// Code by JeeLabs http://news.jeelabs.org/code/

// Released to the public domain! Enjoy!

#ifndef \_RTCLIB\_H\_

#define \_RTCLIB\_H\_

#include <Arduino.h>

class TimeSpan;

// Simple general-purpose date/time class (no TZ / DST / leap second handling!)

class DateTime

{

public:

    DateTime(uint32\_t t =0);

    DateTime(uint16\_t year, uint8\_t month, uint8\_t day,

             uint8\_t hour =0, uint8\_t min =0, uint8\_t sec =0);

    DateTime(const DateTime& copy);

    DateTime(const char \* date, const char \* time);

    DateTime(const \_\_FlashStringHelper \* date, const \_\_FlashStringHelper \* time);

    uint16\_t year() const       { return 2000 + yOff; }

    uint8\_t month() const       { return m; }

    uint8\_t day() const         { return d; }

    uint8\_t hour() const        { return hh; }

    uint8\_t minute() const      { return mm; }

    uint8\_t second() const      { return ss; }

    uint8\_t dayOfTheWeek() const;

    // 32-bit times as seconds since 1/1/2000

    long secondstime() const;

    // 32-bit times as seconds since 1/1/1970

    uint32\_t unixtime(void) const;

    DateTime operator+(const TimeSpan& span);

    DateTime operator-(const TimeSpan& span);

    TimeSpan operator-(const DateTime& right);

protected:

    uint8\_t yOff, m, d, hh, mm, ss;

};

// Timespan which can represent changes in time with seconds accuracy.

class TimeSpan

{

public:

    TimeSpan (int32\_t seconds = 0);

    TimeSpan (int16\_t days, int8\_t hours, int8\_t minutes, int8\_t seconds);

    TimeSpan (const TimeSpan& copy);

    int16\_t days() const         { return \_seconds / 86400L; }

    int8\_t  hours() const        { return \_seconds / 3600 % 24; }

    int8\_t  minutes() const      { return \_seconds / 60 % 60; }

    int8\_t  seconds() const      { return \_seconds % 60; }

    int32\_t totalseconds() const { return \_seconds; }

    TimeSpan operator+(const TimeSpan& right);

    TimeSpan operator-(const TimeSpan& right);

protected:

    int32\_t \_seconds;

};

// RTC based on the DS1307 chip connected via I2C and the Wire library

enum Ds1307SqwPinMode { OFF = 0x00, ON = 0x80, SquareWave1HZ = 0x10, SquareWave4kHz = 0x11, SquareWave8kHz = 0x12, SquareWave32kHz = 0x13 };

class RTC\_DS1307

{

public:

    boolean begin(void);

    static void adjust(const DateTime& dt);

    uint8\_t isrunning(void);

    static DateTime now();

    static Ds1307SqwPinMode readSqwPinMode();

    static void writeSqwPinMode(Ds1307SqwPinMode mode);

    uint8\_t readnvram(uint8\_t address);

    void readnvram(uint8\_t\* buf, uint8\_t size, uint8\_t address);

    void writenvram(uint8\_t address, uint8\_t data);

    void writenvram(uint8\_t address, uint8\_t\* buf, uint8\_t size);

};

// RTC using the internal millis() clock, has to be initialized before use

// NOTE: this clock won't be correct once the millis() timer rolls over (>49d?)

class RTC\_Millis

{

public:

    static void begin(const DateTime& dt) { adjust(dt); }

    static void adjust(const DateTime& dt);

    static DateTime now();

protected:

    static long offset;

};

#endif // \_RTCLIB\_H