Trinket RGB Shield Clock

Created by Anne Barela

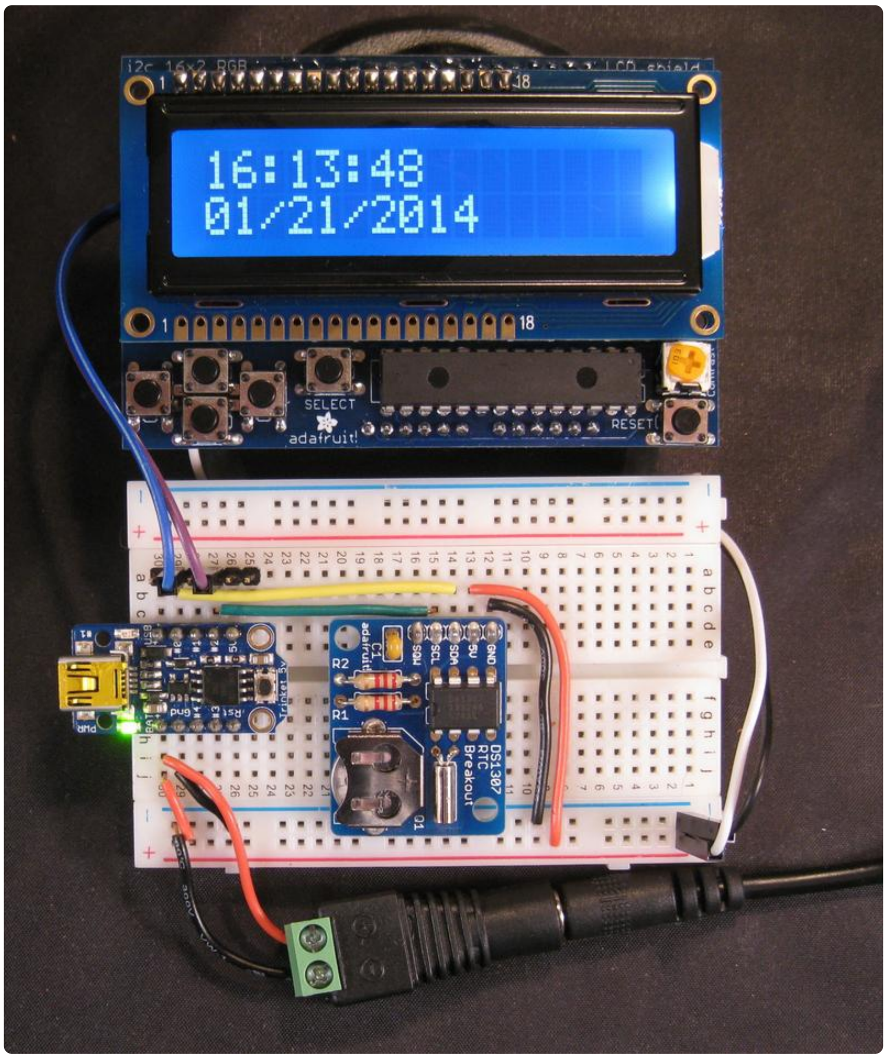
https://learn.adafruit.com/trinket-rgb-shield-clock

Last updated on 2022-12-01 02:07:20 PM EST

Table of Contents

Overview

This project was inspired by a forum member who asked a simple question: Can you interface a Trinket mini microcontroller to the Adafruit RGB LCD Shield. The shield is made to interface with more "classic" Arduino microcontrollers with an Arduino standard shield pin layout. Obviously the shield cannot stack onto Trinket but with four wires, the display shield can hook up to a Trinket project well. This is accomplished as both use the I2C or two-wire bus to communicate. As a further demonstration, the Adafruit I2C based DS1307 real-time clock module is used to display the time and date. The display shield's buttons allow for changing the hour in case of daylight savings time and toggle the backlight.



Update 2016: Adafruit has updated their libraries to allow the Trinket's ATtiny85 to use the same libraries as the Arduino Uno and other boards. This tutorial updates the libraries and code to reflect this.

