A GPS OLED Clock Project

The GPS OLED Clock project describes a highly accurate 24 h clock using a UBLOX-NEO6M GPS module, a 1.3″ OLED display and an Atmel ATmega328P micro-controller. The parts are readily available from internet stores at reasonable cost. The clock shows the local time on the OLED display in a 24 h format with a Verdana 24 pixel high,variable width font. The project is easy to build and the executable file (Intel hex format) is provided for programming to the micro-controller's flash memory. The project is written in assembly language and the source code file is included.

Date: 2018 January 13

Revision 0, 2018-01-13

Table of Contents

1 Introduction

This project describes a simple GPS based clock with time shown on a 1.3" OLED display. Time is shown in a 24 h format, corrected for local time. The GPS NEO-6M module provides accurate UTC (Coordinated Universal Time) data and the software converts this to local time using a hard coded time zone offset. The time zone offset can be changed to match the time zone where the clock is used. A shorting jumper selects daylight saving (DST) time adjustment where DST is used.

The clock runs off an externally supplied +5 Vdc power. The GPS clock project is relatively easy to build and is completely stand-alone in operation. The clock has a default time zone of -5 h, a value chosen because the clock was tested in the Eastern Standard Time zone.

The GPS clock is based on the UBlox NEO6M GPS module and an Atmel ATmega328P-PU micro-controller chip. (You could use an Arduino ATmega328P controller board in place of the ATmega328P chip, but this is overkill for such a simple project). All the hardware components used for this project are readily available from numerous sellers on eBay and other on-line retailers.

The executable file (gps-oled-clock.hex) for the GPS clock project is provided, ready for flashing to the ATmega328P device. Program source code is also included. A method for programming the ATmega328P chip using a USB ISP programmer and the AVRDude software tool is described in this manual.

The NEO-6M GPS module is a complete GPS receiver module with 3.3 V Rx and Tx input/output signals. The Tx output pin on the GPS module transmits several NMEA "sentences" once a second. This output is connected to the micro-controller's Rx input pin. The RX input on the GPS module is not used in this application and is left unconnected.

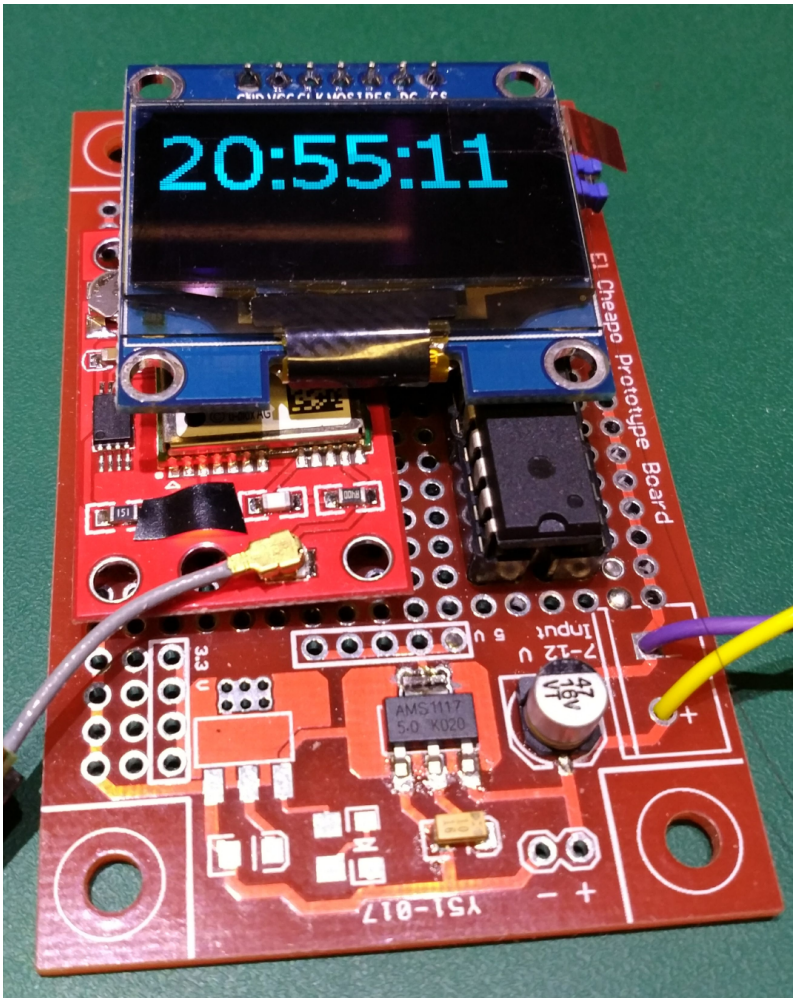
The clock program scans the incoming burst of NMEA "sentences" for a specific $GPRMC sentence. An example of this sentence is shown here:

