NetMedia

Basic Express
Application Note

Decoding GPS Text Data Using BasicX

Introduction

GPS receivers are generally capable of transmitting data in several formats. One format, called *Simple Text Output* protocol, is one of several modes available from GARMIN® receivers such as eTrex™. The format is documented here:

http://www.garmin.com/support/text_out.html

In this format, GPS data consists of a simple 7 bit ASCII text string with a constant length of 57 bytes. On eTrex receivers, the string is transmitted about once per second.

A BasicX microcontroller can be used to extract GPS time, position and velocity data embedded in the text strings. Single precision floating point variables are used to store 3D position and velocity fixes.

Hardware interface

Figure 1 (below) illustrates the hardware interface between a GPS receiver and BasicX system:

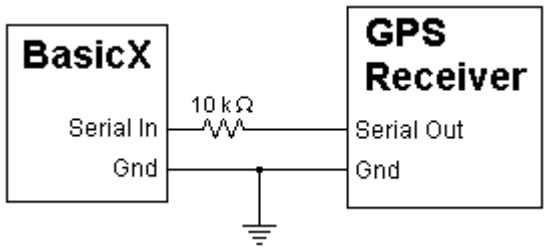

Figure 1

The BasicX serial input pin number on depends on the system. For BX-01 systems, port Com2 is used on pin 12. For BX-24 and BX-35 systems, port Com3 is used on pin 16.

If you use a GARMIN eTrex or similar receiver, it should be set to “text out” mode at 1200 baud.

Software Interface

In the BasicX demonstration program, GPS data is decoded and written to the following arrays:

```
Name       Type     Dimensions
-----------------------------
TimeGPS    Byte         6
Position   Float        3
Velocity   Float        3
```

Each array element is defined as follows:

```
Array
Element       Definition
------------------------

TimeGPS(1)    Year (2 digit)
TimeGPS(2)    Month
TimeGPS(3)    Day
TimeGPS(4)    Hour
TimeGPS(5)    Minute
TimeGPS(6)    Second

Position(1)   Longitude (deg)
Position(2)   Latitude (deg)
Position(3)   Altitude (m)

Velocity(1)   East (m/s)
Velocity(2)   North (m/s)
Velocity(3)   Up (m/s)
```

For position fixes, the sign convention is such that North latitude is positive and South is negative. Similarly East longitude is positive and West is negative. Position and velocity vectors are oriented assuming a right-hand coordinate system.

Whenever a GPS fix is decoded, selected elements are also transmitted out the BasicX Com1 serial port configured to 19 200 baud, 8 data bits, 1 stop bit, no parity. Example format:

```
Time:  16:27:48
 Lat:  32.33132°
 Lon:  -111.0172°
 Alt:  741 m
  Ve:  1.5 m/s
  Vn:  0.2 m/s
  Vu:  0.01 m/s
```

Sometimes the GPS receiver may not pick up signals from enough satellites for complete position and velocity fixes. In these cases, data may not be available, or may be only partially available. The text string typically includes underscore characters in fields that are not available.

If this happens, the BasicX program displays asterisks in fields that are affected. As an example, if time is available, but position and velocity are not, the following format is displayed:

```
Time:  16:27:48
 Lat:  *
 Lon:  *
 Alt:  *
  Ve:  *
  Vn:  *
  Vu:  *
```

Raw text display. Instead of decoded data, you can choose to display untranslated text as received from the GPS receiver. In the main program, the Boolean constant RawText controls this selection. A typical line of raw text looks like this:

```
@021021183228N3224250W11056803G006+00782E0000N0000U0001
```

Program listing. The program GPSTextDecoder consists of the following 4 modules, which are included as separate files along with this application note:

GPSTextDecoder.bas – Includes the main program and other high level code

GPSDriver.bas – Does the low level decoding of text data

GPSSerialPort.bas – Handles the GPS serial port

GPSPinAssignments.bas **--** Defines the serial I/O pin number and port number

Each file is listed on the following pages.

The program is written for a BX-24 system. If you want to port the program to other BasicX systems, most code changes should be localized to module GPSPinAssignments, and are flagged by comments.

