

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.

```
1 /*
2  Start with a simple program to download the gps feed and tease out two strings.
3  The RMC and GGA strings which will give one date, time, speed, lat and long as well
4  as altitude, geoid and dilution.
5 */
6 #include "simpletools.h"           // Include simple tools
7 #include "fdserial.h"           // Include fdserial for the serial feed
8 //Declare variables
9 char GPFEED[100];
10 char RMC[100];
11 char GGA[100];
12 char gpsraw;
13 volatile char nmea[100];
14 //Forward declarations for the two routines that will do the job.
15 void getdata2(); //receive data from gps
16 void start_cog();
17
18 //Declare the rx/tx pins, mode and baud rate
19 int rx_pin = 0;
20 int tx_pin = 27;
21 int mode_ = 0;
22 int Baud_rate = 9600;
23 //Since we are going to kick the heavy stuff to a cog to make this work as we need to
24 // keep up with the feed
25 //we need to set up the cog.
26 int cog;
27 unsigned int stack[128]; //Stack needed to run the cog
28 fdserial *GPS; // FDSERIAL INPUT
29
30
```

```
35
36 int main()                                // Main function
37
38 {
39
40     start_cog();
41     print("cog %d\n", cog); //verify that a new cog is started
42     while(1)
43     {
44         //print out the two strings
45         print(RMC);
46         print(GGA);
47         //only two lines will print
48         print("%c", CLREOL);
49         print("%c", HOME);
50         pause(500);
51     }
52
53 }
54
55 void start_cog()
56
57 {
58
59     //open in new cog
60     cog=cogstart(getdata2, NULL, stack, sizeof(stack));
61 }
62
63
64
```

```

64
65 void getdata2()
66
67 {
68     //start gps feed
69
70     {
71
72     GPS = fdserial_open(rx_pin,tx_pin,mode_ ,Baud_rate);
73     }
74
75     while(1)
76
77 {
78     //read in characters from the GPS
79     int idx = 0;
80     do
81     {
82         gpsraw = fdserial_rxChar(GPS);
83         GPFEED[idx++] = gpsraw;
84     } while(gpsraw != 13 && gpsraw != 10);
85     GPFEED[idx] = 0; //null terminate
86
87     if(strncmp(GPFEED,"$GPRMC",6) == 0)
88         strcpy(RMC,GPFEED);
89     if(strncmp(GPFEED,"$GPGGA",6) == 0)
90         strcpy(GGA,GPFEED);
91 }
92
93 }

```

Figure 1

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.

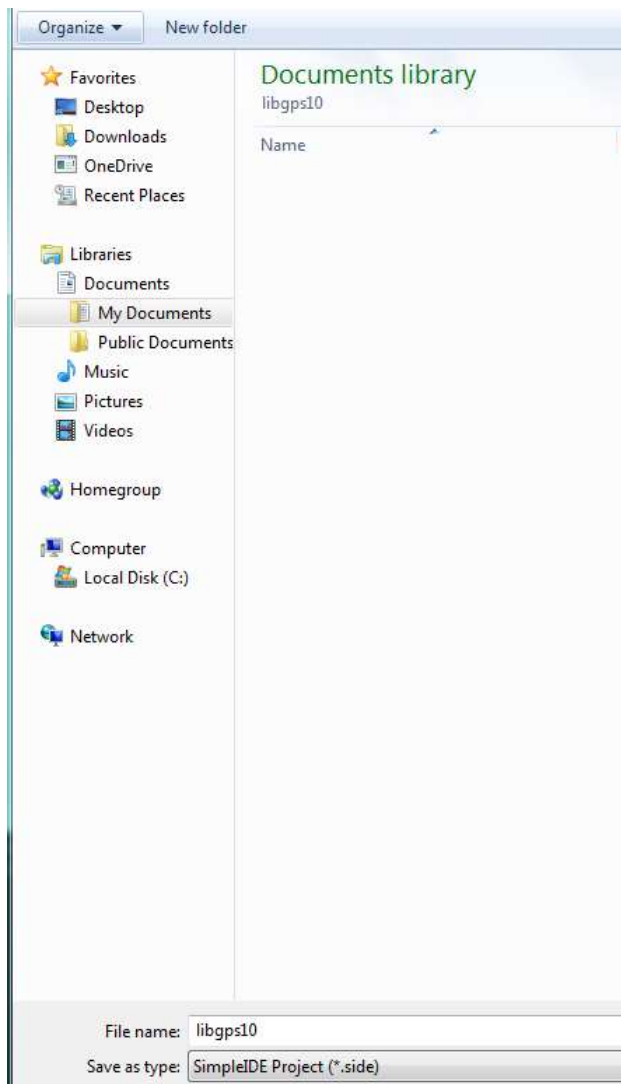
1. Save the project via the following path which in my case looked like this creating a folder in the simple libraries folder and a subfolder then saving the program as "libgps10" in my case, you can use what you want but the saved project MUST HAVE "lib" in front of it found, via the forums. to a common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



\\Users\\mm\\Documents\\SimpleIDE\\Learn\\Simple Libraries\\gps10\\libgps10

Name	Date modified	Type	Size
Audio	3/17/2018 1:42 PM	File folder	
Convert	3/17/2018 1:42 PM	File folder	
Display	3/17/2018 1:42 PM	File folder	
fresh gps	5/20/2018 2:35 PM	File folder	
gps	6/23/2018 4:14 PM	File folder	
gps2	5/20/2018 2:42 PM	File folder	
gps3	5/20/2018 2:43 PM	File folder	
gps4	5/26/2018 10:27 AM	File folder	
gps5	5/26/2018 12:12 PM	File folder	
gps6	6/9/2018 11:27 AM	File folder	
gps7	6/9/2018 12:52 PM	File folder	
gps9	6/10/2018 2:33 PM	File folder	
gps10	6/23/2018 4:15 PM	File folder	
Light	3/17/2018 1:42 PM	File folder	
Misc	3/17/2018 1:42 PM	File folder	
Motor	5/6/2018 11:47 AM	File folder	
my libraries	4/14/2018 4:29 PM	File folder	
new gps	5/20/2018 2:58 PM	File folder	
PropellerGCC	5/6/2018 11:47 AM	File folder	
Protocol	3/17/2018 1:42 PM	File folder	
Remote	3/17/2018 1:42 PM	File folder	
Robotics	3/17/2018 1:42 PM	File folder	
Sensor	3/17/2018 1:42 PM	File folder	
Social	3/17/2018 1:42 PM	File folder	
TextDevices	3/17/2018 1:42 PM	File folder	
Time	3/17/2018 1:42 PM	File folder	
Utility	3/17/2018 1:42 PM	File folder	

- 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.
- Verify via your file viewer that you have something that looks like figure 7.

