LinksPoint Documentation

Using GlobalPoint GPS Receivers in NMEA Mode

October 2006

LINKSPOINT

One Selleck Street
Suite 330
Norwalk, CT 06855
www.linkspoint.com

Phone 203.853.4600
Fax 413.473.4600
Latitude: 41.10705
Longitude: -73.40905

Using GlobalPoint GPS Receivers in NMEA Mode

Introduction

Most GPS application developers can use the Links Point GPS Toolkit to simplify the development process by interfacing to the GPS hardware either in its native SiRF protocol, or in the configurable NMEA mode. The Toolkit is available as a C style API, compatible with applications developed in C, C++ and .NET, or as an ActiveX control, compatible with Embedded Visual Basic.

The GPS Toolkit, however, does not support certain rapid application development tools. In these cases, the application must parse data from the GPS receiver directly. This document identifies a method to interface to the GlobalPoint GPS receiver without using the LinksPoint GPS Toolkit.

Changing the Protocol

The default GPS protocol for LinksPoint GPS receivers is NMEA at 4800 baud so it should be unnecessary to change the protocol as described in this section. Some of LinksPoint's older GPS receivers output data in the SiRF binary protocol at 38400 baud by default. If you are not sure, you can confirm which protocol is in use by downloading and using our GlobalPoint GPS software (available at www.linkspoint.com/hardware.asp). Install the software, tap on "Tools" and then "Configure." Use the "Auto Detect" command and the software will determine which protocol and baud rate is in use. If the protocol is SiRF, LinksPoint recommends switching the protocol to SiRF. By using the method explained below, each time the GPS receiver is powered on, the software will send a simple binary command (shown in Table 1) to the receiver to switch the protocol to NMEA, a simpler ASCII protocol, at 4800 baud. The command will have no impact on most occasions, since most power cycles have no effect on the receiver protocol.

- Table 1 – Switch to NMEA Command

Byte Number	Binary Value (Hexadecimal)	Description
1	A0	Start sequence
2	A2	
3	00	Payload length
4	18	
5	81	Message ID
6	02	Set mode to NMEA
7	01	Transmit GGA messages at 1 second intervals
8	01	Transmit checksum with GGA message
9	00	Do not transmit GLL messages
10	01	Transmit checksum with GLL message
11	05	Transmit GSA messages at 5 second intervals
12	01	Transmit checksum with GSA message
13	05	Transmit GSV messages at 5 second intervals
14	01	Transmit checksum with GSV message
15	00	Do not transmit RMC
16	01	Transmit checksum with RMC message
17	00	Do not transmit VTG
18	01	Transmit checksum with VTG message
19	00	Unused fields
20	01	
21	00	
22	01	
23	00	
24	01	
25	00	
26	01	
27	12	Transmit data at a baud rate of 4800
28	C0	
29	01	Checksum
30	6A	
31	B0	End sequence
32	B3	

Sending the binary command in Embedded Visual Basic 3.0

Microsoft Embedded Visual Basic 3.0 has a documented bug in its CE Comm control that makes it impossible to send binary data using this control. To work around the problem, eVB developers have to make an API call to SendFile. The following code accomplishes the task. The code assumes that a Comm control named Comm1 and a Command Button named Command1 are on the form.

```
Option Explicit

Declare Function WriteFileL Lib "Coredll" Alias "WriteFile" _
   (ByVal hFile As Long, lpBuffer As Byte, ByVal nNumberOfBytesToWrite As Long, _
   lpNumberOfBytesWritten As Long, ByVal lpOverlapped As Long) As Long

Public Sub SendArrayData(ByVal hCommID As Long, baData)
   Dim i, lRet, iWrite
   For i = LBound(baData) To UBound(baData)
      lRet = WriteFileL(hCommID, baData(i), 1, iWrite, 0)
   Next
End Sub

Private Sub Command1_Click()
   ' Set up the "Switch to NMEA" command
   Dim Command(32) As Byte
   Command(1) = &HA0
   Command(2) = &HA2
   Command(3) = &H0
   Command(4) = &H18
   Command(5) = &H81
   Command(6) = &H2
   Command(7) = &H1
   Command(8) = &H1
   Command(9) = &H0
   Command(10) = &H1
   Command(11) = &H5
   Command(12) = &H1
   Command(13) = &H5
   Command(14) = &H1
   Command(15) = &H0
   Command(16) = &H1
   Command(17) = &H0
   Command(18) = &H1
   Command(19) = &H0
   Command(20) = &H1
   Command(21) = &H0
   Command(22) = &H1
   Command(23) = &H0
   Command(24) = &H1
   Command(25) = &H0
   Command(26) = &H1
   Command(27) = &H12
   Command(28) = &HC0
   Command(29) = &H1
   Command(30) = &H6A
   Command(31) = &HB0
   Command(32) = &HB3

   ' Configure the comm port
   Comm1.InputLen = 0
   Comm1.InputMode = comInputModeText
   Comm1.RThreshold = 1
   Comm1.SThreshold = 1
   Comm1.CommPort = 1                  'Assume the GPS is connected to COM1:
   Comm1.Settings = "38400,N,8,1"   'Assume the GPS is in SiRF mode

   'Send the command
   Comm1.PortOpen = True
   SendArrayData Comm1.CommID, Command
   Comm1.PortOpen = False
End Sub
```