Module GPSTextDecoder

```
Attribute VB_Name = "GPSTextDecoder"
'--------------------------------------------------------------------------------
Option Explicit

Private Const YearIndex   As Byte = 1
Private Const MonthIndex  As Byte = 2
Private Const DayIndex    As Byte = 3
Private Const HourIndex   As Byte = 4
Private Const MinuteIndex As Byte = 5
Private Const SecondIndex As Byte = 6

Private TimeGPS(1 To 6) As Byte
Private Position(1 To 3) As Single
Private Velocity(1 To 3) As Single

Private TimeIsValid As Boolean
Private PositionIsValid As Boolean
Private VelocityIsValid As Boolean

Private Const EastIndex  As Byte = 1
Private Const NorthIndex As Byte = 2
Private Const UpIndex    As Byte = 3
'--------------------------------------------------------------------------------
Public Sub Main()

' This program decodes GPS data in Garmin Simple Text Output format. The GPS
' data is received over a serial port.

    ' Set this flag to true to display raw text from the GPS receiver.
    Const RawText As Boolean = False

    Call Initialize

    Debug.Print
    Debug.Print "GPS text decoder"
    Delay 0.5
    Debug.Print
    Debug.Print "Waiting for GPS data..."
    Debug.Print

    If (RawText) Then
        Call DisplayRawText
    End If

    Do
        Call GetFixGPS(TimeGPS, Position, Velocity, TimeIsValid, _
            PositionIsValid, VelocityIsValid)

        Call DisplayData
    Loop

End Sub
'--------------------------------------------------------------------------------
```

```
'--------------------------------------------------------------------------------
Private Sub DisplayRawText()

' This procedure displays raw text received from the GPS receiver.

    Do
        Debug.Print Chr(GetByte);
    Loop

End Sub
'--------------------------------------------------------------------------------
Private Sub Initialize()

    Dim N As Byte

    TimeIsValid = False
    PositionIsValid = False
    VelocityIsValid = False

    Call InitializeGPS

End Sub
'--------------------------------------------------------------------------------
Private Sub DisplayData()

' This procedure displays decoded GPS data.

    Const Deg As Byte = 176 ' Degree symbol.

    Debug.Print

    Debug.Print " Time:  ";
    If (TimeIsValid) Then
        Call DisplayTime
    Else
        Debug.Print "*"
    End If

    If (PositionIsValid) Then
        Debug.Print "  Lat:  "; CStr(Position(NorthIndex)); Chr(Deg)
        Debug.Print "  Lon:  "; CStr(Position(EastIndex));  Chr(Deg)
        Debug.Print "  Alt:  "; CStr(Position(UpIndex));    " m"
    Else
        Debug.Print "  Lat:  *"
        Debug.Print "  Lon:  *"
        Debug.Print "  Alt:  *"
    End If

    If (VelocityIsValid) Then
        Debug.Print "   Ve:  "; CStr(Velocity(EastIndex));  " m/s"
        Debug.Print "   Vn:  "; CStr(Velocity(NorthIndex)); " m/s"
        Debug.Print "   Vu:  "; CStr(Velocity(UpIndex));    " m/s"
    Else
        Debug.Print "   Ve:  *"
        Debug.Print "   Vn:  *"
        Debug.Print "   Vu:  *"
    End If

End Sub
'--------------------------------------------------------------------------------
```

```
'--------------------------------------------------------------------------------
Public Sub DisplayTime()

' This procedure displays time in HH:MM:SS format.

    Dim B As Byte
    Dim MSDig As Byte
    Dim LSDig As Byte

    B = TimeGPS(HourIndex)
    LSDig = B Mod 10
    B = B \ 10
    MSDig = B Mod 10
    Debug.Print Chr(MSDig + 48) & Chr(LSDig + 48) & ":";

    B = TimeGPS(MinuteIndex)
    LSDig = B Mod 10
    B = B \ 10
    MSDig = B Mod 10
    Debug.Print Chr(MSDig + 48) & Chr(LSDig + 48) & ":";

    B = TimeGPS(SecondIndex)
    LSDig = B Mod 10
    B = B \ 10
    MSDig = B Mod 10
    Debug.Print Chr(MSDig + 48) & Chr(LSDig + 48)

End Sub
'--------------------------------------------------------------------------------
```