Documents library			
gps10			
Name	Date modified	Type	Size
libgps10	6/23/2018 4:17 PM	File folder	



Documents library					Arrange by: Folder ▾	
libgps10						
Name	Date modified	Type	Size			
 libgps10	6/23/2018 4:19 PM	C File	2 KB			
 libgps10	6/23/2018 4:19 PM	SimpleIDE Applica...	1 KB			

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

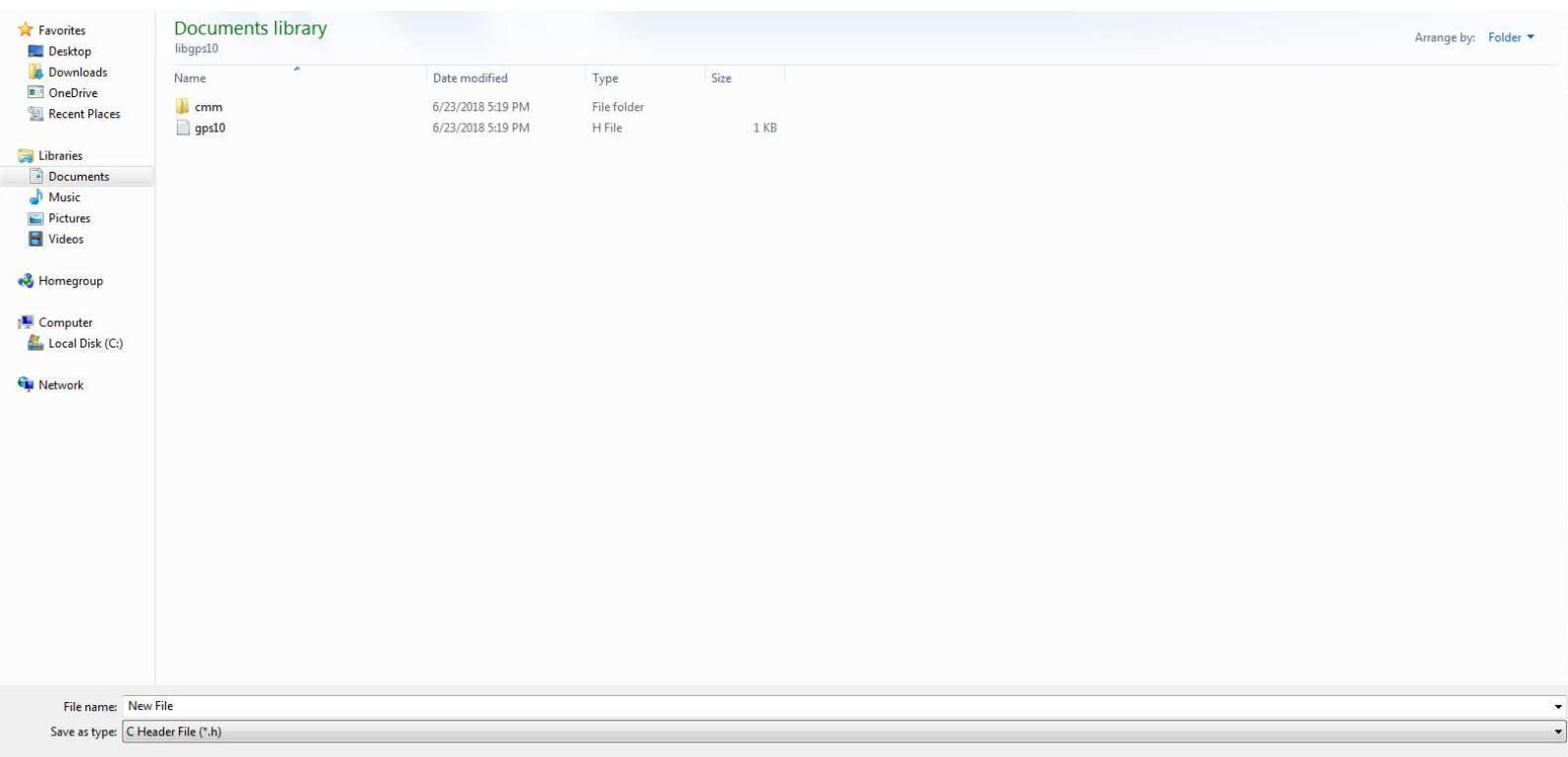
NO OFFENCE

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” .

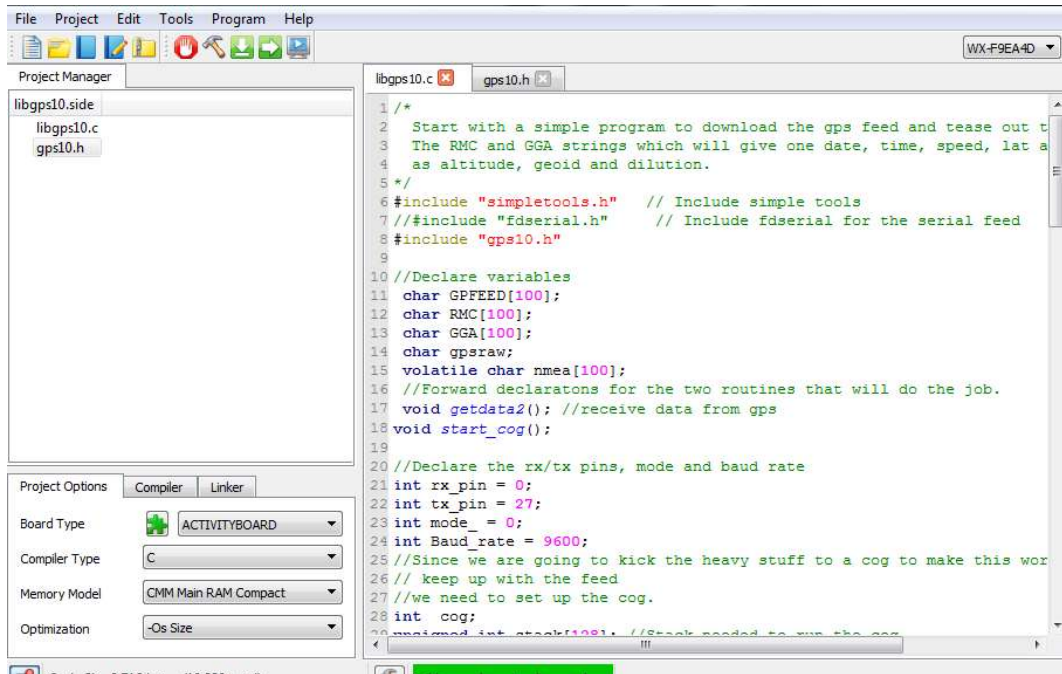
The “dot h” file is where all of the forward function declarations and variables that will be passed from the library to your working project will live.

