DS1307 RTC & AT24C32

Interfacing to the Arduino Mega 2560

Ric Morte, 06 August 2012

Document Control

Date	Status	Filename
2012-08-06	First Draft	DS1307 RTC Libraries And Examples V2.1.Docx

Contents

1 Introduction

The DS1307 provides an interesting project involving:

- Interfacing to the Arduino
- Using the I2C bus
- Programming the DS1307
- Reading and using RTC data
- Using the AT24C32 storage capacity

The DS1307 is not a particularly accurate clock. Those wanting a more accurate clock would do well to check out the DS3231 from Maxim. Here is the datasheet for this chip:

http://datasheets.maxim-ic.com/en/ds/DS3231.pdf

If you are interested in a board to use with the Arduino then here is a suitable supply for the UK:

https://www.loveelectronics.co.uk/products/137/ds3231-real-time-clock-module

It is believed (and I may yet try to confirm this) that software developed for the DS1307 will still function with the DS3231. The latter RTC will make a good 2nd project to follow on from the DS1307 project.

2 Background

Several articles have been published around the theme of using the DS1307 RTC. All use different approaches, some more easy to configure and understand than others and all with varying degrees of success.

Each approach uses different libraries and, after a while, these proliferate and it is easy to forget which library is which, what each does and which is best for developing and taking further. This document is simply a record of the different approaches and the outcomes.

This document focuses on the DS1307; using the AT24C32 is a much lower priority (and may never be documented!)

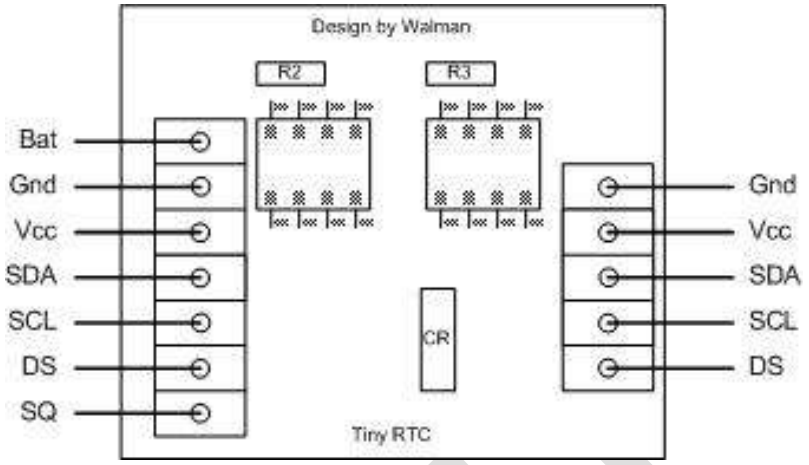
3 The RTC Board

Mine was bough from eBay, cost £4.80 and delivered free from China. The board is shown opposite. I think I could have bought one cheaper but... hey ho...

It was listed as “Version 2”, whatever that means... This “Version 2” appears to have a smaller crystal profile than similar boards. Other than that I can’t see that much difference between them. The board has two sets of solder pads each side for the Power supply and I2C interface; the other set has battery and square wave clock pulse breakouts.

3.1 DS1307 & AT24C32 Pin Connections

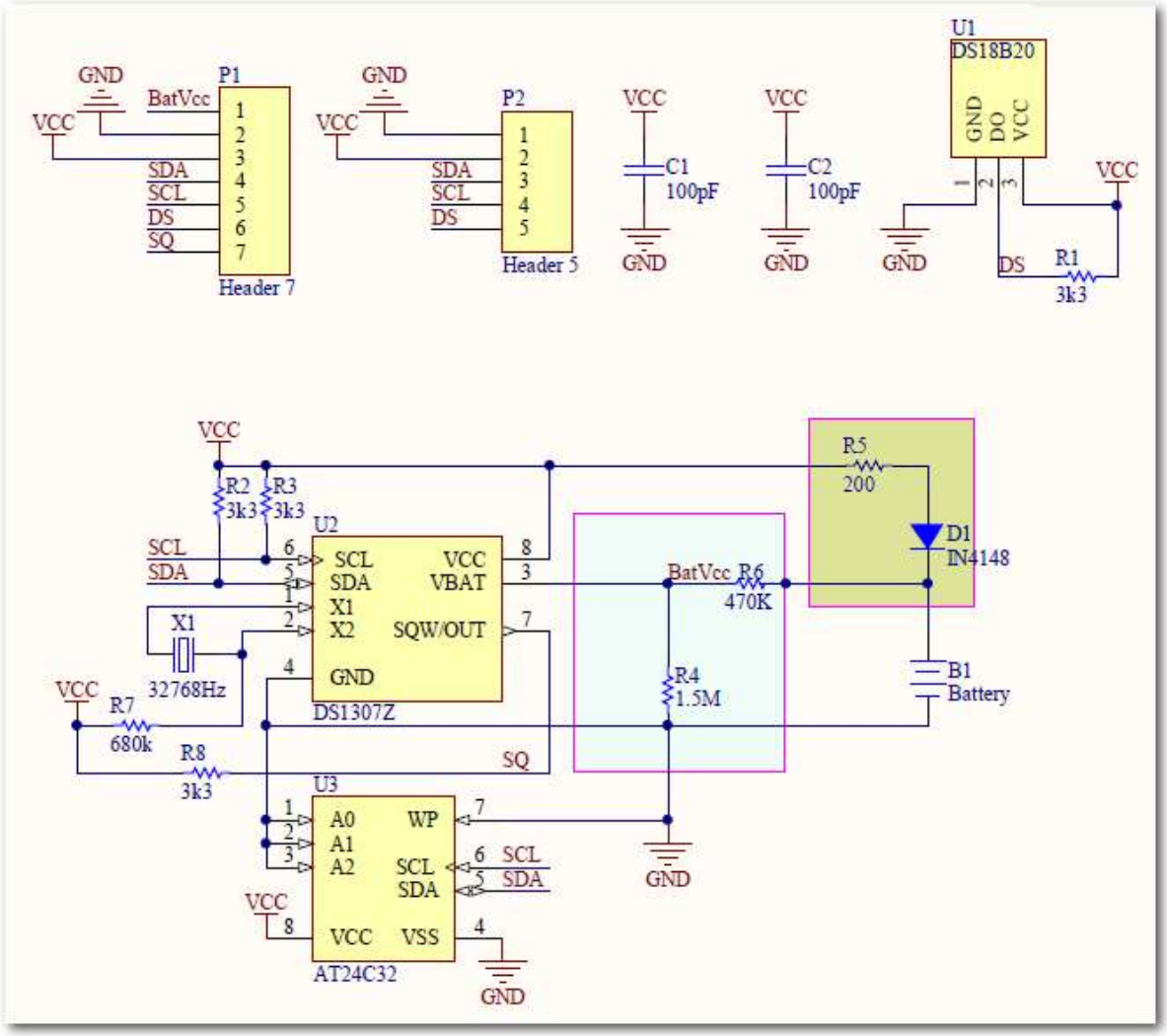
Here are the connections for this board, looking from the component side:



R2 and R3 are the 3K3 pull-up resistors on the SDA and SCL i2C bus lines.

3.2 DS1307 & AT24C32 Circuit (Schematic)

The circuit (schematic) diagram below appears to relate to this board:



I say "appears" because there is no way I can trace the tracks underneath the battery which is situated on the underside of the board. There is (almost) exactly the same number of components and they are numbered correspondingly.

My guess is that the circuit diagram does indeed relate to the board with the exception that U1, the DS18B20 1-wire digital thermometer, is not present on the board although there are solder pads to take this device.

3.3 DS1307 Specifications

There are various data sheets on the DS1307. Here are a few useful links:

Link	Comments
http://www.sparkfun.com/datasheets/Components/DS1307.pdf	The Dallas technical datasheet. This is your bible to understanding exactly what is (and more importantly, what is not) in the chip,
http://www.maxim-ic.com/datasheet/index.mvp/id/2688	Maxim website technical information

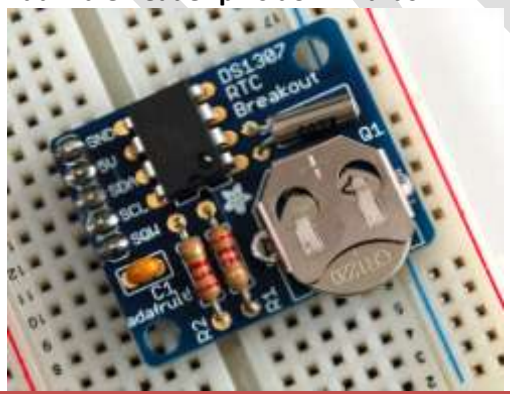
http://www.microbot.it/documents/mr005-001_datasheet.pdf	This datasheet relates to a different implementation of the DS1307. This is a simple 2-page datasheet containing useful descriptions.
http://www.ladyada.net/learn/breakoutplus/ds1307rtc.html	Construction of an RTC board using the DS1307. Links to files at GitHub. Note that most files are now very old (> 2 years) and not really relevant to the Mega 2560.
http://www.sparkfun.com/products/99	SparkFun implementation of the DS1307 using their proprietary board. Build it or buy it.

4 Setting up the DS1307 Board

I thought this would be straightforward... I should have known!

4.1 Step 1: Add Connectors

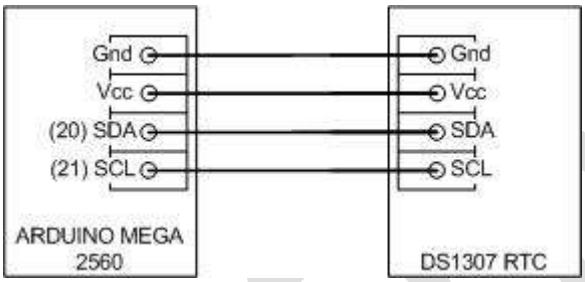
My board had no connectors and there were three choices I could have made:

Option	Advantages / Disadvantages
Add male header pins upwards or right-angled male header pins	Adding pins pointing upwards allows me to connect individual Dupont female-male jumpers between the RTC board and the Arduino
Add male header pins downwards 	Adding pins pointing downwards allows me to plug the RTC board into 0.1” matrix breadboard and from there to the Arduino: Note this board is not the one I used, it is courtesy: http://www.ladyada.net/learn/breakoutplus/ds1307rtc.html This format also allows a direct connection to the Arduino Duemilanove as shown oppoite

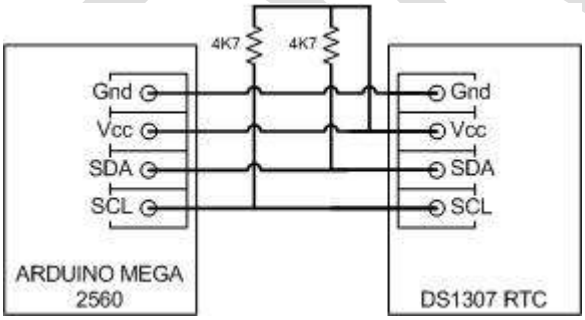
	
Add female header sockets	This allows Dupont male header pins to be inserted in exactly the same way as the Arduino.

4.2 Step 2: Connecting to the Mega 2560

The 2560 comes with dedicated SDA and SCL pins for the I2C serial communication bus. The RTC requires a +5V and GND connections:



Some boards do not have dedicated pull-up resistors for the I2C bus and must be added separately. Only one set of resistors is required regardless of the number of devices connected to the bus. If resistors are required they go here:



5 Code Comparisons

It was pointed out in section 2 “Background” that a number of sketch examples exist. These are documented in the following sections together with the results. As a rule there are two separate stages in programming the RTC:

1. Setting the date-time
2. Reading the date-time

Ideally setting the time needs to occur just once; thereafter the clock, keeping perfect time, will never need resetting. In practice the DS1307 is not that accurate and due to component mismatch and particularly temperature changes, the clock is not guaranteed to be more accurate than to within a minute per month. This is quite poor by modern standards. Setting the time (or rather, re-setting the time) on the DS1307 will therefore be a more common practice.

Reading the data is relatively straightforward and requires interrogating the appropriate registers and performing the BCD (binary-coded-decimal) conversion.

A further word about setting the time is required. There are different approaches to this task and care must be taken to ensure the date-time is not continually being reset by the date-time set program.

If the date-time set program is resident in memory and the date-time is hard-coded (as in the first example, 6.1 ”Simple ‘Bildr’ Example”), the date-time will be reset to the hard-coded values each time the program runs. Such a date-time set sketch should be called only ONCE and then control given over to a date-time read sketch.

Most of the example sketches available on the internet use separate sketches for setting and reading (or require a “set time” section of code to be commented-out when reading the time from the RTC). With these sketches, every time the RTC needs to be reset the date-time set program must be re-sent to the board (or un-commented) with new hard-coded values and the date-time updated. The date-time set program must be replaced immediately with a date-time read program so that values can be retrieved.

Such an approach is OK for testing but useless for any serious real-world situation.

A better approach is to write a program that accepts a serial input. The serial input string is nothing more than the date-time set arguments. This allows the one sketch to be sent to the Arduino and the date-time updated by sending an “update” string. There are two examples of this approach documented here [TODO] and here [TODO]. Just one of the examples uses command line arguments that will accept user input and date-time values. This one sketch can be sent to the Arduino board and perform both reads and writes without further complication.

A third approach would be a much more fully-fledged system with an interactive interface whereby the date-time can be set “on-the-fly”. Suitable interfaces might be a LCD display where values can be selected and used to configure the RTC. Such an approach is documented here:

[TODO]

In working my way through the various examples the most difficult aspect has been to determine the historical sequence in which sketches and libraries have been developed. Many of the earlier ones have few credits and no dates. Almost none have a change history and there seems to have been little or no control over version history. For this reason variants stand side-by-side with others that have bugs because it is not clear which version supersedes which.

This is one of the biggest problems with Arduino sketches and libraries for those new to the arena. It is not surprising to see the number of forum posts on the theme “Newbie needs help! RTC doesn’t work!”.

Two final notes before the examples that follow:

I came to the Arduino relatively late and my first board was the Mega 2560. For this reason I have absolutely no interest in the Mega 1280 or any other precursors including the Duemilanove and the

Uno. I have absolutely no idea if any of the following examples will work for, or produce consistent results with, those boards. Heck, for the former, I'm not even sure how to pronounce the name!

I am also using Arduino IDE version 1.0.1. Every example listed below has been updated to work with version 1.0.1 and cannot therefore be relied upon to work with version 023 and earlier.

6 Sketches with no Library Dependencies

6.1 Simple 'Bildr' Example

This example uses the existing wire library and a bare minimum of code. The example can be found here: http://bildr.org/2011/03/ds1307-arduino/ and was first announced on 01 Mar 2011.

This sketch requires no specialist libraries and depends only on the Wire library. It is well worth following the link to the article as it contains a lot of useful information about the DS1307 including its limitations. The author states that the code is "oversimplified" but this in itself creates a very good starting point.

6.1.1 Set Time Sketch

```
//Arduino 1.0+ Only
//Arduino 1.0+ Only

#include "Wire.h"
#define DS1307_ADDRESS 0x68
byte zero = 0x00; //workaround for issue #527

void setup(){
Wire.begin();
Serial.begin(9600);
setDateTime(); //MUST CONFIGURE IN FUNCTION
}

void loop(){
printDate();
delay(1000);
}

void setDateTime(){
byte second =00; //0-59
byte minute =57; //0-59
byte hour =18; //0-23
byte weekDay = 1; //1-7 (0-6, Sunday -□ Saturday)
byte monthDay =15; //1-31
byte month = 7; //1-12
byte year= 12; //0-99 (msd's 4-digit year ignored

Wire.beginTransmission(DS1307_ADDRESS);
Wire.write(zero); //stop Oscillator
Wire.write(decToBcd(second));
Wire.write(decToBcd(minute));
Wire.write(decToBcd(hour));
Wire.write(decToBcd(weekDay));
```

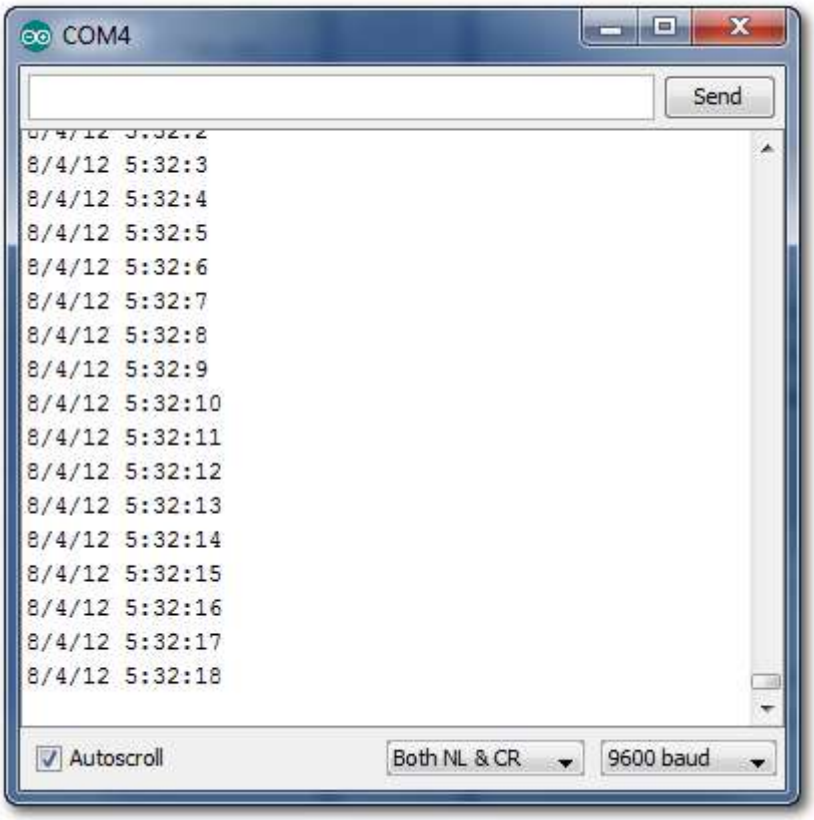
```
Wire.write(decToBcd(monthDay));
Wire.write(decToBcd(month));
Wire.write(decToBcd(year));
Wire.write(zero); //start
Wire.endTransmission();
}

byte decToBcd(byte val){
// Convert normal decimal numbers to binary coded decimal
return ( (val/10*16) + (val%10) );
}

byte bcdToDec(byte val){
// Convert binary coded decimal to normal decimal numbers
return ( (val/16*10) + (val%16) );
}

void printDate(){
// Reset the register pointer
Wire.beginTransmission(DS1307_ADDRESS);
Wire.write(zero);
Wire.endTransmission();

Wire.requestFrom(DS1307_ADDRESS, 7);

int second = bcdToDec(Wire.read());
int minute = bcdToDec(Wire.read());
int hour = bcdToDec(Wire.read() & 0b111111); //24 hour time
int weekDay = bcdToDec(Wire.read()); //0-6 -> sunday -
Saturday
int monthDay = bcdToDec(Wire.read());
int month = bcdToDec(Wire.read());
int year = bcdToDec(Wire.read());

//print the date EG 3/1/11 23:59:59
Serial.print(month);
Serial.print("/");
Serial.print(monthDay);
Serial.print("/");
Serial.print(year);
Serial.print(" ");
Serial.print(hour);
Serial.print(":");
Serial.print(minute);
Serial.print(":");
Serial.println(second);
}
```

6.1.2 Read Time Sketch

```
//Arduino 1.0+ Only
//Arduino 1.0+ Only

#include "Wire.h"
#define DS1307_ADDRESS 0x68
```

```
void setup(){
Wire.begin();
Serial.begin(9600);
}

void loop(){
printDate();
delay(1000);
}

byte bcdToDec(byte val) {
// Convert binary coded decimal to normal decimal numbers
return ( (val/16*10) + (val%16) );
}

void printDate(){

// Reset the register pointer
Wire.beginTransmission(DS1307_ADDRESS);

byte zero = 0x00;
Wire.write(zero);
Wire.endTransmission();

Wire.requestFrom(DS1307_ADDRESS, 7);

int second = bcdToDec(Wire.read());
int minute = bcdToDec(Wire.read());
int hour = bcdToDec(Wire.read() & 0b111111); //24 hour time
int weekDay = bcdToDec(Wire.read()); //0-6 -> sunday -
Saturday
int monthDay = bcdToDec(Wire.read());
int month = bcdToDec(Wire.read());
int year = bcdToDec(Wire.read());

//print the date EG 3/1/11 23:59:59
Serial.print(month);
Serial.print("/");
Serial.print(monthDay);
Serial.print("/");
Serial.print(year);
Serial.print(" ");
Serial.print(hour);
Serial.print(":");
Serial.print(minute);
Serial.print(":");
Serial.println(second);
}
```

6.1.3 Sample Output



The display of date & time can be reconfigured easily by changing the order of statements in date print section of code. The following example produces output in the form “yyyy-mm-dd hh:mm:ss”:

```
Serial.print(year);
Serial.print("-");
Serial.print(month);
Serial.print("-");
Serial.print(monthDay);
Serial.print(" ");
Serial.print(hour);
Serial.print(":");
Serial.print(minute);
Serial.print(":");
Serial.println(second);
```

6.1.4 Notes

The delay(1000) line in both the set and read programs may cause the display to skip seconds occasionally. This will happen if the program is running “fast”. A more accurate value can be determined by following the interval between skipped seconds and adjusting the delay() value. This is a crude approach because the RTC’s timekeeping is temperature dependent and the reported time WILL drift.

