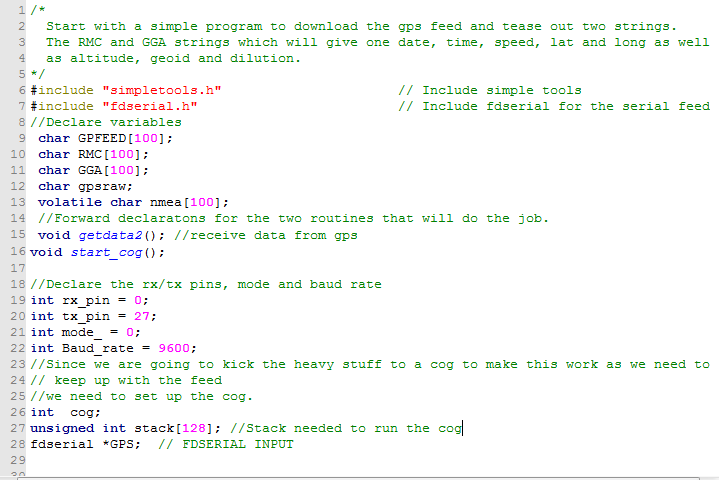
Propeller C library tutorial.

Since I had some difficulties with writing libraries and following the Learn tutorial and as well as having posted in the forums, and getting good feedback, which helped in finally creating a library that worked and re creating the library with success, I decided to re write the tutorial using an asynchronous serial GPS unit with the math to do all of the conversions.

I will do this in BABY steps! So don’t get annoyed and I will post most of the cmm files

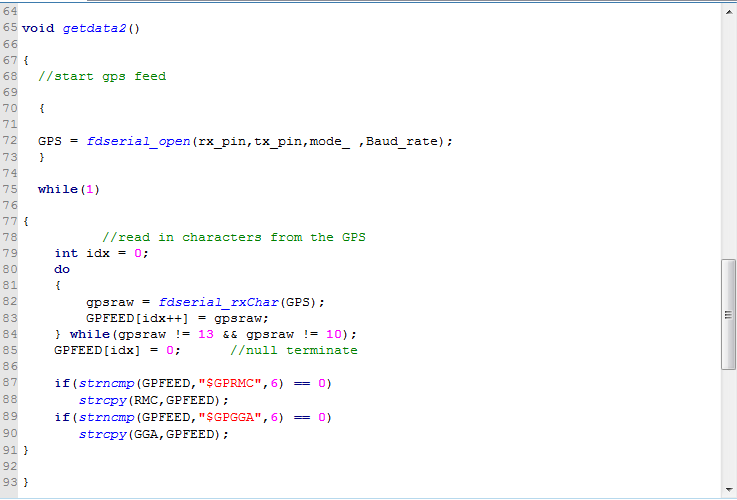
Step one let us start with a simple routine to capture the GPS data that is downloaded once a second.



Figure



Figure



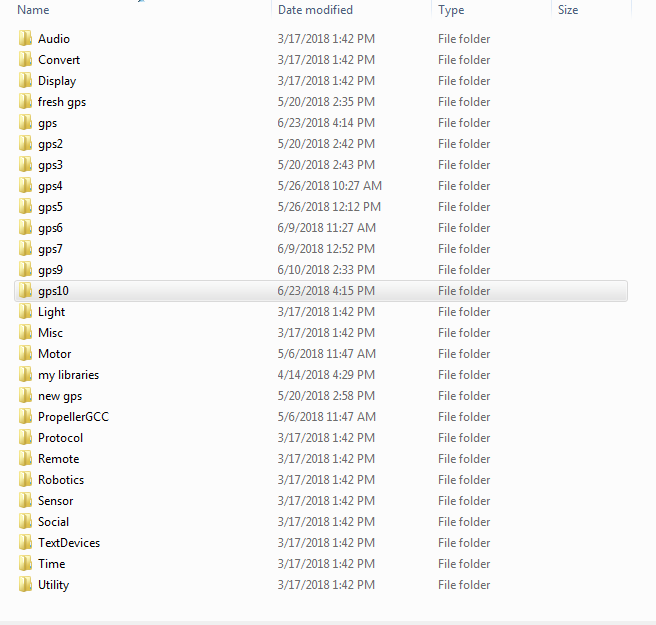
Figure

I am using the PAM Q but this will work on others: <https://www.parallax.com/product/28509>.

