```
#include<I2C_constant.h>
/************************************************************************//**
 * @file I2C.c
 * @brief I2C implementation for ATmega.

 * This file implement an I2C bus for Atmel ATmega series. It is base on a application
 * note gave by ATMEL on implementing a TWI in assembler for all the AVR family.
 * "AVR 300: Software TWI Master Interface"

 * @author Patrick Grogan
 * @date 2008-01-30
 * @date 2008-02-05
 * @warning You must define these constant before using those functions
 * \li \c SPEED : Speed of the clk (SLOW, FAST)
 ************************************************************************/


 /*Function prototypes'*/
 void I2C_write(uint8_t I2C_data);
 void I2C_put_ack();
 void I2C_stop();
 uint8_t I2C_get_ack();
 uint8_t I2C_start(uint8_t I2C_slave_addr);
 uint8_t I2C_rep_start(uint8_t I2C_slave_addr);
 uint8_t I2C_read();

/************************************************************************//**
 * @fn uint8_t I2C_start(uint8_t I2C_slave_addr)
 * @brief This function start a communication with slave specified. Clear the
 *          carry flag if a slave respond. You must make I2C_write follow this
 *          function.
 * \param I2C_slave_addr slave adress to transfer data to (+1 read, +0 write)
 * @return 0 if slave respond, 1 if it did not
 ************************************************************************/

 uint8_t I2C_start(uint8_t I2C_slave_addr)
 {

    I2CDDR |= (1<<SDAP); /*put data line as output*/
    I2CDDR |= (1<<SCLP); /*put serial clock as output*/

    I2CPORT |= (1<<SCLP); /*force serial clock high*/
    I2CPORT &= (0<<SDAP); /*put 0 on SDA, this is the starting condition*/

    _delay_us(0.6); /* delay*/

    I2C_write(I2C_slave_addr);

    if(!I2C_get_ack()) /*if slave respond*/
        return 0;
    else
        return 1;   /*if not*/



 }


/************************************************************************//**
 * @fn uint8_t I2C_rep_start(uint8_t I2C_slave_addr)
 * @brief This function start a new communication with the slave specified
 *          A repeated START can only be given after a byte has been read
 *          or written.
 * \param I2C_slave_addr slave adress to transfer data to
 * @return 0 if slave responded, 1 if not
 ************************************************************************/

 uint8_t I2C_rep_start(uint8_t I2C_slave_addr)
```

```
 {

    return (I2C_start(I2C_slave_addr));

 }


/*************************************************************************//**
 * @fn void I2C_write(uint8_t I2C_data)
 * @brief Writes data (one byte) to the I2C bus. This function is also
 *       used for sending the address.
 * @param I2C_data data to be writen on the I2C bus
 * @warning This function must be followed by I2C_get_ack()
 *************************************************************************/

 void I2C_write(uint8_t I2C_data)
 {

    I2CDDR |= (1<<SCLP); /*control of clock*/
    I2CPORT |= (0<<SCLP); /*force clock to low*/

    I2CDDR |= (1<<SDAP); /*place sda as output*/

    for(int i = 0 ; i < 8 ; i++)
    {

        asm volatile("clc"::); /*clear the Carry Bit*/
        asm volatile("SBRC %0,%1"::"r" (I2C_data), "I" (PIN7)); /*If MSB = 1*/
        asm volatile("sec"::);//set Carry Bit
        asm volatile("rol %0" : "=r" (I2C_data) : "0" (I2C_data)); /*rotate left with
                                                                    CARRY*/

        if(I2C_data & (1<<PIN0)) /*if bit to transfer is set*/
            I2CPORT |= (1<<SDAP);/*set SDA*/
        else                     /*if bit to transer if clear*/
            I2CPORT &= ~(1<<SDAP);/*clear SDA*/

        _delay_us(1.3); /*delay of 1.3 µs*/
        I2CPORT |= (1<<SCLP); /*put clk high*/

        _delay_us(1.3); /*delay of 1.3 µs*/
        I2CPORT &= (0<<SCLP);/*put clk low*/
    }

 }

/*************************************************************************//**
 * @fn uint8_t I2C_get_ack()
 * @brief Wait for the aknowledge of the slave
 * @return 0 if slave acknowledged, 1 if not
 *************************************************************************/

 uint8_t I2C_get_ack()
 {
    uint8_t return_val = 0; /*value to return, 0 by default*/

    I2CDDR &= (0<<SDAP); /* data line as output*/

    _delay_us(1.3);
    I2CPORT |= (1<<SCLP); /* put clk high*/

    if((I2CPIN & (1<<SDAP))) /*if SDA is high*/
        return_val = 1; /*no acknowledge*/

    _delay_us(1.3); /*delay 1.3 µs*/
    I2CPORT &= (0<<SCLP);

    I2CDDR &= (0<<SCLP); /*release clock line*/
```

```
    while(!(I2CPIN & (1<<SCLP))); /*wait until slave not busy*/

    I2CDDR |= (1<<SCLP);    /*take control of serial clk*/
    I2CDDR |= (1<<SDAP);    /*take control of data line*/

    return return_val;


 }


/*************************************************************************//**
 * @fn uint8_t I2C_read()
 * @brief This function read data transferd by a slave
 * @return The byte recevied from the xfer
 * @warning Must be followed by
 *************************************************************************/

 uint8_t I2C_read()
 {
    uint8_t received_byte = 0; /*byte received to return*/

    I2CDDR |= (1<<SCLP); /*Serial clock as output*/
    I2CDDR &= (0<<SDAP); /*Data line as input*/

    I2CPORT &= (0<<SCLP); /*clock to low*/

    for (int i = 0; i < 8; i++)
    {
        _delay_us(1.3); /*half period delay*/
        I2CPORT |= (1<<SCLP); /*clock high*/

        received_byte = received_byte << 1; /*left shift one bit*/

        if(I2CPIN & (1<<SDAP))
            received_byte |= (1<<PIN0);
        else
            received_byte &= (0<<PIN0);

        _delay_us(1.3); /*half period delay*/
        I2CPORT &= (0<<SCLP); /*Clock low*/

    }

    return received_byte;
 }


/*************************************************************************//**
 * @fn void I2C_put_ack()
 * @brief This function put an aknowledge on the I2C bus
 *************************************************************************/

 void I2C_put_ack()
 {
    I2CDDR |= (1<<SCLP); /*control of serial clock*/
    I2CDDR |= (1<<SDAP); /*control of data line*/

    I2CPORT &= (0<<SDAP); /*force Serial data low*/
    I2CPORT |= (1<<SCLP); /*force clk to high*/
    _delay_us(0.6); /*quarter period delay*/
    I2CPORT |= (1<<SDAP); /*put serial data high*/
 }


/*************************************************************************//**
 * @fn void I2C_stop()
```

```
 * @brief This function read data transferd by a slave
 * @return The data recevied from the xfer
 ***************************************************************************/

 void I2C_stop()
 {
    I2CDDR |= (1<<SCLP); /*control of clock*/
    I2CDDR |= (1<<SDAP); /*take control of data line*/

    I2CPORT |= (1<<SCLP); /*force clk to high*/
    _delay_us(0.6); /*delay quarter of period*/
    I2CPORT |= (1<<SDAP);/*put data line to high*/
 }
```