The ONLY way round this problem is to use interrupts.

The above code does not explain why the set time function has to be a function rather than part of the setup code: “setDateTime(); //MUST CONFIGURE IN FUNCTION”.

7 Sketches using the DS1307RTC Library

The examples provided with Arduino 1.0.1 fall into this category and need not be documented further. DS1307RTC is the latest library and should be used preferentially to many of the older libries documented below.

If your requirements are for simple “read time” and “set time” then these libraries will provide 95% of what you want 95% of the time. The example sketches are easily modified or extended and there are many applications available on the web that utilise the DS1307RTC library. A few examples are shown below:

Temperasture Logger:

http://www.airsensor.co.uk/component/zoo/item/temperature-logger-with-2-sensors.html

Seeeduino Stalker Variant:

http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1273413449

HT1632c Dot Matrix Display:

http://code.google.com/p/ht1632c/

8 Sketches using the DS1307 Library

8.1 Library Files

The DS1307 library is authored by Matt Joyce and is documented here:

https://code.google.com/p/libds1307/

The DS1307.h and DS1307.cpp files are available here :

1. http://libds1307.googlecode.com/svn-history/r3/trunk/DS1307.h
2. http://libds1307.googlecode.com/svn-history/r3/trunk/DS1307.cpp

The two files have been modified to make them compatible with Arduino IDE 1.0 and those reproduced below should be used in preference to those obtainable through the previous links:

8.1.1 DS1307.h

```
/*
  DS1307.h - library for DS1307 rtc
 */

// ensure this library description is only included once
#ifndef DS1307_h
#define DS1307_h

// include types & constants of Wiring core API
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
```

```
#include "WProgram.h"
#endif

// include types & constants of Wire ic2 lib
#include <Wire.h>

#define DS1307_SEC 0
#define DS1307_MIN 1
#define DS1307_HR 2
#define DS1307_DOW 3
#define DS1307_DATE 4
#define DS1307_MTH 5
#define DS1307_YR 6

#define DS1307_BASE_YR 2000

#define DS1307_CTRL_ID B1101000  //DS1307

// Define register bit masks
#define DS1307_CLOCKHALT B10000000

#define DS1307_LO_BCD  B00001111
#define DS1307_HI_BCD  B11110000

#define DS1307_HI_SEC  B01110000
#define DS1307_HI_MIN  B01110000
#define DS1307_HI_HR   B00110000
#define DS1307_LO_DOW  B00000111
#define DS1307_HI_DATE B00110000
#define DS1307_HI_MTH  B00110000
#define DS1307_HI_YR   B11110000

// library interface description
class DS1307
{
    // user-accessible "public" interface
public:
    DS1307();
    void get(int *, boolean);
    int get(int, boolean);
    void set(int, int);
    void start(void);
    void stop(void);

    // library-accessible "private" interface
private:
    byte rtc_bcd[7]; // used prior to read/set ds1307
registers;
    void read(void);
    void save(void);
};

extern DS1307 RTC;

#endif
```

The main modification to the h file has been to:

- At line 10 replace “#include <WConstants.h>” by “Arduino.h”
- Add the reference to WProgram.h: “#include "WProgram.h"”

8.1.2 DS1307.cpp

```
#include <Wire.h>

#include "DS1307.h"

DS1307::DS1307()
{
  Wire.begin();
}

DS1307 RTC=DS1307();

// PRIVATE FUNCTIONS

// Aquire data from the RTC chip in BCD format
// refresh the buffer
void DS1307::read(void)
{
  // use the Wire lib to connect to tho rtc
  // reset the resgiter pointer to zero
  Wire.beginTransmission(DS1307_CTRL_ID);
    Wire.write(0x00);
  Wire.endTransmission();

  // request the 7 bytes of data    (secs, min, hr, dow,
date. mth, yr)
  Wire.requestFrom(DS1307_CTRL_ID, 7);
  for(int i=0; i<7; i++)
  {
    // store data in raw bcd format
    rtc_bcd[i]=Wire.read();
  }
}

// update the data on the IC from the bcd formatted data in
the buffer
void DS1307::save(void)
{
  Wire.beginTransmission(DS1307_CTRL_ID);
  Wire.write(0x00); // reset register pointer
  for(int i=0; i<7; i++)
  {
    Wire.write(rtc_bcd[i]);
  }
  Wire.endTransmission();
}
```

```
// PUBLIC FUNCTIONS
void DS1307::get(int *rtc, boolean refresh)   // Aquire
data from buffer and convert to int, refresh buffer if
required
{
  if(refresh) read();
  for(int i=0;i<7;i++)  // cycle through each component,
create array of data
  {
 rtc[i]=get(i, 0);
  }
}

int DS1307::get(int c, boolean refresh)  // aquire
individual RTC item from buffer, return as int, refresh
buffer if required
{
  if(refresh) read();
  int v=-1;
  switch(c)
  {
  case DS1307_SEC:
    v=(10*((rtc_bcd[DS1307_SEC] &
DS1307_HI_SEC)>>4))+(rtc_bcd[DS1307_SEC] & DS1307_LO_BCD);
 break;
  case DS1307_MIN:
    v=(10*((rtc_bcd[DS1307_MIN] &
DS1307_HI_MIN)>>4))+(rtc_bcd[DS1307_MIN] & DS1307_LO_BCD);
 break;
  case DS1307_HR:
    v=(10*((rtc_bcd[DS1307_HR] &
DS1307_HI_HR)>>4))+(rtc_bcd[DS1307_HR] & DS1307_LO_BCD);
 break;
  case DS1307_DOW:
    v=rtc_bcd[DS1307_DOW] & DS1307_LO_DOW;
 break;
  case DS1307_DATE:
    v=(10*((rtc_bcd[DS1307_DATE] &
DS1307_HI_DATE)>>4))+(rtc_bcd[DS1307_DATE] &
DS1307_LO_BCD);
 break;
  case DS1307_MTH:
    v=(10*((rtc_bcd[DS1307_MTH] &
DS1307_HI_MTH)>>4))+(rtc_bcd[DS1307_MTH] & DS1307_LO_BCD);
 break;
  case DS1307_YR:
    v=(10*((rtc_bcd[DS1307_YR] &
DS1307_HI_YR)>>4))+(rtc_bcd[DS1307_YR] &
DS1307_LO_BCD)+DS1307_BASE_YR;
 break;
  } // end switch
  return v;
}
```

```
void DS1307::set(int c, int v)  // Update buffer, then
update the chip
{
  switch(c)
  {
  case DS1307_SEC:
    if(v<60 && v>-1)
    {
 //preserve existing clock state (running/stopped)
 int state=rtc_bcd[DS1307_SEC] & DS1307_CLOCKHALT;
 rtc_bcd[DS1307_SEC]=state | ((v / 10)<<4) + (v % 10);
    }
    break;
  case DS1307_MIN:
    if(v<60 && v>-1)
    {
 rtc_bcd[DS1307_MIN]=((v / 10)<<4) + (v % 10);
    }
    break;
  case DS1307_HR:
  // TODO : AM/PM  12HR/24HR
    if(v<24 && v>-1)
    {
 rtc_bcd[DS1307_HR]=((v / 10)<<4) + (v % 10);
    }
    break;
  case DS1307_DOW:
    if(v<8 && v>-1)
    {
 rtc_bcd[DS1307_DOW]=v;
    }
    break;
  case DS1307_DATE:
    if(v<32 && v>-1)
    {
 rtc_bcd[DS1307_DATE]=((v / 10)<<4) + (v % 10);
    }
    break;
  case DS1307_MTH:
    if(v<13 && v>-1)
    {
 rtc_bcd[DS1307_MTH]=((v / 10)<<4) + (v % 10);
    }
    break;
  case DS1307_YR:
    if(v<99 && v>-1)   // we are OK up to 2099
    {
 rtc_bcd[DS1307_YR]=((v / 10)<<4) + (v % 10);
    }
    break;
  } // end switch
  save();
}

void DS1307::stop(void)
```

```
{
 // set the ClockHalt bit high to stop the rtc
 // this bit is part of the seconds byte
    rtc_bcd[DS1307_SEC]=rtc_bcd[DS1307_SEC] |
DS1307_CLOCKHALT;
    save();
}

void DS1307::start(void)
{
 // unset the ClockHalt bit to start the rtc
 // TODO : preserve existing seconds
    rtc_bcd[DS1307_SEC]=0;
 save();
}
```

The main modification to the cpp file has been to:

- Change the reference to “extern "C" {#include <../Wire/Wire.h>}”
- Replace “Wire.send” with “Wire.write”
- Replace “Wire.receive” with “Wire.read”
- Correct the conditional at line 120 by replacing “if(v<32 && v>-1)” by “if(v<31 && v>-1)”

8.2 Sample Sketch using DS1307

8.2.1 DS1307.ino

This simple sketch hard-codes the date-time values to be sent to the Arduino. All this happens in the setup routine, ensuring it happens only once. Thereafter control passes to loop where the RTC date-time values are read and printed to the Monitor.

The following parameters are set

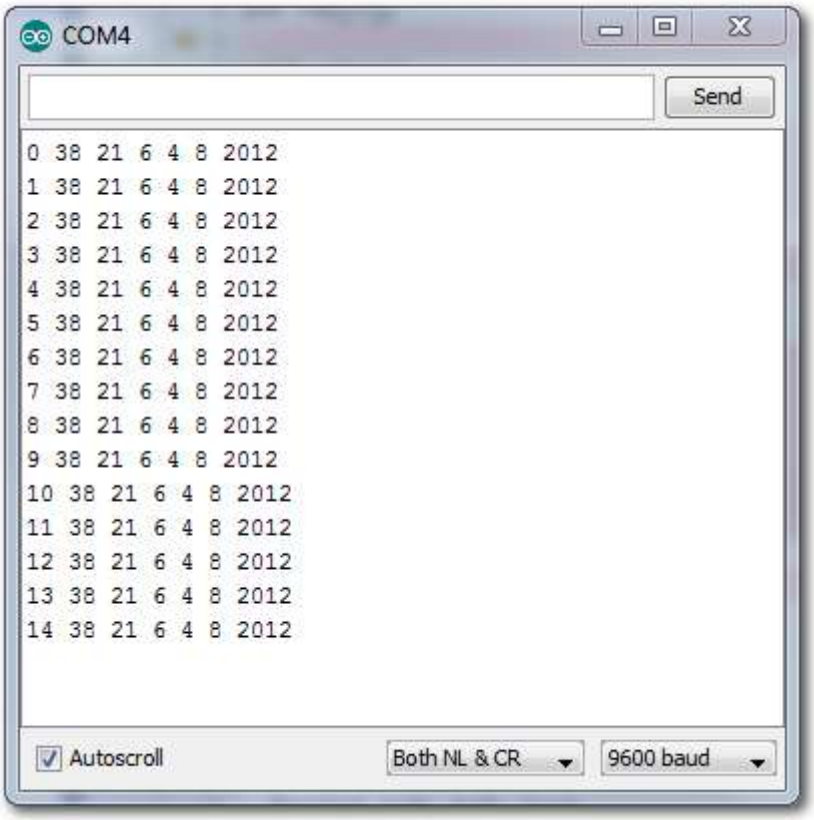
```
#include <Wire.h>
#include <DS1307.h>

int rtc[7];

void setup()
{
  Serial.begin(9600);
  // un-comment following to set the time...
  // test values will give
/*
  RTC.stop();
  RTC.set(DS1307_SEC,1);          // seconds         (0-59)
  RTC.set(DS1307_MIN,38);         // minutes         (0-59)
  RTC.set(DS1307_HR,21);          // hours           (0-23)
  RTC.set(DS1307_DOW,6);          // Day of the week (0-6)
  RTC.set(DS1307_DATE,4);         // Day number      (1-31)
  RTC.set(DS1307_MTH,8);          // Month Number    (1-12)
  RTC.set(DS1307_YR,12);          // Year number     (2-
digit)
```

```
  RTC.start();
*/
}

void loop()
{
  RTC.get(rtc,true);

  for(int i=0; i<7; i++)
  {
    Serial.print(rtc[i]);
    Serial.print(" ");
  }
  Serial.println();

  delay(1000);

}
```

The set parameters require explanation:

1. The Day of the Week (DOW) MUST be set by this method (other methods documented here do not require this parameter to be set). The reason is that 7 arguments are expected – a feature that can be seen in the read section where a simple loop using 0-6 (i.e., “i<7”) is used.
2. For DOW, 0=Sunday even though the set program will accept any value. (To be verified: In fact I think almost any definition can be used because later examples use the case where Sunday=1; Saturday=7. Go figure…).
3. The Day number should be in the interval 1-31. A bug in an earlier version did not allow the 31st of a month to be set.
4. The year should be specified as two digits only.

8.2.2 Sample Output



The output sequence is the same as the SET sequence of arguments.

8.2.3 Notes

The time is reset every time the code runs (power applied or reset button pressed or new Monitor window is opened). Great for testing if the RTC works.

This can be used to set the correct date and time if you anticipate the length of time to send the sketch and set the number of seconds that far ahead.

At the time of writing the migration to version 1.0+ proved problematic. A post has been initiated on the Arduino forum: (http://arduino.cc/forum/index.php/topic,117202.0.html) Thanks and acknowledgement to “Riva” on that forum for the eventual solution.

9 Sketches using the SoftDS1307RTC Library

This new library was announced in January 2011 here:

http://arduino.cc/forum/index.php?action=printpage;topic=38068.0

The various links provided in that announcement are reproduced below:

1. *Software library:*

   https://github.com/kamermans/SoftDS1307RTC

2. *There is also a compact RTC design that I've created:*

https://github.com/kamermans/SoftDS1307RTC/raw/master/1307_Front.JPG

3. *I put an SMD crystal on the back to ensure very accurate timing:*

   https://github.com/kamermans/SoftDS1307RTC/raw/master/1307_Back.JPG

The first link takes you to a bunch of files including images of the board, the circuit used, etc. It is a good starting point.

9.1 Library Files

In the Arduino libraries folder create a new folder called "SoftDS1307RTC". Copy SoftDS1307RTC.cpp and SoftDS1307RTC.h into that folder. Above these links on the web page is an "Examples" folder: https://github.com/kamermans/SoftDS1307RTC/tree/master/Examples

These are the sample sketch files to get you started and documented further down. The libraries are somewhat deprecated so this soft library is included for completeness only. The header file references the deprecated WProgram.h. Additional libraries (both deprecated bearing dates as far back as 2009) are required:

1. TwoWireBase
2. SoftI2cMaster

These can be found here:
http://code.google.com/p/neuroelec/source/browse/trunk/SoftI2cMaster/?r=2

A folder "SoftI2cMaster" should be created within the Arduino "libraries" folder and the corresponding files (3 in total) copied into that folder. Note that the code shown here for those three files:

1. SoftI2CMaster.cpp
2. SoftI2cMaster.h
3. TwoWireBase.h

…has been modified for compatibility with version 1.0.1 of the IDE. Accordingly the sample library files documented below should be used.

