```
// I2Cdev library collection - Main I2C device class header file
// Abstracts bit and byte I2C R/W functions into a convenient class
// 6/9/2012 by Jeff Rowberg <jeff@rowberg.net>
//
// Changelog:
//     2012-06-09 - fix major issue with reading > 32 bytes at a time
with Arduino Wire
//                - add compiler warnings when using outdated or IDE or
limited I2Cdev implementation
//     2011-11-01 - fix write*Bits mask calculation (thanks sasquatch @
Arduino forums)
//     2011-10-03 - added automatic Arduino version detection for ease of
use
//     2011-10-02 - added Gene Knight's NBWire TwoWire class
implementation with small modifications
//     2011-08-31 - added support for Arduino 1.0 Wire library (methods
are different from 0.x)
//     2011-08-03 - added optional timeout parameter to read* methods to
easily change from default
//     2011-08-02 - added support for 16-bit registers
//                - fixed incorrect Doxygen comments on some methods
//                - added timeout value for read operations (thanks mem @
Arduino forums)
//     2011-07-30 - changed read/write function structures to return
success or byte counts
//                - made all methods static for multi-device memory
savings
//     2011-07-28 - initial release

/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a
copy
of this software and associated documentation files (the "Software"), to
deal
in the Software without restriction, including without limitation the
rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

```
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
===============================================
*/

#ifndef _I2CDEV_H_
#define _I2CDEV_H_

// -----------------------------------------------------------------------
-------
// I2C interface implementation setting
// -----------------------------------------------------------------------
-------
#define I2CDEV_IMPLEMENTATION       I2CDEV_ARDUINO_WIRE

// comment this out if you are using a non-optimal IDE/implementation
setting
// but want the compiler to shut up about it
#define I2CDEV_IMPLEMENTATION_WARNINGS

// -----------------------------------------------------------------------
-------
// I2C interface implementation options
// -----------------------------------------------------------------------
-------
#define I2CDEV_ARDUINO_WIRE         1 // Wire object from Arduino
#define I2CDEV_BUILTIN_NBWIRE       2 // Tweaked Wire object from Gene
Knight's NBWire project
                                      // ^^^ NBWire implementation is
still buggy w/some interrupts!
#define I2CDEV_BUILTIN_FASTWIRE     3 // FastWire object from Francesco
Ferrara's project
                                      // ^^^ FastWire implementation in
I2Cdev is INCOMPLETE!

// -----------------------------------------------------------------------
-------
// Arduino-style "Serial.print" debug constant (uncomment to enable)
// -----------------------------------------------------------------------
-------
#define I2CDEV_SERIAL_DEBUG

#ifdef ARDUINO
    #if ARDUINO < 100
        #include "WProgram.h"
    #else
        #include "Arduino.h"
    #endif
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        #include <Wire.h>
    #endif
#else
```

```
    #include "ArduinoWrapper.h"
#endif

// 1000ms default read timeout (modify with "I2Cdev::readTimeout =
[ms];")
#define I2CDEV_DEFAULT_READ_TIMEOUT     1000

class I2Cdev {
    public:
        I2Cdev();

        static int8_t readBit(uint8_t devAddr, uint8_t regAddr, uint8_t
bitNum, uint8_t *data, uint16_t timeout=I2Cdev::readTimeout);
        static int8_t readBitW(uint8_t devAddr, uint8_t regAddr, uint8_t
bitNum, uint16_t *data, uint16_t timeout=I2Cdev::readTimeout);
        static int8_t readBits(uint8_t devAddr, uint8_t regAddr, uint8_t
bitStart, uint8_t length, uint8_t *data, uint16_t
timeout=I2Cdev::readTimeout);
        static int8_t readBitsW(uint8_t devAddr, uint8_t regAddr, uint8_t
bitStart, uint8_t length, uint16_t *data, uint16_t
timeout=I2Cdev::readTimeout);
        static int8_t readByte(uint8_t devAddr, uint8_t regAddr, uint8_t
*data, uint16_t timeout=I2Cdev::readTimeout);
        static int8_t readWord(uint8_t devAddr, uint8_t regAddr, uint16_t
*data, uint16_t timeout=I2Cdev::readTimeout);
        static int8_t readBytes(uint8_t devAddr, uint8_t regAddr, uint8_t
length, uint8_t *data, uint16_t timeout=I2Cdev::readTimeout);
        static int8_t readWords(uint8_t devAddr, uint8_t regAddr, uint8_t
length, uint16_t *data, uint16_t timeout=I2Cdev::readTimeout);

        static bool writeBit(uint8_t devAddr, uint8_t regAddr, uint8_t
bitNum, uint8_t data);
        static bool writeBitW(uint8_t devAddr, uint8_t regAddr, uint8_t
bitNum, uint16_t data);
        static bool writeBitWADS(uint8_t devAddr, uint8_t regAddr,
uint8_t bitNum, uint16_t data); // for TI ADS1115
        static bool writeBits(uint8_t devAddr, uint8_t regAddr, uint8_t
bitStart, uint8_t length, uint8_t data);
        static bool writeBitsW(uint8_t devAddr, uint8_t regAddr, uint8_t
bitStart, uint8_t length, uint16_t data);
        static bool writeByte(uint8_t devAddr, uint8_t regAddr, uint8_t
data);
        static bool writeWord(uint8_t devAddr, uint8_t regAddr, uint16_t
data);
        static bool writeWordADS(uint8_t devAddr, uint8_t regAddr,
uint16_t data); //For TI ADS1115
        static bool writeBytes(uint8_t devAddr, uint8_t regAddr, uint8_t
length, uint8_t *data);
        static bool writeWords(uint8_t devAddr, uint8_t regAddr, uint8_t
length, uint16_t *data);
        static bool writeWordsADS(uint8_t devAddr, uint8_t regAddr,
uint8_t length, uint16_t *data);

        static uint16_t readTimeout;
```

```
};