Verify that the program works and then we can proceed.

Now we are going to save the project in a fashion that start our quest to create a simple library.

1. Save the project via the following path which in my case looked like this creating a folder in the simple libraries folder and a subfolder then saving the program as “libgps10” in my case, you can use what you want but the saved project MUST HAVE “lib” in front of it found, via the forums. to a common mistake. Refer to figures 4, 5 and 6. \Users\mm\Documents\SimpleIDE\Learn\Simple Libraries\gps10\libgps10

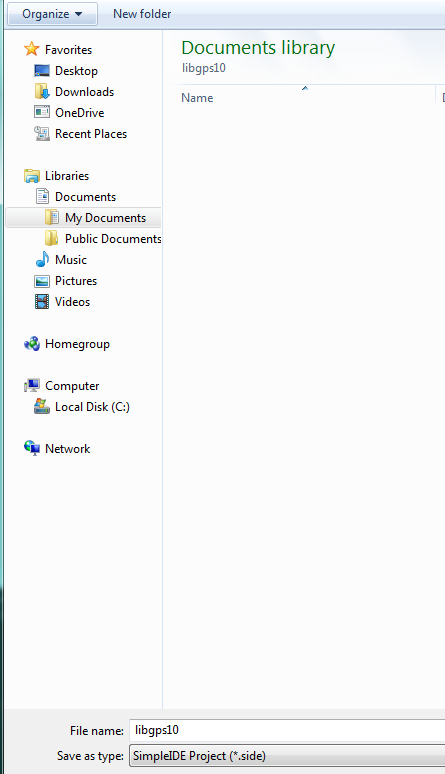


Figure

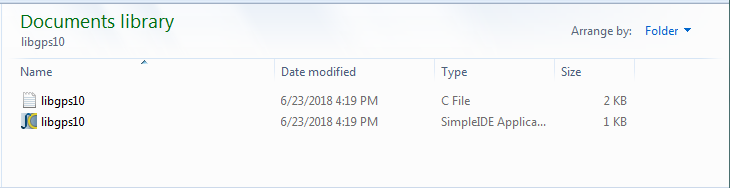


Figure

1. Open, as in my example gps10 and create a new folder, as in my example libgps10. Then save the project. As in my example it will be libgps10.
2. Verify via your file viewer that you have something that looks like figure 7.



Figure



Figure

**Verify that the project is still working prior to moving on to the next step which will be the start of creating the library.**

