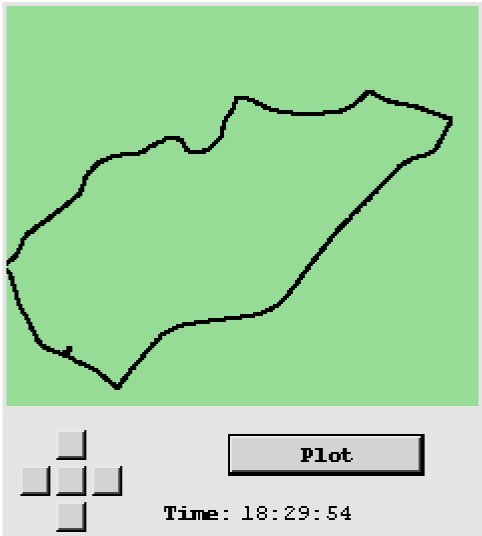
GPS Series - Part 3

By Michael Simpson



Interface to a GPS Module or Receiver

As seen in December 2007 of Servo Magazine
Pick up an issue at
www.servomagazine.com

At this point you should understand how to connect each of the modules to your PC. You can also use the various applications I have provided to check the connection status. This month I'm going to show you how to parse the positional data from the NMEA protocol. I'm also going to present you with a data logger program that will allow you to capture and store NMEA data on your PC.

More on the NMEA 0183 Protocol

Back in Part 1, we looked at the GSV and GSA NMEA commands. While those commands are invaluable for determining your GPS lock status, they won't yield any positional data. Let's take a look at two additional commands:

- GGA: Time, Position, Fix Type
- RMC: Time, Date, Position, Course, Speed

Remember you can download a complete NMEA 0183 reference manual here:
http://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf

Just To Recap:
A NMEA 0183 message begins with a $GP and ends with a carriage return. It looks something like this:

$GPGSV,3,1,12,20,00,000,,10,00,000,,25,00,000,,27,00,000,*79

The message name, which is also referred to as the option, are the characters just following the $GP. Each data element is separated by a comma. The data elements are terminated by the * character, followed by the checksum. There is an 8-bit XOR of each character between the $ and * to form the checksum. The last two characters in the message are a hex representation of the calculated checksum.

GGA: Global Positioning System Fixed Data

Field 1, UTC Time in the format of hhmmss.sss
Field 2, Latitude in the format of ddmm.mmmm
Fields 3, N/S Indicator (N=North, S=South)
Field 4, Longitude in the format of dddmm.mmmm
Field 5, E/W Indicator (E=East, W=West)
Field 6, Position Fix Indicator (0=No Fix, 1=SPS Fix, 2=DGPS Fix)
Field 7, Satellites Used (0-12)
Field 8, Horizontal Dilution of Precision
Field 9, MSL Altitude
Field 10, MSL Units (M=Meters)
Field 11, Geoid Separation
Field 12, Geoid Units (M=Meters)
Field 13, Age of Diff Correction in seconds
Field 14, Diff Reference

RMC: Recommended Minimum Specific GNSS Data

Field 1, UTC Time in the format of hhmmss.sss
Field 2, Status (A=Valid Data, B=Invalid Data)
Field 3, Latitude in the format of ddmm.mmmm
Fields 4, N/S Indicator (N=North, S=South)
Field 5, Longitude in the format of dddmm.mmmm
Field 6, E/W Indicator (E=East, W=West)
Field 7, Speed over ground in knots
Field 8, Course over ground in degrees
Field 9, Date in the format of ddmmyy
Field 10, Magnetic Variation in degrees
Field 11, Mode (A=Autonomous, D=DGPS, E=DR)

Both the GGA and RMC fields will give you the Longitude and Latitude, but only the GGA will report the Altitude and Fix Type. The RMC command will report your course and speed. So it's clear that we need to parse both of these commands to gain all the information.

Data Logger

To help you understand the GGA and RMC commands a little better, let's start out by building a data logger. Data loggers are invaluable because they let you collect test data that you can later use to help you test and refine your projects without having to resort to field tests.