Module GPSDriver

```
Attribute VB_Name = "GPSDriver"
'--------------------------------------------------------------------------------
Option Explicit

' TimeGPS indexes.
Private Const YearIndex   As Byte = 1
Private Const MonthIndex  As Byte = 2
Private Const DayIndex    As Byte = 3
Private Const HourIndex   As Byte = 4
Private Const MinuteIndex As Byte = 5
Private Const SecondIndex As Byte = 6

' FixGPS indexes.
Private Const LonDegIndex As Byte = 1
Private Const LonMinIndex As Byte = 2
Private Const LatDegIndex As Byte = 3
Private Const LatMinIndex As Byte = 4
Private Const AltIndex    As Byte = 5
Private Const VelEIndex   As Byte = 6
Private Const VelNIndex   As Byte = 7
Private Const VelUIndex   As Byte = 8

Private PositionStatus As Byte
Private EPH As Integer

Private Const ChrPlus  As Byte = 43   ' +
Private Const ChrEast  As Byte = 69   ' E
Private Const ChrNorth As Byte = 78   ' N
Private Const ChrUp    As Byte = 85   ' U

Private Const MaxDigits As Byte = 5
Private PowTenLookup(1 To MaxDigits) As Integer

Private DataIsValid As Boolean
'--------------------------------------------------------------------------------
Public Sub InitializeGPS()

    PowTenLookup(1) = 1
    PowTenLookup(2) = 10
    PowTenLookup(3) = 100
    PowTenLookup(4) = 1000
    PowTenLookup(5) = 10000

    Call OpenSerialPortGPS

End Sub
'--------------------------------------------------------------------------------
```

```
'-------------------------------------------------------------------------------
Public Sub GetFixGPS( _
    ByRef TimeGPS() As Byte, _
    ByRef Position() As Single, _
    ByRef Velocity() As Single, _
    ByRef TimeIsValid As Boolean, _
    ByRef PositionIsValid As Boolean, _
    ByRef VelocityIsValid As Boolean)

    Const SentenceStart As Byte = 64    ' "@"

    Dim N As Byte
    Dim Temp As Byte
    Dim Sign As Integer
    Dim FixGPS(LonDegIndex To VelUIndex) As Integer

    '---------------
    ' Start sentence                            ' Character
    '---------------                            ' Position

    Do Until (GetByte = SentenceStart)          ' 1
        ' Null
    Loop

    '-----
    ' Time
    '-----

    DataIsValid = True
    TimeIsValid = True

    For N = 1 To 6
        TimeGPS(N) = CByte(GetNumber(2))        ' 2 .. 13
    Next

    ' Error check.
    If (Not DataIsValid) Then
        TimeIsValid = False
    End If

    '---------
    ' Position
    '---------

    DataIsValid = True
    PositionIsValid = True

    ' Latitude.
    Sign = GetSign(ChrNorth)                    ' 14
    FixGPS(LatDegIndex) = GetNumber(2) * Sign   ' 15 .. 16
    FixGPS(LatMinIndex) = GetNumber(5)          ' 17 .. 21

    ' Longitude
    Sign = GetSign(ChrEast)                     ' 22
    FixGPS(LonDegIndex) = GetNumber(3) * Sign   ' 23 .. 25
    FixGPS(LonMinIndex) = GetNumber(5)          ' 26 .. 30

    PositionStatus = GetByte                    ' 31

    EPH = GetNumber(3)                          ' 32 .. 34
```

```
    ' Altitude.
    Sign = GetSign(ChrPlus)                         ' 35
    FixGPS(AltIndex) = GetNumber(5) * Sign          ' 36 .. 40

    ' Error check.
    If (Not DataIsValid) Then
        PositionIsValid = False
    End If

    '---------
    ' Velocity
    '---------

    DataIsValid = True
    VelocityIsValid = True

    ' Velocity east/west.
    Sign = GetSign(ChrEast)                         ' 41
    FixGPS(VelEIndex) = GetNumber(4) * Sign         ' 42 .. 45

    ' Velocity north/south.
    Sign = GetSign(ChrNorth)                        ' 46
    FixGPS(VelNIndex) = GetNumber(4) * Sign         ' 47 .. 50

    ' Velocity up/down.
    Sign = GetSign(ChrUp)                           ' 51
    FixGPS(VelUIndex) = GetNumber(4) * Sign         ' 52 .. 55

    ' Error check.
    If (Not DataIsValid) Then
        VelocityIsValid = False
    End If

    '-------------
    ' End sentence
    '-------------

    Temp = GetByte                                  ' 56
    Temp = GetByte                                  ' 57

    '-----------------
    ' Unit conversions
    '-----------------

    Position(1) = ToDegrees(FixGPS(LonDegIndex), FixGPS(LonMinIndex))
    Position(2) = ToDegrees(FixGPS(LatDegIndex), FixGPS(LatMinIndex))
    Position(3) = CSng(FixGPS(AltIndex))

    Velocity(1) = CSng(FixGPS(VelEIndex)) / 10.0
    Velocity(2) = CSng(FixGPS(VelNIndex)) / 10.0
    Velocity(3) = CSng(FixGPS(VelUIndex)) / 100.0

End Sub
'-------------------------------------------------------------------------------
```

```
'-------------------------------------------------------------------------------
Public Function GetNumber( _
    ByVal DigitCount As Byte) As Integer

' This function converts a decimal string to an integer. The string is of
' length DigitCount.
'
' Each character must be a decimal digit -- otherwise the character is
' illegal. If any illegal characters are found, they're treated as
' equivalent to "0" and the error flag DataIsValid is set to False.

    Dim PowTen As Integer
    Dim N As Byte
    Dim InByte As Byte

    PowTen = PowTenLookup(DigitCount)
    GetNumber = 0

    For N = 1 To DigitCount

        InByte = GetByte

        ' Check for legal decimal digit.
        If (InByte >= 48) And (InByte <= 57) Then
            GetNumber = GetNumber + (CInt(InByte - 48) * PowTen)
        Else
            DataIsValid = False
        End If

        PowTen = PowTen \ 10
    Next

End Function
'-------------------------------------------------------------------------------
Public Function GetSign( _
    ByVal Value As Byte) As Integer

' This function reads a byte and returns a sign value that depends on the
' specified number Value. If the new byte matches, the sign is positive.
' Otherwise the sign is negative.

    If (GetByte = Value) Then
        GetSign = 1
    Else
        GetSign = -1
    End If

End Function
'-------------------------------------------------------------------------------
```

```
'-------------------------------------------------------------------------------
Public Function GetByte() As Byte

' This function reads the next byte from the input device. If the byte matches
' InvalidTag, the flag DataIsValid is set to False.

    Const InvalidTag As Byte = 95  ' "_"

    GetByte = GetByteGPS

    If (GetByte = InvalidTag) Then
        DataIsValid = False
    End If

End Function
'-------------------------------------------------------------------------------
Private Function ToDegrees( _
    ByVal Degrees As Integer, _
    ByVal mMinutes As Integer) As Single

' Converts integer degrees/minutes to float degrees. Note that mMinutes are
' in units of 10^3 minutes.

    Dim Deg As Single
    Dim Frac As Single

    Deg = CSng(Degrees)

    If (Deg >= 0.0) Then
        Frac = CSng(mMinutes)
    Else
        Frac = -CSng(mMinutes)
    End If

    ' Convert to fractional degrees.
    Frac = Frac / 60000.0

    ToDegrees = Deg + Frac

End Function
'-------------------------------------------------------------------------------
```