Libraries

The project uses three code libraries:

- Wire.h - the Arduino standard library for I2C/two wire communication (already installed with the Arduino IDE)
- RTClib () - Used to communicate with the DS1307 real-time clock
- Adafruit_RGBLCDShield () - communicates with the Adafruit RGB LCD Shield

You may click on the links to download from the Adafruit Github repository.

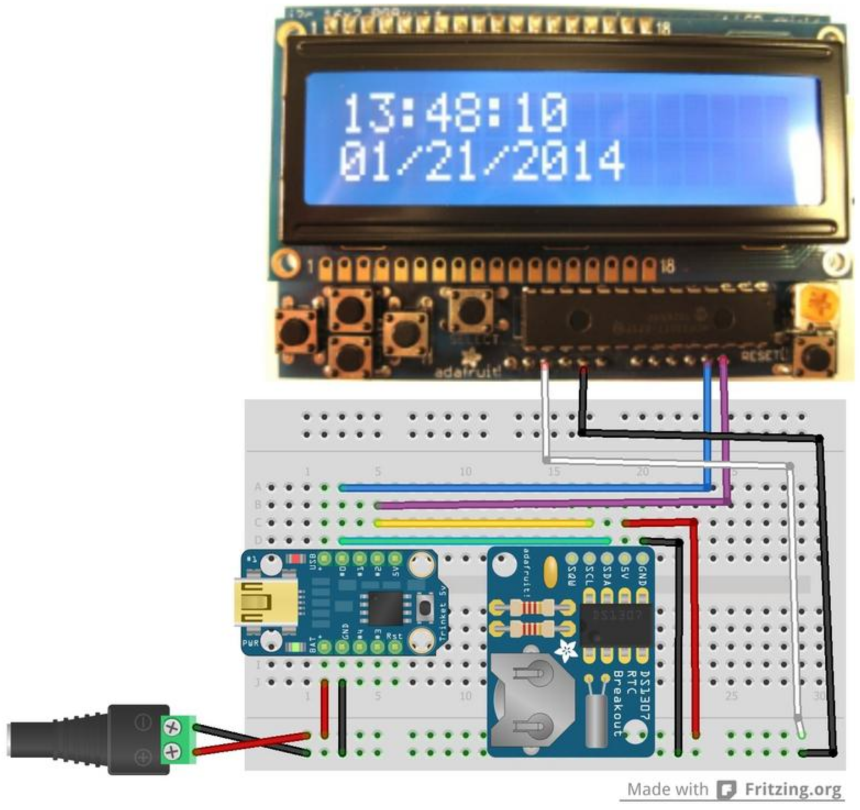
These should be installed in your Arduino folder where your sketches are stored in the libraries folder. For an in-depth discussion of installing and managing Arduino libraries, see tutorial All About Arduino Libraries ().

Getting Ready

If you use the Arduino IDE version 1.6.5 or later, you can download the Adafruit board file in the Tools -> Board -> Board Manager. You will then be able to select "Trinket 8 MHz" as a board and all is well.

If you use an older Arduino IDE, you must follow the instructions in the Introducing Trinket tutorial to set up your Arduino development environment properly. Failure to do all the steps will not allow you to program a Trinket properly.

Hook-up



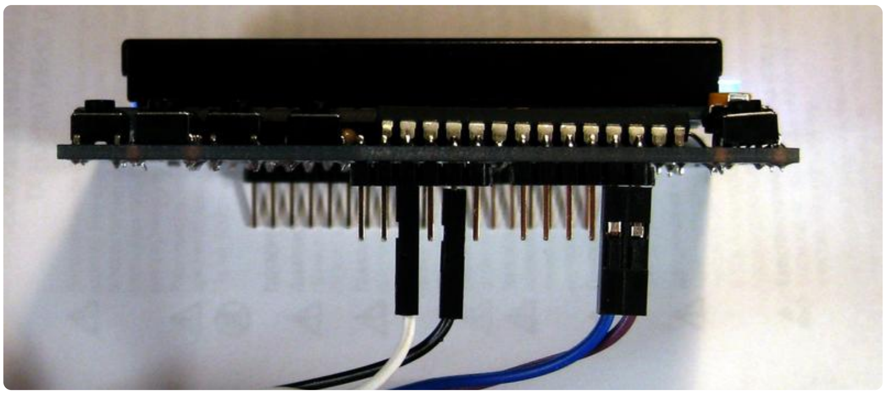
Since we're using I2C for the shield and real time clock, hookup is fairly straightforward. Don't forget, I2C allows you to use multiple devices on two shared pins, perfect for when you don't have a lot of pins like the Trinket!

