```
/* Microchip Technology Inc. and its subsidiaries.  You may use this software
 * and any derivatives exclusively with Microchip products.
 *
 * THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS".  NO WARRANTIES, WHETHER
 * EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
 * WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
 * PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
 * WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.
 *
 * IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
 * INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
 * WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
 * BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE.  TO THE
 * FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS
 * IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF
 * ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.
 *
 * MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
 * TERMS.
 */

/*
 * File:
 * Author:
 * Comments:
 * Revision history:
 */

// This is a guard condition so that contents of this file are not included
// more than once.

#ifndef I2C_H
#defineI2C_H

#define I2C_BAUDRATE 100000

#define _XTAL_FREQ 20000000
//Write Sequence

//Function Proto types

void I2C_Start(void); //Start the Communication

void I2C_Wait(void); //Monitoring the idle state of wire

void I2C_Stop(void); //Terminate the connection

void I2C_ACK(void); //Master Acknowledgment

void I2C_Repeat_Start(void); //Repeated start for bus holding

int I2C_Sent_Data(unsigned char); //Write the character to Respective location

void I2C_NACK(void); //Master Negative-Acknowledgment

void I2C_Master_Init(void); //Initial configuration for Master

unsigned char I2C_Read_Data(int); //Read the Data from the Respective Location

void I2C_Page_Read(unsigned char*, int, unsigned char, unsigned char);
//Read the Sequence of Byte

void I2C_Page_Write(unsigned char *, unsigned char, unsigned char);
//Write the Sequence of Byte

int I2C_Device_Select_R(unsigned char, unsigned char);
```

```
int I2C_Device_Select_W(unsigned char, unsigned char);


int I2C_Data_Write(unsigned char);

unsigned char I2C_Char_Read(unsigned char, unsigned char, unsigned char, int);

unsigned char buff; //Temp buff

int I2C_Device_Select_W(unsigned char Device_add, unsigned char Reg_add) {

    while (I2C_Sent_Data((Device_add & 0xFE))) {

        I2C_Repeat_Start();

    }

    I2C_Sent_Data(Reg_add);

    return 1;


}

int I2C_Device_Select_R(unsigned char Device_add, unsigned char Reg_add) {

    while (I2C_Sent_Data((Device_add & 0xFE))) {

        I2C_Repeat_Start();

    }

    I2C_Sent_Data(Reg_add);

    I2C_Repeat_Start();

    while (I2C_Sent_Data(Device_add | 0x01))
        I2C_Repeat_Start();



    return 1;

}

int I2C_Data_Write(unsigned char data) {

    return I2C_Sent_Data(data);


}

//Sequence write

void I2C_Page_Write(unsigned char *data, unsigned char Device_add,
        unsigned char Reg_add) {

    while (I2C_Sent_Data((Device_add & 0xFE))) {

        I2C_Repeat_Start();

    }

    I2C_Sent_Data(Reg_add >> 8);

    I2C_Sent_Data((unsigned char) Reg_add);

    while (*data) {
```

```
        I2C_Sent_Data(*data);

        data++;

    }
    __delay_ms(10);

}


//single Character Read

unsigned char I2C_char_Read(unsigned char data, unsigned char Device_add, unsigned char Reg_add, int Ack) {

    while (I2C_Sent_Data(Device_add & 0xFE)) {
        I2C_Repeat_Start();

    }


    I2C_Sent_Data((unsigned char) Reg_add);


    I2C_Repeat_Start();

    while (I2C_Sent_Data(Device_add | 0x01))
        I2C_Repeat_Start();

    return (unsigned char) I2C_Read_Data(Ack);
}

//Read Sequence

void I2C_Page_Read(unsigned char* result, int Size,
        unsigned char Device_add, unsigned char Reg_add) {

    while (I2C_Sent_Data(Device_add & 0xFE)) {
        I2C_Repeat_Start();

    }

    I2C_Sent_Data(Reg_add >> 8);

    I2C_Sent_Data((unsigned char) Reg_add);


    I2C_Repeat_Start();

    while (I2C_Sent_Data(Device_add | 0x01))
        I2C_Repeat_Start();


    for (int i = Size; i >= 0; i--) {

        result[Size - i] = (unsigned char) I2C_Read_Data(i);
    }

    result[Size + 1] = 0;


}



//notice the IDLE

void I2C_Wait() {
```

```
    while (READ_WRITE || SSPCON2 & 0X1F);
}

//Initiate the Communication
void I2C_Start() {
    I2C_Wait();
    SEN = 1;
}
//Terminate the Communication
void I2C_Stop() {
    I2C_Wait();
    PEN = 1;
}
//Repeated_Start
void I2C_Repeat_Start() {
    I2C_Wait();
    RSEN = 1;
}
//Initial Configuration of Master Node
void I2C_Master_Init() {
    TRISC3 = 1;
    TRISC4 = 1; //SCL,SDL
    SMP = 1; //for 100 khz //SSPSTAT 7 bit for slew rate
    CKE = 0; //For disable the SMBUS standard (System management bus)
    SSPEN = 1; //bit 5 enable the I2c serial communication
    SSPCON |= 0X08; //Enable the I2C 7-bit address start and stop and interrupt enable
    SSPADD = ((_XTAL_FREQ) / (4 * I2C_BAUDRATE)) - 1;
}


//Master Receiver mode only
void I2C_ACK() {
    ACKDT = 0; //ACK
    I2C_Wait();
    ACKEN = 1; //Send the Signal
```

```
}

//Master Receiver mode only

void I2C_NACK() {

    ACKDT = 1; //NACK

    I2C_Wait();

    ACKEN = 1; //Send the signal


}

//Writing the Character

int I2C_Sent_Data(unsigned char data) {

    I2C_Wait();

    SSPBUF = data;

    I2C_Wait();

    return (int) ACKSTAT;
}

//Reading the Character

unsigned char I2C_Read_Data(int flag) {

    I2C_Wait();

    RCEN = 1;

    I2C_Wait();

    while (!SSPIF);

    buff = SSPBUF;

    SSPIF = 0;

    (flag != 0) ? I2C_ACK() : I2C_NACK();

    return buff;
}



// TODO Insert declarations or function prototypes (right here) to leverage
// live documentation

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */


    void I2C_Start(void);

    void I2C_Wait(void);

    void I2C_Stop(void);

    void I2C_Master_RCEN(void);
```

```
    void I2C_ACK(void);

    void I2C_Repeat_Start(void);

    int I2C_Sent_Data(unsigned char);

    void I2C_NACK(void);

    void I2C_Master_Init(void);

    unsigned char I2C_Read_Data(int);

    void I2C_Page_Read(unsigned char*, int, unsigned char, unsigned char);

    void I2C_Page_Write(unsigned char *, unsigned char, unsigned char);

    void I2C_Char_Write(unsigned char, unsigned char, unsigned char);

    unsigned char I2C_Char_Read(unsigned char, unsigned char, unsigned char, int);

    int I2C_Device_Select_R(unsigned char, unsigned char);


    int I2C_Device_Select_W(unsigned char, unsigned char);

    int I2C_Data_Write(unsigned char);

    // TODO If C++ is being used, regular C code needs function names to have C
    // linkage so the functions can be used by the c code.

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif/* XC_HEADER_TEMPLATE_H */
```