#if I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    ////////////////////
    // FastWire 0.2
    // This is a library to help faster programs to read I2C devices.
    // Copyright(C) 2011
    // Francesco Ferrara
    ////////////////////

    /* Master */
    #define TW_START                0x08
    #define TW_REP_START            0x10

    /* Master Transmitter */
    #define TW_MT_SLA_ACK           0x18
    #define TW_MT_SLA_NACK          0x20
    #define TW_MT_DATA_ACK          0x28
    #define TW_MT_DATA_NACK         0x30
    #define TW_MT_ARB_LOST          0x38

    /* Master Receiver */
    #define TW_MR_ARB_LOST          0x38
    #define TW_MR_SLA_ACK           0x40
    #define TW_MR_SLA_NACK          0x48
    #define TW_MR_DATA_ACK          0x50
    #define TW_MR_DATA_NACK         0x58

    #define TW_OK                   0
    #define TW_ERROR                1

    class Fastwire {
        private:
            static boolean waitInt();

        public:
            static void setup(int khz, boolean pullup);
            static byte write(byte device, byte address, byte value);
            static byte readBuf(byte device, byte address, byte *data,
byte num);
    };
#endif

#if I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_NBWIRE
    // NBWire implementation based heavily on code by Gene Knight
<Gene@Telobot.com>
    // Originally posted on the Arduino forum at
http://arduino.cc/forum/index.php/topic,70705.0.html
    // Originally offered to the i2cdevlib project at
http://arduino.cc/forum/index.php/topic,68210.30.html

    #define NBWIRE_BUFFER_LENGTH 32

    class TwoWire {
```

```
    private:
        static uint8_t rxBuffer[];
        static uint8_t rxBufferIndex;
        static uint8_t rxBufferLength;

        static uint8_t txAddress;
        static uint8_t txBuffer[];
        static uint8_t txBufferIndex;
        static uint8_t txBufferLength;

        // static uint8_t transmitting;
        static void (*user_onRequest)(void);
        static void (*user_onReceive)(int);
        static void onRequestService(void);
        static void onReceiveService(uint8_t*, int);

    public:
        TwoWire();
        void begin();
        void begin(uint8_t);
        void begin(int);
        void beginTransmission(uint8_t);
        //void beginTransmission(int);
        uint8_t endTransmission(uint16_t timeout=0);
        void nbendTransmission(void (*function)(int)) ;
        uint8_t requestFrom(uint8_t, int, uint16_t timeout=0);
        //uint8_t requestFrom(int, int);
        void nbrequestFrom(uint8_t, int, void (*function)(int));
        void send(uint8_t);
        void send(uint8_t*, uint8_t);
        //void send(int);
        void send(char*);
        uint8_t available(void);
        uint8_t receive(void);
        void onReceive(void (*)(int));
        void onRequest(void (*)(void));
};

#define TWI_READY   0
#define TWI_MRX     1
#define TWI_MTX     2
#define TWI_SRX     3
#define TWI_STX     4

#define TW_WRITE    0
#define TW_READ     1

#define TW_MT_SLA_NACK      0x20
#define TW_MT_DATA_NACK     0x30

#define CPU_FREQ            16000000L
#define TWI_FREQ            100000L
#define TWI_BUFFER_LENGTH   32
```

```
    /* TWI Status is in TWSR, in the top 5 bits: TWS7 - TWS3 */

    #define TW_STATUS_MASK
(_BV(TWS7)|_BV(TWS6)|_BV(TWS5)|_BV(TWS4)|_BV(TWS3))
    #define TW_STATUS                   (TWSR & TW_STATUS_MASK)
    #define TW_START                    0x08
    #define TW_REP_START                0x10
    #define TW_MT_SLA_ACK               0x18
    #define TW_MT_SLA_NACK              0x20
    #define TW_MT_DATA_ACK              0x28
    #define TW_MT_DATA_NACK             0x30
    #define TW_MT_ARB_LOST              0x38
    #define TW_MR_ARB_LOST              0x38
    #define TW_MR_SLA_ACK               0x40
    #define TW_MR_SLA_NACK              0x48
    #define TW_MR_DATA_ACK              0x50
    #define TW_MR_DATA_NACK             0x58
    #define TW_ST_SLA_ACK               0xA8
    #define TW_ST_ARB_LOST_SLA_ACK      0xB0
    #define TW_ST_DATA_ACK              0xB8
    #define TW_ST_DATA_NACK             0xC0
    #define TW_ST_LAST_DATA             0xC8
    #define TW_SR_SLA_ACK               0x60
    #define TW_SR_ARB_LOST_SLA_ACK      0x68
    #define TW_SR_GCALL_ACK             0x70
    #define TW_SR_ARB_LOST_GCALL_ACK    0x78
    #define TW_SR_DATA_ACK              0x80
    #define TW_SR_DATA_NACK             0x88
    #define TW_SR_GCALL_DATA_ACK        0x90
    #define TW_SR_GCALL_DATA_NACK       0x98
    #define TW_SR_STOP                  0xA0
    #define TW_NO_INFO                  0xF8
    #define TW_BUS_ERROR                0x00

    //#define _MMIO_BYTE(mem_addr) (*(volatile uint8_t *)(mem_addr))
    //#define _SFR_BYTE(sfr) _MMIO_BYTE(_SFR_ADDR(sfr))

    #ifndef sbi // set bit
        #define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
    #endif // sbi

    #ifndef cbi // clear bit
        #define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
    #endif // cbi

    extern TwoWire Wire;

#endif // I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_NBWIRE

#endif /* _I2CDEV_H_ */
```