As shown in Figure 2 the data logger is straight forward. I have included both PC and Pocket PC versions that will handle all the modules and receivers discussed in this series. You select the device using the Device menu shown in Figure 3. This will set the correct baud rate and enable special setup commands needed for the Etek and Copernicus modules.

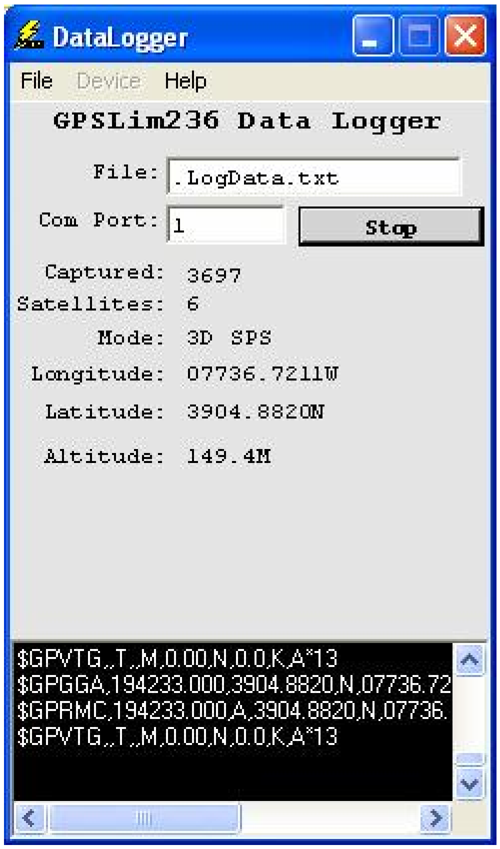


Figure 2

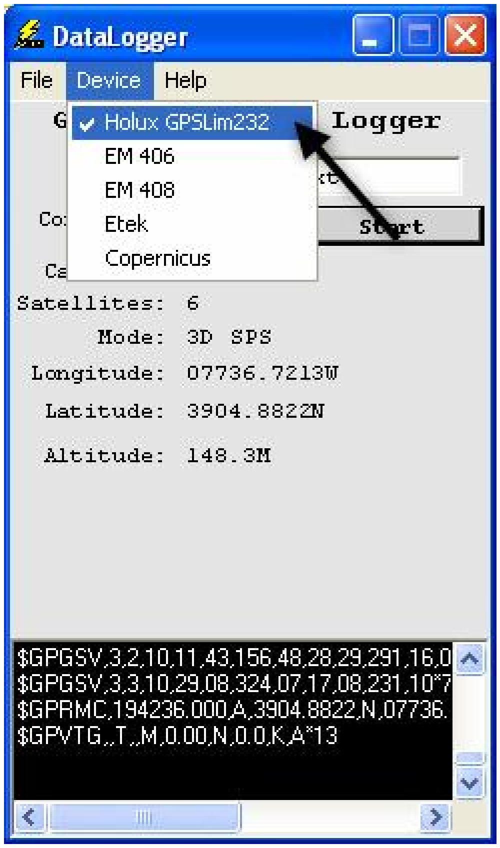


Figure 3

You start the data collection by hitting the start button shown in Figure 4. The program will then open the com port indicated and initialize the GPS module if needed. Collected data will be saved to the file indicated. If you want to save the file into the same directory as the GPSDataLogger program, precede the filename with a decimal point as shown in Figure 4.

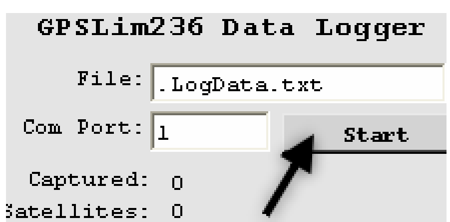


Figure 4

As data is collected and saved, it is also parsed. The NMEA commands GGA, GSV, GSA and RMC are all parsed. The pertinent information is displayed on the form as shown in Figure 5. The actual number of bytes captured and saved will also be displayed. If you see the captured number go up but none of the data fields are updated, you have selected the wrong device.