Module GPSSerialPort

```
Attribute VB_Name = "GPSSerialPort"
'--------------------------------------------------------------------------------
Option Explicit

Private Const InputQueueSize As Integer = 10    ' 1 byte buffer.
Private Const OutputQueueSize As Integer = 10   ' 1-byte buffer.

Private InputQueue(1 To InputQueueSize) As Byte
Private OutputQueue(1 To OutputQueueSize) As Byte

Private Const Baud As Long = 1200

Private Const NullOutputPin As Byte = 0
'--------------------------------------------------------------------------------
Public Sub OpenSerialPortGPS()

    Call OpenQueue(InputQueue, InputQueueSize)

    Call OpenQueue(OutputQueue, OutputQueueSize)

    If (PortNumber = 3) Then

        ' Inverted logic, no parity, 8 data bits.
        Call DefineCom3(SerialInputPin, NullOutputPin, bx1000_1000)
    End If

    Call OpenCom(PortNumber, Baud, InputQueue, OutputQueue)

End Sub
'--------------------------------------------------------------------------------
Public Function GetByteGPS() As Byte

    Dim Value As Byte

    Call GetQueue(InputQueue, Value, 1)

    GetByteGPS = Value

End Function
'--------------------------------------------------------------------------------
```

Module GPSPinAssignments

```
Attribute VB_Name = "GPSPinAssignments"
'--------------------------------------------------------------------------------
Option Explicit

' This module defines port and I/O pin numbers.

' BX-01 assignments.
'>>Public Const PortNumber As Byte = 2
'>>Public Const SerialInputPin As Byte = 12

' BX-24 assignments.
Public Const PortNumber As Byte = 3
Public Const SerialInputPin As Byte = 16

' BX-35 assignments.
'>>Public Const PortNumber As Byte = 3
'>>Public Const SerialInputPin As Byte = 16
'--------------------------------------------------------------------------------
```