1) Assemble the RGB LCD Shield per this tutorial ()
2) Assemble the DS1307 clock per this tutorial ()
3) Solder supplied male headers to Trinket

Wiring

All components are hooked to 5 volts and ground. Note the ground pin used for the shield (the ground pin next to the 5 volt pin will NOT work, it isn't connected!)

Hookup of the display is easier with some male-female extension wires (http://adafru.it/824) especially if the shield has male headers already soldered on. For breadboarding this allows the display to just plug in.



The clock and display are connected to Pin 0 and Pin 2 on Trinket. Pin 0 is the I2c data line (SDA). Pin 2 is the I2C clock line (SCL). Both the display and the clock can share this bus as they each have a different address assigned during manufacture. The sharing of the bus saves pins in a Trinket project. Use of the buttons on the LCD shield is also through the I2C bus, again saving pins.

If an LCD display with I2C backpack were substituted for the shield, other data pins on Trinket could be used for buttons. Use of an I2C expander chip such as the MCP23008 (http://adafru.it/593) or MCP23017 (http://adafru.it/732) to gain an additional 8 or 16 pins respectively.

You can easily fit the parts for permanent mounting on a half perma-proto board (http://adafru.it/1609), perhaps a quarter perma-proto board (http://adafru.it/1608) with some moving of parts. If you decide on a permanent mount, I suggest use of female header (http://adafru.it/598) to mount the Trinket, perhaps even the clock module, in case the part has issues you can quickly swap it out.

Code

Two programs are used to save code space. The first one should be needed only once to set the battery-backed DS1307 real-time clock. It may also be needed if the battery runs out but the battery life is expected to be quite long, like 7 years or so. It sets the clock according to the Arduino date and time so the system clock on your computer should be accurate for this to set correctly.

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

Sketch to set the time and date for the DS1307 Real Time Clock
  with an Adafruit Trinket mini microcontroller

**********************/

// include the library code:
#include &lt;Wire.h&gt;
#include &lt;RTClib.h&gt;

RTC_DS1307 RTC;
```

```
void setup() {
  RTC.begin();
  if(!RTC.isrunning()) {
     RTC.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop() {
}
```

The main clock code is below.

```
/*********************
Trinket RGB LCD Shield Clock

Example code for the Adafruit RGB Character LCD Shield and Library
for Trinket

The DS1307 Real Time Clock must be initialized with a separate sketch.

Version 2.0 Use with Arduino IDE Version 1.6.5 or later and Adafruit
   libraries modified December 2015 or later.  Mike Barela for Adafruit.
**********************/

// include the library code:

#include &lt;Wire.h&gt;
#include &lt;Adafruit_RGBLCDShield.h&gt;  // RGB LCD Shield communications
#include &lt;RTClib.h&gt;                 // DS1307 clock communications

// These defines make it easy to set the backlight color
#define OFF 0x0
#define RED 0x1
#define YELLOW 0x3
#define GREEN 0x2
#define TEAL 0x6
#define BLUE 0x4
#define VIOLET 0x5
#define WHITE 0x7

// The shield uses the I2C SCL and SDA pins.
Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();

RTC_DS1307 RTC;      // Establish clock object
DateTime Clock;      // Holds current clock time
int8_t offset = 0;   // Hour offset set by user with buttons
uint8_t backlight = WHITE;  // Backlight state

void setup() {
  lcd.begin(16, 2);         // initialize display colums and rows
  RTC.begin();              // Initialize clock
  lcd.setBacklight(WHITE);  // Set to OFF if you do not want backlight on boot
}

void loop() {
  uint8_t buttons;                        // button read value
  uint8_t hourval, minuteval, secondval; // holds the time