**REMEMBER WE WILL DO THIS**

**IN**

**BABY STEPS**

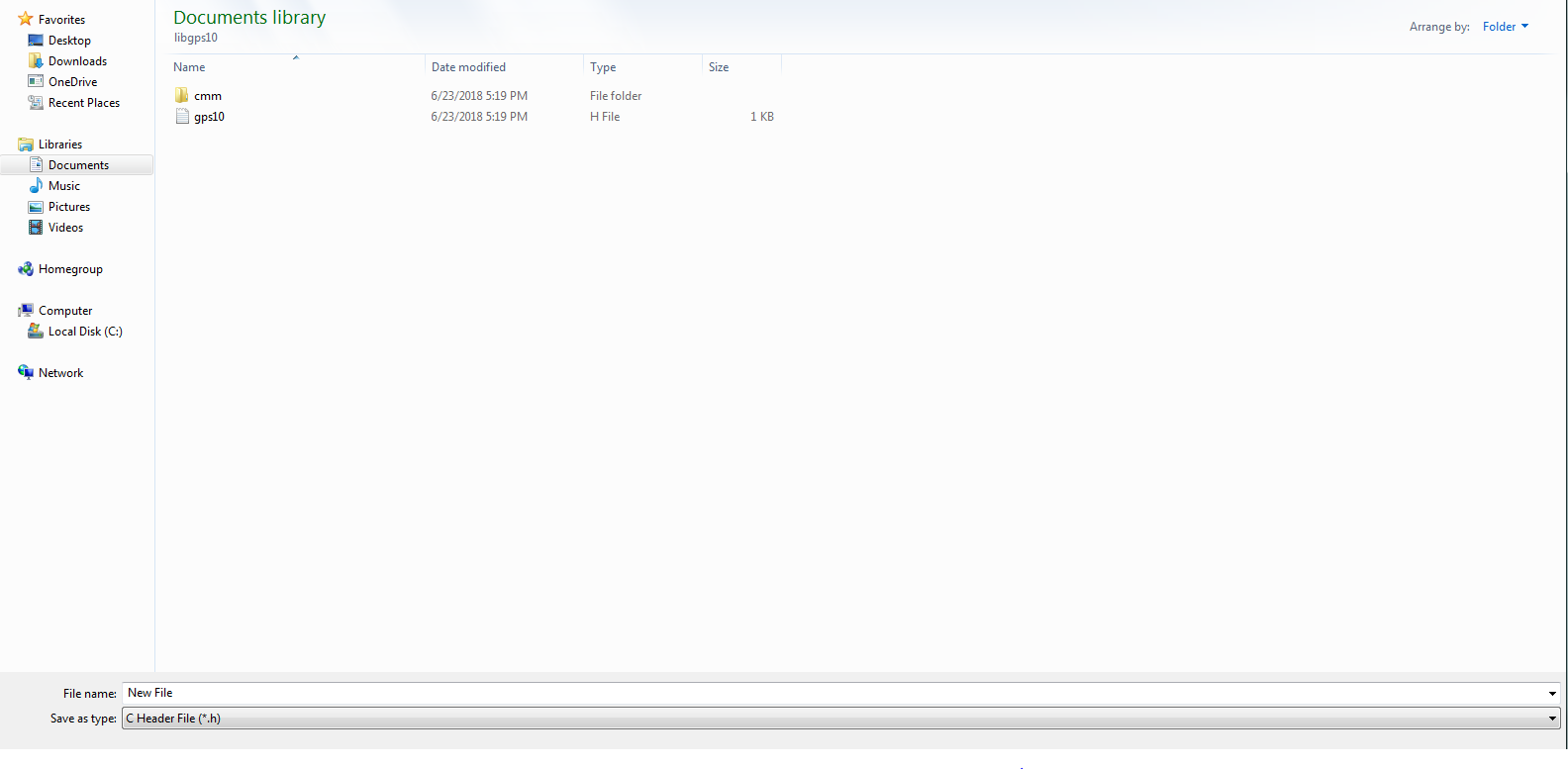
**NO OFFENCE**

Now that all is working we are going to take the first step by making simple move of a small section of the program to a header file with the extension of “.h” .

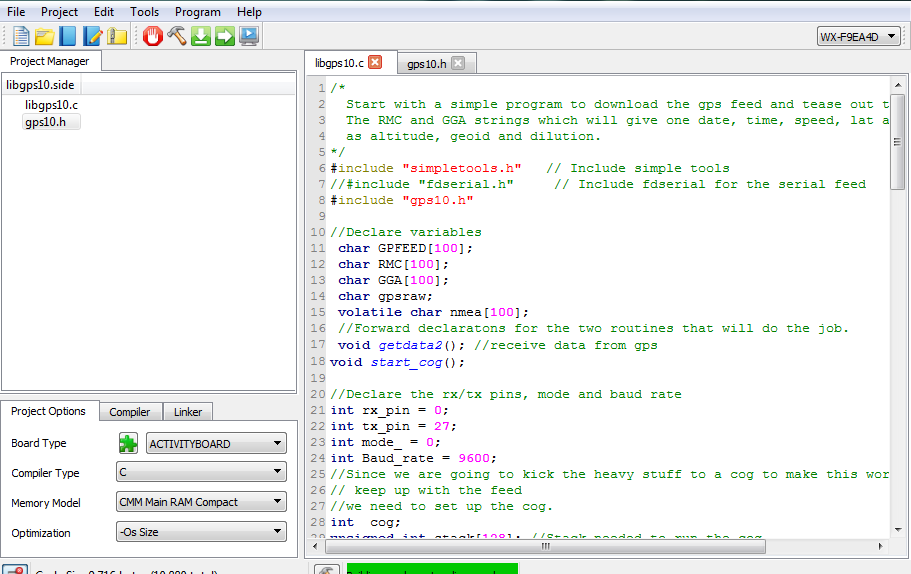
The “dot h” file is where all of the forward function declarations and variables that will be passed from the library to your working project will live.