Figure 5

Data Plotter

You will want to view the data you collected. I have created two programs to allow you to do just that. The GPSLogDisplay program shown in Figure 6 will display all the pertinent information. You select the log file captured with the GPSDataLogger program by selecting the File Menu as shown in Figure 7.

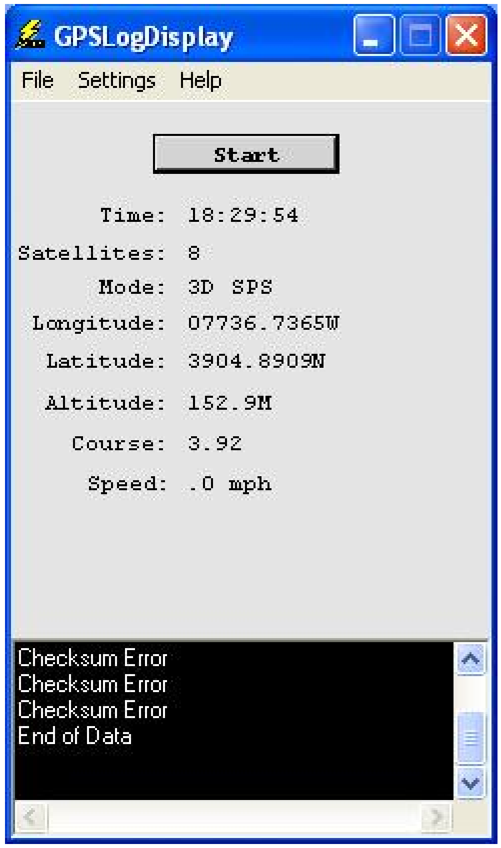


Figure 6

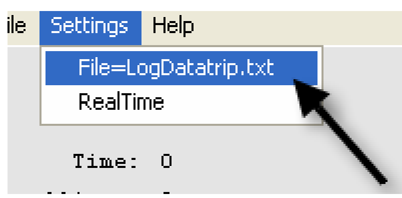


Figure 7

You have the option of displaying the data as fast as your computer can process the data, or in real-time by setting the RealTime menu shown in Figure 8. When in real-time, the data will be processed based on the UTC time stamp in the message. What the program does is look for differences in the seconds in the UTC field. When it sees a discrepancy it delays the program for one second.

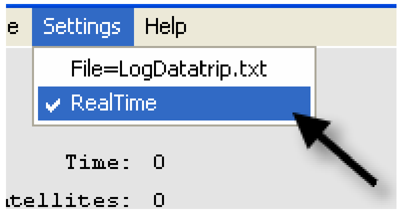


Figure 8

For actual plotting you can use the program called GPSLogPlot shown in Figure 9. This program will allow you to plot your actual trip. By default, the program sets the scale to 200. This divides plot points by 200 thus shrinking the plot to fit on the display. You can change this using the settings menu. When plotting short distances, use a smaller scale.

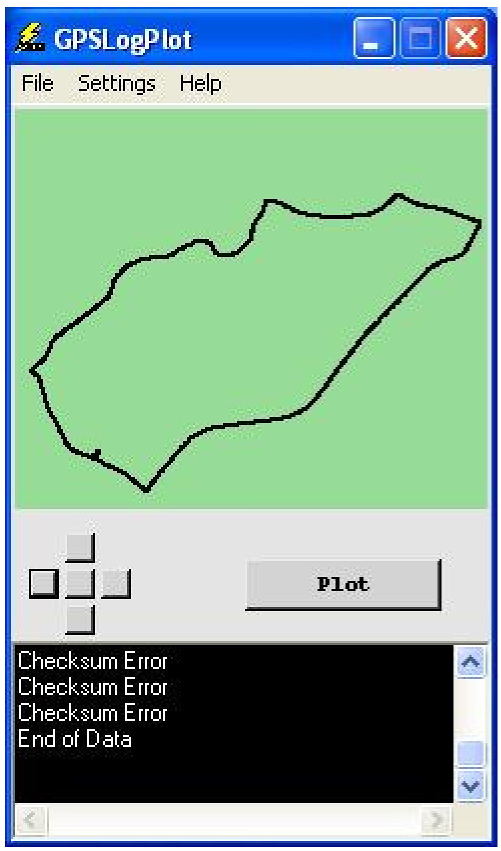


Figure 9

When you start the plot, the first valid point becomes the reference starting point that will be, by default, the center point on the display. You can change this point by changing the Start x and Start y points in the settings menu. The actual plot area is a 1000 x 1000 grid. You can change the view of this grid by using the small pad on the form shown in Figure 10. The center button will center the view to its default.