  DateTime Clock;                     // variable to hold our time
  char* colon = ":";                  // static characters save a bit
  char* slash = "/";                  //   of memory

  Clock = RTC.now();                  // get the RTC time

  hourval = Clock.hour()+offset;      // calculate hour to display
  if(hourval &gt; 23) hourval-=24;      // adjust for over 23 hour
  else if(hourval &lt; 0) hourval+=24;  //   or under 0 hours

  minuteval = Clock.minute();        // This block prints the time
  secondval = Clock.second();        //  to the LCD Shield
  lcd.setCursor(0,0);
  if(hourval &lt; 10) printzero();      // print function does not print
  lcd.print(hourval);                //   leading zeros so this will
  lcd.print(colon);
  if(minuteval &lt; 10) printzero();
  lcd.print(minuteval);
  lcd.print(colon);
  if(secondval &lt; 10) printzero();
  lcd.print(secondval);

  buttons = lcd.readButtons();  // read the buttons on the shield

  if(buttons!=0) {                     // if a button was pressed
      if (buttons &amp; BUTTON_UP) {       // if up pressed, increment hours
         offset +=1;
       }
      if (buttons &amp; BUTTON_DOWN) {     // if down pressed, decrement hours
         offset -=1;
       }
      if (buttons &amp; BUTTON_SELECT) {   // if select button pressed
         if(backlight)                 // if the backlight is on
            backlight=OFF;             //   set it to off
         else                          // else turn on the backlight if off
            backlight=WHITE;           //   (you can select any color)
         lcd.setBacklight(backlight);  // set the new backlight state
      }
   }
  lcd.setCursor(0,1);                  // This block prints the date
  if(Clock.month()&lt;10) printzero();    //   to the LCD Shield
  lcd.print(Clock.month());
  lcd.print(slash);
  if(Clock.day()&lt;10) printzero();
  lcd.print(Clock.day());
  lcd.print(slash);
  lcd.print(Clock.year());

  delay(1000);  // wait one second
}

void printzero() {  // prints a zero to the LCD for leading zeros
  lcd.print("0");   // a function saves multiple calls to the print function
}
```

The code displays the clock value and polls the buttons. If the up or down buttons are pressed, the value offset is incremented/decremented. This is added to the RTC clock time to form the hour. A more robust program would have the hour written back to the DS1307 but that one function takes about 300+ bytes of code which is too much for our mighty Trinket.

Use and Going Further

Use It!

Supply 5 volts to the 2.1mm jack via a wall supply, or use a USB wall adapter and a long USB cable. Remember that initially the real-time clock module will not be set, if you have never programmed it. Run the first program to set the clock! The coin cell battery will provide timekeeping for 7 years.

Load the main program. You should see the time and date on the display after the 10 second bootloader time. If not, check your connections.

You can use the UP and DOWN buttons to adjust the hour for daylight savings time.

You can use the SELECT button to turn on or off the LCD backlight (useful for night time use). If you want a different color display, the RGB shield provides 8 different colors noted in the code.

Since the buttons are only polled every second, the buttons may not seem to register easily. Press and hold for a second and the desired function should happen.

Going Further

The combination of Trinket and the RGB LCD Shield is a good combination for display and input. There is enough code space to hook a number of sensors for real-time readout. If you believe the shield form factor is not ideal, use of the LCD with the I2C backpack is a good combination. See the tutorial for the Trinket Ultrasonic Rangefinder () as an example.

The combination of the Trinket, display, and DS1307 clock makes for some trade-offs for projects. The amount of code space is limited to 5,310 bytes so you can easily go over if you include too many display or clock function calls. Even updating the real-time clock with RTC.adjust takes precious code, hence the use of an offset in the program here. The same with alarm functionality. Code optimization is crucial: use of floating point numbers is probably not possible. Use of signed integers even adds code. If you have heavy code, moving up to Arduino Uno might be necessary. Trinket provides a significant benefit in terms of cost and size if your project can fit in the code space available.

If you want a more precise clock, you can swap the DS1307 for a Chronodot (http://adafru.it/255), its code-compatible and is ultra-precise!