Click on the following path:

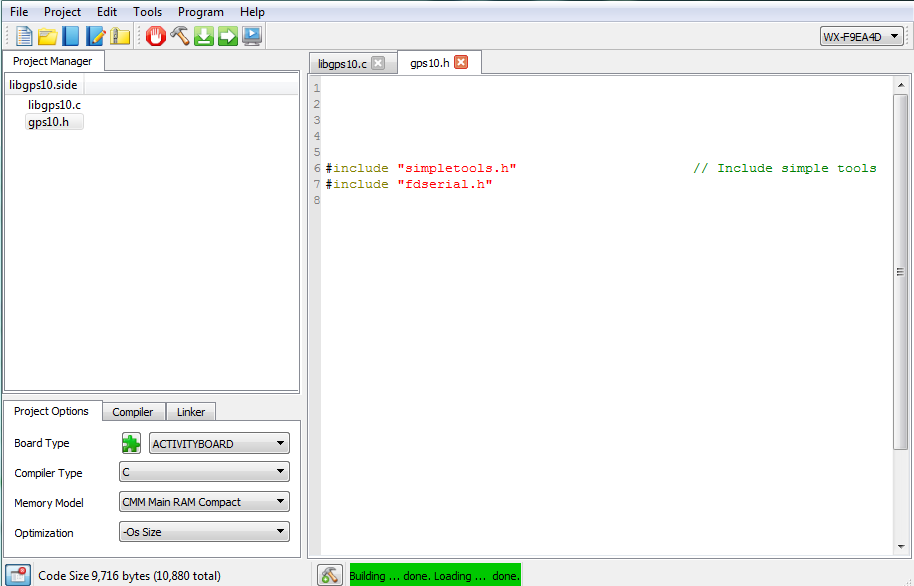
Project ->> Add tab to project and the following window should open.

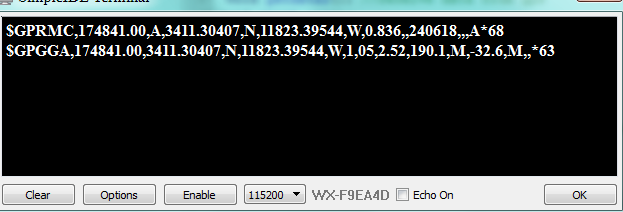


Change the save as type to “C header file (\*.h) name it and save it. In this example I used “gps10” so you should see something similar to the following:



Comment out the #include “fdserial.h” and copy it to the .h file with the #include “simpletools.h” add #include “gps10.h” to the libgps.c save and run.