Click on the following path:

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

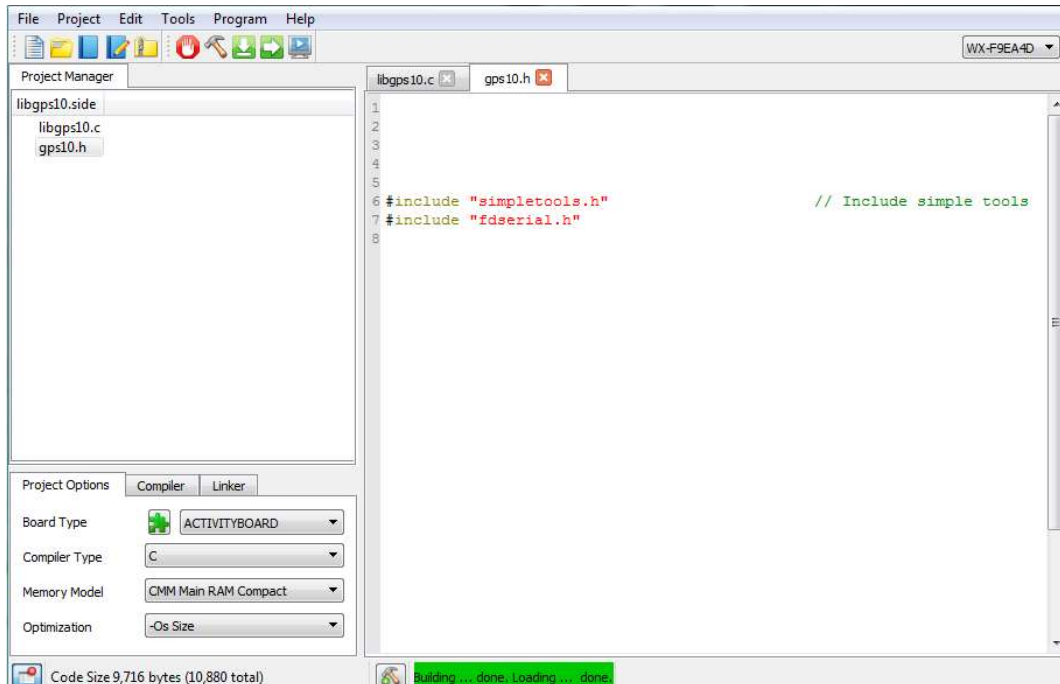


Change the save as type to “C header file (*.h) name it and save it. In this example I used “gps10” so you should see something similar to the following:



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.