"$GPRMC,213226.00,V,,,,,,,250616,,,N*7D"

The $GPRMC sentence is the "Recommended minimum specific GPS/Transit data", and is identified by the $GPRMC header. This sentence contains

several fields including the time (213226.00) field and the date (250716) field. All NMEA sentences consists of ASCII printable characters. The clock program extracts the time fields on the fly and saves the ASCII encoded data in internal RAM buffers.

The received time ASCII encoded string data in the RAM buffers is processed to apply time zone offset and DST corrections, then sent to the OLED module for display.



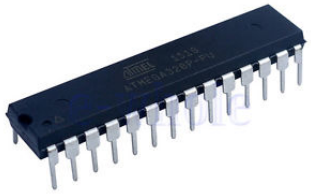
The finished GPS OLED clock.

2 Parts list

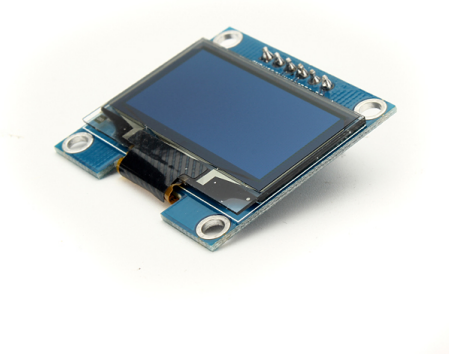
You need the following parts for the gps oled clock project:

1. Atmel ATmega328P-PU microcontroller chip, 28 pin DIP package.
2. uBLOX-6M GPS module. Module runs on 5 V supply. Only the TX signal is connected to the micro-controller in this application.
3. 5 V power supply. A 5 V Cell phone charger is suitable.
4. 1.3″ OLED display module. Modules with either the SSD1306 or SH1106 graphic controller chips can be used. The smaller size 0.96″ OLED displays can be used as well.
5. Miscellaneous hardware:
    1. Small piece of perforated prototyping board.
    2. One 2 pin header and a shorting jumper.
    3. One 47 uF/16 V tantalum or electrolytic capacitor.
    4. One 10 k, 1/4 W resistor.
    5. Wire-wrap wire, 20 pin and 8 pin DIP sockets, etc.