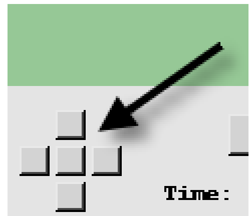


Figure 10

The plots shown in Figure 11 were all captured with the GPSDataLogger and my pocket PC using the BT359W shown in Figure 12. This is the most accurate GPS I have ever owned. The main reason I have not showcased it in this series is that it is a Bluetooth only receiver. You can use the same interface program as the Holux GPSLim236. Unlike the GPSLim236 the BT359W does supports WAAS.

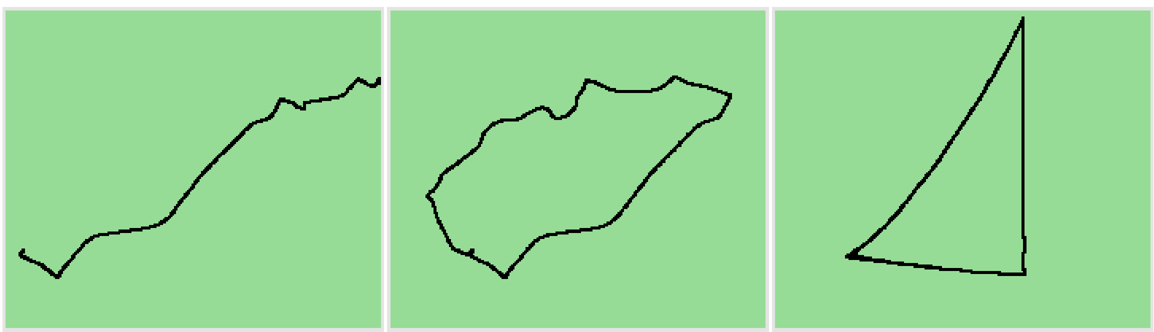


Figure 11

Figure 12

GPS Parsing Software

While I have included the compiled version of the programs presented in this article, I have included the source code for those that may want to roll their own. Each of the programs parses the GGA, RMC, GSV, and GSA NMEA commands. The main NMEA processor function is called **ProcNMEA**. This function calls four functions to handle the parsing of these commands. Each function populates a set of global variables as shown in Table 1. These variables map to the fields in the NMEA specification. One exception is the **GGA_FIXtxt** variable, which contains an actual description of the FIX type.

Function	Variable Populated
procGGA	GGA_UTCTime GGA_Latitude GGA_NS GGA_Longitude GGA_EW GGA_FIX GGA_FIXtxt GGA_Sats GGA_HDOP GGA_AltValue GGA_AltUnit GGA_Sep GGA_SepUnits GGA_Age GGA_Diff
procRMC	RMC_UTC RMC_Status RMC_Latitude RMC_NS RMC_Longitude RMC_EW RMC_SOG RMC_COG RMC_Date RMC_Variation RMC_Mode
procGSV	GSV_SATSINVIEW GSV_NOM GSV_MSG GSV_SATIDS(x) GSV_SATELE(x) GSV_SATAZ(x) GSV_SATSNR(x) When GSV_NOM = GSV_MSG then all data has been collected. At that point you should set GSV_NOM = 0