9.1.1 SoftDS1307RTC.h

```
/*
* SoftDS1307RTC.h - library for DS1307 RTC using Software
I2c
* This library is intended to be uses with Arduino Time.h
library functions
*/
#ifndef DS1307RTC_h
#define DS1307RTC_h
#include <TwoWireBase.h>
#include <SoftI2cMaster.h>
#include <Time.h>
// library interface description
class SoftDS1307RTC {
// user-accessible "public" interface
public:
SoftDS1307RTC();
```

```
static time_t get();
static void set(time_t t);
static void read(tmElements_t &tm);
static void write(tmElements_t &tm);
private:
static SoftI2cMaster _i2c;
static uint8_t dec2bcd(uint8_t num);
static uint8_t bcd2dec(uint8_t num);
};
extern SoftDS1307RTC RTC;
#endif
```

9.1.2 SoftDS1307RTC.cpp

```
/*
* SoftDS1307RTC.cpp - library for DS1307 RTC using Software
I2c
Software I2c based DS1307 RTC library by Steve Kamerman
2010
Based almost entirely on Michael Margolis' 12/30/2009
library "DS1307RTC"

Copyright (c) Steve Kamerman 2010
This library is intended to be uses with Arduino Time.h
library functions

The library is free software; you can redistribute it
and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation;
either
version 2.1 of the License, or (at your option) any later
version.

This library is distributed in the hope that it will be
useful,
but WITHOUT ANY WARRANTY; without even the implied warranty
of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public
License along with this library; if not, write to the Free
Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301 USA
30 Dec 2010 - Modified by Steve Kamerman to support
SoftI2cMaster
*/

#include "SoftDS1307RTC.h"

#define DS1307_CTRL_ID 0xD0
```

```
#ifndef SCL_PIN
#define SCL_PIN 21
#endif

#ifndef SDA_PIN
#define SDA_PIN 20
#endif

SoftI2cMaster SoftDS1307RTC::_i2c;

SoftDS1307RTC::SoftDS1307RTC()
{
  _i2c.init(SCL_PIN, SDA_PIN);
}

// PUBLIC FUNCTIONS
time_t SoftDS1307RTC::get() // Aquire data from buffer and
convert to time_t
{
  tmElements_t tm;
  read(tm);
  return(makeTime(tm));
}

void SoftDS1307RTC::set(time_t t)
{
  tmElements_t tm;
  breakTime(t, tm);
  tm.Second |= 0x80; // stop the clock
  write(tm);
  tm.Second &= 0x7f; // start the clock
  write(tm);
}

// Aquire data from the RTC chip in BCD format
void SoftDS1307RTC::read( tmElements_t &tm)
{
  _i2c.start(DS1307_CTRL_ID | I2C_WRITE);
  _i2c.write(0x00);
  _i2c.restart(DS1307_CTRL_ID | I2C_READ);

  tm.Second = bcd2dec(_i2c.read(false));
  tm.Minute = bcd2dec(_i2c.read(false));
  tm.Hour = bcd2dec(_i2c.read(false));
  tm.Wday = bcd2dec(_i2c.read(false));
  tm.Day = bcd2dec(_i2c.read(false));
  tm.Month = bcd2dec(_i2c.read(false));
  tm.Year = y2kYearToTm((bcd2dec(_i2c.read(true))));

  _i2c.stop();
}

void SoftDS1307RTC::write(tmElements_t &tm)
{
```

```
  _i2c.start(DS1307_CTRL_ID | I2C_WRITE);
  _i2c.write(0x00);

  _i2c.write(dec2bcd(tm.Second));
  _i2c.write(dec2bcd(tm.Minute));
  _i2c.write(dec2bcd(tm.Hour));
  _i2c.write(dec2bcd(tm.Wday));
  _i2c.write(dec2bcd(tm.Day));
  _i2c.write(dec2bcd(tm.Month));
  _i2c.write(dec2bcd(tmYearToY2k(tm.Year)));

  _i2c.stop();
}
// PRIVATE FUNCTIONS

// Convert Decimal to Binary Coded Decimal (BCD)
uint8_t SoftDS1307RTC::dec2bcd(uint8_t num)
{
  return ((num/10 * 16) + (num % 10));
}

// Convert Binary Coded Decimal (BCD) to Decimal
uint8_t SoftDS1307RTC::bcd2dec(uint8_t num)
{
  return ((num/16 * 10) + (num % 16));
}
SoftDS1307RTC RTC = SoftDS1307RTC(); // create an instance
for the user
```

9.1.3 SoftI2cMaster.h

```
/* Arduino SoftI2cMaster Library
 * Copyright (C) 2009 by William Greiman
 *
 * This file is part of the Arduino SoftI2cMaster Library
 *
 * This Library is free software: you can redistribute it
and/or modify
 * it under the terms of the GNU General Public License as
published by
 * the Free Software Foundation, either version 3 of the
License, or
 * (at your option) any later version.
 *
 * This Library is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
```

```
 * along with the Arduino SoftI2cMaster Library.  If not,
see
 * <http://www.gnu.org/licenses/>.
 */
#ifndef SOFT_I2C_MASTER
#define SOFT_I2C_MASTER
#include <TwoWireBase.h>

// delay used to tweek signals
#define I2C_DELAY_USEC 10

class SoftI2cMaster : public TwoWireBase {
  uint8_t sclPin_;
  uint8_t sdaPin_;
public:
  /** init bus */
  void init(uint8_t sclPin, uint8_t sdaPin);

  /** read a byte and send Ack if last is false else Nak to
terminate read */
  uint8_t read(uint8_t last);

  /** send new address and read/write bit without stop */
  uint8_t restart(uint8_t addressRW);

  /** issue a start condition for i2c address with
read/write bit */
  uint8_t start(uint8_t addressRW);

  /** issue a stop condition */
  void stop(void);

  /** write byte and return true for Ack or false for Nak
*/
  uint8_t write(uint8_t b);

  /** write byte and return true for Ack or false for Nak
*/
  uint8_t ldacwrite(uint8_t b, uint8_t);

};
#endif //SOFT_I2C_MASTER
```

9.1.4 SoftI2cMaster.cpp

```
/* Arduino SoftI2cMaster Library
 * Copyright (C) 2009 by William Greiman
 *
 * This file is part of the Arduino SoftI2cMaster Library
 *
 * This Library is free software: you can redistribute it
and/or modify
 * it under the terms of the GNU General Public License as
published by
```

```
 * the Free Software Foundation, either version 3 of the
License, or
 * (at your option) any later version.
 *
 * This Library is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
 * along with the Arduino SoftI2cMaster Library.  If not,
see
 * <http://www.gnu.org/licenses/>.
 */
#include "SoftI2cMaster.h"
//------------------------------------------------------------
----------------------
// WARNING don't change anything unless you verify the
change with a scope
//------------------------------------------------------------
----------------------
// init pins and set bus high
void SoftI2cMaster::init(uint8_t sclPin, uint8_t sdaPin)
{
  sclPin_ = sclPin;
  sdaPin_ = sdaPin;
  pinMode(sclPin_, OUTPUT);
  pinMode(sdaPin_, OUTPUT);
  digitalWrite(sclPin_, HIGH);
  digitalWrite(sdaPin_, HIGH);
}
//------------------------------------------------------------
----------------------
// read a byte and send Ack if last is false else Nak to
terminate read
uint8_t SoftI2cMaster::read(uint8_t last)
{
  uint8_t b = 0;
  // make sure pullup enabled
  digitalWrite(sdaPin_, HIGH);
  pinMode(sdaPin_, INPUT);
  // read byte
  for (uint8_t i = 0; i < 8; i++) {
    // don't change this loop unless you verify the change
with a scope
    b <<= 1;
    delayMicroseconds(I2C_DELAY_USEC);
    digitalWrite(sclPin_, HIGH);
    if (digitalRead(sdaPin_)) b |= 1;
    digitalWrite(sclPin_, LOW);
  }
```

```
  // send Ack or Nak
  pinMode(sdaPin_, OUTPUT);
  digitalWrite(sdaPin_, last);
  digitalWrite(sclPin_, HIGH);
  delayMicroseconds(I2C_DELAY_USEC);
  digitalWrite(sclPin_, LOW);
  digitalWrite(sdaPin_, HIGH);
  return b;
}
//------------------------------------------------------------
---------------------
// send new address and read/write without stop
uint8_t SoftI2cMaster::restart(uint8_t addressRW)
{
  digitalWrite(sclPin_, HIGH);
  return start(addressRW);
}
//------------------------------------------------------------
---------------------
// issue a start condition for i2c address with read/write
bit
uint8_t SoftI2cMaster::start(uint8_t addressRW)
{
  digitalWrite(sdaPin_, LOW);
  delayMicroseconds(I2C_DELAY_USEC);
  digitalWrite(sclPin_, LOW);
  return write(addressRW);
}
//------------------------------------------------------------
---------------------
// issue a stop condition
void SoftI2cMaster::stop(void)
{
  delayMicroseconds(I2C_DELAY_USEC);
  digitalWrite(sclPin_, HIGH);
  delayMicroseconds(I2C_DELAY_USEC);
  digitalWrite(sdaPin_, HIGH);
  delayMicroseconds(I2C_DELAY_USEC);
}
//------------------------------------------------------------
---------------------
// write byte and return true for Ack or false for Nak
uint8_t SoftI2cMaster::write(uint8_t b)
{
  // write byte
  for (uint8_t m = 0X80; m != 0; m >>= 1) {
    // don't change this loop unless you verify the change
with a scope
    digitalWrite(sdaPin_, m & b);
    digitalWrite(sclPin_, HIGH);
    delayMicroseconds(I2C_DELAY_USEC);
    digitalWrite(sclPin_, LOW);
  }
  // get Ack or Nak
  digitalWrite(sdaPin_, HIGH);
```

```
  pinMode(sdaPin_, INPUT);
  digitalWrite(sclPin_, HIGH);
  b = digitalRead(sdaPin_);
  digitalWrite(sclPin_, LOW);
  pinMode(sdaPin_, OUTPUT);
  return b == 0;
}

//------------------------------------------------------------
---------------------
// write byte and return true for Ack or false for Nak
uint8_t SoftI2cMaster::ldacwrite(uint8_t b, uint8_t
ldacpin)
{
  // write byte
  for (uint8_t m = 0X80; m != 0; m >>= 1) {
    // don't change this loop unless you verify the change
with a scope
    digitalWrite(sdaPin_, m & b);
    digitalWrite(sclPin_, HIGH);
    delayMicroseconds(I2C_DELAY_USEC);
    digitalWrite(sclPin_, LOW);
  }
  // get Ack or Nak
  digitalWrite(sdaPin_, HIGH);
  digitalWrite(ldacpin, LOW);
  pinMode(sdaPin_, INPUT);
  digitalWrite(sclPin_, HIGH);
  b = digitalRead(sdaPin_);
  digitalWrite(sclPin_, LOW);
  pinMode(sdaPin_, OUTPUT);
  return b == 0;
}
```

9.1.5 TwoWireBase.h

```
/* Arduino SoftI2cMaster and TwiMaster Libraries
 * Copyright (C) 2009 by William Greiman
 *
 * This file is part of the Arduino SoftI2cMaster and
TwiMaster Libraries
 *
 * This Library is free software: you can redistribute it
and/or modify
 * it under the terms of the GNU General Public License as
published by
 * the Free Software Foundation, either version 3 of the
License, or
 * (at your option) any later version.
 *
 * This Library is distributed in the hope that it will be
useful,
 * but WITHOUT ANY WARRANTY; without even the implied
warranty of
```

```
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
Public License
 * along with the Arduino SoftI2cMaster and TwiMaster
Libraries.
 * If not, see <http://www.gnu.org/licenses/>.
 */
#ifndef TWO_WIRE_BASE_H
#define TWO_WIRE_BASE_H

// include types & constants of Wiring core API
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

// R/W direction bit to OR with address for start or
restart
#define I2C_READ 1
#define I2C_WRITE 0

class TwoWireBase {
public:
  /** read a byte and send Ack if last is false else Nak to
terminate read */
  virtual uint8_t read(uint8_t last) = 0;

  /** send new address and read/write bit without stop */
  virtual uint8_t restart(uint8_t addressRW) = 0;

  /** issue a start condition for i2c address with
read/write bit */
  virtual uint8_t start(uint8_t addressRW) = 0;

  /** issue a stop condition */
  virtual void stop(void) = 0;

  /** write byte and return true for Ack or false for Nak
*/
  virtual uint8_t write(uint8_t data) = 0;
};
#endif // TWO_WIRE_BASE_H
```

Note that the SoftI2CMaster libraries perform correct handshaking with the I2C bus: hence the directives not to change any values without an oscilloscope. The timings are critical for the transmission of data.

9.2 Sample Sketches using SoftDS1307RTC

9.2.1 Set Time using SoftDS1307RTC

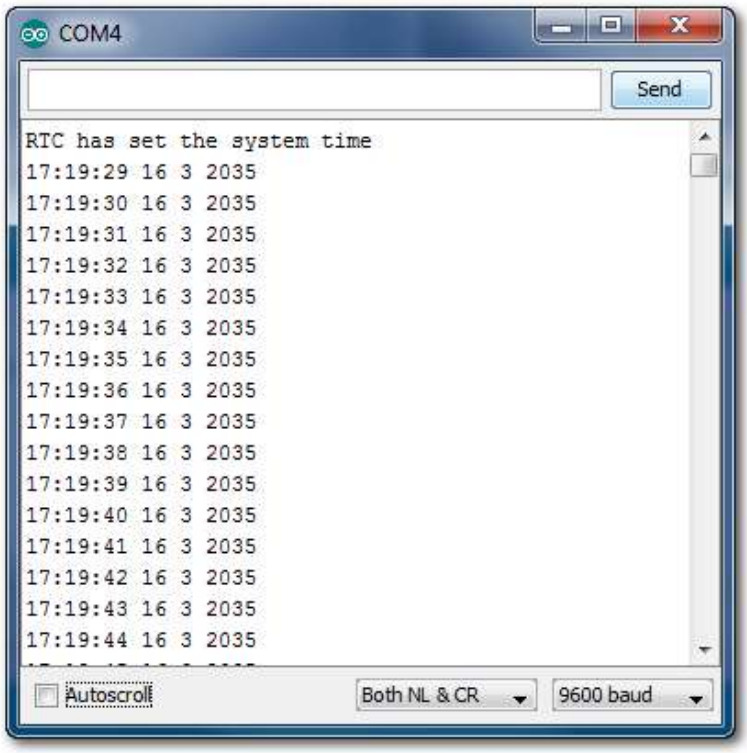
Here is the set time sketch:

```
/*
* TimeRTCSet.ino
* example code illustrating Time library with Real Time
Clock.
*
* RTC clock is set in response to serial port time message
* A Processing example sketch to set the time is inclided
in the download
*/

#include <Time.h>
#include <TwoWireBase.h>
#include <SoftI2cMaster.h>
#include <SoftDS1307RTC.h> // a basic DS1307 library that
returns time as a time_t

#define SDA_PIN 20    // 20 on the Mega 2560
#define SCL_PIN 21    // 21 on the Mega 2560

void setup() {
  Serial.begin(9600);
  setSyncProvider(RTC.get); // the function to get the time
from the RTC
  if(timeStatus()!= timeSet)
     Serial.println("Unable to sync with the RTC");
  else
     Serial.println("RTC has set the system time");
}

void loop()
{
  if(Serial.available())
  {
     time_t t = processSyncMessage();
     if(t >0)
     {
        RTC.set(t); // set the RTC and the system time to
the received value
        setTime(t);
     }
  }
   digitalClockDisplay();
   delay(1000);
}

void digitalClockDisplay(){
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
```

```
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){
  // utility function for digital clock display: prints
preceding colon and leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

/* code to process time sync messages from the serial port
*/
#define TIME_MSG_LEN 11 // time sync to PC is HEADER
followed by unix time_t as ten ascii digits
#define TIME_HEADER 'T' // Header tag for serial time sync
message

time_t processSyncMessage() {
  // return the time if a valid sync message is received on
the serial port.
  while(Serial.available() >= TIME_MSG_LEN ){ // time
message consists of a header and ten ascii digits
    char c = Serial.read() ;
    Serial.print(c);
    if( c == TIME_HEADER ) {
      time_t pctime = 0;
      for(int i=0; i < TIME_MSG_LEN -1; i++){
        c = Serial.read();
        if( c >= '0' && c <= '9'){
          pctime = (10 * pctime) + (c - '0') ; // convert
digits to a number
        }
      }
      return pctime;
    }
  }
  return 0;
}
```

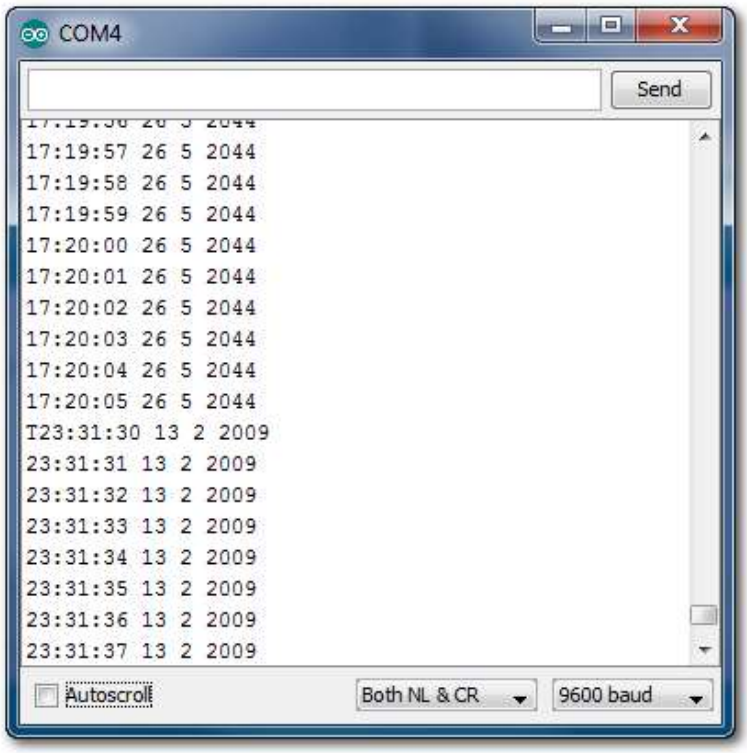
9.2.1.1 Sample Output of the SET Sketch

This screen shot shows the default start-up:



Note the indeterminate date-time values.

This screen capture shows the results of sending the time string in the example shown in the Notes, below:



Setting the time is achieved by sending a header followed by 10 numbers. In this example:

```
T1234567890
```

…is sent via the serial monitor.

The resulting time is:

```
23:31:30 13/02/2009
```

9.2.1.2 Notes

Every time the set sketch runs or the Arduino is reset the SET function kicks in and changes the RTC date-time. In the absence of a valid time string header, the set routine leaves the RTC is an indeterminate state.

This means that even after setting the time, a reset or a power-down destroys that time and the output becomes indeterminate. This happens even if the Serial Monitor is closed and then re-opened without downloading a sketch or powering down. For this reason alone, the sketch is best avoided.