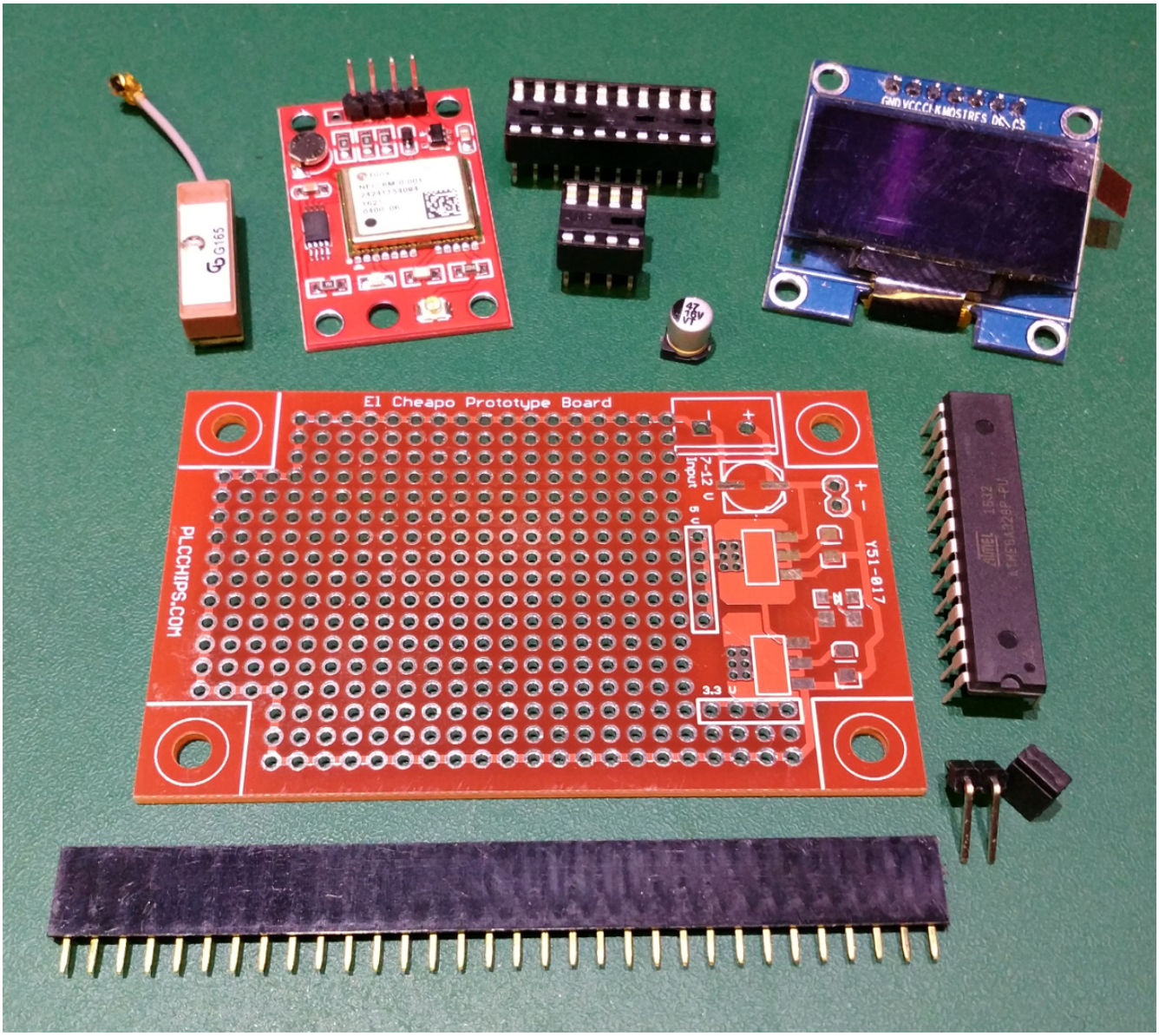
The uBLOX-NEO-6M GPS module and antenna.



The ATmega328P micro-controller



OLED Display module



GPS OLED clock project parts.

3 Construction details

The clock is built on a small piece of vector board on which the modules and components are mounted and wired with 30 AWG wire-wrap kynar insulated wire. The board used in the prototype clock has solder pads arranged on a 0.1 x 0.1 inch grid pattern with plated-though holes.

The NEO-6M is a 3.3 V device and the Tx and Rx signal pins operate at a 3.3 V level. The NEO-6M module operates on +5 V power. The module has an on-board voltage regulator.