procGSA	GSA_SATMODE GSA_SATCOUNT

Table 1

Take a look at the Dispit function shown in Program Snipit 1. This is the heart of the GPSLogDisplay program. This function is called when the Start button is pressed. The function opens the log file you have selected, then enters a processing loop. In each iteration of the loop, the abort button is checked ann a line of data is retrieved from the log file. If the end of file is reached or the abort button is hit, the file is closed and the function exits. Each line retrieved from the log file is passed to the procNMEA function and only when a GGA message is received does the display get updated.

```
'-----------------------------
'Get and display the data
'-----------------------------
func Dispit()

   dim tstr as string
   dim newtime as string
   dim oldgpstime as string

   FormMenu(0,0,0,"")
   FormButton(Disp_Start,-1,-1,-1,-1,"Abort")

  'First Open the File
   if FileOpen(1,gfname,Open) = 0 then
      msgbox("Unable to open file: "+gfname,0,"Open File")
      FormMenu(0,0,1,"")
      FormButton(Disp_Start,-1,-1,-1,-1,"Start")
      exit()
   endif

'===========================================================
'----- Main Data Display Loop ------------------------------
loop:

   if FormButton(Disp_Start,0) > 0 then
     FileClose(1)
     FormMenu(0,0,1,"")
     FormButton(Disp_Start,-1,-1,-1,-1,"Start")
     exit()
   endif


   if FileEOF(1) = 1 then
      FileClose(1)
      Print "End of Data"
```

```
    FormMenu(0,0,1,"")
    FormButton(Disp_Start,-1,-1,-1,-1,"Start")
    exit()
  endif

 '----- Read a Line of data from Log File ----------
  procNMEA(FileReadLine(1))

 '----- If we get a GGA message lets update the display
  strif NMEAmsg = "GGA" then
    newtime=converttime(GGA_UTCTime,-5))
    FormLabel(Disp_time,-1,-1,-1,-1,newtime)
    Formlabel(Disp_Fix,-1,-1,-1,-1,GGA_FIXtxt)
    Formlabel(Disp_mode,-1,-1,-1,-1,GSA_SATMODE)
    Formlabel(Disp_sats,-1,-1,-1,-1,GSA_SATCOUNT)
    GSV_NOM=0
    GSV_MSG=0

    if GGA_Fix <> 0 then
       Formlabel(Disp_Longitude,-1,-1,-1,-1,GGA_Longitude+GGA_EW)
       Formlabel(Disp_Latitude,-1,-1,-1,-1,GGA_Latitude+GGA_NS)
       Formlabel(Disp_Alt,-1,-1,-1,-1,GGA_AltValue+GGA_AltUnit)
       Formlabel(Disp_Course,-1,-1,-1,-1,RMC_COG)
       Formlabel(Disp_Speed,-1,-1,-1,-1,Format(float(RMC_SOG * 1.1508),".0")+" mph")
     else
       Formlabel(Disp_Longitude,-1,-1,-1,-1,"")
       Formlabel(Disp_Latitude,-1,-1,-1,-1,"")
       Formlabel(Disp_Alt,-1,-1,-1,-1,"")
       Formlabel(Disp_Course,-1,-1,-1,-1,"")
       Formlabel(Disp_Speed,-1,-1,-1,-1,"")
     endif

    '--- Used for realtime display option
     strif oldgpstime <> newtime then
       oldgpstime = newtime
       if realtime = 1 then pause(1000)
     endif

  endif

  goto loop

endfunc
```

Program Snipit 1

The plotit function in the GPSLogPlot program is very similar to the dispit function, with the exception of how the GPS information is presented. The plotit function uses a special command built into the Zeus languages called GPSCVTLongitudedec and GPSCVTLatitudedec to convert the GPS positional string data to an integer value in degrees * 100000. This is a whole number that can be used for plotting.

One final variation of the dispit function is the StartCapture function used in the GPSDataLogger program. In this function, a com port is opened and its parameters are set based on the actual device selected. The function also calls various setup functions to place the device into the correct mode when needed. Instead of calling the procNMEA function directly, data from the device is added to a global variable called rxdat when it is received. A call is then made to a function called procdata. This functions pulls a single line from the rxdat variable one at a time and passes them to the procNMEA command as before.

Sending Log Data

Plotting and displaying data is cool to play with, but the main reason we want to capture the data is so that we can simulate an actual GPS module or receiver. I have included a program called GPSLogOutput shown in Figure 13. GPSLogOutput allows you to play back the captured log data to a serial port. The program looks and operates much like the GPSLogDisplay program, but also sends a copy of the captured data to a serial com port. You select the com port via the Settings menu shown in Figure 14. You can also set the baud rate and flag the data to be sent in real time.