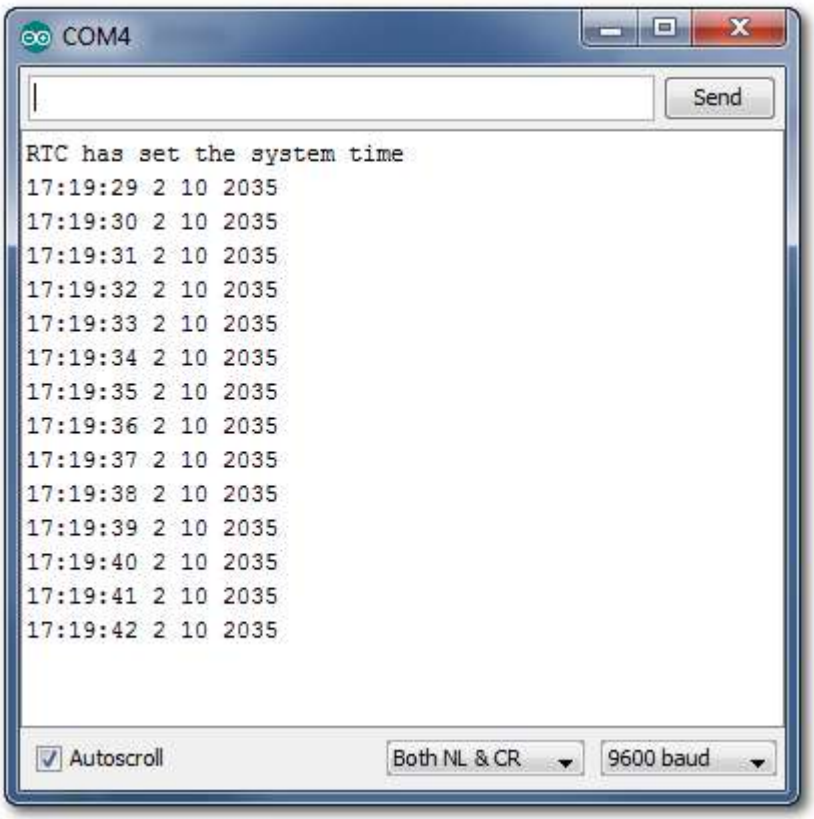
9.2.2 Read Time using SoftDS1307RTC

Here is the read sketch:

```
/*
* TimeRTC.pde
* example code illustrating Time library with Real Time
Clock.
*
*/

#include <Time.h>
#include <TwoWireBase.h>
#include <SoftI2cMaster.h>
#include <SoftDS1307RTC.h> // a basic DS1307 library that
returns time as a time_t

//#define SDA_PIN 10    // Pin 20 on Mega 2560
//#define SCL_PIN 11    // Pin 21 on Mega 2560
#define SDA_PIN 20
#define SCL_PIN 21

void setup() {
  Serial.begin(9600);
  setSyncProvider(RTC.get); // the function to get the time
from the RTC
  if(timeStatus()!= timeSet)
     Serial.println("Unable to sync with the RTC");
  else
     Serial.println("RTC has set the system time");
}

void loop()
{
   digitalClockDisplay();
   delay(1000);
}

void digitalClockDisplay(){
```

```
  // digital clock display of the time
  Serial.print(hour());
  printDigits(minute());
  printDigits(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}

void printDigits(int digits){
  // utility function for digital clock display: prints
preceding colon and leading 0
  Serial.print(":");
  if(digits < 10)
    Serial.print('0');
  Serial.print(digits);
}
```

9.2.2.1 Sample Output



9.2.2.2 Notes

The read example could never be tested because each time the serial monitor window was closed the current date-time (set by the TimeRTCSet.ino sketch) was destroyed. The output shown above is no different from the output of the set sketch.

Of all the libraries tested and sample sketches provided the SoftDS1307RTC was the most problematic, required the greatest amount of effoirt to get working and uses what is now deprecated code.

Given the advantages of other libraries and examples I cannot recommend this one.

10 Sketches using the RTCLib Library

This is available here:

https://github.com/adafruit/RTClib

The code samples and associated project are from Adafruit Industries, aka “ladyada”. The associated tutorial has been referred to before and can be found here:

http://www.ladyada.net/learn/breakoutplus/ds1307rtc.html

The original code appears to have been written by Jeelabs:

http://news.jeelabs.org/code/

10.1 Library Files

10.1.1 RTCLib.h

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

// 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 char* date, const char* 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 dayOfWeek() 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;

protected:
    uint8_t yOff, m, d, hh, mm, ss;
};
```

```
// RTC based on the DS1307 chip connected via I2C and the
Wire library
class RTC_DS1307 {
public:
  static uint8_t begin(void);
    static void adjust(const DateTime& dt);
    uint8_t isrunning(void);
    static DateTime now();
};

// 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;
};
```

10.1.2 RTCLib.cpp

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

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

#define DS1307_ADDRESS 0x68
#define SECONDS_PER_DAY 86400L

#define SECONDS_FROM_1970_TO_2000 946684800

#if (ARDUINO >= 100)
 #include <Arduino.h> // capital A so it is error prone on
case-sensitive filesystems
#else
 #include <WProgram.h>
#endif

int i = 0; //The new wire library needs to take an int when
you are sending for the zero register
////////////////////////////////////////////////////////////
////////////////////
// utility code, some of this could be exposed in the
DateTime API if needed
```

```
const uint8_t daysInMonth [] PROGMEM = {
31,28,31,30,31,30,31,31,30,31,30,31 }; //has to be const or
compiler compaints

// 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 = date2days(yOff, m, d);
    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;
}


#if (ARDUINO >= 100)

uint8_t RTC_DS1307::isrunning(void) {
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(i);
  Wire.endTransmission();

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

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

DateTime RTC_DS1307::now() {
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.write(i);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_ADDRESS, 7);
```

```
  uint8_t ss = bcd2bin(Wire.read() & 0x7F);
  uint8_t mm = bcd2bin(Wire.read());
  uint8_t hh = bcd2bin(Wire.read());
  Wire.read();
  uint8_t d = bcd2bin(Wire.read());
  uint8_t m = bcd2bin(Wire.read());
  uint16_t y = bcd2bin(Wire.read()) + 2000;

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

#else

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

DateTime RTC_DS1307::now() {
  Wire.beginTransmission(DS1307_ADDRESS);
  Wire.send(i);
  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);
}

#endif
```

```
////////////////////////////////////////////////////////////////
//////////////////////
// RTC_Millis implementation

long RTC_Millis::offset = 0;

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

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

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

10.2 Sample Sketches) using RTCLib

The repository at github has three sample sketches, each of which is documented below.

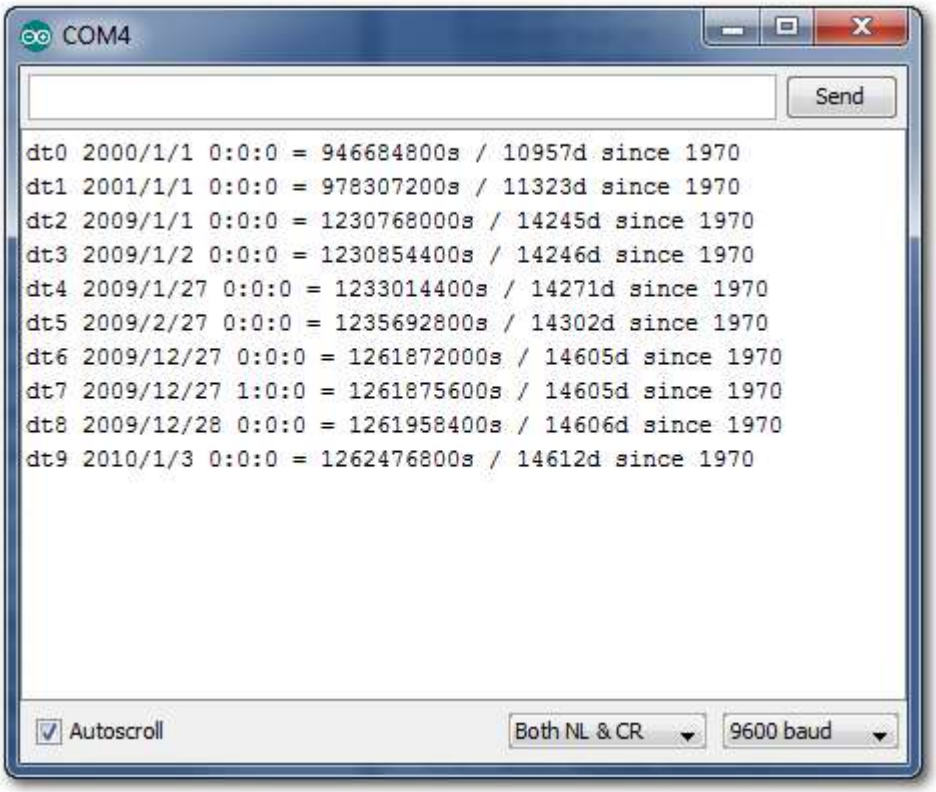
10.2.1 datecalc.ino

This sketch demonstrates the conversion of, and calculations with, date-time strings

```
// Simple date conversions and calculations

#include <Wire.h>
#include "RTClib.h"

void showDate(const char* txt, const DateTime& dt) {
    Serial.print(txt);
    Serial.print(' ');
    Serial.print(dt.year(), DEC);
    Serial.print('/');
    Serial.print(dt.month(), DEC);
    Serial.print('/');
    Serial.print(dt.day(), DEC);
    Serial.print(' ');
    Serial.print(dt.hour(), DEC);
    Serial.print(':');
    Serial.print(dt.minute(), DEC);
    Serial.print(':');
    Serial.print(dt.second(), DEC);

    Serial.print(" = ");
    Serial.print(dt.unixtime());
    Serial.print("s / ");
    Serial.print(dt.unixtime() / 86400L);
    Serial.print("d since 1970");

    Serial.println();
}
```

```
void setup () {
    Serial.begin(57600);

    DateTime dt0 (0, 1, 1, 0, 0, 0);
    showDate("dt0", dt0);

    DateTime dt1 (1, 1, 1, 0, 0, 0);
    showDate("dt1", dt1);

    DateTime dt2 (2009, 1, 1, 0, 0, 0);
    showDate("dt2", dt2);

    DateTime dt3 (2009, 1, 2, 0, 0, 0);
    showDate("dt3", dt3);

    DateTime dt4 (2009, 1, 27, 0, 0, 0);
    showDate("dt4", dt4);

    DateTime dt5 (2009, 2, 27, 0, 0, 0);
    showDate("dt5", dt5);

    DateTime dt6 (2009, 12, 27, 0, 0, 0);
    showDate("dt6", dt6);

    DateTime dt7 (dt6.unixtime() + 3600); // one hour later
    showDate("dt7", dt7);

    DateTime dt8 (dt6.unixtime() + 86400L); // one day
later
    showDate("dt8", dt8);

    DateTime dt9 (dt6.unixtime() + 7 * 86400L); // one week
later
    showDate("dt9", dt9);
}

void loop () {
}
```

10.2.1.1 Sample Output



10.2.1.2 Notes

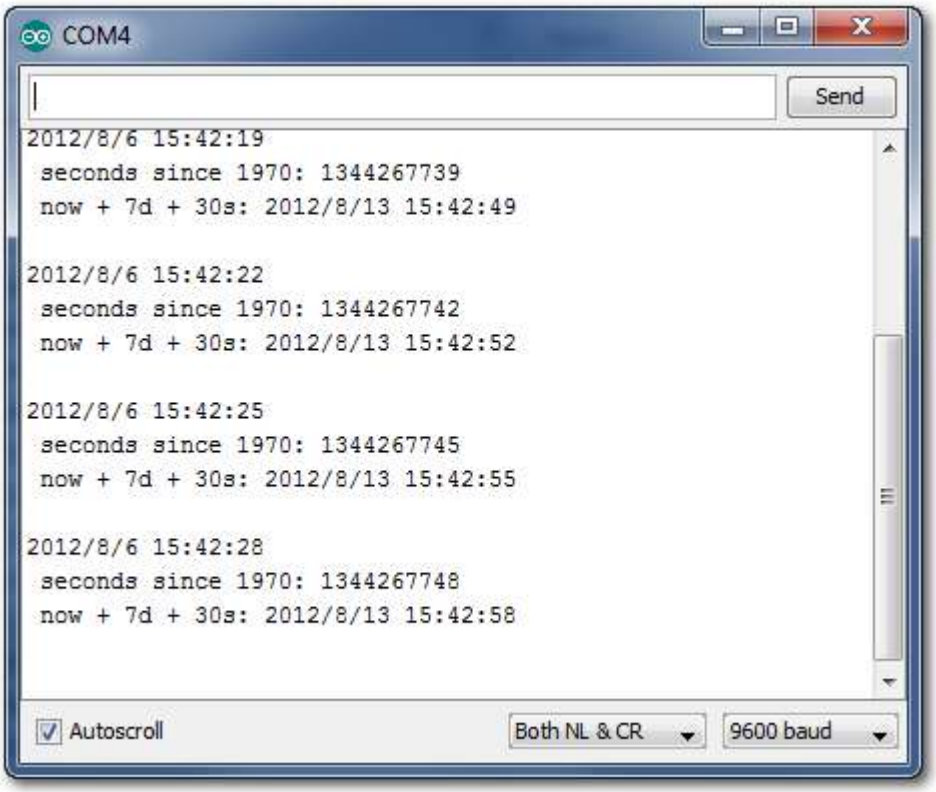
Well, it's very good news: it works straight out of the box with IDE 1.0.1

10.2.2 softrtc.ino

```
// Date and time functions using just software, based on
millis() & timer

#include <Wire.h>
#include "RTClib.h"

RTC_Millis RTC;

void setup () {
    Serial.begin(57600);
    // following line sets the RTC to the date & time this
sketch was compiled
    RTC.begin(DateTime(__DATE__, __TIME__));
}

void loop () {
    DateTime now = RTC.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
```

```
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.print(" seconds since 1970: ");
    Serial.println(now.unixtime());

    // calculate a date which is 7 days and 30 seconds into
the future
    DateTime future (now.unixtime() + 7 * 86400L + 30);

    Serial.print(" now + 7d + 30s: ");
    Serial.print(future.year(), DEC);
    Serial.print('/');
    Serial.print(future.month(), DEC);
    Serial.print('/');
    Serial.print(future.day(), DEC);
    Serial.print(' ');
    Serial.print(future.hour(), DEC);
    Serial.print(':');
    Serial.print(future.minute(), DEC);
    Serial.print(':');
    Serial.print(future.second(), DEC);
    Serial.println();

    Serial.println();
    delay(3000);
}
```

10.2.2.1 Sample Output



10.2.2.2 Notes

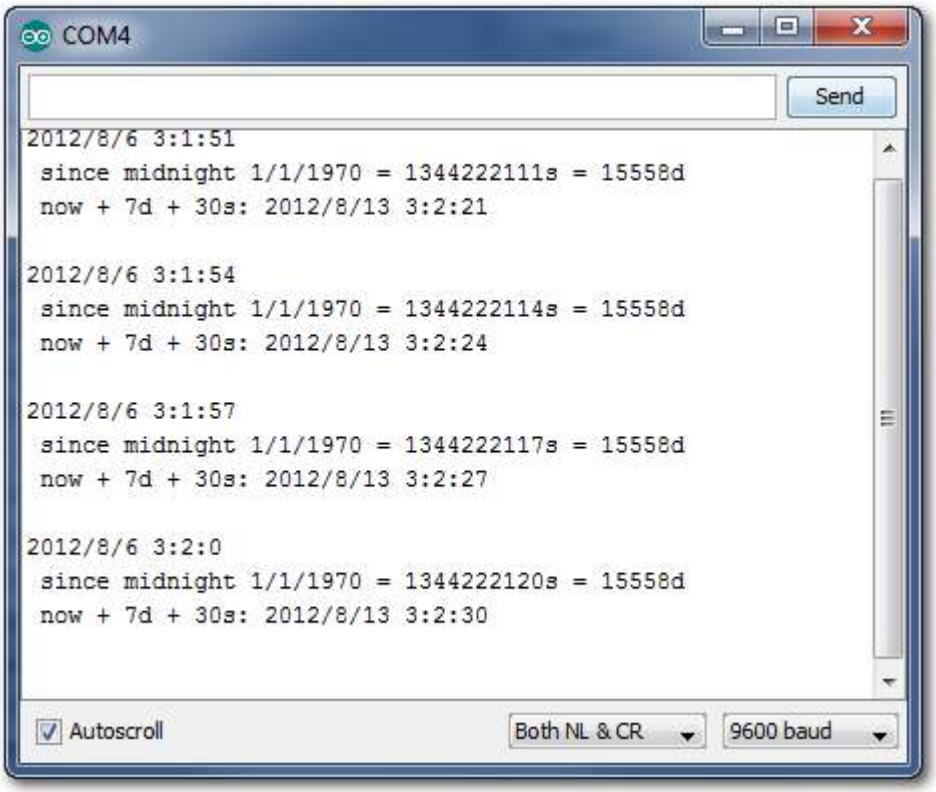
Again, it's very good news: it works straight out of the box with IDE 1.0.1

10.2.3 DS1307.ino

```
// Date and time functions using a DS1307 RTC connected via
I2C and Wire lib

#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
    Serial.begin(9600);
    Wire.begin();
    RTC.begin();

  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this
sketch was compiled
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
```

```
    DateTime now = RTC.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.print(" since midnight 1/1/1970 = ");
    Serial.print(now.unixtime());
    Serial.print("s = ");
    Serial.print(now.unixtime() / 86400L);
    Serial.println("d");

    // calculate a date which is 7 days and 30 seconds into
the future
    DateTime future (now.unixtime() + 7 * 86400L + 30);

    Serial.print(" now + 7d + 30s: ");
    Serial.print(future.year(), DEC);
    Serial.print('/');
    Serial.print(future.month(), DEC);
    Serial.print('/');
    Serial.print(future.day(), DEC);
    Serial.print(' ');
    Serial.print(future.hour(), DEC);
    Serial.print(':');
    Serial.print(future.minute(), DEC);
    Serial.print(':');
    Serial.print(future.second(), DEC);
    Serial.println();

    Serial.println();
    delay(3000);
}
```

10.2.3.1 Sample Output



10.2.3.2 Notes

Finally, it's very good news again: it works straight out of the box with IDE 1.0.1

11 Sketches using the DS1307RTCnew Library

The DFRobot site is a mine of useful information. One of the projects listed there is a Real Time Clock using the DS1307 clock chip. Here is the link:

http://www.dfrobot.com/wiki/index.php?title=Real_Time_Clock_Module_%28DS1307%29_%28SKU:DFR0151%29

The documentation and explanations are excellent. The project also starts from the point of building your own PCB rather than buying a ready-made one as I did. Note that the connections are not for the Mega 2560; for that use the connections documented at the beginning of this review.

I use these library files a lot and the "Monitor" sketch is brilliant for determining the contents of the NVRam and for writing specific values to specific NVRAM addresses. It also provides one of the best ways (perhaps that should read "most convenient ways") I have seen for setting the RTC current time.

After trying many of the other libraries and sample sketches I felt this library and associated utility sketches was the most reliable and consistent. It is clear the authors have put in a huge amount of work and I am left with the feeling that they have done an extremely good job.

11.1 Library Files

11.1.1 DS1307new.h

```
//
############################################################
##################
// #
// # Scriptname : DS1307new.h
// # Author     : Peter Schmelzer
// # Contributor: Oliver Kraus
// # contact    : schmelle2@googlemail.com
// # Date       : 2010-11-01
// # Version    : 0.2
// # License    : cc-by-sa-3.0
// #
// # Description:
// # Headerfile for the DS1307new Library
// #
//
############################################################
##################
// ******************************************
// DEFINE
// ******************************************
#ifndef DS1307new_h
#define DS1307new_h

// ******************************************
// INCLUDE
// ******************************************
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