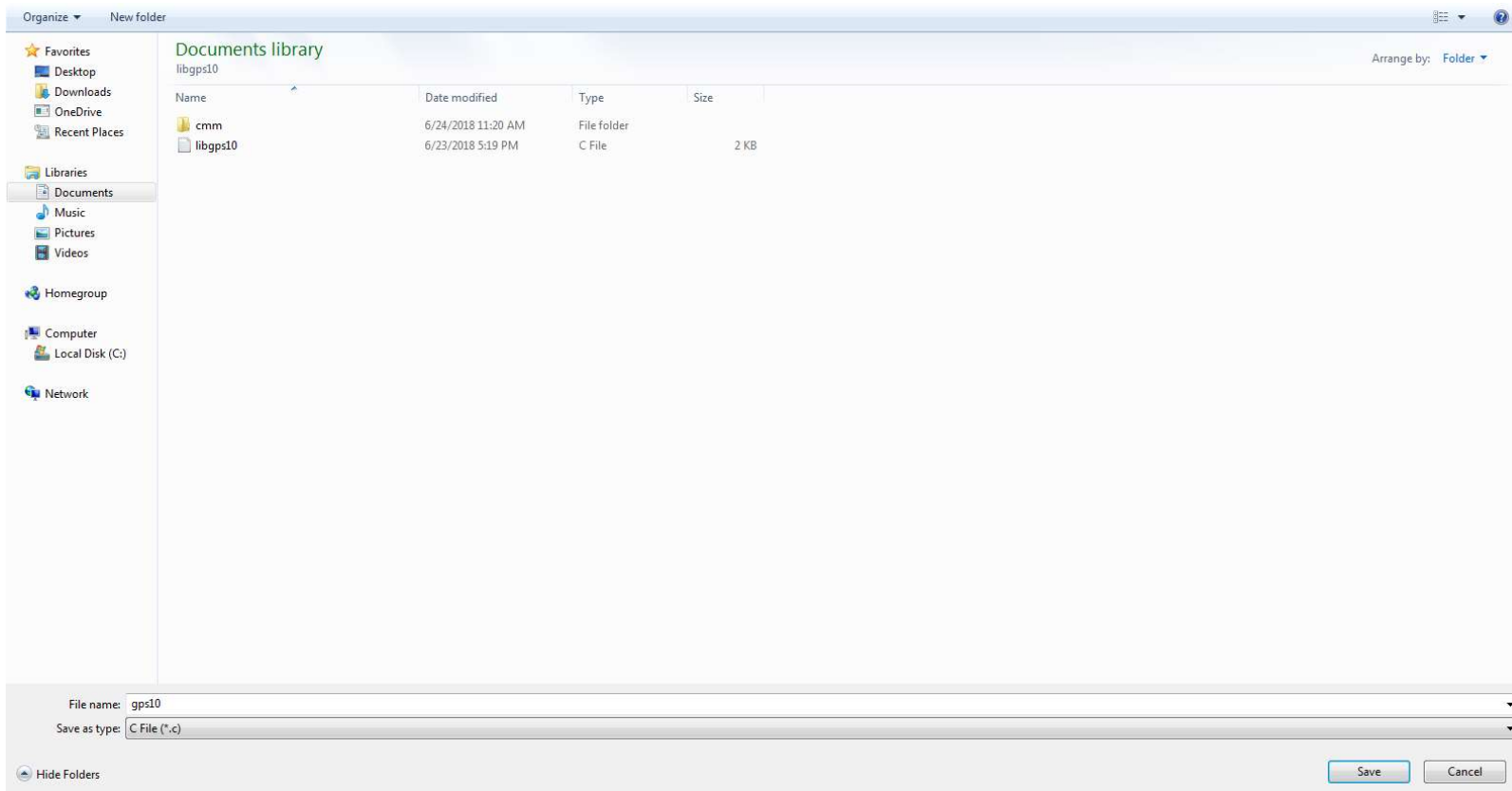
Now what just

happened?

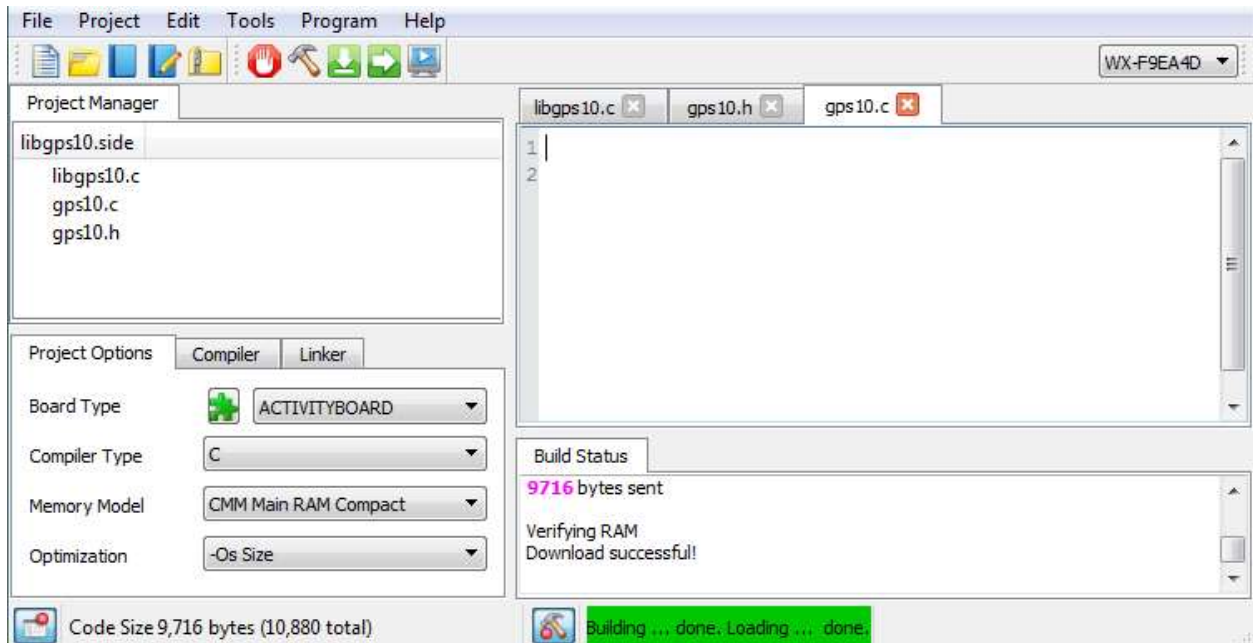
We told the main program that it can find information regarding what" #include's" that are necessary to run the library when adding it your project.

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".

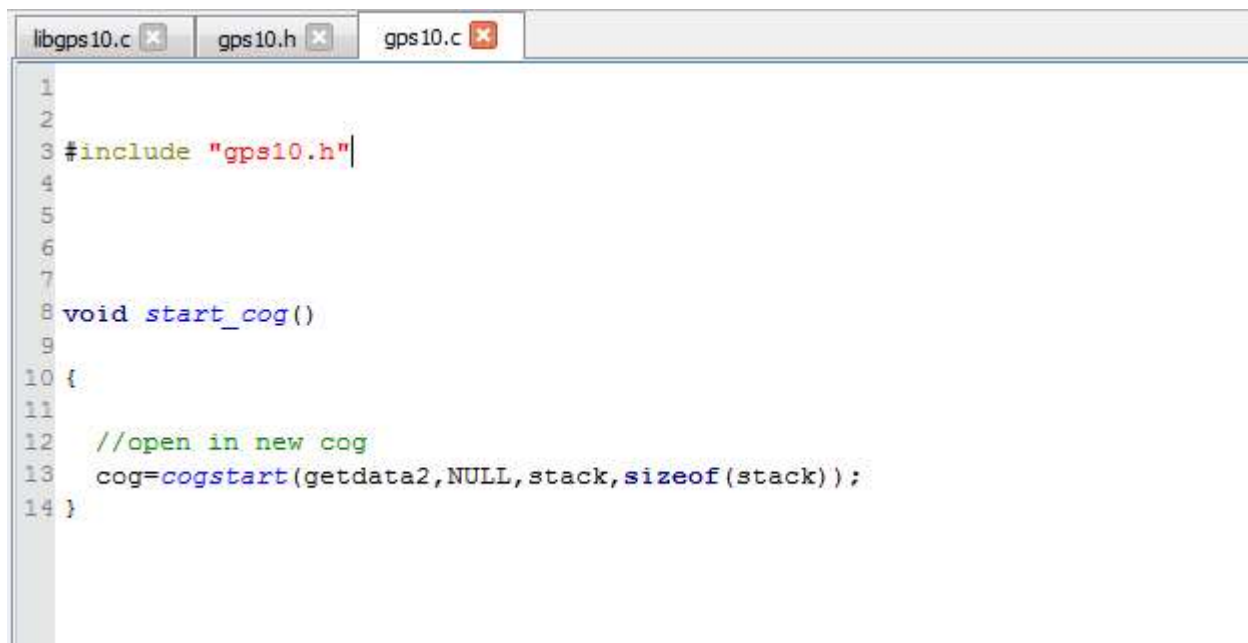
Now let's create a .c file for our library. Same as before click on Project ->> Add tab to project and name it gps10.c.