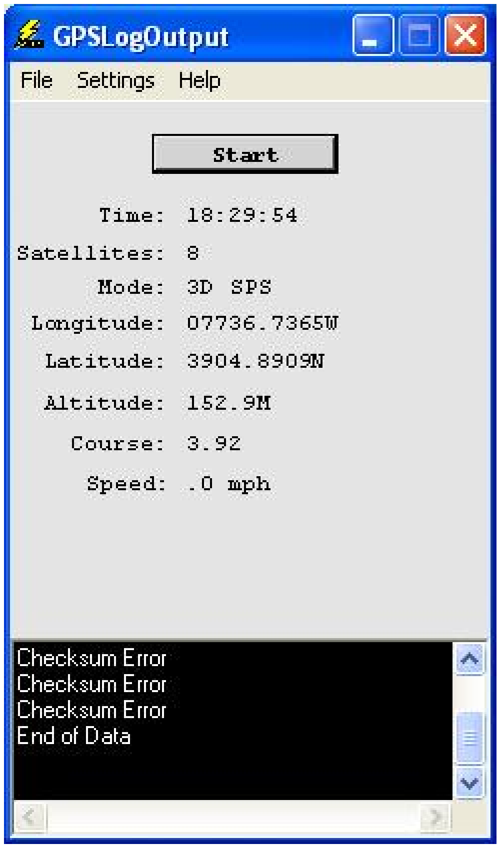


Figure 13

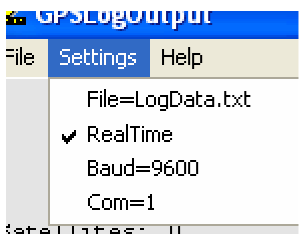


Figure 14

Using the Log Data with a Microcontroller

Next mont,h when we start to interface the GPS modules to a microcontroller, the GPSLogOutput program will be indispensable. In addition to your PC, you will need a DiosPro Microcontroller and a carrier board. I will be using the Dios Workboard Deluxe shown in Figure 15. The DiosPro has a UART built into the chip that has a TTL interface. This is perfect for the modules, but in order to use our PC as a simulator you will need an EZRS232 interface shown in Figure 16.

Figure 15

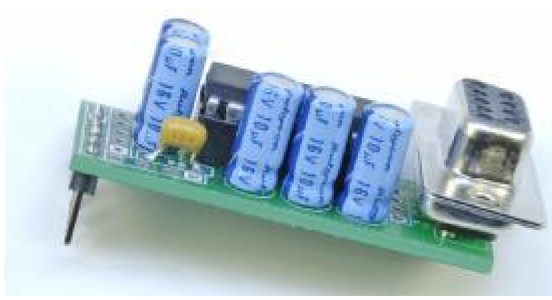


Figure 16

In order to use the GPSLogOutput program you will need two serial ports on your PC. One port will connect to the program port on the Workboard and the other will connect to the EZRS232 module. Connect the following pins on the EZRS232 module to the Dios Workboard as shown in Figure 17.

EZRS232 Pin 1 - Workboard VSS
EZRS232 Pin 2 - Workboard VDD
EZRS232 Pin 3 - Workboard Port 8
EZRS232 Pin 4 - Workboard Port 9