// ******************************************
// Library interface description
// ******************************************
class DS1307new
{
  public:
    DS1307new();
    uint8_t isPresent(void);
    void startClock(void);
    void stopClock(void);
    void setTime(void);
    void getTime(void);
    void getCTRL(void);
    void setCTRL(void);
    void getRAM(uint8_t rtc_addr, uint8_t * rtc_ram,
uint8_t rtc_quantity);
```

```
    void setRAM(uint8_t rtc_addr, uint8_t * rtc_ram,
uint8_t rtc_quantity);
    uint8_t second;
    uint8_t minute;
    uint8_t hour;
    uint8_t dow;             // day of week, 0 = sunday
    uint8_t day;
    uint8_t month;
    uint16_t year;


    uint8_t ctrl;

    uint16_t ydn;      // day within the year (year day
number, starts with 1 = 1. Jan)
    uint16_t cdn;      // days after 2000-01-01 (century
day number, starts with 0)
    uint32_t time2000; // seconds after 2000-01-01 00:00
(max value: 2136-02-07 06:28:15)

    void fillByCDN(uint16_t _cdn);
    void fillByTime2000(uint32_t _time2000);
    void fillByHMS(uint8_t h, uint8_t m, uint8_t s);
    void fillByYMD(uint16_t y, uint8_t m, uint8_t d);
    uint8_t isMEZSummerTime(void);

  private:
    uint8_t is_leap_year(uint16_t y);
    void calculate_ydn(void);                  // calculate
ydn from year, month & day
    void calculate_cdn(void);                  // calculate
cdn from year & ydn
    void calculate_dow(void);                  // calculate
dow from ydn
    void calculate_time2000(void);             // calculate
time2000 from cdn, hour, minute & second

    uint16_t _corrected_year_day_number(void);
    void calculate_month_by_year_and_ydn(void);
    void calculate_day_by_month_year_and_ydn(void);

    uint8_t dec2bcd(uint8_t num);
    uint8_t bcd2dec(uint8_t num);
};

extern DS1307new RTC;

#endif
```

11.1.2 DS1307new.cpp

```
//
###########.read###############################################
######################
// #
```

```
// # Scriptname : DS1307new.cpp
// # Author     : Peter Schmelzer
// # Contributor: Oliver Kraus
// # contact    : info@schmelle2.de
// # Date       : 2010-11-01
// # Version    : 1.00
// # License    : cc-by-sa-3.0
// #
// # Description:
// # The DS1307new Library
// #
// # Naming Convention:
// #   get...     Get information from the DS1307 hardware
// #   set...     Write information to the DS1307 hardware
// #   fill...          Put some information onto the
object, but do not access DS1307 hardware
// #
// # Notes on the date calculation procedures
// #  Written 1996/97 by Oliver Kraus
// #  Published by Heinz Heise Verlag 1997 (c't 15/97)
// #  Completly rewritten and put under GPL 2011 by Oliver
Kraus
// #
//
############################################################
##################
// ********************************************
// INCLUDE
// ********************************************
#include "Wire.h"
#include "DS1307new.h"

// ********************************************
// DEFINE
// ********************************************
#define DS1307_ID 0x68

// ********************************************
// Public functions
// ********************************************
DS1307new::DS1307new()
{
  Wire.begin();
}

uint8_t DS1307new::isPresent(void)         // check if the
device is present
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);
  if (Wire.endTransmission() == 0) return 1;
  return 0;
}
```

```
void DS1307new::stopClock(void)         // set the
ClockHalt bit high to stop the rtc
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);                    //
Register 0x00 holds the oscillator start/stop bit
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ID, 1);
  second = Wire.read() | 0x80;       // save actual seconds
and OR sec with bit 7 (sart/stop bit) = clock stopped
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);
  Wire.write((uint8_t)second);                  // write
seconds back and stop the clock
  Wire.endTransmission();
}

void DS1307new::startClock(void)        // set the
ClockHalt bit low to start the rtc
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);                    //
Register 0x00 holds the oscillator start/stop bit
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ID, 1);
  second = Wire.read() & 0x7f;       // save actual seconds
and AND sec with bit 7 (sart/stop bit) = clock started
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);
  Wire.write((uint8_t)second);                  // write
seconds back and start the clock
  Wire.endTransmission();
}

// Aquire time from the RTC chip in BCD format and convert
it to DEC
void DS1307new::getTime(void)
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ID, 7);       // request secs,
min, hour, dow, day, month, year
  second = bcd2dec(Wire.read() & 0x7f);// aquire seconds...
  minute = bcd2dec(Wire.read());     // aquire minutes
  hour = bcd2dec(Wire.read());       // aquire hours
  dow = bcd2dec(Wire.read());        // aquire dow (Day Of
Week)
  dow--;                                        //  correction
from RTC format (1..7) to lib format (0..6). Useless,
because it will be overwritten
  day = bcd2dec(Wire.read());       // aquire day
  month = bcd2dec(Wire.read());      // aquire month
  year = bcd2dec(Wire.read());       // aquire year...
```

```
  year = year + 2000;                    // ...and assume
that we are in 21st century!

  // recalculate all other values
  calculate_ydn();
  calculate_cdn();
  calculate_dow();
  calculate_time2000();
}

// Set time to the RTC chip in BCD format
void DS1307new::setTime(void)
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x00);
  Wire.write(dec2bcd(second) | 0x80);   // set seconds
(clock is stopped!)
  Wire.write(dec2bcd(minute));           // set minutes
  Wire.write(dec2bcd(hour) & 0x3f);      // set hours (24h
clock!)
  Wire.write(dec2bcd(dow+1));              // set dow (Day
Of Week), do conversion from internal to RTC format
  Wire.write(dec2bcd(day));              // set day
  Wire.write(dec2bcd(month));            // set month
  Wire.write(dec2bcd(year-2000));           // set year
  Wire.endTransmission();
}

// Aquire data from the CTRL Register of the DS1307 (0x07)
void DS1307new::getCTRL(void)
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x07);                     // set
CTRL Register Address
  Wire.endTransmission();
  Wire.requestFrom(DS1307_ID, 1);       // read only CTRL
Register
  while(!Wire.available())
  {
    // waiting
  }
  ctrl = Wire.read();                // ... and store it in
ctrl
}

// Set data to CTRL Register of the DS1307 (0x07)
void DS1307new::setCTRL(void)
{
  Wire.beginTransmission(DS1307_ID);
  Wire.write((uint8_t)0x07);                     // set
CTRL Register Address
  Wire.write((uint8_t)ctrl);                     // set
CTRL Register
  Wire.endTransmission();
}
```

```
// Aquire data from RAM of the RTC Chip (max 56 Byte)
void DS1307new::getRAM(uint8_t rtc_addr, uint8_t * rtc_ram,
uint8_t rtc_quantity)
{
  Wire.beginTransmission(DS1307_ID);
  rtc_addr &= 63;                          // avoid wrong
adressing. Adress 0x08 is now address 0x00...
  rtc_addr += 8;                           // ... and address
0x3f is now 0x38
  Wire.write(rtc_addr);                     // set CTRL
Register Address
  if ( Wire.endTransmission() != 0 )
    return;
  Wire.requestFrom(DS1307_ID, (int)rtc_quantity);
  while(!Wire.available())
  {
    // waiting
  }
  for(int i=0; i<rtc_quantity; i++)     // Read x data from
given address upwards...
  {
    rtc_ram[i] = Wire.read();          // ... and store it in
rtc_ram
  }
}

// Write data into RAM of the RTC Chip
void DS1307new::setRAM(uint8_t rtc_addr, uint8_t * rtc_ram,
uint8_t rtc_quantity)
{
  Wire.beginTransmission(DS1307_ID);
  rtc_addr &= 63;                          // avoid wrong
adressing. Adress 0x08 is now address 0x00...
  rtc_addr += 8;                           // ... and address
0x3f is now 0x38
  Wire.write(rtc_addr);                     // set RAM start
Address
  for(int i=0; i<rtc_quantity; i++)     // Send x data from
given address upwards...
  {
    Wire.write(rtc_ram[i]);                // ... and send it
from rtc_ram to the RTC Chip
  }
  Wire.endTransmission();
}

/*
  Variable updates:
    cdn, ydn, year, month, day
*/
void DS1307new::fillByCDN(uint16_t _cdn)
{
  uint16_t y, days_per_year;
  cdn = _cdn;
```

```
  y = 2000;
  for(;;)
  {
    days_per_year = 365;
    days_per_year += is_leap_year(y);
    if ( _cdn >= days_per_year )
    {
      _cdn -= days_per_year;
      y++;
    }
    else
      break;
  }
  _cdn++;
  year = y;
  ydn = _cdn;
  calculate_dow();
  calculate_month_by_year_and_ydn();
  calculate_day_by_month_year_and_ydn();
  calculate_time2000();
}

/*
  Variable updates:
    time2000, cdn, ydn, year, month, day, hour, minute,
second
*/
void DS1307new::fillByTime2000(uint32_t _time2000)
{
  time2000 = _time2000;
  second = _time2000 % 60;
  _time2000 /= 60;
  minute = _time2000 % 60;
  _time2000 /= 60;
  hour = _time2000 % 24;
  _time2000 /= 24;
  fillByCDN(_time2000);
}

void DS1307new::fillByHMS(uint8_t h, uint8_t m, uint8_t s)
{
  // assign variables
  hour = h;
  minute = m;
  second = s;
  // recalculate seconds since 2000-01-01
  calculate_time2000();
}

void DS1307new::fillByYMD(uint16_t y, uint8_t m, uint8_t d)
{
  // assign variables
  year = y;
  month = m;
  day = d;
```

```
  // recalculate depending values
  calculate_ydn();
  calculate_cdn();
  calculate_dow();
  calculate_time2000();
}

// check if current time is european summer time
uint8_t DS1307new::isMEZSummerTime(void)
{
  uint32_t current_time, summer_start, winter_start;
  current_time = time2000;

  // calculate start of summer time
  fillByYMD(year, 3, 30);
  fillByHMS(2,0,0);
  fillByCDN(RTC.cdn - RTC.dow);   // sunday before
  summer_start = time2000;

  // calculate start of winter
  fillByYMD(year, 10, 31);
  fillByHMS(3,0,0);
  fillByCDN(RTC.cdn - RTC.dow);   // sunday before
  winter_start = time2000;

  // restore time
  fillByTime2000(current_time);

  // return result
  if ( summer_start <= current_time && current_time <
winter_start )
    return 1;
  return 0;
}


// ********************************************
// Private functions
// ********************************************
// Convert Decimal to Binary Coded Decimal (BCD)
uint8_t DS1307new::dec2bcd(uint8_t num)
{
  return ((num/10 * 16) + (num % 10));
}

// Convert Binary Coded Decimal (BCD) to Decimal
uint8_t DS1307new::bcd2dec(uint8_t num)
{
  return ((num/16 * 10) + (num % 16));
}

/*
  Prototype:
    uint8_t DS1307new::is_leap_year(uint16_t y)
  Description:
```

```
    Calculate leap year
  Arguments:
    y             year, e.g. 2011 for year 2011
  Result:
    0           not a leap year
    1           leap year
*/
uint8_t DS1307new::is_leap_year(uint16_t y)
{
  //uint16_t y = year;
   if (
          ((y % 4 == 0) && (y % 100 != 0)) ||
          (y % 400 == 0)
      )
      return 1;
   return 0;
}

/*
  Prototype:
    void calculate_ydn(void)
  Description:
    Calculate the day number within a year. 1st of Jan has
the number 1.
    "Robertson" Algorithm
  Arguments:
    this->year              year, e.g. 2011 for year 2011
    this->month             month with 1 = january to 12 =
december
    this->day               day starting with 1
  Result:
    this->ydn          The "day number" within the year: 1
for the 1st of Jan.

*/
void DS1307new::calculate_ydn(void)
{
  uint8_t tmp1;
  uint16_t tmp2;
  tmp1 = 0;
  if ( month >= 3 )
    tmp1++;
  tmp2 = month;
  tmp2 +=2;
  tmp2 *=611;
  tmp2 /= 20;
  tmp2 += day;
  tmp2 -= 91;
  tmp1 <<=1;
  tmp2 -= tmp1;
  if ( tmp1 != 0 )
    tmp2 += is_leap_year(year);
  ydn = tmp2;
}
```

```
/*
  Prototype:
    uint16_t to_century_day_number(uint16_t y, uint16_t
ydn)
  Description:
    Calculate days since January, 1st, 2000
  Arguments:
    this->y           year, e.g. 2011 for year 2011
    this->ydn     year day number (1st of Jan has the number
1)
  Result
    this->cdn     days since 2000-01-01 (2000-01-01 has the
cdn 0)
*/
void DS1307new::calculate_cdn(void)
{
  uint16_t y = year;
  cdn = ydn;
  cdn--;
  while( y > 2000 )
  {
    y--;
    cdn += 365;
    cdn += is_leap_year(y);
  }
}

/*
  calculate day of week (dow)
  0 = sunday .. 6 = saturday
  Arguments:
    this->cdn     days since 2000-01-01 (2000-01-01 has the
cdn 0 and is a saturday)
*/
void DS1307new::calculate_dow(void)
{
  uint16_t tmp;
  tmp = cdn;
  tmp += 6;
  tmp %= 7;
  dow = tmp;
}

/*
  Calculate the seconds after 2000-01-01 00:00. The largest
possible
  time is 2136-02-07 06:28:15
  Arguments:
    this->h         hour
    this->m minutes
    this->s       seconds
    this->cdn     days since 2000-01-01 (2000-01-01 has the
cdn 0)
*/
void DS1307new::calculate_time2000(void)
```

```
{
  uint32_t t;
  t = cdn;
  t *= 24;
  t += hour;
  t *= 60;
  t += minute;
  t *= 60;
  t += second;
  time2000 = t;
}


uint16_t DS1307new::_corrected_year_day_number(void)
{
   uint8_t a;
   uint16_t corrected_ydn = ydn;
   a = is_leap_year(year);
   if ( corrected_ydn > (uint8_t)(((uint8_t)59)+a) )
   {
      corrected_ydn += 2;
      corrected_ydn -= a;
   }
   corrected_ydn += 91;
   return corrected_ydn;
}

/*
  Variables reads:
    ydn, year
  Variable updates:
    month
*/
void DS1307new::calculate_month_by_year_and_ydn(void)
{
  uint8_t a;
  uint16_t c_ydn;
  c_ydn = _corrected_year_day_number();
  c_ydn *= 20;
  c_ydn /= 611;
  a = c_ydn;
  a -= 2;
  month = a;
}

/*
  Variables reads:
    ydn, year, month
  Variable updates:
    day
*/
void DS1307new::calculate_day_by_month_year_and_ydn(void)
{
  uint8_t m;
  uint16_t tmp, c_ydn;
```

```
  m = month;
  m += 2;
  c_ydn = _corrected_year_day_number();
  tmp = 611;
  tmp *= m;
  tmp /= 20;
  c_ydn -= tmp;
  day = c_ydn;
}



// ***************************************
// Define user object
// ***************************************
class DS1307new RTC;
```

11.2 Sample Sketches using DS1307new

The sketches found by following the previous DFRobot link are ".pde" files. They will still work and compile successfully using IDE 1.0.1. The files can be renamed with a ".ino" extension (as I have done below) if that is your preference.

11.2.1 DS1307_Test.ino

```
//
############################################################
##################
// #
// # Scriptname : DS1307_Test.ino
// # Author     : Peter Schmelzer, Oliver Kraus
// # Date       : 2011-04-08
// # Version    : 1.21
// # License    : cc-by-sa-3.0
// #
// # Description:
// # Test file for the DS1307new library. Assumes that you
have a DS1307
// # connected to the I2C-Bus of your Arduino and that it
has a battery backup.
// #
//
############################################################
##################
// ***************************************
// INCLUDE
// ***************************************
#include <Wire.h>                       // For some strange
reasons, Wire.h must be included here
#include <DS1307new.h>

// ***************************************
// DEFINE
// ***************************************
```

```
// ********************************************
// VARIABLES
// ********************************************
uint16_t startAddr = 0x0000;            // Start address to
store in the NV-RAM
uint16_t lastAddr;                      // new address for
storing in NV-RAM
uint16_t TimeIsSet = 0xaa55;            // Helper that time
must not set again

// ********************************************
// SETUP
// ********************************************
void setup()
{
  pinMode(2, INPUT);                    // Test of the SQW
pin, D2 = INPUT
  digitalWrite(2, HIGH);                // Test of the SQW
pin, D2 = Pullup on

  Serial.begin(9600);

/*
   PLEASE NOTICE: WE HAVE MADE AN ADDRESS SHIFT FOR THE NV-
RAM!!!
                  NV-RAM ADDRESS 0x08 HAS TO ADDRESSED WITH
ADDRESS 0x00=0
                  TO AVOID OVERWRITING THE CLOCK REGISTERS
IN CASE OF
                  ERRORS IN YOUR CODE. SO THE LAST ADDRESS
IS 0x38=56!
*/
  RTC.setRAM(0, (uint8_t *)&startAddr, sizeof(uint16_t));//
Store startAddr in NV-RAM address 0x08

/*
   Uncomment the next 2 lines if you want to SET the clock
   Comment them out if the clock is set.
   DON'T ASK ME WHY: YOU MUST UPLOAD THE CODE TWICE TO LET
HIM WORK
   AFTER SETTING THE CLOCK ONCE.
*/
//  TimeIsSet = 0xffff;
//  RTC.setRAM(54, (uint8_t *)&TimeIsSet,
sizeof(uint16_t));