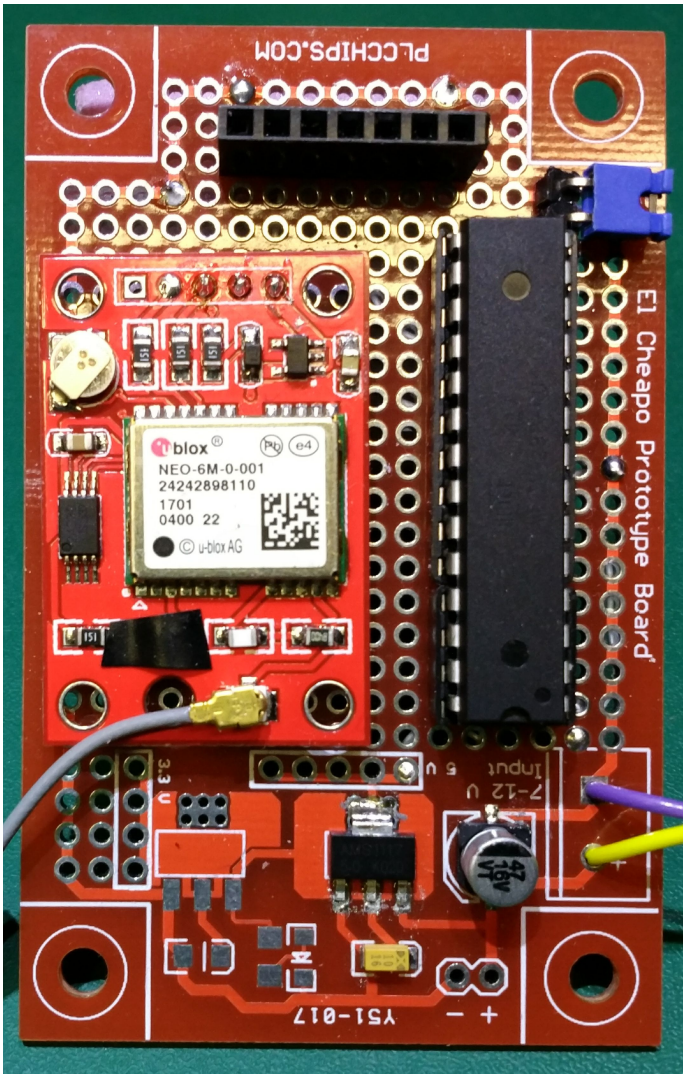
The Rx and Tx I/O on the ATmega328P are 5 V compatible. The Tx output from the NEO-6M can drive the Rx input on the ATmega328P as this input can accept the 3.3 V level output from the NEO-6M.

Refer to the GPS-oled-clock schematic diagram for connection details between NEO-6M and the ATmega328P micro-controller. Only three connections are required, Vcc, GND and TX.

The picture below shows the prototype clock layout. A 5 V regulator and SMT bypass capacitor is shown on the prototype board although these are not shown on the schematic diagram. These additional parts are not required as the clock runs directly from an externally connected 5 V(dc). The custom board in the prototype supports 5 V and 3.3V regulators for general purpose use.

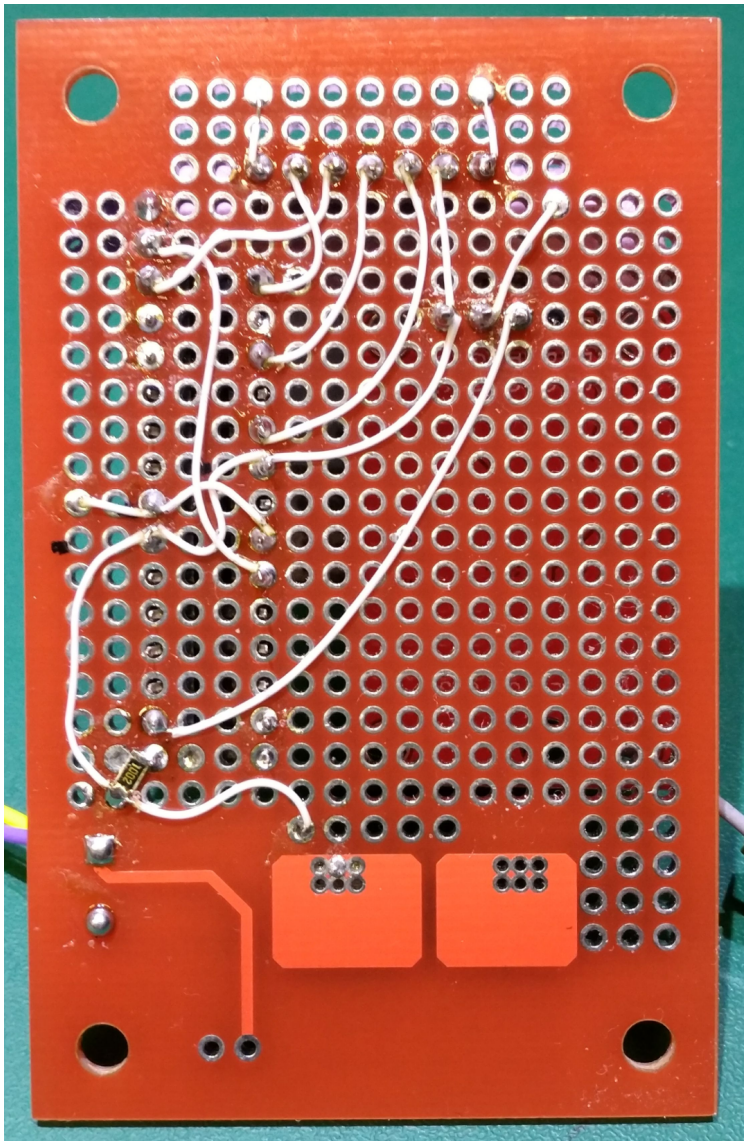
SMT components are used in the prototype but you can use leaded parts instead as these are much easier to handle. The 10 K SMT resistor is soldered directly to the RESET* pin of the ATmega328P device, between two through hole pads.