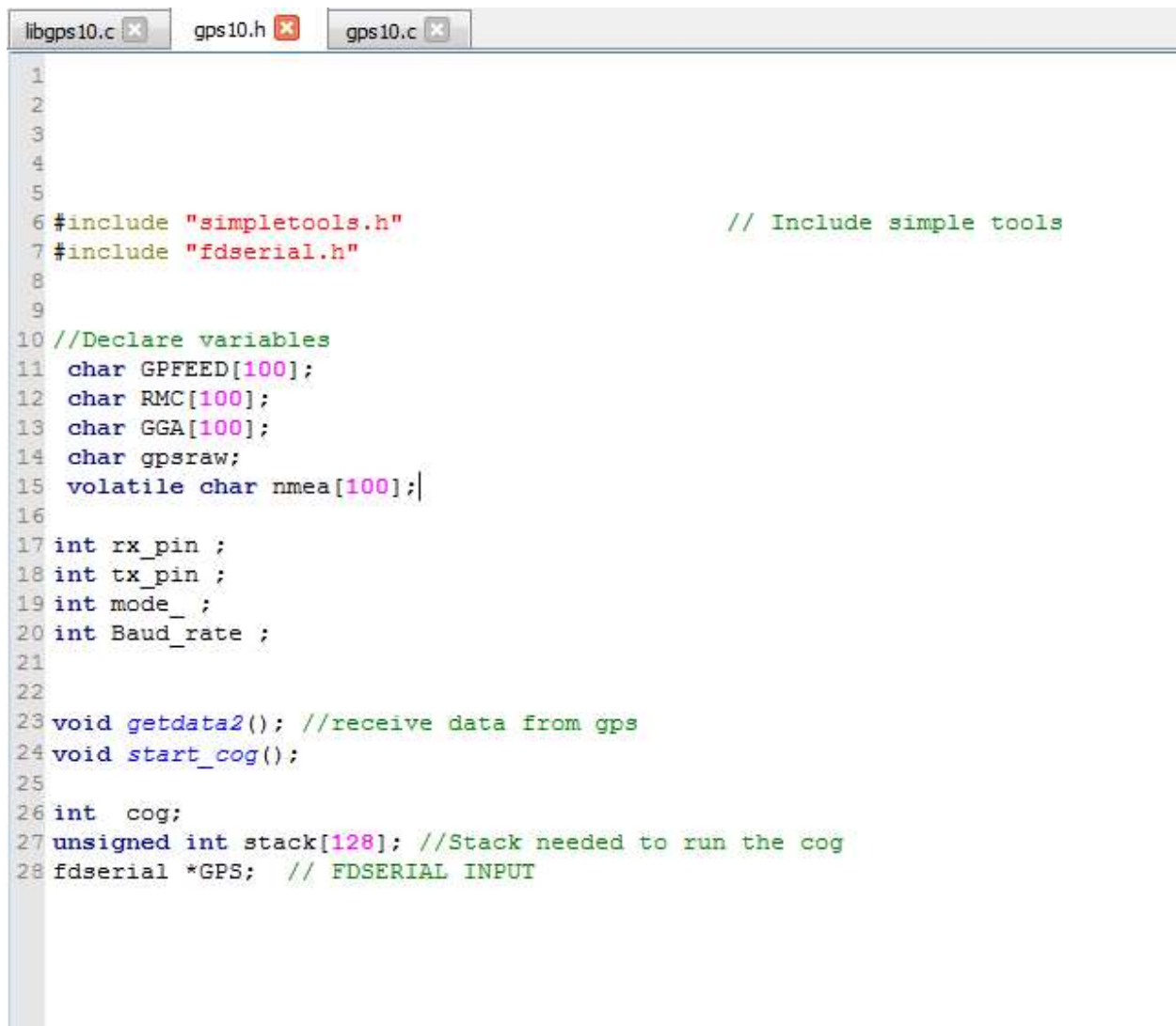
You should get something like this:



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

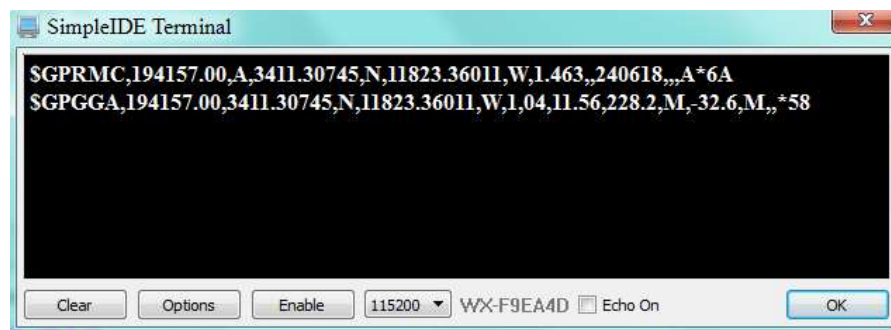


Copy the following to the gps.h file:



```
1
2
3
4
5
6 #include "simpletools.h"           // Include simple tools
7 #include "fdserial.h"
8
9
10 //Declare variables
11 char GPFEED[100];
12 char RMC[100];
13 char GGA[100];
14 char gpsraw;
15 volatile char nmea[100];|
16
17 int rx_pin ;
18 int tx_pin ;
19 int mode_ ;
20 int Baud_rate ;
21
22
23 void getdata2(); //receive data from gps
24 void start_cog();
25
26 int cog;
27 unsigned int stack[128]; //Stack needed to run the cog
28 fdserial *GPS; // FDSERIAL INPUT
```

Test run and check the results.