/*
  Control the clock.
  Clock will only be set if NV-RAM Address does not contain
0xaa.
  DS1307 should have a battery backup.
*/
  RTC.getRAM(54, (uint8_t *)&TimeIsSet, sizeof(uint16_t));
  if (TimeIsSet != 0xaa55)
```

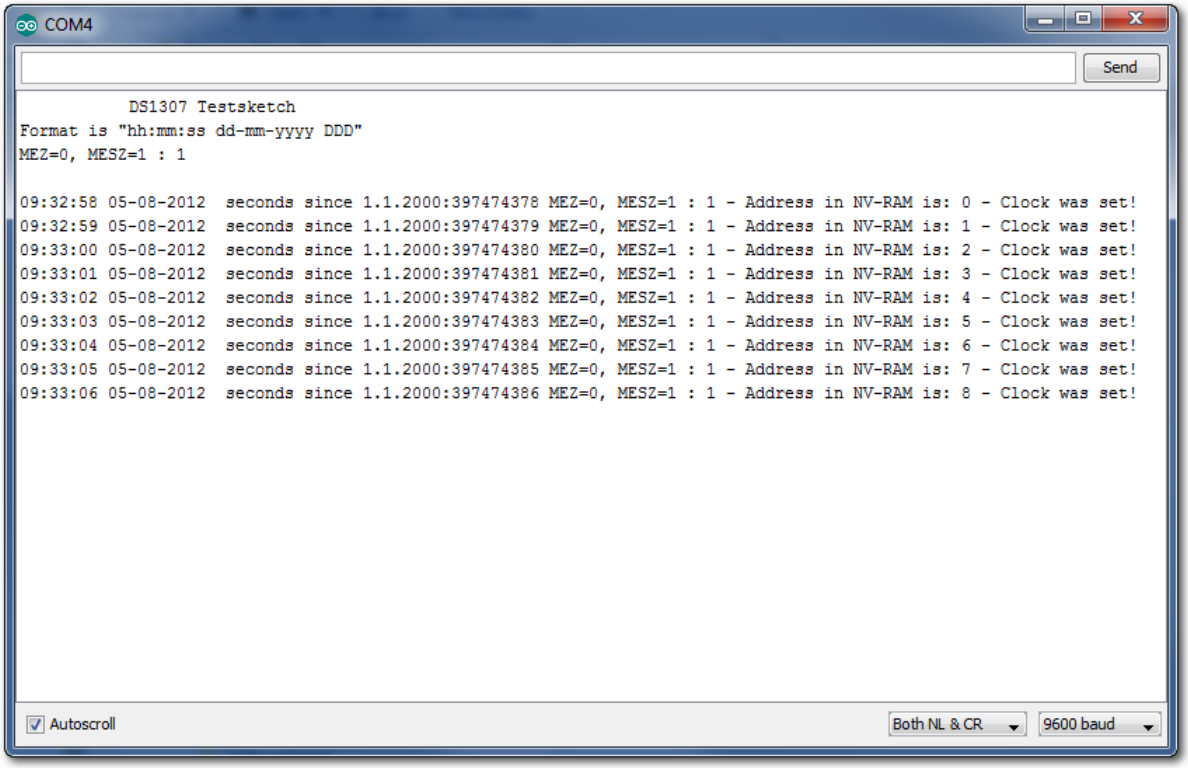
```
  {
    RTC.stopClock();

    RTC.fillByYMD(2011,4,8);
    RTC.fillByHMS(22,7,0);

    RTC.setTime();
    TimeIsSet = 0xaa55;
    RTC.setRAM(54, (uint8_t *)&TimeIsSet,
sizeof(uint16_t));
    RTC.startClock();
  }
  else
  {
    RTC.getTime();
  }

/*
   Control Register for SQW pin which can be used as an
interrupt.
*/
  RTC.ctrl = 0x00;                          // 0x00=disable SQW
pin, 0x10=1Hz,
                                            // 0x11=4096Hz,
0x12=8192Hz, 0x13=32768Hz
  RTC.setCTRL();

  Serial.println("DS1307 Testsketch");
  Serial.println("Format is \"hh:mm:ss dd-mm-yyyy DDD\"");

  uint8_t MESZ;

  MESZ = RTC.isMEZSummerTime();
  Serial.print("MEZ=0, MESZ=1 : ");
  Serial.println(MESZ, DEC);
  Serial.println();
}

// ******************************************
// MAIN (LOOP)
// ******************************************
void loop()
{
  RTC.getTime();
  if (RTC.hour < 10)                        // correct hour if
necessary
  {
    Serial.print("0");
    Serial.print(RTC.hour, DEC);
  }
  else
  {
    Serial.print(RTC.hour, DEC);
  }
  Serial.print(":");
```

```
  if (RTC.minute < 10)                    // correct minute
if necessary
  {
    Serial.print("0");
    Serial.print(RTC.minute, DEC);
  }
  else
  {
    Serial.print(RTC.minute, DEC);
  }
  Serial.print(":");
  if (RTC.second < 10)                    // correct second
if necessary
  {
    Serial.print("0");
    Serial.print(RTC.second, DEC);
  }
  else
  {
    Serial.print(RTC.second, DEC);
  }
  Serial.print(" ");
  if (RTC.day < 10)                      // correct date if
necessary
  {
    Serial.print("0");
    Serial.print(RTC.day, DEC);
  }
  else
  {
    Serial.print(RTC.day, DEC);
  }
  Serial.print("-");
  if (RTC.month < 10)                     // correct month if
necessary
  {
    Serial.print("0");
    Serial.print(RTC.month, DEC);
  }
  else
  {
    Serial.print(RTC.month, DEC);
  }
  Serial.print("-");
  Serial.print(RTC.year, DEC);            // Year need not to
be changed
  Serial.print(" ");
  switch (RTC.dow)                        // Friendly
printout the weekday
  {
    case 1:
      Serial.print("MON");
      break;
    case 2:
      Serial.print("TUE");
```

```
      break;
    case 3:
      Serial.print("WED");
      break;
    case 4:
      Serial.print("THU");
      break;
    case 5:
      Serial.print("FRI");
      break;
    case 6:
      Serial.print("SAT");
      break;
    case 7:
      Serial.print("SUN");
      break;
  }
  Serial.print(" seconds since 1.1.2000:");
  Serial.print(RTC.time2000, DEC);
  uint8_t MESZ = RTC.isMEZSummerTime();
  Serial.print(" MEZ=0, MESZ=1 : ");
  Serial.print(MESZ, DEC);

  Serial.print(" - Address in NV-RAM is: ");
  RTC.getRAM(0, (uint8_t *)&lastAddr, sizeof(uint16_t));
  Serial.print(lastAddr, HEX);
  lastAddr = lastAddr + 1;                  // we want to use
it as addresscounter for example
  RTC.setRAM(0, (uint8_t *)&lastAddr, sizeof(uint16_t));
  RTC.getRAM(54, (uint8_t *)&TimeIsSet, sizeof(uint16_t));
  if (TimeIsSet == 0xaa55)                  // check if the
clock was set or not
  {
    Serial.println(" - Clock was set!");
  }
  else
  {
    Serial.println(" - Clock was NOT set!");
  }
  delay(1000);                              // wait a second
}
```

Note that this example sketch uses 1 through 7 for the “Day of the Week” with Sunday=7.

11.2.1.1 Sample Output



11.2.1.2 Notes

This is the output “straight-out-of-the box”.

11.2.2 DS1307_Monitor.ino

```
//
#############################################################
##################
// #
// # Scriptname : DS1307_Monitor.ino
// # Author     : Oliver Kraus
// # Date       : 21.03.2011
// # License    : cc-by-sa-3.0
// #
// # Description:
// # Inspection and monitoring tool for the DS1307 hardware
// #
//
#############################################################
##################
// ********************************************
// INCLUDE
// ********************************************
#include <Wire.h>                          // For some strange
reasons, Wire.h must be included here
#include <DS1307new.h>

// ********************************************
```

```
// DEFINE
// ********************************************

// ********************************************
// VARIABLES
// ********************************************

#define MON_BUF_LEN 128

char mon_buf[MON_BUF_LEN];
uint8_t mon_buf_cnt = 0;
char *mon_curr;

uint8_t mon_is_date;
uint16_t mon_year;
uint8_t mon_month;
uint8_t mon_day;

uint8_t mon_is_time;
uint8_t mon_hour;
uint8_t mon_minute;
uint8_t mon_second;

/*
  1:   missing '-' after year ("yyyy-mm-dd")
  2:   missing '-' after month ("yyyy-mm-dd")
  3:   month not within 1..12
  4:   day not within 1..31
  5:   missing ':' after hour ("hh:mm")
  6:   hour not within 0..23
  7:   minute not within 0..59
  8:   second not within 0..59
*/
uint8_t mon_error;


// ********************************************
// PARSER
// ********************************************

void mon_skip_space(void)
{
  for(;;)
  {
    if ( *mon_curr == '\0' )
      break;
    if ( *mon_curr > ' ')
      break;
    mon_curr++;
  }
}

uint16_t mon_get_value(void)
{
  uint16_t v = 0;
```

```
  uint8_t c;
  for(;;)
  {
    if ( *mon_curr >= '0' && *mon_curr <= '9' )
    {
      c = *mon_curr;
      c -= '0';
      v *= 10;
      v += c;
      mon_curr++;
    }
    else
      break;
  }
  mon_skip_space();
  return v;
}

uint8_t mon_check(char c)
{
  if ( *mon_curr != c )
    return 0;
  mon_curr++;
  mon_skip_space();
  return 1;
}

/*
  yyyy-mm-dd
*/
uint8_t mon_get_date(void)
{
  mon_year = mon_get_value();
  if ( mon_year < 100 )
    mon_year += 2000;
  if ( mon_check('-') == 0 )
  {
    mon_error = 1;
    return 0;
  }
  mon_month = mon_get_value();
  if ( mon_month == 0 || mon_month > 12 )
  {
    mon_error = 3;
    return 0;
  }
  if ( mon_check('-') == 0 )
  {
    mon_error = 2;
    return 0;
  }
  mon_day = mon_get_value();
  if ( mon_day == 0 || mon_day > 31 )
  {
    mon_error = 4;
```

```
    return 0;
  }
  mon_is_date = 1;
  return 1;
}

/*
  hh:mm
  hh:mm:ss
*/
uint8_t mon_get_time(void)
{
  mon_second = 0;
  mon_hour = mon_get_value();
  if ( mon_hour > 23 )
  {
    mon_error = 6;
    return 0;
  }
  if ( mon_check(':') == 0 )
  {
    mon_error = 5;
    return 0;
  }
  mon_minute = mon_get_value();
  if ( mon_minute > 59 )
  {
    mon_error = 7;
    return 0;
  }
  if ( mon_check(':') == 0 )
  {
    mon_is_time = 1;
    return 1;
  }
  mon_second = mon_get_value();
  if ( mon_second > 59 )
  {
    mon_error = 8;
    return 0;
  }
  mon_is_time = 1;
  return 1;
}


/*
  yyyy-mm-dd hh:mm:ss
*/

uint8_t mon_get_date_time(void)
{
  char *mon_ptr;
  uint16_t v;
```

```
  mon_is_date = 0;
  mon_year = 2000;
  mon_month = 1;
  mon_day = 1;

  mon_is_time = 0;
  mon_hour = 0;
  mon_minute = 0;
  mon_second = 0;


  for(;;)
  {
    mon_ptr = mon_curr;
    v = mon_get_value();
    if ( mon_check(':') != 0 )
    {
      mon_curr = mon_ptr;
      if ( mon_get_time() == 0 )
 return 0;
    }
    else if ( mon_check('-') != 0 )
    {
      mon_curr = mon_ptr;
      if ( mon_get_date() == 0 )
 return 0;
    }
    else
      break;
  }
  return 1;
}

void mon_help(void)
{
  Serial.println("Available commands:");
  Serial.println("i         read and display current date
and time");
  Serial.println("s <d> <t>  set date <d> and/or time
<t>");
  Serial.println("l         list clock NVRAM");
  Serial.println("d         print daylight saving date
(Europa)");
  Serial.println("m <a> <v>  write value <v> to NVRAM
location <a>");
  Serial.println("h         this help");
}

void mon_print_error(void)
{
  Serial.print("Errorcode: ");
  Serial.println(mon_error, DEC);
}

void mon_print_2d(uint8_t v)
```

```
{
  if ( v < 10 )
    Serial.print("0");
  Serial.print(v, DEC);
}

void mon_print_3d(uint8_t v)
{
  if ( v < 10 )
    Serial.print(" ");
  if ( v < 100 )
    Serial.print(" ");
  Serial.print(v, DEC);
}

void mon_print_date(uint16_t y, uint8_t m, uint8_t d)
{
    Serial.print(y, DEC);
    Serial.print("-");
    mon_print_2d(m);
    Serial.print("-");
    mon_print_2d(d);
}

void mon_print_time(uint16_t h, uint8_t m, uint8_t s)
{
    mon_print_2d(h);
    Serial.print(":");
    mon_print_2d(m);
    Serial.print(":");
    mon_print_2d(s);
}

void mon_info(void)
{
  char wd[7][3] = { "So", "Mo", "Tu", "We", "Th", "Fr",
"Sa" };
  RTC.getTime();
  Serial.print("RTC date: ");
  mon_print_date(RTC.year, RTC.month, RTC.day);
  Serial.print(" ");
  Serial.print(wd[RTC.dow]);
  Serial.print("  time: ");
  mon_print_time(RTC.hour, RTC.minute, RTC.second);
  Serial.println("");
  Serial.print(RTC.cdn, DEC);
  Serial.print(" days or ");
  Serial.print(RTC.time2000, DEC);
  Serial.println(" seconds since 2000-01-01 00:00:00");
}

void mon_dst(void)
{
  long m;
  uint8_t isSummerTime;
```

```
  uint16_t i;

  m = millis();
  for( i = 0; i < 1000; i++ )
    isSummerTime = RTC.isMEZSummerTime();
  m = millis() - m;

  Serial.print("Result from isMEZSummerTime(): ");
  Serial.print(isSummerTime, DEC);
  Serial.print(" (");
  Serial.print(m, DEC);
  Serial.print("ns + ");

  m = millis();
  for( i = 0; i < 1000; i++ )
    RTC.fillByTime2000(RTC.time2000);            // speed
measue
  m = millis() - m;
  Serial.print(m, DEC);
  Serial.println("us)");

  m = millis();
  RTC.fillByYMD(RTC.year, 4, 1);       // first of April
  if ( RTC.dow == 0 )
    RTC.fillByCDN(RTC.cdn - 7);              // sunday
before
  else
    RTC.fillByCDN(RTC.cdn - RTC.dow);  // sunday before
  m = millis() - m;
  Serial.print("Summer time (turn forward the clock): ");
  mon_print_date(RTC.year, RTC.month, RTC.day);
  Serial.print(" (");
  Serial.print(RTC.cdn, DEC);
  Serial.print(" days or ");
  Serial.print(RTC.time2000, DEC);
  Serial.print(" seconds since 2000-01-01 00:00:00)");

  Serial.println("");

  RTC.fillByYMD(RTC.year, 11, 1);      // first of november
  if ( RTC.dow == 0 )
    RTC.fillByCDN(RTC.cdn - 7);              // sunday
before
  else
    RTC.fillByCDN(RTC.cdn - RTC.dow);  // sunday before
  Serial.print("Winter time (turn back the clock): ");
  mon_print_date(RTC.year, RTC.month, RTC.day);
  Serial.print(" (");
  Serial.print(RTC.cdn, DEC);
  Serial.print(" days or ");
  Serial.print(RTC.time2000, DEC);
  Serial.print(" seconds since 2000-01-01 00:00:00)");
}
```

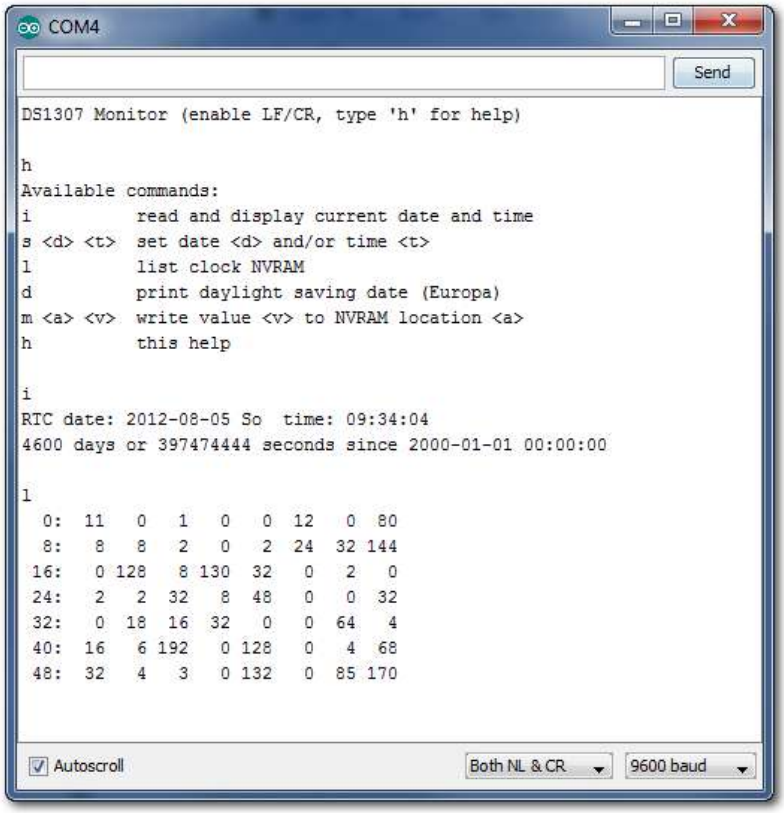
```
void mon_set_date_time(void)
{
  if ( mon_get_date_time() == 0 )
  {
    mon_print_error();
    return;
  }
  RTC.getTime();
  Serial.print("Assign ");
  if ( mon_is_date != 0 )
  {
    mon_print_date(mon_year, mon_month, mon_day);
    Serial.print(" ");
    RTC.fillByYMD(mon_year, mon_month, mon_day);
  }
  if ( mon_is_time != 0 )
  {
    mon_print_time(mon_hour, mon_minute, mon_second);
    Serial.print(" ");
    RTC.fillByHMS(mon_hour, mon_minute, mon_second);
  }
  RTC.setTime();
  RTC.startClock();
  Serial.println("to RTC");
}

void mon_list(void)
{
  uint8_t i, addr;
  uint8_t ram[8];

  for( addr = 0; addr < 56; addr+= 8 )
  {
    RTC.getRAM(addr, ram, 8);
    mon_print_3d(addr);
    Serial.print(": ");
    for( i = 0; i < 8; i++ )
    {
      mon_print_3d(ram[i]);
      Serial.print(" ");
    }
    Serial.println("");
  }
}

void mem_memory(void)
{
    uint8_t a, m;
    a = mon_get_value();
    m = mon_get_value();
    Serial.print("Write ");
    Serial.print(m, DEC);
    Serial.print(" to memory location ");
    Serial.println(a, DEC);
```

```
    RTC.setRAM(a, &m, 1);
}


void mon_cmd(void)
{
  mon_skip_space();
  if ( *mon_curr == '\0' )
    return;
  if ( mon_check('?') )
    mon_help();
  else if ( mon_check('h') )
    mon_help();
  else if ( mon_check('s') )
    mon_set_date_time();
  else if ( mon_check('i') )
    mon_info();
  else if ( mon_check('l') )
    mon_list();
  else if ( mon_check('m') )
    mem_memory();
  else if ( mon_check('d') )
    mon_dst();
  else
    ;

}



// *********************************************
// SETUP
// *********************************************
void setup()
{
  delay(1000);
  pinMode(2, INPUT);                    // Test of the SQW
pin, D2 = INPUT
  digitalWrite(2, HIGH);                // Test of the SQW
pin, D2 = Pullup on
  Serial.begin(9600);
  Serial.println("DS1307 Monitor (enable LF/CR, type 'h'
for help)");
  Serial.println();
}

// *********************************************
// EXEC
// *********************************************

void exec(void)
{
  Serial.println(mon_buf);
  mon_curr = mon_buf;
  mon_cmd();
```

```
}


// ****************************************
// LED flashing
// ****************************************
uint8_t LED_state = 0;
uint32_t LED_next_change = 0L;
uint32_t LED_on_time = 100L;
uint32_t LED_off_time = 1000L;

void LED_flashing(void)
{
  if ( LED_next_change < millis() )
  {
    if ( LED_state == 0 )
    {
      LED_next_change = millis() + LED_on_time;
      LED_state = 1;
    }
    else
    {
      LED_next_change = millis() + LED_off_time;
      LED_state = 0;
    }
    digitalWrite(13, LED_state);
  }
}

// ****************************************
// MAIN (LOOP)
// ****************************************
void loop()
{
  LED_flashing();

  if ( RTC.isPresent() == 0 )
    LED_off_time = 100L;                // fast flashing if
device is not available

  if ( Serial.available() )
  {
    char c;
    c = Serial.read();
    if ( mon_buf_cnt >= MON_BUF_LEN-1 || c == '\n' || c ==
'\r'  )
    {
      exec();
      mon_buf_cnt = 0;
      mon_buf[mon_buf_cnt] = '\0';
    }
    else
    {
      mon_buf[mon_buf_cnt] = c;
      mon_buf_cnt++;
```

```
        mon_buf[mon_buf_cnt] = '\0';
      }
    }
  }
```

11.2.2.1 Sample Output



11.2.2.2 Notes

Not documented nor shown here is the ability of the sketch to write to specific RAM locations. These are locations outside the area reserved for the RTC registers. Shown in the above screen capture is the function to display the full NVRAM contents.

You should consult the data sheet to determine which locations are safe to write to. As far as I know, this library and accompanying Monitor sketch is the only one to provide the functionality to read and write NVRAM. (Note: this is different from reading and writing to the additional AT24C32 4K RAM area provided with some RTC boards.

12 Sketches using the RealTimeClockDS1307 Library

This library has been written by David H Brown and is available here:

https://github.com/davidhbrown/RealTimeClockDS1307

According to the author, the reason for creating “yet another” RTC library was:

> *My goal in creating yet another DS1307 library was to provide*

easy access to some of the other functions I needed from the chip, specifically its square wave output and its battery-backed RAM.

12.1 Library Files

12.1.1 RealTimeClockDS1307.h

```
/*
RealTimeClockDS1307 - library to control a DS1307 RTC
module
Copyright (c) 2011 David H. Brown. All rights reserved

v0.92 Updated for Arduino 1.00; not re-tested on earlier
versions
Much thanks to John Waters and Maurice Ribble for their
earlier and very helpful work (even if I didn't wind up
using any of their code):
- http://combustory.com/wiki/index.php/RTC1307_-
_Real_Time_Clock
- http://www.glacialwanderer.com/hobbyrobotics/?p=12

This library is free software; you can redistribute it
and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation;
either
version 2.1 of the License, or (at your option) any later
version.