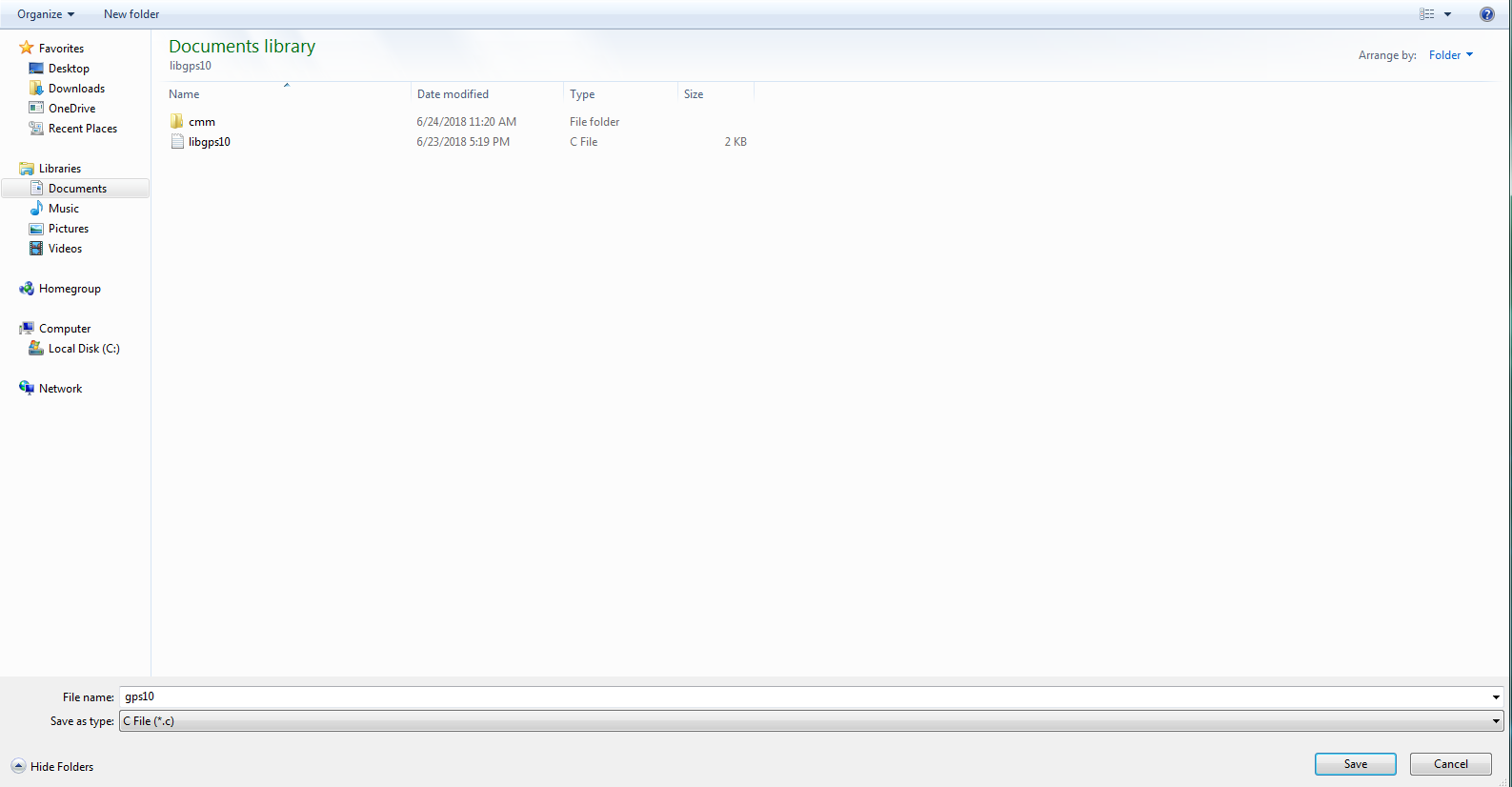


You should have the program running as before as seen in the above figure.

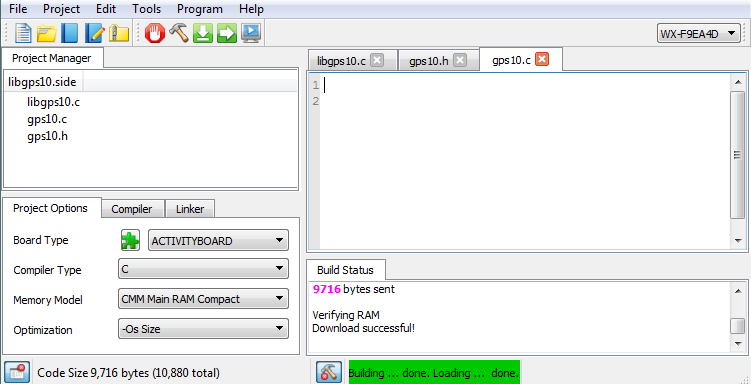
Now what just happened?

We told the main program that it can find information regarding what” #include’s” that are necessary to run the library when adding it your project.

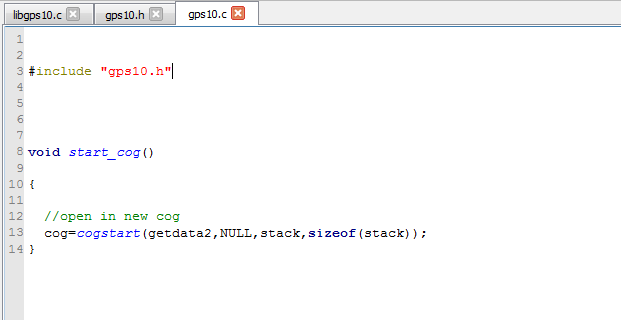
I am relatively new to C programming but come from the old days of Fortran and Basic. I cut my teeth on the IBM 1130 with rope core memory and the HP 2000C which used BASIC. There we used subroutines and would call them up using as in FORTRAN ->> “GOSUB”.

Now let’s create a .c file for our library. Same as before click on Project ->> Add tab to project and name it gps10.c.