On the top right hand corner of the board you can see the purple jumper on the two pin header for the DST_en selection. There is a small piece of black electrical tape covering the LED on the NEO-6M module to block off the light which was annoying in low ambient light conditions.



GPS OLED clock layout — front view.

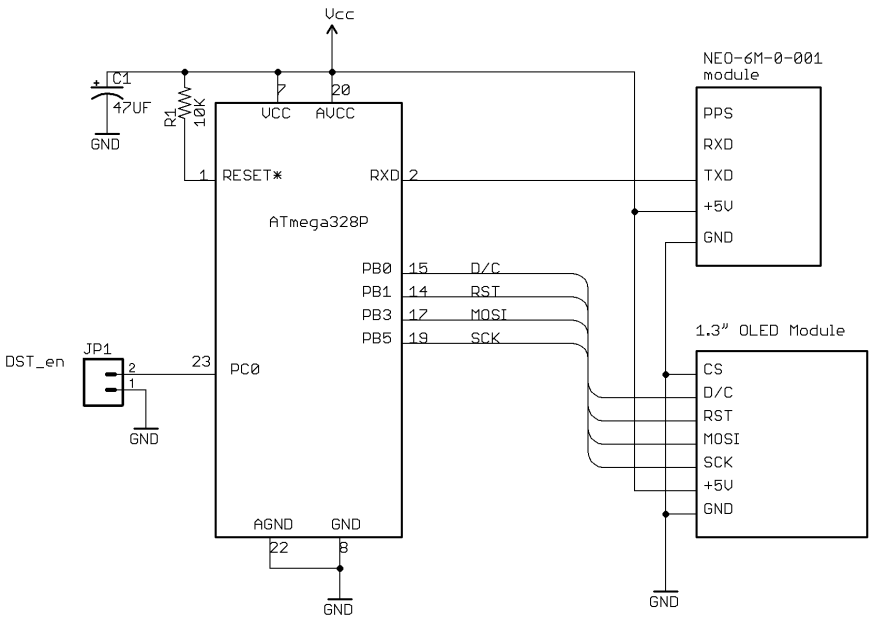
The photograph below shows the wiring details at the back of the clock.



GPS OLED clock wiring — back view.

4 Putting the project together

The shcematic diagram below shows the wiring connections between the ATmega328P MICRO-controller, the NEO-6M module, the OLED module and other components.



Notes: 1. Remove DST_en ground jumper to enable DST
2. Arduino pin numbers are for ATmega328P-PU device.
3. OLED module (SPI interface) with SSD1306 or SH1106 controllers.