This library is distributed in the hope that it will be
useful,
but WITHOUT ANY WARRANTY; without even the implied warranty
of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public
License along with this library; if not, write to the Free
Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301 USA
*/

#ifndef RealTimeClockDS1307_h
#define RealTimeClockDS1307_h

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
```

```
//#include <HardwareSerial.h>
//#include <WConstants.h> //need/want 'boolean' and 'byte'
types used by Arduino
//#undef round is required to avoid a compile-time
//"expected unqualified-id before 'double'" error in math.h
//see: http://www.arduino.cc/cgi-
bin/yabb2/YaBB.pl?num=1247924528/3
#undef round
#include <Wire.h>

#define ARDUINO_PIN_T uint8_t

class RealTimeClockDS1307
{
  private:
    byte _reg0_sec;
    byte _reg1_min;
    byte _reg2_hour;
    byte _reg3_day;
    byte _reg4_date;
    byte _reg5_month;
    byte _reg6_year;
    byte _reg7_sqw;
    byte decToBcd(byte);
    byte bcdToDec(byte);
    char lowNybbleToASCII(byte);
    char highNybbleToASCII(byte);
  public:
    RealTimeClockDS1307();
    void readClock();//read registers (incl sqw) to local
store
    void setClock();//update clock registers from local
store
    void stop();//immediate; does not require setClock();
    void start();//immediate; does not require setClock();
    void sqwEnable(byte);//enable the square wave with the
specified frequency
    void sqwDisable(boolean);//disable the square wave,
setting output either high or low
    void writeData(byte, byte);//write a single value to a
register
    void writeData(byte, void *, int);//write several
values consecutively
    byte readData(byte);//read a single value from a
register
    void readData(byte, void *, int);//read several values
into a buffer

    int getHours();
    int getMinutes();
    int getSeconds();
    int getYear();
    int getMonth();
    int getDate();
```

```
    int getDayOfWeek();
    boolean is12hour();
    boolean isPM();
    boolean isStopped();
    //getFormatted writes into a char array provided by
you. Format is:
    // YY-MM-DD HH:II:SS ... plus "A" or "P" if in 12-hour
mode
    //and of course a NULL terminator. So, [18] for 24h or
[19] for 12h
    void getFormatted(char *);//see comment above
    void getFormatted2k(char *);//as getFormatted, but with
"20" prepended

    //must also call setClock() after any of these
    //before next readClock(). Note that invalid dates are
not
    //corrected by the clock. All the clock knows is when
it should
    //roll over to the next month rather than the next date
in the same month.
    void setSeconds(int);
    void setMinutes(int);
    //setHours rejects values out of range for the current
12/24 mode
    void setHours(int);
    void setAM();//does not consider hours; see
switchTo24()
    void setPM();//does not consider hours; see
switchTo24()
    void set24h();//does not consider hours; see
switchTo24()
    void switchTo24h();//returns immediately if already 24h
    void switchTo12h();//returns immediately if already 12h
    void setDayOfWeek(int);//incremented at midnight; not
set by date (no fixed meaning)
    void setDate(int);//allows 1-31 for *all* months.
    void setMonth(int);
    void setYear(int);

    //squarewave frequencies:
    static const byte SQW_1Hz=0x00;
    static const byte SQW_4kHz=0x01;//actually 4.096kHz
    static const byte SQW_8kHz=0x02;//actually 8.192kHz
    static const byte SQW_32kHz=0x03;//actually 32.768kHz
};

extern RealTimeClockDS1307 RTC;

#endif
```

12.1.2 RealTimeClockDS1307.cpp

```
/*
```

```
RealTimeClockDS1307 - library to control a DS1307 RTC
module
Copyright (c) 2011 David H. Brown. All rights reserved

v0.92 Updated for Arduino 1.00; not re-tested on earlier
versions
Much thanks to John Waters and Maurice Ribble for their
earlier and very helpful work (even if I didn't wind up
using any of their code):
- http://combustory.com/wiki/index.php/RTC1307_-
_Real_Time_Clock
- http://www.glacialwanderer.com/hobbyrobotics/?p=12

This library is free software; you can redistribute it
and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation;
either
version 2.1 of the License, or (at your option) any later
version.

This library is distributed in the hope that it will be
useful,
but WITHOUT ANY WARRANTY; without even the implied warranty
of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public
License along with this library; if not, write to the Free
Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301 USA
*/

/**************************************************************
********************
* Includes
**************************************************************
******************/

#include "RealTimeClockDS1307.h"
#include <Wire.h>

/**************************************************************
********************
* Definitions
**************************************************************
******************/

#define DS1307_I2C_ADDRESS 0x68 // This is the I2C address
```

```
/**********************************************************
*******************
* Constructors
**********************************************************
******************/

RealTimeClockDS1307::RealTimeClockDS1307()
{
  Wire.begin();
  //must NOT attempt to read the clock before
  //Wire.begin() has not been called; readClock() will
hang.
  //Fortunately, it seems that you can call Wire.begin()
  //multiple times with no adverse effect).
}

/**********************************************************
*******************
* User API
**********************************************************
******************/

/***** CHIP READ/WRITE ******/

void RealTimeClockDS1307::readClock()
{
  // Reset the register pointer
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write((uint8_t) 0x00);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 8);
  _reg0_sec = Wire.read();
  _reg1_min = Wire.read();
  _reg2_hour = Wire.read();
  _reg3_day = Wire.read();
  _reg4_date = Wire.read();
  _reg5_month = Wire.read();
  _reg6_year = Wire.read();
  _reg7_sqw = Wire.read();
}

void RealTimeClockDS1307::setClock()
{
  //to be paranoid, we're going to first stop the clock
  //to ensure we don't have rollovers while we're
  //writing:
  writeData(0,0x80);
  //now, we'll write everything *except* the second
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write((uint8_t) 0x01);
  Wire.write(_reg1_min);
  Wire.write(_reg2_hour);
  Wire.write(_reg3_day);
  Wire.write(_reg4_date);
```

```
  Wire.write(_reg5_month);
  Wire.write(_reg6_year);
  Wire.endTransmission();
  //now, we'll write the seconds; we didn't have to keep
  //track of whether the clock was already running, because
  //_reg0_sec already knows what we want it to be. This
  //will restart the clock as it writes the new seconds
value.
  writeData(0,_reg0_sec);
}

void RealTimeClockDS1307::stop()
{
  //"Bit 7 of register 0 is the clock halt (CH) bit.
  //When this bit is set to a 1, the oscillator is
disabled."
  _reg0_sec = _reg0_sec | 0x80;
  writeData(0,_reg0_sec);
}
void RealTimeClockDS1307::start()
{
  //"Bit 7 of register 0 is the clock halt (CH) bit.
  //When this bit is set to a 1, the oscillator is
disabled."
  _reg0_sec = _reg0_sec & ~0x80;
  writeData(0,_reg0_sec);
}

void RealTimeClockDS1307::writeData(byte regNo, byte value)
{
  if(regNo > 0x3F) { return; }
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.write(regNo);
   Wire.write(value);
   Wire.endTransmission();
}

void RealTimeClockDS1307::writeData(byte regNo, void *
source, int length)
{
  char * p = (char*) source;
  if(regNo > 0x3F || length > 0x3F) { return; }
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(regNo);
  for(int i=0; i<length; i++) {
    Wire.write(*p);
    p++;
  }
  Wire.endTransmission();
}

byte RealTimeClockDS1307::readData(byte regNo)
{
  if(regNo > 0x3F) { return 0xff; }
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
```

```
  Wire.write(regNo);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, 1);
  return Wire.read();
}

void RealTimeClockDS1307::readData(byte regNo, void * dest,
int length)
{
  char * p = (char*) dest;
  if(regNo > 0x3F || length > 0x3F) { return; }
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(regNo);
  Wire.endTransmission();
  Wire.requestFrom(DS1307_I2C_ADDRESS, length);
  for(int i=0; i<length; i++) {
    *p=Wire.read();
    p++;
  }
}


void RealTimeClockDS1307::sqwEnable(byte frequency)
{
  if(frequency > 3) { return; }
  //bit 4 is enable (0x10);
  //bit 7 is current output state if disabled
  _reg7_sqw = _reg7_sqw & 0x80 | 0x10 | frequency;
  writeData(0x07, _reg7_sqw);
}

void RealTimeClockDS1307::sqwDisable(boolean outputLevel)
{
  //bit 7 0x80 output + bit 4 0x10 enable both to zero,
  //the OR with the boolean shifted up to bit 7
  _reg7_sqw = _reg7_sqw & ~0x90 | (outputLevel << 7);
  writeData(0x07, _reg7_sqw);
  //note: per the data sheet, "OUT (Output control): This
bit controls
  //the output level of the SQW/OUT pin when the square
wave
  //output is disabled. If SQWE = 0, the logic level on the
  //SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0."
  //"The SQW/OUT pin is open drain and requires an external
  //pull-up resistor."
  //It is worth mentioning that on the Sparkfun breakout
board,
  //BOB-00099, a LED connected to the SQW pin through a
resistor to
  //Vcc+5V illuminated when OUT=0 and was dark when OUT=1,
the
  //opposite of what I expected until I remembered that it
is
  //an open drain (google it if you need to). Basically,
they don't
```

```
  //so much mean a logic level (e.g., +3.3V rel Gnd) as
they mean
  //high or low *impeadance* to ground (drain). So High is
basically
  //an open switch. Low connects to ground.
}


/***** GETTERS ******/

boolean RealTimeClockDS1307::is12hour()
{
  //12-hour mode has bit 6 of the hour register set high
  return ((_reg2_hour & 0x40) == 0x40);
}
boolean RealTimeClockDS1307::isPM()
{
  //if in 12-hour mode, but 5 of the hour register
indicates PM
  if(is12hour()) {
    return ((_reg2_hour & 0x20) == 0x20);
  }
  //otherwise, let's consider any time with the hour >11 to
be PM:
  return (getHours() > 11);
}
boolean RealTimeClockDS1307::isStopped()
{
  //bit 7 of the seconds register stopps the clock when
high
  return ((_reg0_sec & 0x80) == 0x80);
}

int RealTimeClockDS1307::getHours()
{
  if(is12hour()) {
    //do not include bit 5, the am/pm indicator
    return bcdToDec(_reg2_hour & 0x1f);
  }
  //bits 4-5 are tens of hours
  return bcdToDec(_reg2_hour & 0x3f);
}
int RealTimeClockDS1307::getMinutes()
{
  //could mask with 0x7f but shouldn't need to
  return bcdToDec(_reg1_min);
}
int RealTimeClockDS1307::getSeconds()
{
  //need to mask oscillator start/stop bit 7
  return bcdToDec(_reg0_sec & 0x7f);
}
int RealTimeClockDS1307::getYear()
{
  return bcdToDec(_reg6_year);
```

```
}
int RealTimeClockDS1307::getMonth()
{
  //could mask with 0x1f but shouldn't need to
  return bcdToDec(_reg5_month);
}
int RealTimeClockDS1307::getDate()
{
  //could mask with 0x3f but shouldn't need to
  return bcdToDec(_reg4_date);
}
int RealTimeClockDS1307::getDayOfWeek()
{
  //could mask with 0x07 but shouldn't need to
  return bcdToDec(_reg3_day);
}

void RealTimeClockDS1307::getFormatted(char * buffer)
{
  int i=0;
  //target string format: YY-MM-DD HH:II:SS
  buffer[i++]=highNybbleToASCII(_reg6_year);
  buffer[i++]=lowNybbleToASCII(_reg6_year);
  buffer[i++]='-';
  buffer[i++]=highNybbleToASCII(_reg5_month & 0x1f);
  buffer[i++]=lowNybbleToASCII(_reg5_month);
  buffer[i++]='-';
  buffer[i++]=highNybbleToASCII(_reg4_date & 0x3f);
  buffer[i++]=lowNybbleToASCII(_reg4_date);
  buffer[i++]=' ';
  if(is12hour()) {
    buffer[i++]=highNybbleToASCII(_reg2_hour & 0x1f);
  } else {
    buffer[i++]=highNybbleToASCII(_reg2_hour & 0x3f);
  }
  buffer[i++]=lowNybbleToASCII(_reg2_hour);
  buffer[i++]=':';
  buffer[i++]=highNybbleToASCII(_reg1_min & 0x7f);
  buffer[i++]=lowNybbleToASCII(_reg1_min);
  buffer[i++]=':';
  buffer[i++]=highNybbleToASCII(_reg0_sec & 0x7f);
  buffer[i++]=lowNybbleToASCII(_reg0_sec);
  if(is12hour()) {
    if(isPM()) {
      buffer[i++]='P';
    } else {
      buffer[i++]='A';
    }
  }
  buffer[i++]=0x00;
}

void RealTimeClockDS1307::getFormatted2k(char * buffer)
{
  buffer[0]='2';
```

```
  buffer[1]='0';
  getFormatted(&buffer[2]);
}

/**** SETTERS *****/

void RealTimeClockDS1307::setSeconds(int s)
{
  if (s < 60 && s >=0)
  {
    //need to preserve oscillator bit
    _reg0_sec = decToBcd(s) | (_reg0_sec & 0x80);
  }
}
void RealTimeClockDS1307::setMinutes(int m)
{
  if (m < 60 && m >=0)
  {
    _reg1_min = decToBcd(m);
  }
}
void RealTimeClockDS1307::setHours(int h)
{
  if (is12hour())
  {
    if (h >= 1 && h <=12)
    {
      //preserve 12/24 and AM/PM bits
      _reg2_hour = decToBcd(h) | (_reg2_hour & 0x60);
    }
  } else {
    if (h >= 0 && h <=24)
   {
      //preserve 12/24 bit
      _reg2_hour = decToBcd(h) | (_reg2_hour & 0x40);
   }
  }//else
}//setHours

void RealTimeClockDS1307::set24h()
{
  //"Bit 6 of the hours register is defined as the
  //"12- or 24-hour mode select bit.
  //"When high, the 12-hour mode is selected"
  //So, mask the curent value with the complement turn off
that bit:
  _reg2_hour = _reg2_hour & ~0x40;
}
void RealTimeClockDS1307::setAM()
{
  //"In the 12-hour mode, bit 5 is the AM/PM bit with logic
high being PM"
  //so we need to OR with 0x40 to set 12-hour mode and also
  //turn off the PM bit by masking with the complement
  _reg2_hour = _reg2_hour & ~0x20 | 0x40;
```

```
}
void RealTimeClockDS1307::setPM()
{
  //"In the 12-hour mode, bit 5 is the AM/PM bit with logic
high being PM"
  //so we need to OR with 0x40 and 0x20 to set 12-hour mode
and also
  //turn on the PM bit:
  _reg2_hour = _reg2_hour | 0x60;
}

void RealTimeClockDS1307::switchTo12h()
{
  if(is12hour()) { return; }
  int h = getHours();
  if (h < 12) {
    setAM();
  } else {
    h = h-12;
    setPM();
  }
  if (h==0)
  {
    h=12;
  }
  setHours(h);
}
void RealTimeClockDS1307::switchTo24h()
{
  if(!is12hour()) { return ; }
  int h = getHours();
  if(h==12) {//12 PM is just 12; 12 AM is 0 hours.
    h = 0;
  }
  if (isPM())
  {//if it was 12 PM, then h=0 above and so we're back to
12:
    h = h+12;
  }
  set24h();
  setHours(h);
}

void RealTimeClockDS1307::setDayOfWeek(int d)
{
  if (d > 0 && d < 8)
  {
    _reg3_day = decToBcd(d);
  }
}

void RealTimeClockDS1307::setDate(int d)
{
  if (d > 0 && d < 32)
  {
```

```
    _reg4_date = decToBcd(d);
  }
}
void RealTimeClockDS1307::setMonth(int m)
{
  if (m > 0 && m < 13)
  {
    _reg5_month = decToBcd(m);
  }
}
void RealTimeClockDS1307::setYear(int y)
{
  if (y >= 0 && y <100)
  {
    _reg6_year = decToBcd(y);
  }
}



/****************************************
* Private methods
****************************************/
byte RealTimeClockDS1307::decToBcd(byte b)
{
  return ( ((b/10) << 4) + (b%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte RealTimeClockDS1307::bcdToDec(byte b)
{
  return ( ((b >> 4)*10) + (b%16) );
}

char RealTimeClockDS1307::lowNybbleToASCII(byte b)
{
  b = b & 0x0f;
  if(b < 10) {
    //0 is ASCII 48
    return 48+b;
  }
  //A is ASCII 55
  return 55+b;
}
char RealTimeClockDS1307::highNybbleToASCII(byte b)
{
  return lowNybbleToASCII(b >> 4);
}

/***** INSTANCE *******/

RealTimeClockDS1307 RTC = RealTimeClockDS1307();
```

12.2 Sample Sketch using RealTimeClockDS1307

12.2.1 RealTimeClockDS1307_Test.ino

This is David Brown's comprehensive test sketch

```
/*
RealTimeClockDS1307 - library to control a DS1307 RTC
module
Copyright (c) 2011 David H. Brown. All rights reserved
Much thanks to John Waters and Maurice Ribble for their
earlier and very helpful work (even if I didn't wind up
using any of their code):
- http://combustory.com/wiki/index.php/RTC1307_-
_Real_Time_Clock
- http://www.glacialwanderer.com/hobbyrobotics/?p=12

This library is free software; you can redistribute it
and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation;
either
version 2.1 of the License, or (at your option) any later
version.