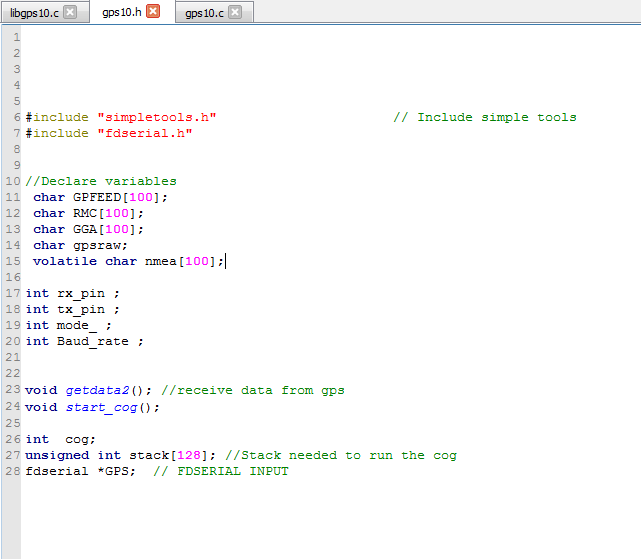
You should get something like this:

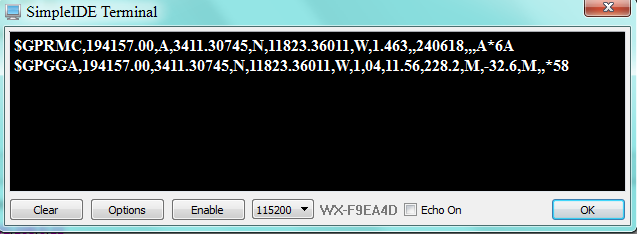


The next item is to populate the gps10.c with the working functions. Copy the Start\_cog() function to the gps10.c file.

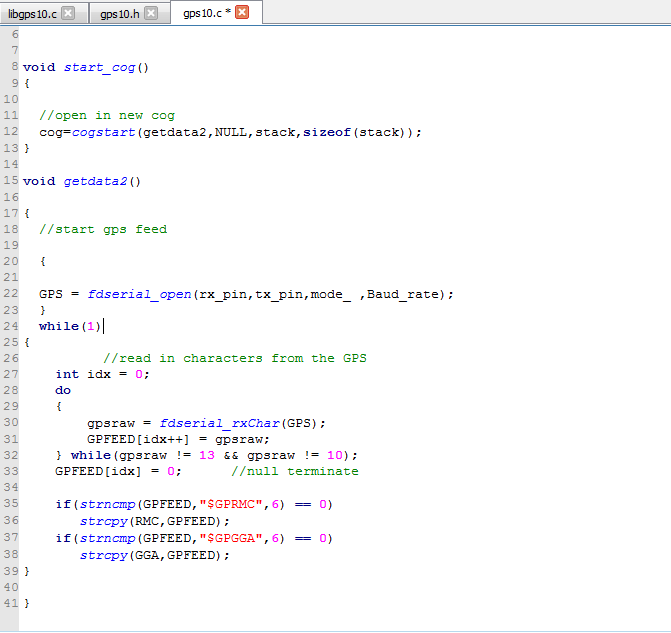


Copy the following to the gps.h file:

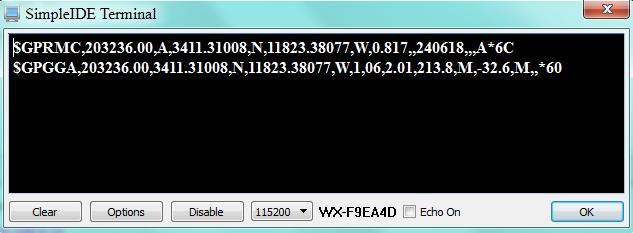
Test run and check the results.



Now copy the getdata2() function to the gps10.c as follows and test run.

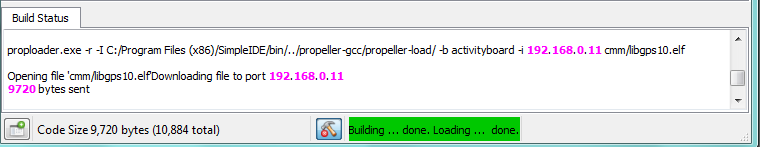


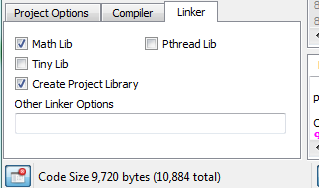
It should work:



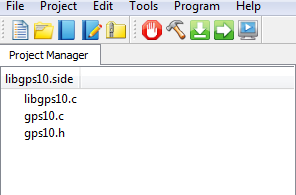
Now that it works we have to use the linker function in SimpleIDE so we can use it with other programs.