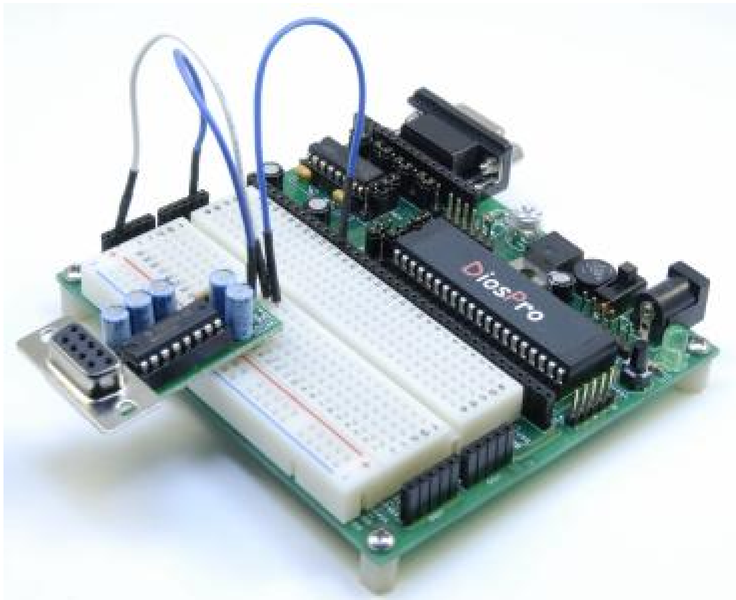


Figure 17

Load code shown in Program 1 into the DiosPro compiler and program the chip. Once loaded, start the GPSLogOutput program and load up one of the LogData files I have included. Set the GPSLogOutput com port to the one that is connected to the EZRS232 module. Set the baud rate to 4800 as shown in Figure 18.

```
'DiosProg1.txt
func main()
  dim val
  hsersetup baud,HBAUD4800,start,txon

nodata:
  hserin nodata,val
  debug val

  goto nodata

endfunc
```

Program 1

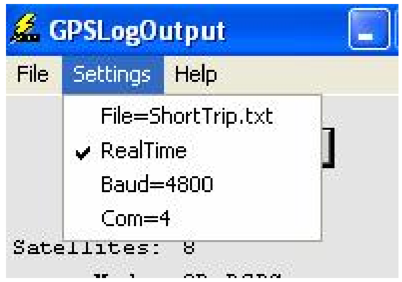


Figure 18

Once this is done, hit the start program. You should see NMEA data in the debug terminal of the Dios compiler as shown in Figure 19.

