```c
/* Name: David (DongYun) Kim
 * SID: 200405213
 * Course: ENEL351
 * Description: ENEL351 Project - Smart Parking System
 * File name: i2c.c
 */

#include <stm32f10x.h>
#include "i2c.h"
/* This source file is derived from example code at nicerland.com */

/* i2c_init applies to controller and peripheral initialization */
/* turns on clocks, configures I/O, configures I2C1 in standard mode */
void i2c_init()
{
 RCC->APB2ENR |= RCC_APB2ENR_IOPBEN | RCC_APB2ENR_AFIOEN; /* enable clocks for I2C related
GPIOs */
 RCC->APB1ENR |= RCC_APB1ENR_I2C1EN; /* enable clock for I2C1 */

 GPIOB->CRL |= GPIO_CRL_CNF7 | GPIO_CRL_CNF6 | GPIO_CRL_MODE7 | GPIO_CRL_MODE6 ; //0xFF000000
/* configure PB6 and PB7 as alt. func. open drain */

//  Setup for 24MHZ APB1 Clock
// I2C1->CR2 |= I2C_CR2_FREQ_4 | I2C_CR2_FREQ_3 ; //0x0018 = 24 MHz
// I2C1->CCR = 0x0078 ; // 100KHz SCL = 10uS period with 5uS Thigh and 5 uS Tlow. 5 uS is 120
cycles (0x0078) at 24 MHZ;
// I2C1->TRISE = 25; // INT(1000nS/41ns) + 1 = 25;

//  Setup for 36MHZ APB1 Clock
 I2C1->CR2 |= I2C_CR2_FREQ_5 | I2C_CR2_FREQ_2;//36 = 32 + 4
 I2C1->CCR = 180; // CCR = (36MHZ/100KHz)/2 = 180 (0x00B4)
 I2C1->TRISE = 37; // (36MHZ/1MHZ) + 1 = 37;

}


void i2c_periph_set_ack()
{
 I2C1->CR1 |= I2C_CR1_ACK ; // Enable Peripheral ACK
}
void i2c_periph_set_ownaddr()
{
 I2C1->OAR1 = (0x68<<1) ; // Enable Peripheral ACK
}

void i2c_enable()
{
 I2C1->CR1 |=  I2C_CR1_PE ; /* PE = 1 */
}


void i2c_waitForReady()
{
  while((I2C1->SR2& I2C_SR2_BUSY) != 0); /* check bus busy */
}

void i2c_sendStart()
{
 int stat;
 I2C1->CR1 |= (I2C_CR1_START); /* start */
 while((I2C1->SR1&(I2C_SR1_SB)) == 0); /* wait for SB */
 stat = I2C1->SR2;
}

void i2c_sendStop()
{
 I2C1->CR1 |= (I2C_CR1_STOP); /* stop */
 while((I2C1->SR2&(I2C_SR2_MSL)) != 0); /* wait for becoming slave */
}

uint8_t i2c_sendAddr(uint8_t addr)
{
 int stat;
 I2C1->DR = addr;

 do{
  stat = I2C1->SR1;
  if((stat&(I2C_SR1_ARLO)) != 0) /* arbitration lost */
   return 1;

  if((stat&(I2C_SR1_ADDR)) != 0) /* address sent */
  {
   stat = I2C1->SR2; /* read SR2 to clear ADDR flag */
   return 0;
  }
 }while(1);
}

uint8_t i2c_sendAddrForRead(uint8_t addr)
{
 return i2c_sendAddr((addr<<1) + 1); /* addr+Read(1) */
}

uint8_t i2c_sendAddrForWrite(uint8_t addr)
{
 return i2c_sendAddr((addr<<1)); /* addr+Write(0) */
}

uint8_t i2c_sendData(uint8_t data)
{
 int stat1;
 I2C1->DR = data;

 do{
  stat1 = I2C1->SR1;
  if((stat1&(I2C_SR1_TXE)) != 0) /* TxE = 1 */
   return 0;
 } while(1);
}

uint8_t i2c_readData(uint8_t ack)
{
 if(ack!= 0)
  I2C1->CR1 |=  I2C_CR1_ACK;//1<<10;
 else
  I2C1->CR1 &=  ~(I2C_CR1_ACK);//~(1<<10);
 while((I2C1->SR1&(I2C_SR1_RXNE)) == 0); /* waiting for RxNE */
 return (I2C1->DR);
 //return (uint8_t)(I2C1->DR);
}

void i2c_sendbyte(uint8_t addr, uint8_t data)
{
do{
  i2c_waitForReady();
  i2c_sendStart();
 }while(i2c_sendAddrForWrite(addr) != 0); // If arbitration is lost i2c_sendAddrForWrite
returns 1. If address transmission is successful, it returns 0

  i2c_sendData(data);
  i2c_sendStop();

}
```