Click on the bottom left corner button to open the project options, compiler and linker dropdowns:

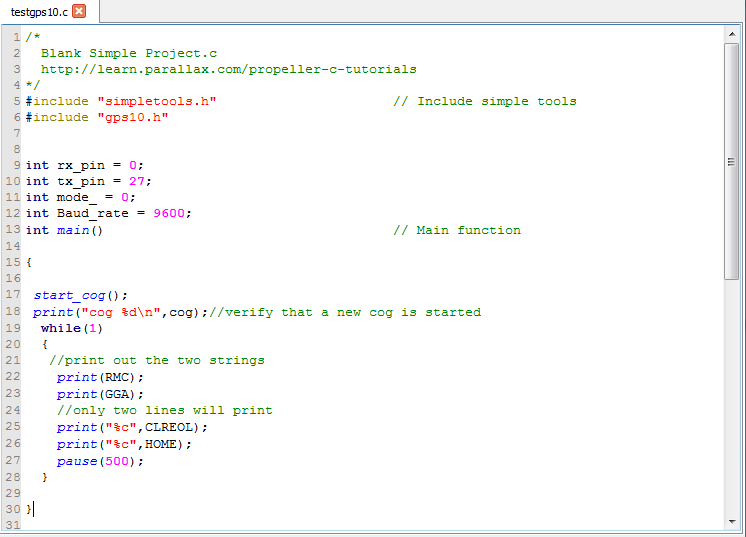


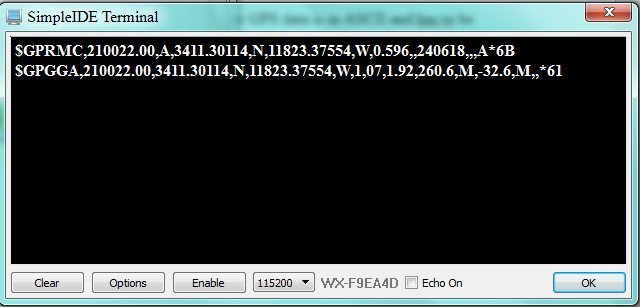


Click on linker and make sure the Create Project Library box is checked. Then click the hammer in the top of the menu buttons.



We can now test and see if our library is really going to work. Open up a new project and copy and past the following with an #include “gps10.h”:

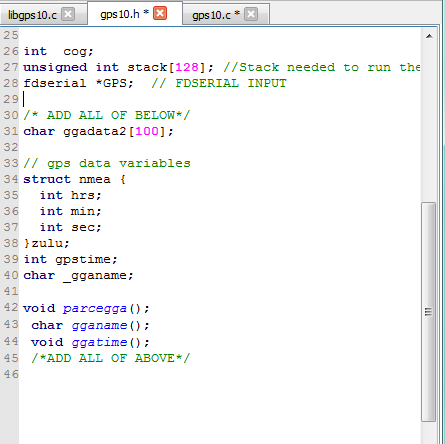




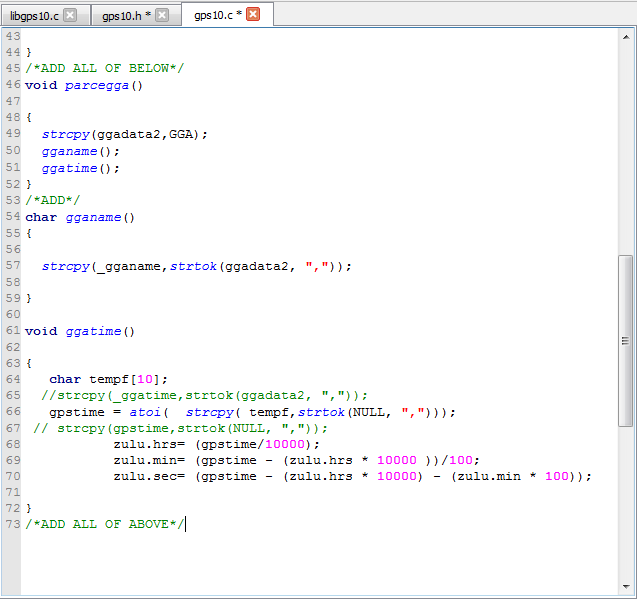
We are now going to parse out one of the strings. Raw GPS data is in ASCII and has to be converted to decimal so as to be able to reformat the data into usable forms. I am using it for astronomical calculations for a robotic German Equatorial Mount to use with my telescope.