TITLE: GPS-OLED-clock	
Design by: Lynf	REV: 0
Date: 1/14/18 10:11 AM	Sheet: 1/1

4.1 Assembly steps

The schematic diagram shows the number of wiring connections are small so the effort required to wire the clock should be small. Please take the time to carefully arrange all the modules and components on the board before you start the wiring. You can use the prototype clock layout shown in the photographs as a guide.

When you have completed the wiring of the components and modules, carefully check the wiring against the schematic diagram again and make sure there are no wiring errors. Look especially at the +5 V power and ground wiring as this type of check-out is one of the easient ways to avoid experiencing a “smoke” event.

Do not apply power to the CLOCK board as yet. At this time you are ready to program the gps-oled-clock.hex file you downloaded from the github.com web site to the ATmega328P's flash memory. The programming method is described in the next section.

Go to the github repository at the link below and download all the files for the gps-oled-clock project.

https://github.com/lynf/GPS-OLED-clock

4.2 Burning the gps-oled-clock.hex program.

The programming method used to program the executable hex file to the ATmega328P's flash memory uses the AVRDude programming software and a USB-ISP-programmer avaialable from many on-line retailers.

The easiest way to connect the programmer to the ATmega328P micro-controller is to plug the chip into an Arduino UNO controller board, then plug in the USB-ISP-programmer cable/connector to the 6-pin ISP header on the UNO board. After the device and fuse bits are programmed, remove the micro-controller from the UNO and plug it into 28 pin DIP socket on the gps-oled-clock board.

First, download and install on your PC the AVRDude software. There are numerous excellent articles on the web describing how to install

and configure AVRdude so we'll skip repeating the same thing here. If you have trouble finding the software, take a look at this site:

http://winavr.sourceforge.net/download.html

The tricky part to programming the ATmega328P using AVRDude is to invoke the proper command line instruction. We'll get to this in a moment.

You need a USB-ASP-Programmer, a typical one is shown in the picture below. You should also get a 10-pin socket to 6-pin plug adapter to make it easier to connect the 10-pin ribbon cable from the programmer to the 6 pin ISP header on the ATmega328P board. The USB programmer is recognized by AVRDude as programmer type usbasp. See the following site for further details of this programmer.

http://www.fischl.de/usbasp/

We'll assume you are running in a Windows environment and that you placed the gps-oled-clock.hex executable program image file in a sub-directory such as:

```
c:\Users\yourname\My Documents\avr\gps-oled-clock.hex
```

When you invoke AVRDude be sure to specify the full path to the directory where avrdude.exe resides. We'll asume for this discussion that the avrdude files are in:

```
c:\opt\avr\avrdude.exe
```

The avrdude.conf and avrdude.rc files are also in this sub-directory.

When you are ready to program the ATmega328P's flash with the gps-oled-clock.hex executable image, plug the programmer into a USB port on the PC and plug the ribbon cable with the 10-to-6 pin adapter onto the 6-pin ISP programming header on the Arduino UNO board. Make sure you connect the adapter the right way round (adapter socket 1 connected to ISP header pin 1). The programmer will supply 5 V power to the UNO board and the power indicator LED will light up.

Open up a command line interface in a Windows terminal and go to the subdirectory c:\Users\yourname\My Documents\avr\ where the gps-oled-clock.hex file is located. Type in the following command line on a windows terminal:

"c:\opt\avr\avrdude.exe" -C"c:\opt\avr\avrdude.conf" -p m328 -c usbasp -u -U flash:w:gps-oled-clock.hex:i

AVRDude will perform a number of programming steps and print some messages on the console screen while it uploads the gps-oled-clock.hex file to the ATmega328P's flash memory. If the programming is successful, you'll see the program verification step completed.

As a side note to AVRDude use, a very useful companion program to have in your toolbox is the AVRdudess, a GUI for AVRDude. Check out this link if you are interested:

http://blog.zakkemble.co.uk/avrdudess-a-gui-for-avrdude/

4.3 Programming the ATmega328P Fuse Bits

The ATmega328P micro-controller is used with the internal R/C oscillator in this project to avoid the need for an external clock crystal and it is therefore necessary to set the fuse bits correctly.

