface

 Input Parameter: none

 Return Value: uint8_t
  - TWDR: recieved value at TWI/I2C-Interface, 0 at timeout
  - 0:    Error at read
 *****************************************/
uint8_t i2c_readAck(void){
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
    uint16_t timeout = F_CPU/F_I2C*2.0;
    while((TWCR & (1 << TWINT)) == 0 &&
          timeout !=0){
        timeout--;
        if(timeout == 0){
            I2C_ErrorCode |= (1 << I2C_READACK);
            return 0;
        }
    };
    return TWDR;
}

 /*****************************************
 Public Function: i2c_readNAck

 Purpose: read non-acknowledge from TWI/I2C Interface

 Input Parameter: none

 Return Value: uint8_t
  - TWDR: recieved value at TWI/I2C-Interface
  - 0:    Error at read
 *****************************************/
uint8_t i2c_readNAck(void){
    TWCR = (1<<TWINT)|(1<<TWEN);
    uint16_t timeout = F_CPU/F_I2C*2.0;
    while((TWCR & (1 << TWINT)) == 0 &&
          timeout !=0){
        timeout--;
        if(timeout == 0){
            I2C_ErrorCode |= (1 << I2C_READNACK);
            return 0;
        }
    };
    return TWDR;
}

//-------------------------------------------------------------------------------->
  ---------------------------------- I2C-Funktionen aus C-Buch

void Error(void)
{
```

```
    error1 = 1;
}

void TWI_START(void)
{
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != START)
    Error();
}

void TWI_MT_SLA_ACK(void)
{
    TWDR = SLA_W;
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != MT_SLA_ACK)
    Error();
}

void TWI_MT_DATA_ACK(uint8_t daten)
{
    TWDR = daten;
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    //if ((TWSR & 0xF8) != MT_DATA_ACK)
    //Error();
}

void TWI_R_START(void)
{
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != R_START)
    Error();
}


void TWI_MR_SLA_ACK(void)
{
    TWDR = SLA_R;
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != MR_SLA_ACK)
    Error();
}

void TWI_MR_SLA_NACK(void)
{
    TWDR = SLA_R;
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != MR_SLA_NACK)
    Error();
}
```

```
uint8_t TWI_READ_DATABYTE_ACK(void)
{
    TWCR = (1<<TWINT) | (1<<TWEA) | (1<<TWEN) ;
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != MR_DATA_ACK)
    Error();
    return TWDR;
}

uint8_t TWI_READ_DATABYTE_NACK(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
    if ((TWSR & 0xF8) != MR_DATA_NACK)
    Error();
    return TWDR;
}


void TWI_STOP(void)
{
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
}

//-------------------------------------------------------------------------------->
  ------------------------------------
```

```
#else
#error "Micorcontroller not supported now!"
#endif
```