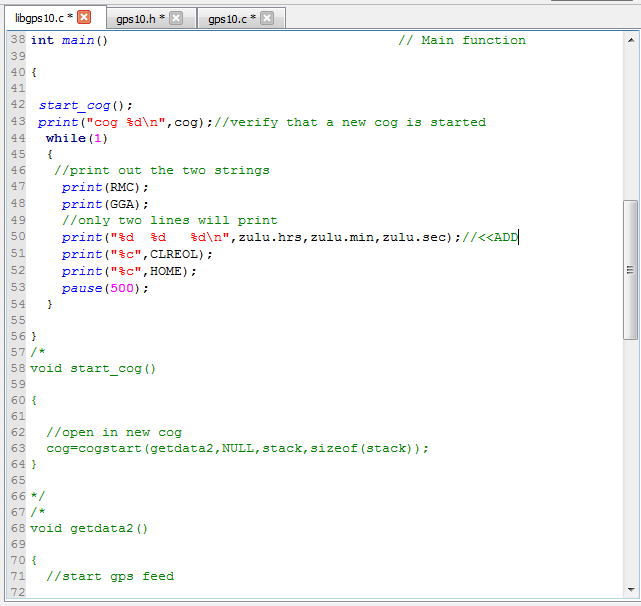
The scope mount uses stepper motors to track the celestial targets and I need accurate data in order to locate objects in the sky. So with that said we are now going to parse out the time in zulu and the present GPS location as an example of adding functions and passing data.

Add the following to the gps.h file:

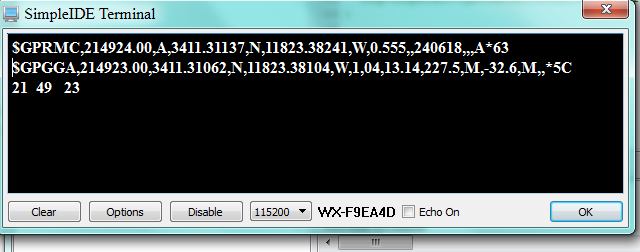


Add the following to the gps10.c , gpd10.h and libgps10.c files:

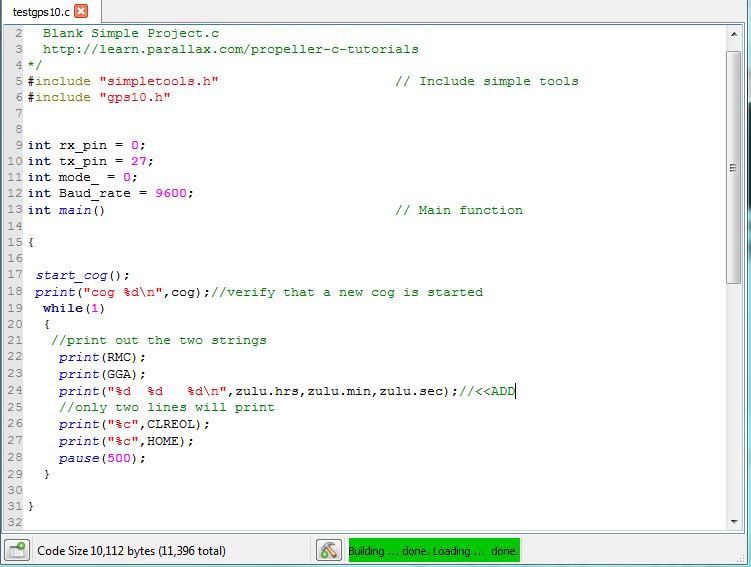


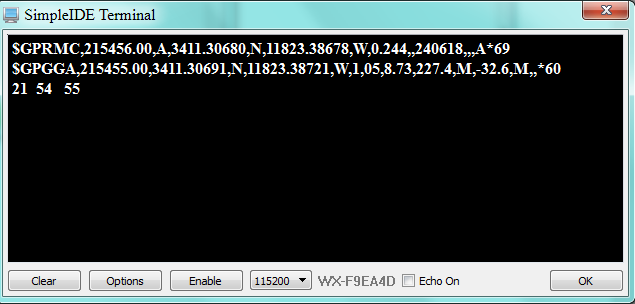


Test and you should get this:



Now go back and use the hammer to link the library. When done add the following to the test harness:





If you got this far it is a success. Repeat the process several times from scratch to be able to get the process right. It took many times for me to be able write a library.