Set the fuse bits in LFUSE byte as indicated below:

- CKSEL[3..0] = 0 0 1 0
- CKDIV = 1

The LFUSE byte and can be programmed to these required settings with

AVRDude by issuing the following command:

"c:\opt\avr\avrdude.exe" -C"c:\opt\avr\avrdude.conf" -p m328 -c usbasp -u -U lfuse:w:0xf2:m

Once you program the device fuse you need not reprogram it again if you need to re-flash the micro-controller with a new or modified executable program image.

Unplug the programmer from the UNO board, remove the flashed ATmega328P and plug it into the DIP socket on the clock board. Now go to the final step of powering up the GPS clock project as described in the following section.

4.4 Powering-up the GPS OLED Clock project

Once the ATmega328P controller is programmed with the gps-oled-clock.hex executable file and plugged into the gps-oled-clock board, you can apply power to the clock board's +5 V input.

When the GPS clock is first powered up it will take a minute or more for the GPS module to acquire the overhead satellites before it starts displaying the GPS derived time information on the OLED display. During this satellite acquisition phase, you may see some garbage numbers displayed. When the satellites are acquired, the display will begin to show the local time using the default -5 h time zone offset correction.

5 Configuring a new Time Zone setting.

The default time zone offset of -5 h is hard coded in the gps-oled-clock program.

If you are in a different time zone to the default time zone, you need to edit the gps-oled-clock.asm assembly source code file and change the following equates:

- .equ TZoff = (your time zone offset)
- .equ TZsign = 0 for negative or 1 for positive offset

Save the modified source code file and build (assemble and link) the assembly language project again. Re-flash the ATmega328P again with the newly generated gps-oled-clock.hex file.

The Atmel Studio 7 development suite was used for developing the gps-oled-clock program and it is highly recommended you use Studio 7 for modifying and re-building the assembly language project.

If you plan to modify the source code to set up a new time zone offset, you should first try out the default gps-oled-clock.hex executable file image from the repository to ensure your hardware is working correctly.

5.1 Adjusting for Daylight Saving Time

If you live in a country that observes daylight saving time (DST) you need to adjust the clock's local time forward by one hour at the beginning of Summer and backward again at the end of Summer.

Daylight Saving Time is observed in the Eastern Standard Time zone. The shorting jumper between the two pins DST_en input (PC0) and GND enables DST correction. When the jumper is installed, the clock shows DST time. When the jumper is removed, normal time is shown.

6 Final Thoughts

The GPS OLED clock project can be assembled by anyone with average construction skills in one or two evening's work. The project is ideal for those who ike building their own clock but may have hesitated because the software complexities involved may be beyond their current programming ability. Since the program is already written and tested, and as the source code is provided, there is no need to worry about the clock not working after it is built.

There are two remaining technical issues you may encounter with this project. The first issue concerns the flashing of the executable image provided to the ATmega328P flash memory using the AVRDude and ISP programmer. Most folks using Arduino boards rely on the built-in bootloader to flash the micro-controller, and may balk at using an unfamiliar programming method such as this.

The AVRDude/ISP programmer method used in this project is not difficult to use and is worthwhile to learn if you wish to step outside the constraints of the Arduino IDE program development environment. If you have questions or run int toruble you can ask for help on the many forums on how to program the ATmega328 device using the AVRDude/ISP programmer method.

The other possible issue with the project arises if you need to change the default time zone offset because you live in a different time zone, and you are not sure if you can handle this task because of your unfamiliarity with dealing with the Studio 7 IDE. There is a bit of a learning curve for Studio 7 but the effort to gain proficiency is well worth-while. Studio 7 can handle C, C++ as well as assembly language projects. The Studio 7 help is fairly detailed, and you can seek help from many on-line forums.

Finally, if you still have problems in modifying the source code to change the time zone offset and generating a new executable image, you can contact me by e-mail and I will do my best to send you a modified gps-oled-clock.hex file.

When your clock is up and running, you will be able to enjoy having a super accurate clock with a nice looking OLED display providing accurate time for years to come. We hope you enjoy building and using your GPS clock as much as we do.