RTClib / RTClib.cpp

100644 186 lines (155 sloc) 5.277 kb

- raw
- blame
- history

```
// Code by JeeLabs http://news.jeelabs.org/code/
// Released to the public domain! Enjoy!

#include <Wire.h>
#include <avr/pgmspace.h>
#include "RTClib.h"
#include <WProgram.h>

#define DS1307_ADDRESS 0x68
#define SECONDS_PER_DAY 86400L

#define SECONDS_FROM_1970_TO_2000 946684800

////////////////////////////////////////////////////////////////////////////////
// utility code, some of this could be exposed in the DateTime API if needed

static uint8_t daysInMonth [] PROGMEM = { 31,28,31,30,31,30,31,31,30,31,30,31 };

// number of days since 2000/01/01, valid for 2001..2099
static uint16_t date2days(uint16_t y, uint8_t m, uint8_t d) {
    if (y >= 2000)
        y -= 2000;
    uint16_t days = d;
    for (uint8_t i = 1; i < m; ++i)
        days += pgm_read_byte(daysInMonth + i - 1);
    if (m > 2 && y % 4 == 0)
        ++days;
    return days + 365 * y + (y + 3) / 4 - 1;
}

static long time2long(uint16_t days, uint8_t h, uint8_t m, uint8_t s) {
    return ((days * 24L + h) * 60 + m) * 60 + s;
}

////////////////////////////////////////////////////////////////////////////////
// DateTime implementation - ignores time zones and DST changes
// NOTE: also ignores leap seconds, see http://en.wikipedia.org/wiki/Leap_second

DateTime::DateTime (uint32_t t) {
  t -= SECONDS_FROM_1970_TO_2000;    // bring to 2000 timestamp from 1970

    ss = t % 60;
    t /= 60;
    mm = t % 60;
    t /= 60;
    hh = t % 24;
    uint16_t days = t / 24;
    uint8_t leap;
    for (yOff = 0; ; ++yOff) {
        leap = yOff % 4 == 0;
        if (days < 365 + leap)
            break;
        days -= 365 + leap;
    }
    for (m = 1; ; ++m) {
        uint8_t daysPerMonth = pgm_read_byte(daysInMonth + m - 1);
        if (leap && m == 2)
            ++daysPerMonth;
        if (days < daysPerMonth)
            break;
        days -= daysPerMonth;
    }
    d = days + 1;
}

DateTime::DateTime (uint16_t year, uint8_t month, uint8_t day, uint8_t hour, uint8_t min, uint8_t sec) {
    if (year >= 2000)
        year -= 2000;
    yOff = year;
    m = month;
    d = day;
    hh = hour;
    mm = min;
    ss = sec;
}

static uint8_t conv2d(const char* p) {
    uint8_t v = 0;
    if ('0' <= *p && *p <= '9')
        v = *p - '0';
    return 10 * v + *++p - '0';
}

// A convenient constructor for using "the compiler's time":
//   DateTime now (__DATE__, __TIME__);
// NOTE: using PSTR would further reduce the RAM footprint
DateTime::DateTime (const char* date, const char* time) {
    // sample input: date = "Dec 26 2009", time = "12:34:56"
    yOff = conv2d(date + 9);
    // Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
    switch (date[0]) {
        case 'J': m = date[1] == 'a' ? 1 : m = date[2] == 'n' ? 6 : 7; break;
        case 'F': m = 2; break;
        case 'A': m = date[2] == 'r' ? 4 : 8; break;
        case 'M': m = date[2] == 'r' ? 3 : 5; break;
        case 'S': m = 9; break;
        case 'O': m = 10; break;
        case 'N': m = 11; break;
        case 'D': m = 12; break;
    }
    d = conv2d(date + 4);
    hh = conv2d(time);
    mm = conv2d(time + 3);
    ss = conv2d(time + 6);
}

uint8_t DateTime::dayOfWeek() const {
    uint16_t day = secondstime() / SECONDS_PER_DAY;
    return (day + 6) % 7; // Jan 1, 2000 is a Saturday, i.e. returns 6
}

uint32_t DateTime::unixtime(void) const {
  uint32_t t;
  uint16_t days = date2days(yOff, m, d);
  t = time2long(days, hh, mm, ss);
  t += SECONDS_FROM_1970_TO_2000;  // seconds from 1970 to 2000

  return t;
}

///////////////////////////////////////////////////////////////////////////////
// RTC_DS1307 implementation

static uint8_t bcd2bin (uint8_t val) { return val - 6 * (val >> 4); }
static uint8_t bin2bcd (uint8_t val) { return val + 6 * (val / 10); }

uint8_t RTC_DS1307::begin(void) {
  return 1;
}

uint8_t RTC_DS1307::isrunning(void) {
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.send(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_ADDRESS, 1);
  uint8_t ss = Wire.receive();
  return !(ss>>7);
}

void RTC_DS1307::adjust(const DateTime& dt) {
    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.send(0);
    Wire.send(bin2bcd(dt.second()));
    Wire.send(bin2bcd(dt.minute()));
    Wire.send(bin2bcd(dt.hour()));
    Wire.send(bin2bcd(0));
    Wire.send(bin2bcd(dt.day()));
    Wire.send(bin2bcd(dt.month()));
    Wire.send(bin2bcd(dt.year() - 2000));
    Wire.send(0);
    Wire.endTransmission();
}

DateTime RTC_DS1307::now() {
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.send(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_ADDRESS, 7);
  uint8_t ss = bcd2bin(Wire.receive() & 0x7F);
  uint8_t mm = bcd2bin(Wire.receive());
  uint8_t hh = bcd2bin(Wire.receive());
  Wire.receive();
  uint8_t d = bcd2bin(Wire.receive());
  uint8_t m = bcd2bin(Wire.receive());
  uint16_t y = bcd2bin(Wire.receive()) + 2000;

  return DateTime (y, m, d, hh, mm, ss);
}

///////////////////////////////////////////////////////////////////////////////
// RTC_Millis implementation

long RTC_Millis::offset = 0;

void RTC_Millis::adjust(const DateTime& dt) {
    offset = dt.secondstime() - millis() / 1000;
}
```

```

DateTime RTC_Millis::now() {
  return (uint32_t)(offset + millis() / 1000);
}

////////////////////////////////////////////////////////////////////////////////
```