This library is distributed in the hope that it will be
useful,
but WITHOUT ANY WARRANTY; without even the implied warranty
of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public
License along with this library; if not, write to the Free
Software
Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301 USA
*/


#include <Wire.h>
#include <RealTimeClockDS1307.h>

//RealTimeClock RTC;//=new RealTimeClock();

#define Display_Clock_Every_N_Seconds 1
#define Display_ShortHelp_Every_N_Seconds 25
//#define TEST_Squarewave
//#define TEST_StopStart
//#define TEST_1224Switch

int count=0;
char formatted[] = "00-00-00 00:00:00x";
```

```
void setup() {
// Wire.begin();
  Serial.begin(9600);
}

void loop() {
  if(Serial.available())
  {
    processCommand();
  }
  delay(1000);
  RTC.readClock();
  count++;
  if(count % Display_Clock_Every_N_Seconds == 0){
    Serial.print(count);
    Serial.print(": ");
    RTC.getFormatted(formatted);
    Serial.print(formatted);
    Serial.println();
  }

  if(count % Display_ShortHelp_Every_N_Seconds == 0) {
    Serial.println("Send ? for a list of commands.");
  }
#ifdef TEST_Squarewave
if(count%10 == 0)
{
  switch(count/10 % 6)
  {
    case 0:
    Serial.print("Squarewave disabled (low impedance): ");
    RTC.sqwDisable(0);
    Serial.println((int) RTC.readData(7));
    break;
    case 1:
    Serial.print("Squarewave disabled (high impedance): ");
    RTC.sqwDisable(1);
    Serial.println((int) RTC.readData(7));
    break;
    case 2:
    Serial.println("Squarewave enabled at 1 Hz");
    RTC.sqwEnable(RTC.SQW_1Hz);
    break;
    case 3:
    Serial.println("Squarewave enabled at 4.096 kHz");
    RTC.sqwEnable(RTC.SQW_4kHz);
    break;
    case 4:
    Serial.println("Squarewave enabled at 8.192 kHz");
    RTC.sqwEnable(RTC.SQW_8kHz);
    break;
    case 5:
    Serial.println("Squarewave enabled at 32.768 kHz");
    RTC.sqwEnable(RTC.SQW_32kHz);
```

```
    break;
    default:
    Serial.println("Squarewave test not defined");
  }//switch
}
#endif

#ifdef TEST_StopStart
if(count%10 == 0)
{
  if(!RTC.isStopped())
  {
    if(RTC.getSeconds() < 45)
    {
      Serial.println("Stopping clock for 10 seconds");
      RTC.stop();
    }//if we have enough time
  } else {
    RTC.setSeconds(RTC.getSeconds()+11);
    RTC.start();
    Serial.println("Adding 11 seconds and restarting
clock");
  }
}//if on a multiple of 10 counts
#endif

#ifdef TEST_1224Switch
  if(count%10 == 0)
  {
    if(count %20 == 0)
    {
      Serial.println("switching to 12-hour time");
      RTC.switchTo12h();
      RTC.setClock();
    }
    else
    {
      Serial.println("switching to 24-hour time");
      RTC.switchTo24h();
      RTC.setClock();
    }
  }
#endif
}

void processCommand() {
  if(!Serial.available()) { return; }
  char command = Serial.read();
  int in,in2;
  switch(command)
  {
    case 'H':
    case 'h':
    in=SerialReadPosInt();
    RTC.setHours(in);
```

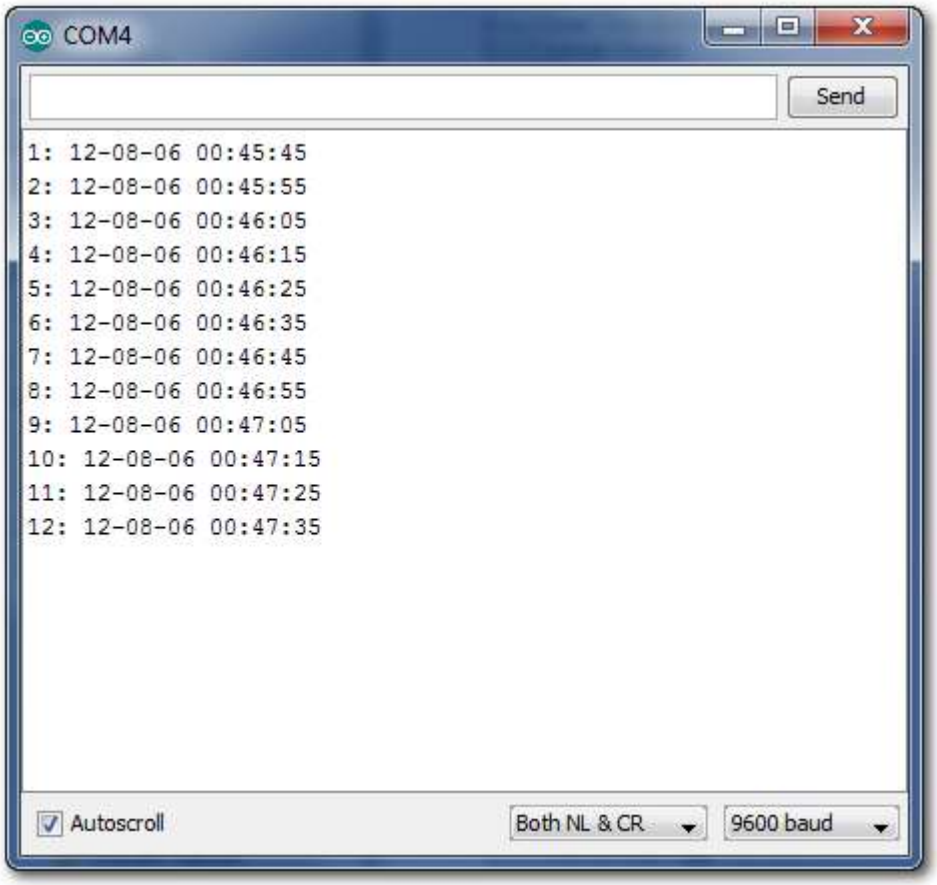
```
    RTC.setClock();
    Serial.print("Setting hours to ");
    Serial.println(in);
    break;
    case 'I':
    case 'i':
    in=SerialReadPosInt();
    RTC.setMinutes(in);
    RTC.setClock();
    Serial.print("Setting minutes to ");
    Serial.println(in);
    break;
    case 'S':
    case 's':
    in=SerialReadPosInt();
    RTC.setSeconds(in);
    RTC.setClock();
    Serial.print("Setting seconds to ");
    Serial.println(in);
    break;
    case 'Y':
    case 'y':
    in=SerialReadPosInt();
    RTC.setYear(in);
    RTC.setClock();
    Serial.print("Setting year to ");
    Serial.println(in);
    break;
    case 'M':
    case 'm':
    in=SerialReadPosInt();
    RTC.setMonth(in);
    RTC.setClock();
    Serial.print("Setting month to ");
    Serial.println(in);
    break;
    case 'D':
    case 'd':
    in=SerialReadPosInt();
    RTC.setDate(in);
    RTC.setClock();
    Serial.print("Setting date to ");
    Serial.println(in);
    break;
    case 'W':
    Serial.print("Day of week is ");
    Serial.println((int) RTC.getDayOfWeek());
    break;
    case 'w':
    in=SerialReadPosInt();
    RTC.setDayOfWeek(in);
    RTC.setClock();
    Serial.print("Setting day of week to ");
    Serial.println(in);
    break;
```

```
case 't':
case 'T':
if(RTC.is12hour()) {
  RTC.switchTo24h();
  Serial.println("Switching to 24-hour clock.");
} else {
  RTC.switchTo12h();
  Serial.println("Switching to 12-hour clock.");
}
RTC.setClock();
break;

case 'A':
case 'a':
if(RTC.is12hour()) {
  RTC.setAM();
  RTC.setClock();
  Serial.println("Set AM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'P':
case 'p':
if(RTC.is12hour()) {
  RTC.setPM();
  RTC.setClock();
  Serial.println("Set PM.");
} else {
  Serial.println("(Set hours only in 24-hour mode.)");
}
break;

case 'q':
RTC.sqwEnable(RTC.SQW_1Hz);
Serial.println("Square wave output set to 1Hz");
break;
case 'Q':
RTC.sqwDisable(0);
Serial.println("Square wave output disabled (low)");
break;

case 'z':
RTC.start();
Serial.println("Clock oscillator started.");
break;
case 'Z':
RTC.stop();
Serial.println("Clock oscillator stopped.");
break;

case '>':
in=SerialReadPosInt();
```

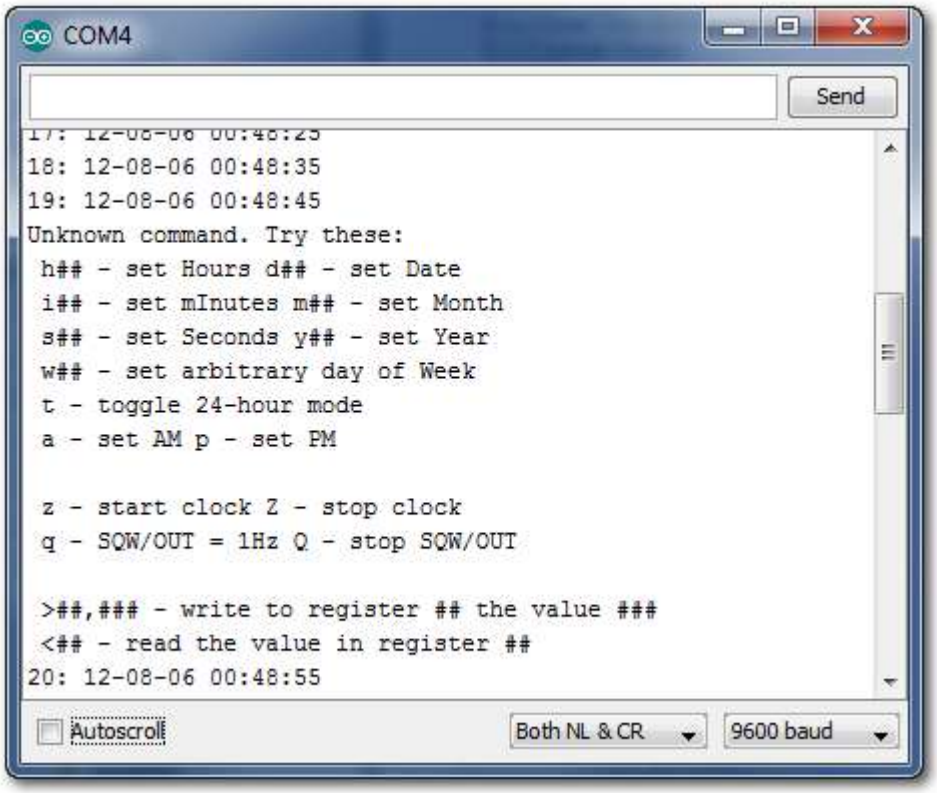
```
    in2=SerialReadPosInt();
    RTC.writeData(in, in2);
    Serial.print("Write to register ");
    Serial.print(in);
    Serial.print(" the value ");
    Serial.println(in2);
    break;
    case '<':
    in=SerialReadPosInt();
    in2=RTC.readData(in);
    Serial.print("Read from register ");
    Serial.print(in);
    Serial.print(" the value ");
    Serial.println(in2);
    break;

    default:
    Serial.println("Unknown command. Try these:");
    Serial.println(" h## - set Hours d## - set Date");
    Serial.println(" i## - set mInutes m## - set Month");
    Serial.println(" s## - set Seconds y## - set Year");
    Serial.println(" w## - set arbitrary day of Week");
    Serial.println(" t - toggle 24-hour mode");
    Serial.println(" a - set AM p - set PM");
    Serial.println();
    Serial.println(" z - start clock Z - stop clock");
    Serial.println(" q - SQW/OUT = 1Hz Q - stop SQW/OUT");
    Serial.println();
    Serial.println(" >##,### - write to register ## the
value ###");
    Serial.println(" <## - read the value in register ##");

  }//switch on command

}

//read in numeric characters until something else
//or no more data is available on serial.
int SerialReadPosInt() {
  int i = 0;
  boolean done=false;
  while(Serial.available() && !done)
  {
    char c = Serial.read();
    if (c >= '0' && c <='9')
    {
      i = i * 10 + (c-'0');
    }
    else
    {
      done = true;
    }
  }
  return i;
}
```

12.2.2 Sample Output

Default output after initialisation:

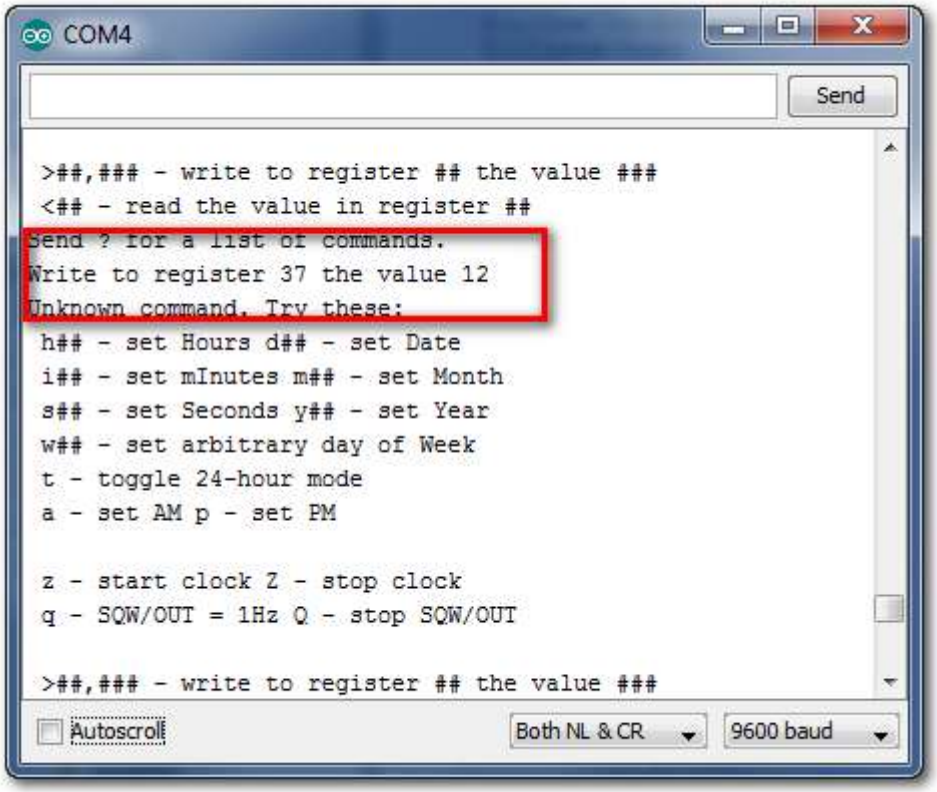


Output after sending "?" to obtain list of available commands:

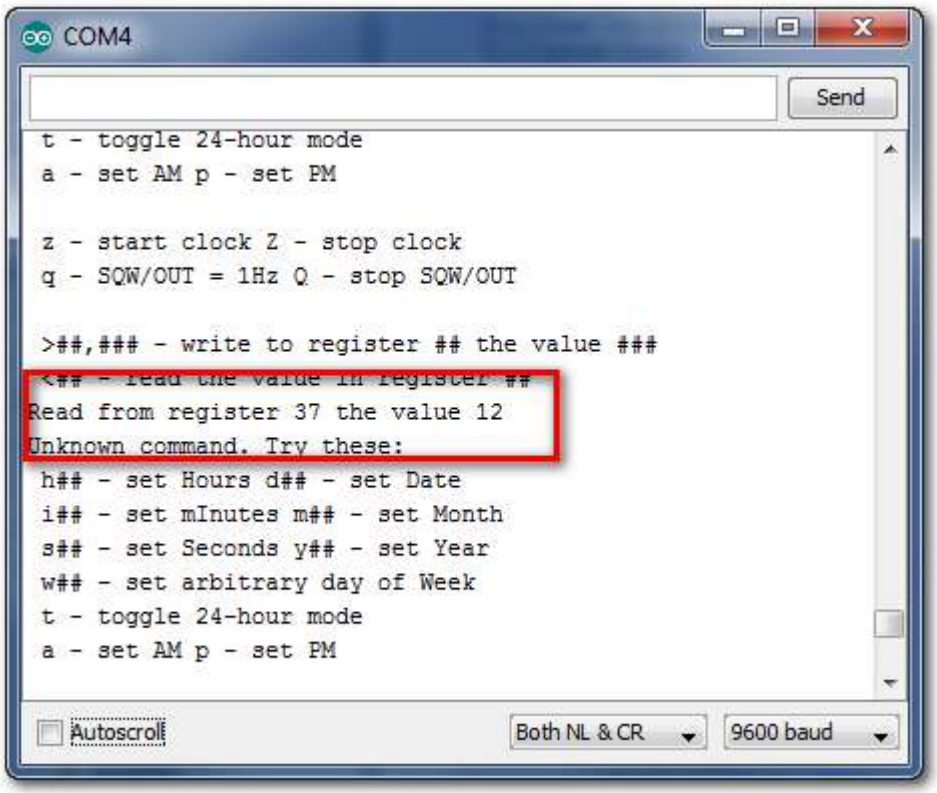


The list of available commands is repeated three times before resuming current date-time output. here is the write, followed by the read commands:

Place the value 12 in register 37 → ">37,12"



Read the value in register 37 → "<37"



12.2.3 Notes

To obtain the sample output shown above I had to reduce the frequency by which the serial monitor display is updated:

Change:

```
void loop() {
  if(Serial.available())
  {
    processCommand();
  }
  delay(1000);
```

to

```
void loop() {
  if(Serial.available())
  {
    processCommand();
  }
  delay(10000);
```

This provides an update every 10 seconds.