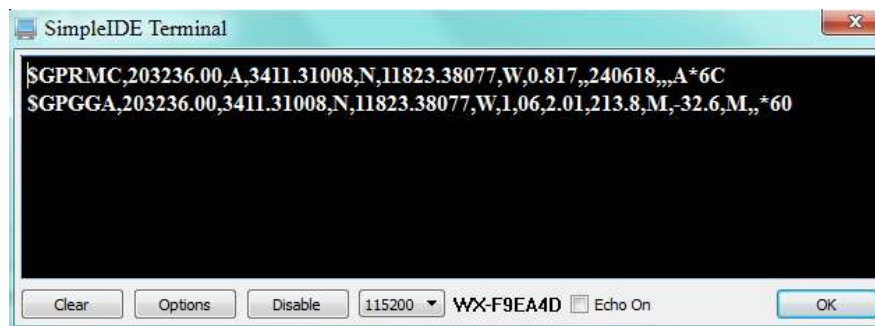


Now
getdata2() function to the gps10.c as follows and test run.

copy the

```
libgps10.c x  gps10.h x  gps10.c * x
6
7
8 void start_cog()
9 {
10
11 //open in new cog
12 cog=cogstart(getdata2,NULL,stack,sizeof(stack));
13 }
14
15 void getdata2()
16 {
17 {
18 //start gps feed
19
20 {
21
22 GPS = fdserial_open(rx_pin,tx_pin,mode_ ,Baud_rate);
23 }
24 while(1)|
25 {
26 //read in characters from the GPS
27 int idx = 0;
28 do
29 {
30 gpsraw = fdserial_rxChar(GPS);
31 GPFEED[idx++] = gpsraw;
32 } while(gpsraw != 13 && gpsraw != 10);
33 GPFEED[idx] = 0; //null terminate
34
35 if(strncmp(GPFEED,"$GPRMC",6) == 0)
36 strcpy(RMC,GPFEED);
37 if(strncmp(GPFEED,"$GPGGA",6) == 0)
38 strcpy(GGA,GPFEED);
39 }
40
41 }
```

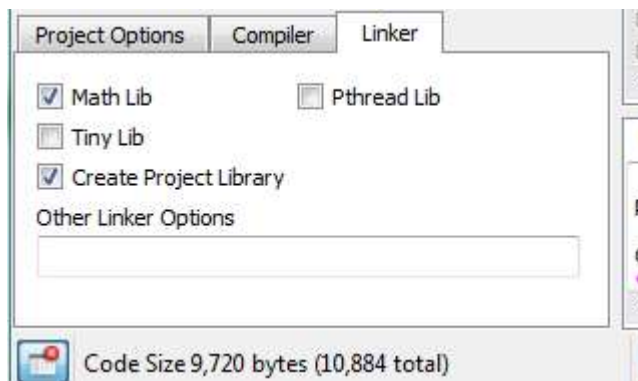
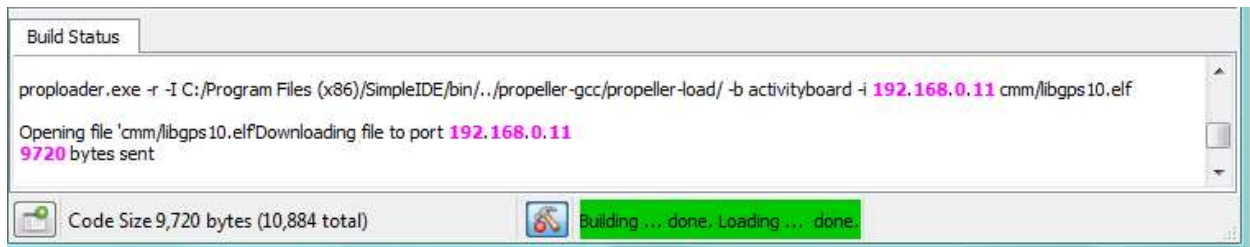
It should work:



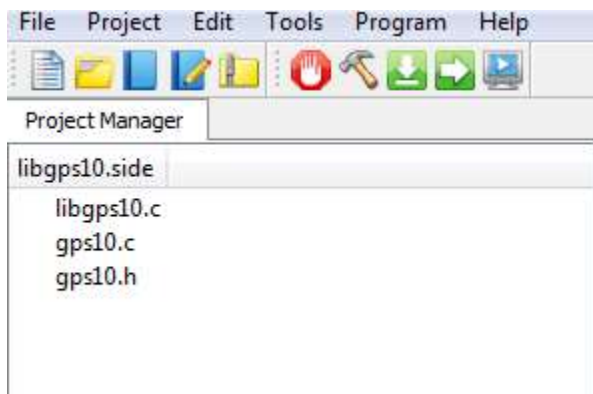
A screenshot of the SimpleIDE Terminal window. The window has a title bar that says "SimpleIDE Terminal" and a close button. The terminal area is black with white text showing two lines of GPS data: "\$GPRMC,203236.00,A,3411.31008,N,11823.38077,W,0.817,,240618,,,A*6C" and "\$GPGGA,203236.00,3411.31008,N,11823.38077,W,1,06,2.01,213.8,M,-32.6,M,,*60". At the bottom of the window, there is a status bar with buttons for "Clear", "Options", "Disable", a baud rate dropdown set to "115200", a device ID "WX-F9EA4D", an "Echo On" checkbox, and an "OK" button.

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:

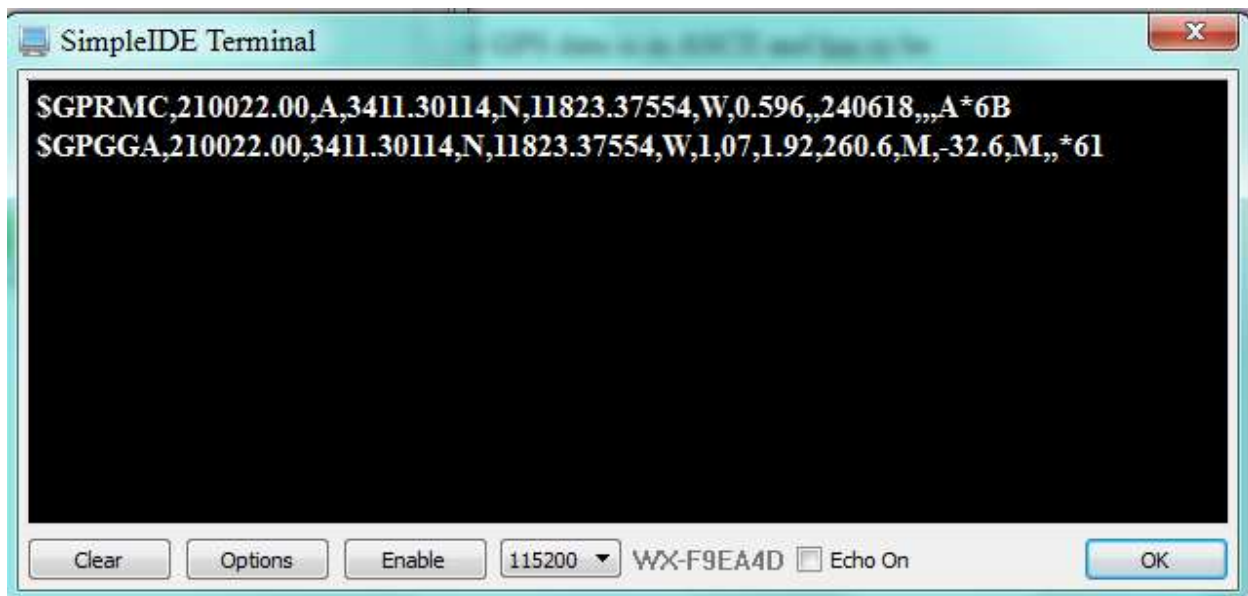


Click on linker and make sure the Create Project Library box is checked. Then click the hammer in the top of the menu buttons.



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"`:

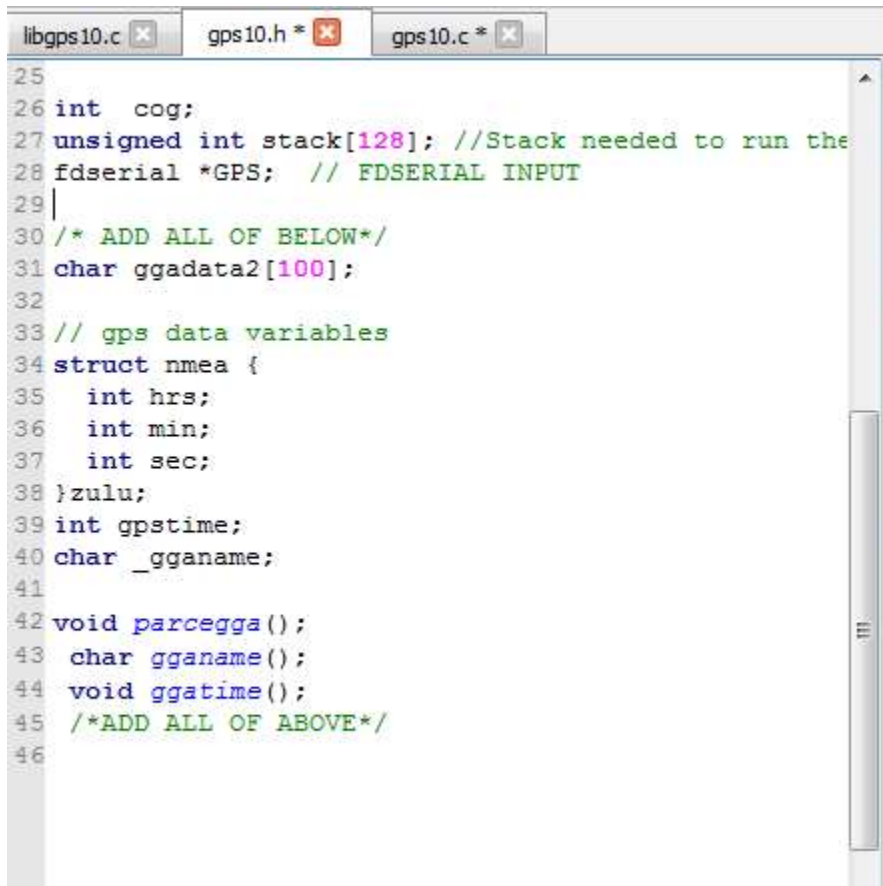
```
testgps10.c
1 /*
2  Blank Simple Project.c
3  http://learn.parallax.com/propeller-c-tutorials
4 */
5 #include "simpletools.h"           // Include simple tools
6 #include "gps10.h"
7
8
9 int rx_pin = 0;
10 int tx_pin = 27;
11 int mode_ = 0;
12 int Baud_rate = 9600;
13 int main()                        // Main function
14
15 {
16
17  start_cog();
18  print("cog %d\n", cog); //verify that a new cog is started
19  while(1)
20  {
21    //print out the two strings
22    print(RMC);
23    print(GGA);
24    //only two lines will print
25    print("%c", CLREOL);
26    print("%c", HOME);
27    pause(500);
28  }
29
30 }
31
```



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:

A screenshot of a code editor window with three tabs: 'libgps10.c', 'gps10.h *', and 'gps10.c *'. The 'gps10.h *' tab is active, displaying the following C code:

```
25
26 int cog;
27 unsigned int stack[128]; //Stack needed to run the
28 fdserial *GPS; // FDSERIAL INPUT
29 |
30 /* ADD ALL OF BELOW*/
31 char gadata2[100];
32
33 // gps data variables
34 struct nmea {
35     int hrs;
36     int min;
37     int sec;
38 }zulu;
39 int gpstime;
40 char _ganame;
41
42 void parcegga();
43 char gganame();
44 void ggatime();
45 /*ADD ALL OF ABOVE*/
46
```

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

```

libgps10.c *  gps10.h *  gps10.c *
43
44 }
45 /*ADD ALL OF BELOW*/
46 void parcegga()
47 {
48 {
49     strcpy(ggadata2, GGA);
50     gganame();
51     ggatime();
52 }
53 /*ADD*/
54 char gganame()
55 {
56 {
57     strcpy(_gganame, strtok(ggadata2, ","));
58 }
59 }
60
61 void ggatime()
62 {
63 {
64     char tempf[10];
65     //strcpy(_ggatime, strtok(ggadata2, ","));
66     gpstime = atoi( strcpy( tempf, strtok(NULL, ",")));
67     // strcpy(gpstime, strtok(NULL, ","));
68     zulu.hrs= (gpstime/10000);
69     zulu.min= (gpstime - (zulu.hrs * 10000 ))/100;
70     zulu.sec= (gpstime - (zulu.hrs * 10000) - (zulu.min * 100));
71 }
72 }
73 /*ADD ALL OF ABOVE*/