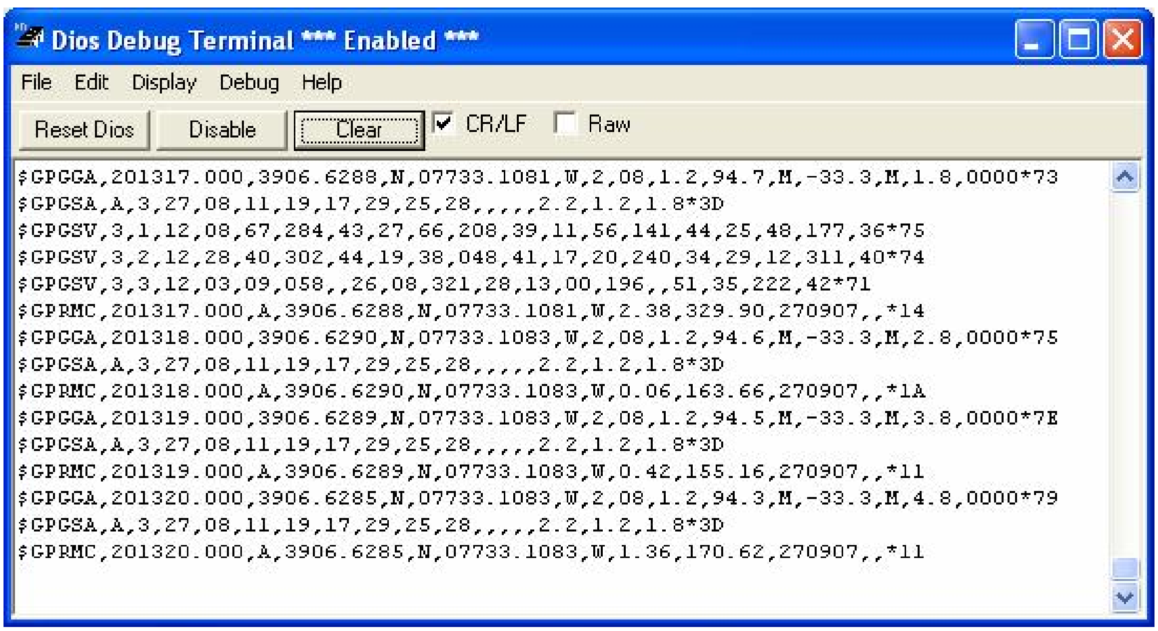


Figure 19

It just so happens that the DiosPro already has a library called DiosNMEA. It is automatically loaded when you place a call to the procNMEA function, in your Dios program as shown in Program 2.

```
'Dios NMEA Proccessor
func main()

  clear
  hsersetup baud,HBAUD4800,start,txon,clear
  print "Mode Lat   Long      Alt    Speed  Dir"
```

```
  print "---- ----- ------    -----  -----  -----"

loop:
  procNMEA()

  if NMEAcmd = 3 then 'GGA
    if NMEAfix > 0 then
        print NMEAfix,":",NMEAsats," ",{-6.0} NMEAlatmin," ",NMEAlongmin;
        print " ",{6.1} NMEAaltitude," ",{4.1} NMEAspeed," ",NMEAdir
    else
        print "No Fix ",NMEAfix,":",NMEAsats
    endif
  endif

  goto loop

endfunc

include \lib\DiosNMEA.lib
```

Program 2

This library will break down the GGA and RMC commands and load up a set of global variables that you can use in your own program. In Program 2, I used the print command to send various pieces of NMEA data to the debug terminal shown in Figure 20.

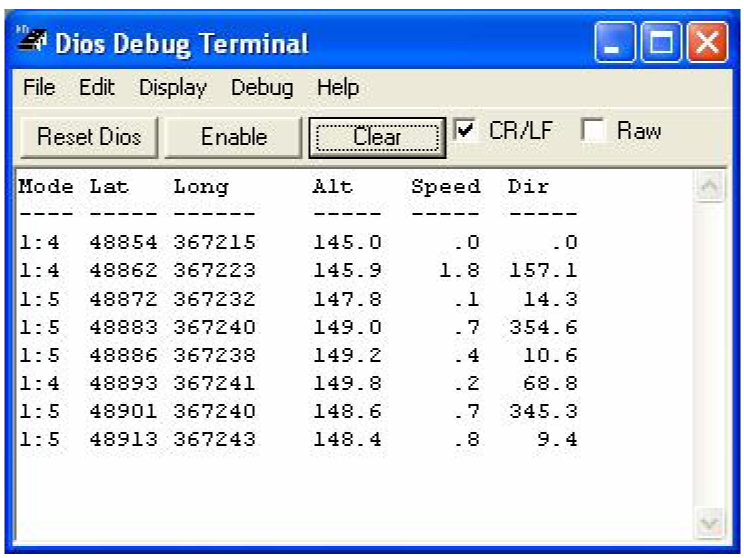


Figure 20

What's Next

Next month I'm going to show you how to connect the various GPS modules to the microcontroller and how to parse the data.

Be sure to check for updates and downloads for this article at:
http://www.kronosrobotics.com/Projects/GPS.shtml

Parts

The following is a breakdown of the source for all the components needed for Parts 2 and 3 of this project.

Spark Fun Electronics

EM-406A GPS module
http://www.sparkfun.com/commerce/product_info.php?products_id=465

EM-406 Evaluation Board
http://www.sparkfun.com/commerce/product_info.php?products_id=653

EM-408 GPS Module
http://www.sparkfun.com/commerce/product_info.php?products_id=8234

Copernicus Evaluation Board
http://www.sparkfun.com/commerce/product_info.php?products_id=8145

9-Pin Serial Cable
http://www.sparkfun.com/commerce/product_info.php?products_id=65

6V AC Adapter
http://www.sparkfun.com/commerce/product_info.php?products_id=737

External Antenna with SMA connector
http://www.sparkfun.com/commerce/product_info.php?products_id=464

SMA to MMCX adapter cable
http://www.sparkfun.com/commerce/product_info.php?products_id=285

KRMicros

ZeusPro

http://www.krmicros.com/Development/ZeusPro/ZeusPro.htm

KronosRobotics

EZRS232
http://www.kronosrobotics.com/xcart/product.php?productid=16167

DiosPro Chip
http://www.kronosrobotics.com/xcart/product.php?productid=16428

Dios WorkBoard Deluxe
http://www.kronosrobotics.com/xcart/product.php?productid=16452