**[RTClib](http://github.com/adafruit/RTClib/tree/master)** / RTClib.cpp RTClib.cpp 

Txt100644 186 lines (155 sloc) 5.277 kb

* [raw](http://github.com/adafruit/RTClib/raw/master/RTClib.cpp)
* [blame](http://github.com/adafruit/RTClib/blame/master/RTClib.cpp)
* [history](http://github.com/adafruit/RTClib/commits/master/RTClib.cpp)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186 | // 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);  }  //////////////////////////////////////////////////////////////////////////////// |