```

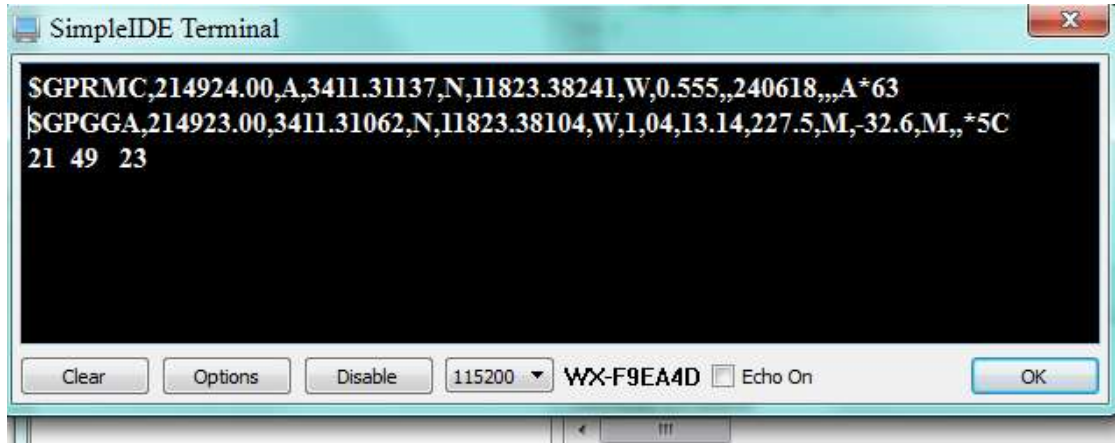
```

libgps10.c *  gps10.h *  gps10.c *
38 int main() // Main function
39 {
40 {
41 {
42     start_cog();
43     print("cog %d\n", cog); //verify that a new cog is started
44     while(1)
45     {
46         //print out the two strings
47         print(RMC);
48         print(GGA);
49         //only two lines will print
50         print("%d %d %d\n", zulu.hrs, zulu.min, zulu.sec); //<<ADD
51         print("%c", CLREOL);
52         print("%c", HOME);
53         pause(500);
54     }
55 }
56 }
57 /*
58 void start_cog()
59 {
60 {
61     //open in new cog
62     cog=cogstart(getdata2, NULL, stack, sizeof(stack));
63 }
64 }
65 */
66 /*
67 */
68 void getdata2()
69 {
70 {
71     //start gps feed
72 }

```

Test
get this:

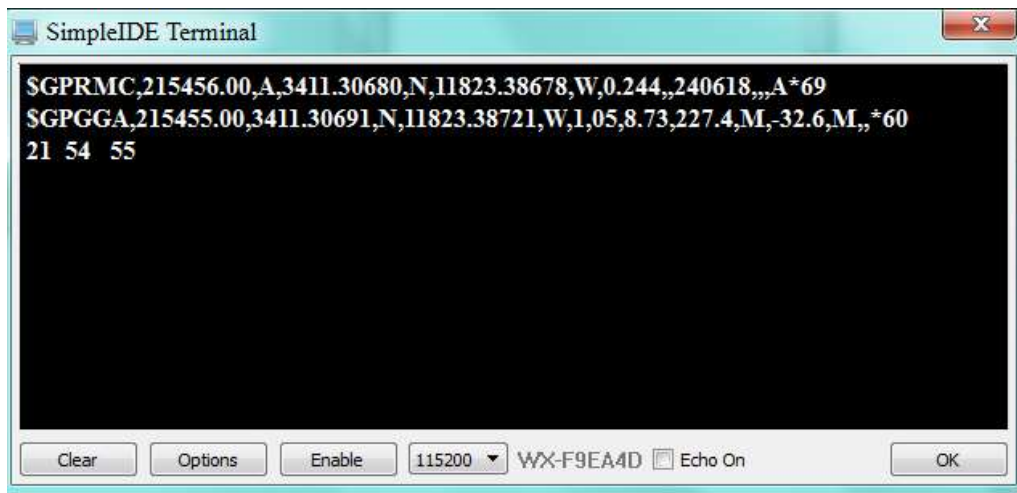
and you should



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

```
testgps10.c
2 Blank Simple Project.c
3 http://learn.parallax.com/propeller-c-tutorials
4 */
5 #include "simpletools.h" // Include simple tools
6 #include "gps10.h"
7
8
9 int rx_pin = 0;
10 int tx_pin = 27;
11 int mode_ = 0;
12 int Baud_rate = 9600;
13 int main() // Main function
14 {
15
16
17 start_cog();
18 print("cog %d\n", cog); //verify that a new cog is started
19 while(1)
20 {
21 //print out the two strings
22 print(RMC);
23 print(GGA);
24 print("%d %d %d\n", zulu.hrs, zulu.min, zulu.sec); //<<ADD
25 //only two lines will print
26 print("%c", CLREOL);
27 print("%c", HOME);
28 pause(500);
29 }
30
31 }
32
```

Code Size 10,112 bytes (11,396 total) Building ... done. Loading ... done.



A screenshot of a 'SimpleIDE Terminal' window. The window has a title bar with a close button. The main area is black with white text showing two NMEA sentences: 'SGPRMC,215456.00,A,3411.30680,N,11823.38678,W,0.244,,240618,,,A*69' and 'SGPGGA,215455.00,3411.30691,N,11823.38721,W,1,05,8.73,227.4,M,-32.6,M,,*60'. Below the sentences, the numbers '21 54 55' are displayed. At the bottom, there is a status bar with buttons for 'Clear', 'Options', 'Enable', a dropdown menu showing '115200', a text field with 'WX-F9EA4D', a checkbox for 'Echo On', and an 'OK' button.

```
SGPRMC,215456.00,A,3411.30680,N,11823.38678,W,0.244,,240618,,,A*69
SGPGGA,215455.00,3411.30691,N,11823.38721,W,1,05,8.73,227.4,M,-32.6,M,,*60
21 54 55
```

Clear Options Enable 115200 WX-F9EA4D ☐ Echo On OK

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.