• Listing 1 – Configuring the Receiver for SiRF in Embedded Visual Basic

Parsing GGA - Global Positioning System Fix Data

When the GlobalPoint receiver has been switched to the NMEA protocol using the command listed above, it will output several NMEA 0183 sentences at 4800 baud. The full NMEA specification is available at www.NMEA.org, but most applications require only the GGA Sentence described here.

All NMEA sentences use the ASCII character set, begin with a '$' character and end with a carriage return/linefeed pair. To parse the message, program your software to look for the '$' character and 'GGA' sentence ID before examining the data fields. Once these have been found, extract and parse each of the comma-delimited fields until the carriage return/linefeed pair is encountered in the data.

Structure:

```
$GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,,,,0000*hh<CR><LF>
1 2    3    4         5   6    7     8 9 10 11  1  13    17  18
```

Example:

```
$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M,,,,0000*18
```

Fields Explained:

1. *Start character* – NMEA sentences always start with '$'.
2. *Talker ID* – Always 'GP'.
3. Sentence ID – 'GGA' for the Global Positioning System Fix Data sentence.
4. *Time of day* - Expressed as Universal Time Coordinated (UTC). 2 fixed digits of hours, 2 fixed digits of minutes and a variable number of digits for decimal fractions of minutes. Leading zeros always included for hours, minutes and seconds to maintain fixed length.
5. *Latitude* – Expressed as WGS84 referenced geodetic latitude. 2 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length.
6. *Direction of latitude* - 'N' for north, or positive, latitude; 'S' for south, or negative, latitude.
7. *Longitude* – Expressed asWGS84 referenced geodetic longitude. 3 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length.
8. *Direction of longitude* - 'E' for east, or positive, longitude; 'W' for west, or negative, longitude.
9. *GPS Quality Indicator* - Possible values are:

0	Fix not available or invalid
1	Fix valid, uncorrected
2	Fix valid, WAAS corrected.
6	Estimated (dead reckoning) mode

10. *Number of satellites in view* - Range 0 to 12.
11. *HDOP* - Horizontal Dilution of Precision. Range 0.0 to 50.0.
12. *Height above below mean geoid*. Expressed in meters. This is not the same as altitude above mean sea level.
13. *Units of height* - always 'M' for meters.
14. *Geoidal separation* - Not supported. Null field.
15. *Units of geoidal separation* - Not supported. Null field.
16. *Age of differential GPS data*. Not supported. Null field.
17. *Differential reference station ID* – Not supported. Always '0000'.
18. *Checksum* – Proceeded by '*'. Calculated as the 8-bit exclusive or sum of all the characters between the '$' delimiter and the '*' delimiter, not including either delimiter. The 8 bit result is represented as two ASCII hexadecimal characters.

Summary

Though straightforward in concept, and easy to implement in its most basic form, there are a number of potential pitfalls involved in including GPS into your application. The LinksPoint GPS Toolkit has, over several years of development, incorporated techniques to conserve battery life, increase GPS signal reception, improve GPS acquisition time and enhance software performance -- all enhancements that could mean the difference between a successful and an unsuccessful implementation. While we provide the information in this document to help when the Toolkit is not an option, we strongly recommend the use of the Toolkit whenever possible.